

# SIMULACIÓN Y CONTROL DE UNA MÁQUINA DE EMBOTELLADO

francisco.marquez@uah.es

**RESUMEN** — El trabajo de la asignatura «Técnicas de la Automatización» (cód. 201987) se realizará en grupos formados por cuatro alumnos y se tendrá en cuenta en la evaluación continua ordinaria con un peso del 20 % de la calificación final. Con este trabajo los alumnos entrenarán algunas de las competencias enumeradas en el documento de verificación de la titulación y, en especial, la competencia «CTecInd8. Capacidad para diseñar y proyectar sistemas de producción automatizados y control avanzado de procesos». Para ello, los alumnos tendrán que diseñar, programar y documentar un problema de complejidad media consistente en la simulación y automatización de una máquina de embotellado. Para la presentación del trabajo se tendrán en cuenta las fechas siguientes: **(i) 22 de diciembre de 2017** en la clase de prácticas es la fecha tope para entregar/probar el simulador de la máquina (PLANTA). Si se entrega con posterioridad la calificación de esta parte se multiplicará por el factor de penalización 0,5. **(ii) 20 de enero de 2018** en la clase de prácticas es la fecha tope para entregar/probar el CONTROLADOR de la PLANTA y la documentación. Si se entregan con posterioridad la calificación de todo el trabajo será de «cero puntos».

**PALABRAS CLAVE** — Simulación, automatización, máquina de embotellado, sistema de eventos discretos, PLANTA, ESPECIFICACIÓN, CONTROLADOR.

## §1. OBJETIVOS DIDÁCTICOS

CON ESTE TRABAJO se pretende que el alumno se enfrente a un problema de automatización de cierta complejidad que le permita entrenar las competencias enumeradas en la «memoria de verificación» del plan de estudios aprobado para el Máster Universitario en Ingeniería Industrial de la UAH (BOE 8 de diciembre de 2014). Entre otras, se entrenarán las siguientes competencias:

- Básicas:
  - CB9. Que los estudiantes sepan comunicar sus conclusiones, y los conocimientos y razones últimas que las sustentan, a públicos especializados y no especializados de un modo claro y sin ambigüedades.
- Generales:
  - CG1. Capacidad de análisis y síntesis.
  - CG2. Capacidad de organización y planificación.
  - CG3. Habilidad para analizar y buscar información de fuentes diversas.
  - CG4. Capacidad de tomar decisiones.
- Transversales:
  - CT1. Capacidad para la resolución de problemas.

- CT2. Compromiso ético con el trabajo.
- CT3. Capacidad para trabajar en equipo.
- CT4. Trabajar en entornos de presión.
- CT5. Motivación por la calidad.

- Específicas:

- CTecInd8. Capacidad para diseñar y proyectar sistemas de producción automatizados y control avanzado de procesos.

Para conseguir estos objetivos el alumno tendrá que resolver (CT1) en equipo (CT3) un problema complejo de automatización (CTecInd8) partiendo de unas especificaciones (CG1) incompletas y puede que ambiguas. Esto le obligará a organizar y planificar el trabajo (CG2), tomar decisiones de diseño (CG4) y buscar información en fuentes alternativas (CG3) para que el trabajo esté finalizado en el plazo establecido (CT4) sin renunciar a la calidad del resultado (CT5). Por último, la documentación y exposición/defensa del trabajo realizado permitirá entrenar la competencia CB9.

A continuación se reproduce el artículo 34 de la «Normativa reguladora de los procesos de evaluación de los aprendizajes» de la UAH:

### «Artículo 34. Originalidad de los trabajos y pruebas.

1. La Universidad transmitirá a los estudiantes que el plagio es una práctica contraria a las normas y a los principios que rigen la formación universitaria.
2. La Universidad proporcionará a los estudiantes la formación necesaria para la elaboración de trabajos u otras pruebas de evaluación con objeto de enseñarles a manejar y citar las fuentes utilizadas, así como a desarrollar y poner en práctica las competencias requeridas.
3. Se entiende por plagio la copia de textos sin citar su procedencia y dándolos como de elaboración propia y conllevará automáticamente la calificación de suspenso (o) en los trabajos o pruebas en los que se hubiera detectado. El profesor que advierta indicios de plagio en los trabajos o pruebas de evaluación que les sean presentados dará cuenta de este hecho al decano o director del centro en un plazo máximo de dos días, para que proceda, en su caso, a ponerlo en conocimiento del Rector por si pudiera ser constitutivo de infracción disciplinaria o de delito.
4. En las guías docentes se puede incluir la previsión de que el estudiante tenga que firmar en los trabajos y materiales entregados para la evaluación de su aprendizaje una declaración explícita en la que asuma la originalidad del trabajo, entendida en el sentido de que no ha utilizado fuentes sin citarlas debidamente.»

Del artículo anterior se desprende que:

1. En la documentación de este trabajo se tienen que citar las fuentes del material no original. Sobre la forma de

hacerlo sirvan de ejemplo las citas que aparecen en este documento.

2. Al presentar textos ajenos como propios se incurre en plagio. Esto significa que «se considerará plagio presentar como propio un trabajo elaborado por otros alumnos».
3. Lo que tradicionalmente se ha llamado «copiar una práctica» ahora se denomina «plagio» y «...conllevará automáticamente la calificación de suspenso (o) en los trabajos o pruebas en los que se hubiera detectado».

Presentar trabajos originales fruto de nuestro esfuerzo y citar adecuadamente los textos/trabajos ajenos, contribuyendo así a su difusión y al reconocimiento de sus autores, es una forma de poner en práctica la competencia CT2.

## §2. ENUNCIADO DEL PROBLEMA

La Fig. 1 (pág. 3) presenta el modelo estructural de una máquina embotelladora de agua que puede realizar las tareas de envasado del líquido y taponado de las botellas. La máquina se compone de tres puestos de trabajo y queremos que su comportamiento responda al siguiente modelo funcional:

- S1. Traspaso de carga: las botellas llegan por una cinta transportadora y son transferidas a la **cinta de traspaso** mediante la acción del cilindro A. Por su parte, el cilindro B se encarga de accionar la cinta para traspasar las botellas a los puestos de llenado y de taponado. Hay que observar que la cremallera acoplada al vástago del cilindro B está diseñada mecánicamente para que actúe sobre la cinta solo en el avance del cilindro.
- S2. Llenado: al retroceder el cilindro C carga la dosis de líquido en el inyector y al avanzar lo transfiere a la botella actuando juntamente con la válvula D.
- S3. Taponado: el cilindro G presenta un tapón para que lo recoja el cilindro E, lo coloque sobre la botella y lo enrosque con el motor neumático F.
- S4. Los sensores CP1, CP2 y CP3 indican la presencia de botella en sus respectivos puestos de trabajo.
- S5. Cada cilindro dispone de dos marcas de posición:  $x_0$  ( $x \in \{a, b, c, g\}$ ) indica «vástago del cilindro X replegado» y  $x_1$  «vástago desplegado». Como  $e_0$  y  $e_1$  son presostatos (se activan a una presión seleccionada), su significado es el contrario al de las marcas anteriores:  $e_0$  se activa al aumentar la presión en la cámara de avance del cilindro con la acción  $E^+$  y  $e_1$  al aumentar la presión en la cámara de retroceso con  $E^-$ .
- S6. Para accionar cada cilindro hay una electroválvula de distribución que responde a las señales YX1 ( $X \in \{A, B, C, E, G\}$ ) para activarla e YX2 para reponerla. Para las válvulas D y F la reposición es por muelle.
- S7. Supondremos que la máquina dispone de un suministro constante de botellas, agua y tapones. Estos elementos se pueden colocar manualmente y su automatización es un problema distinto que no se contempla en este trabajo.

El problema de automatización consiste en diseñar y programar un automatismo que controle a la máquina embotelladora para que trabaje de la «forma adecuada». La solución pasará por implementar un esquema de control realimentado como el que se muestra en la Fig. 2 (pág. 4).

Para aclarar qué se entiende por «trabajar de la forma adecuada» se empleará la definición formal del problema de control para el caso de sistemas de eventos discretos. Empecemos con la definición de «conexión de sistemas» [Willems, 1991, 2007]. Sean los sistemas  $S' = \langle \mathbb{X}', \mathbb{U}', \mathbb{Y}', f', h', x'_0 \rangle$  y  $S'' = \langle \mathbb{X}'', \mathbb{U}'', \mathbb{Y}'', f'', h'', x''_0 \rangle$  que cumplen  $\mathbb{Y}' = \mathbb{U}''$  (la salida de  $S'$  es entrada de  $S''$ ) y cuyos comportamientos manifiestos son  $\mathbb{B}_{S'}$  y  $\mathbb{B}_{S''}$  respectivamente.<sup>1</sup> La *conexión* de  $S'$  y  $S''$  respecto a las variables que toman valor en el conjunto  $\mathbb{C} = \mathbb{Y}' = \mathbb{U}''$  es un nuevo sistema  $S = \langle \mathbb{X}, \mathbb{U}, \mathbb{Y}, f, h, x_0 \rangle$ , que también representaremos como  $S' \times_{\mathbb{C}} S''$ , definido de la forma

$$\begin{aligned} \mathbb{X} &= \mathbb{X}' \times \mathbb{X}'' \\ \mathbb{U} &= \mathbb{U}' \\ \mathbb{Y} &= \mathbb{Y}'' \\ \mathbb{B} &= \mathbb{B}_{S' \times_{\mathbb{C}} S''} \\ &= \left\{ \left( \underbrace{(x', x'')}_x, \underbrace{u'}_u, \underbrace{y''}_y \right) \in ((\mathbb{X}' \times \mathbb{X}'') \times \mathbb{U}' \times \mathbb{Y}'')^{\mathbb{N}} : \right. \\ &\quad \left. \exists (x', u', y') \in \mathbb{B}_{S'} \text{ y } (x'', u'', y'') \in \mathbb{B}_{S''} \text{ tal que } y' = u'' \right\}, \end{aligned}$$

i.e.,  $\mathbb{B}_{S' \times_{\mathbb{C}} S''}$  es un subconjunto de  $\mathbb{B}_{S'} \times \mathbb{B}_{S''}$  donde la salida de  $S'$  toma el mismo valor que la entrada de  $S''$ . En el caso de la conexión con realimentación se fuerza también  $\mathbb{U}' = \mathbb{Y}''$ .

Podemos ahora definir el «problema de control» con los siguientes términos [Tabuada, 2009]. Sean los sistemas P y S denominados **PLANTA** y **ESPECIFICACIÓN** cuyos comportamientos manifiestos son  $\mathbb{B}_P$  y  $\mathbb{B}_S$  respectivamente. *Controlar* la **PLANTA** P consiste en conectarle un sistema C llamado **CONTROLADOR** de forma que el comportamiento manifiesto  $\mathbb{B}_{C \times_{\mathbb{C}} P}$  del sistema formado por la **PLANTA** y el **CONTROLADOR** cumpla

$$\mathbb{B}_{C \times_{\mathbb{C}} P} = \mathbb{B}_S.$$

Este resultado lo expresamos con la frase:

«controlar la **PLANTA** P consiste en conectarle un **CONTROLADOR** C que restringirá su comportamiento para que se ajuste al comportamiento de la **ESPECIFICACIÓN** S».

Como caso particular, pero muy empleado en ingeniería de control, nos encontramos con la conexión con «realimentación», también llamada conexión «en lazo cerrado». En la Fig. 3(b) (pág. 4) podemos ver que no siempre se emplean todas las señales de entrada o salida para conectar el **CONTROLADOR** y la **PLANTA**. En este ejemplo, el espacio de señales de entrada del **CONTROLADOR** está formado por dos subespacios  $\mathbb{U}_C = \mathbb{U}'_C \times \mathbb{U}''_C$ , la conexión se forma con las variables que toman valor en  $\mathbb{C} = \mathbb{U}''_C \times \mathbb{Y}_C = \mathbb{Y}_P \times \mathbb{U}_P$  y la representamos con la expresión  $C \times_{\mathbb{C}} P$ .

<sup>1</sup> Recordemos que los comportamientos «completo»  $\mathbb{B}_f$  y «manifiesto»  $\mathbb{B}$  del sistema de eventos discretos  $(\mathbb{X}, \mathbb{U}, \mathbb{Y}, f, h, x_0)$  son:

$$\begin{aligned} \mathbb{B}_f &= \left\{ (x, u, y) \in (\mathbb{X} \times \mathbb{U} \times \mathbb{Y})^{\mathbb{N}} : (x^+, y) = (f(x, u), h(x, u)) \right\} \text{ y,} \\ \mathbb{B} &= \text{proj}_{(\mathbb{U} \times \mathbb{Y})^{\mathbb{N}}}(\mathbb{B}_f) \\ &= \left\{ (u, y) \in (\mathbb{U} \times \mathbb{Y})^{\mathbb{N}} : \exists x \in \mathbb{X}^{\mathbb{N}} \text{ tal que } (x, u, y) \in \mathbb{B}_f \right\}. \end{aligned}$$

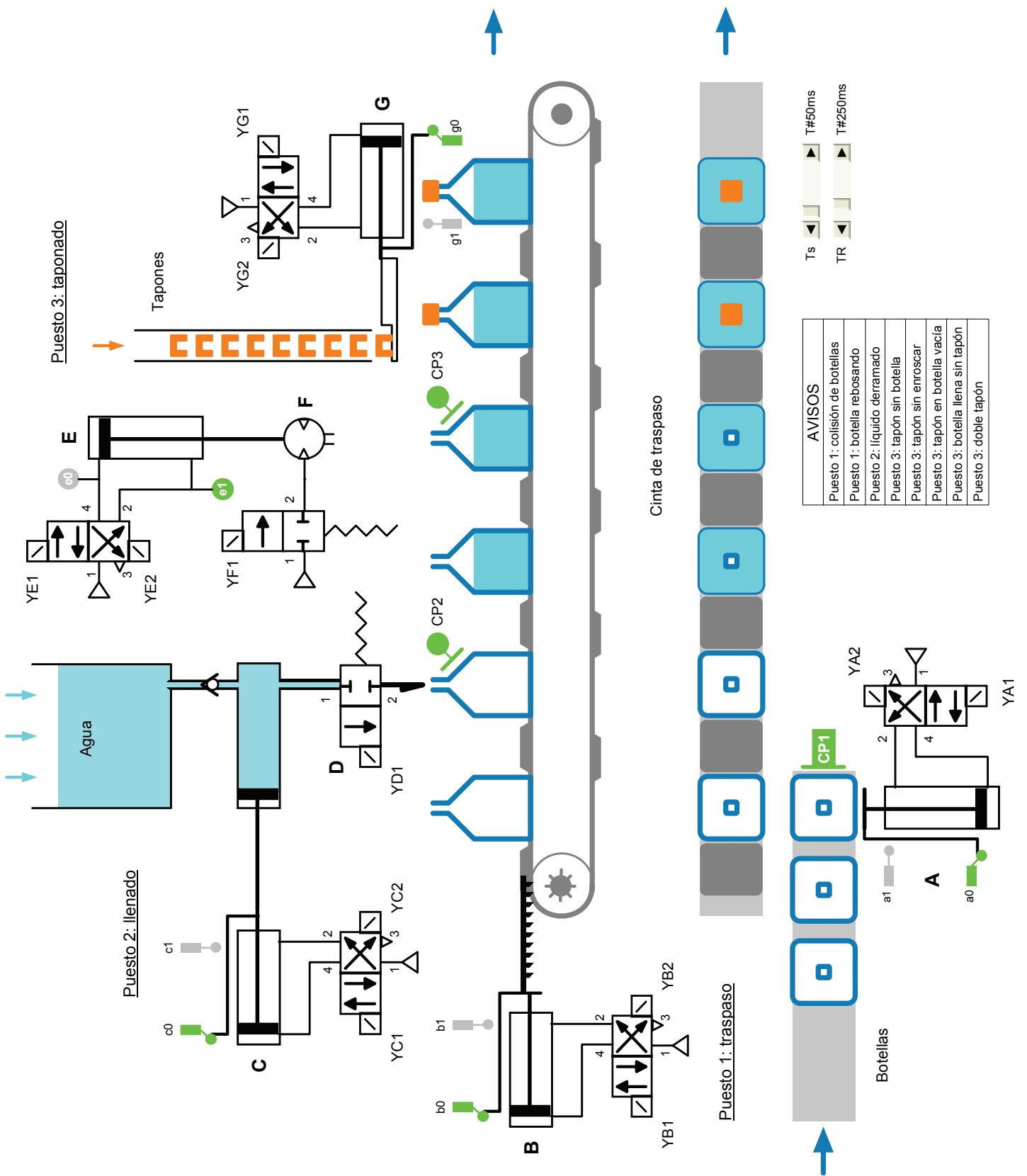


FIG. 1: Modelo estructural de la máquina embotelladora (adaptado de ADEPA [1992, pág.10]).

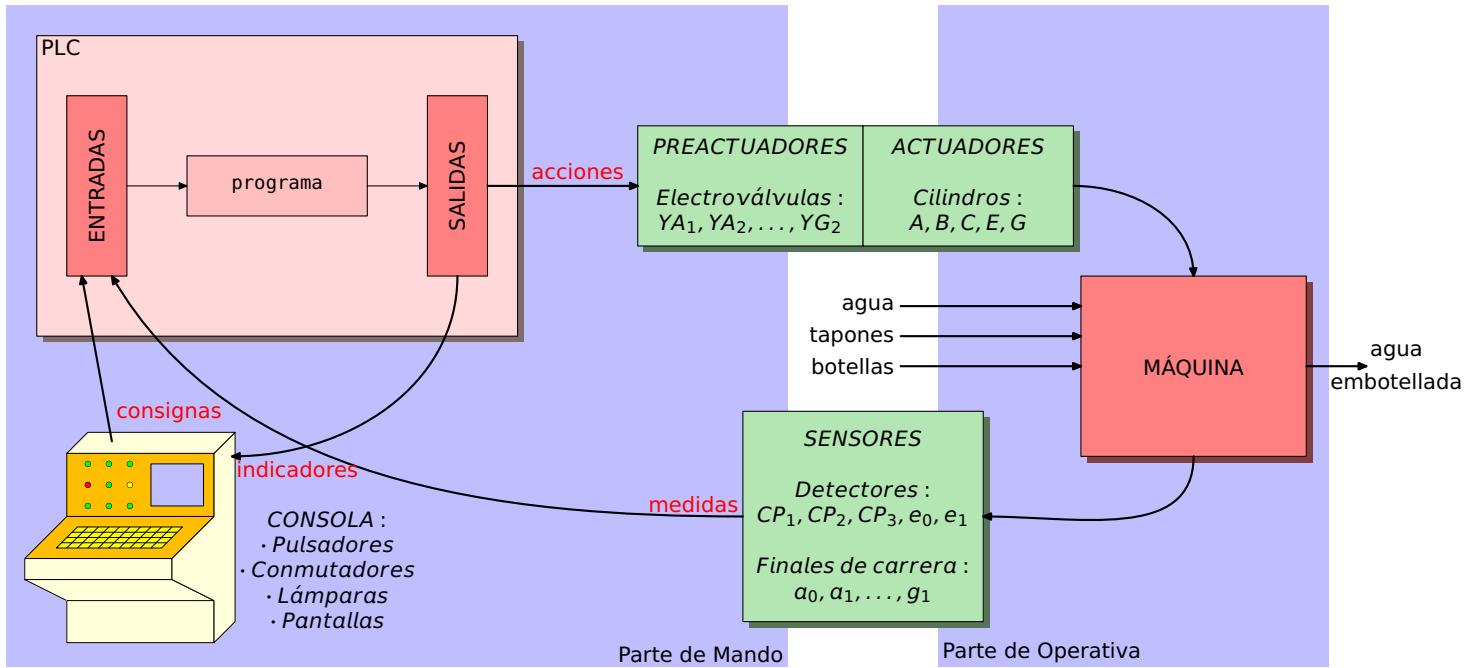


FIG. 2: Esquema de control de la máquina embotelladora.

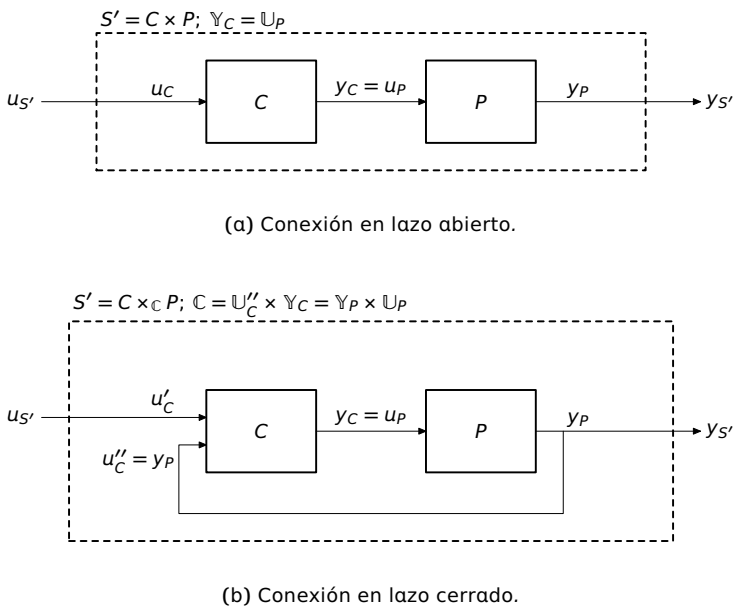


FIG. 3: Conexión de sistemas.

Las especificaciones funcionales S<sub>1</sub>–S<sub>7</sub> enumeradas en la pág. 2 son un ejemplo de descripción del comportamiento B<sub>S</sub>. Sin embargo, estas especificaciones están redactadas para que un diseñador comprenda «en líneas generales» cual es el funcionamiento deseado de la máquina, pero no son un modelo formal de la misma. Se hace así necesario confeccionar una lista exhaustiva de especificaciones que permitan modelar la máquina y obtener B<sub>S</sub>. Esta tarea será posible solo en los casos de máquinas sencillas y, por lo general, habrá que conformarse con obtener un modelo que contemple los estados o modos de funcionamiento más probables y que detecte, aunque no trate de forma individualizada, los numerosos estados poco frecuentes [Miyagi and Riascos, 2006].

Por otro lado nos enfrentamos al impedimento de no disponer

de una máquina real sobre la que probar el CONTROLADOR. No obstante, podemos suplir esta carencia con un simulador de la máquina y, aunque puede parecer alejado de la realidad, es una situación que se da con mucha frecuencia en el desarrollo de sistemas de control: «probar el CONTROLADOR sobre un simulador antes de probarlo sobre el sistema real». Este planteamiento es incluso aconsejable en algunas ocasiones porque puede ahorrar costes mediante una detección temprana de los errores de diseño, sin consumo de energía o materias primas y sin poner en peligro la máquina.

Todo ello justifica la implementación de un simulador de la máquina embotelladora lo más «realista posible».

### §3. EL SIMULADOR DE LA PLANTA

Al hablar de un simulador «lo más realista posible» hay que descartar la posibilidad de simular todos los aspectos físicos de la máquina. Para empezar, la máquina es un sistema híbrido donde hay variables continuas y discretas. Así, p. ej., la velocidad del movimiento del vástago de cada cilindro es una variable continua mientras que sus marcas de posición (x<sub>0</sub> y x<sub>1</sub>) se modelan como variables binarias.

Sin embargo, nuestro modelo de la máquina será una abstracción que tendrá en cuenta únicamente su comportamiento como sistema de eventos discretos donde todas sus variables son binarias. No obstante, se considerarán algunas variables continuas discretizadas de acuerdo con las siguientes especificaciones:

- P1. El desplazamiento del émbolo de cada cilindro se simulará con un movimiento continuo mediante un número fijo (p. ej., 6) de saltos para el avance o retroceso. Como el cilindro B mueve la cinta de traspaso, esta cinta y las botellas que hay en ella se moverán con el mismo número de pasos.
- P2. La velocidad de avance del cilindro B dependerá del número de botellas que haya en la cinta de traspaso y también de si las

botellas están llenas o vacías. Esto simulará la resistencia que ofrece la carga: a mayor resistencia, menor velocidad.

- P3.  $T_s \in [0, 500ms]$ ,  $|\Delta T_s| = 25ms$ , es el tiempo de muestreo de la simulación. Esto quiere decir que los distintos eventos de la simulación están separados por  $T_s$  segundos. Así, si el avance de un cilindro se simula con 6 saltos y  $T_s = 100ms$ , el cilindro tardará  $600ms$  en desplegarse.
- P4.  $T_R \in [0, 1s]$ ,  $|\Delta T_R| = 50ms$ , es el tiempo de respuesta de los electroimanes de mando y reposición de cada electroválvula; i.e., desde que se activa la orden de mando o de reposición de una electroválvula hasta que esta conmuta transcurrirá el tiempo  $T_R$ .
- P5. Los tiempos  $T_s$  y  $T_R$  podrán cambiarse durante la ejecución del simulador con incrementos/decrementos  $\Delta T_s$  y  $\Delta T_R$  respectivamente.

El resto de especificaciones del simulador serán inferidas por el alumno a partir de su nivel de comprensión del funcionamiento de la máquina después de examinar la Fig. 1 (pág. 3).

Será también obligatorio desarrollar una pantalla HMI<sup>2</sup> parecida a la presentada en esa figura y que permita interactuar con el simulador antes de conectarle el CONTROLADOR. Algunas de sus funciones son:

- P6. Activar/reponer el estado de las válvulas de distribución presionando sobre sus mandos YA1, YA2,..., YG1.
- P7. Activar/desactivar los sensores CP1, CP2 y CP3.
- P8. Activar/desactivar la presencia de botellas en cada uno de los 6 puestos de la cinta de traspaso. Esto simulará la acción realizada por un operador de poner o quitar manualmente la botella de la posición correspondiente.
- P9. Avisar de comportamientos no deseados (cf. comentario sobre estados poco frecuentes en la pág. 4) con indicadores como el que se muestra en la Tabla 1.

AVISOS
Puesto 1: colisión de botellas
Puesto 2: botella rebosando
Puesto 2: líquido derramado
Puesto 3: tapón sin botella
Puesto 3: tapón sin enroscar
Puesto 3: tapón en botella vacía
Puesto 3: botella llena sin tapón
Puesto 3: doble tapón

TABLA 1: Avisos del simulador de la máquina.

#### §4. EL MODELO DE LA ESPECIFICACIÓN

Ya hemos visto que la PLANTA P y la ESPECIFICACIÓN S son «datos» del «problema de control» y que este consiste en encontrar un CONTROLADOR C y una conexión  $\times_C$  tales que el comportamiento del sistema  $C \times_C P$  cumpla  $B_{C \times_C P} = B_S$ .

La PLANTA (nuestra máquina embotelladora) tiene un comportamiento  $B_P$  muy amplio y «puede hacer muchas cosas»,

<sup>2</sup>Human Machine Interface.

algunas de ellas «no deseadas»: p.ej., puede tomar una botella vacía y taponarla sin haberla llenado previamente de agua. La misión del CONTROLADOR es restringir el comportamiento de la PLANTA para que haga exactamente aquello para lo que ha sido concebida.

Teniendo esto presente, el siguiente paso en la resolución de un problema de automatización es determinar el comportamiento que se espera de la PLANTA, i.e., determinar  $B_S$ . Las especificaciones S1–S7 (pág. 2) describen parte de  $B_S$  para nuestro problema concreto. Sin embargo, son claramente insuficientes ya que se limitan al «funcionamiento normal» de la máquina; en concreto, al funcionamiento a «pleno rendimiento» donde se produce una botella por ciclo (por cada botella que entra, sale una botella llena y taponada).

Podemos interrogarnos sobre otros modos de funcionamiento, p.ej.:

1. ¿Cómo se detiene la producción:
  - (a) de forma abrupta parando todos los puestos de trabajo?,
  - (b) dando salida a las botellas que hay en la cinta de traspaso sin llenarlas ni taponarlas? o,
  - (c) dando salida a las botellas llenándolas y taponándolas?
2. Qué hacer si faltan tapones o se avería el puesto de taponado:
  - (a) detener la producción o,
  - (b) continuar la producción y taponar las botellas manualmente.
3. Cómo actuar en caso de avería o emergencia: p.ej., se rompe alguna botella e impide el avance de la cinta de traspaso.
4. Después de una reparación o para detectar una avería interesa accionar cada puesto de trabajo individualmente para comprobar su correcto funcionamiento.

Para ayudarnos a determinar todos los posibles «modos de funcionamiento» de la ESPECIFICACIÓN (i.e., la máquina automatizada) será necesario emplear la guía GEMMA [ADEPA, 1992] y seleccionar los modos de «marcha y parada». El resultado se reflejará sobre la plantilla de la guía (cf. ANEXO A, pág. 11) y en una lista numerada de especificaciones que complete la lista S1–S7 (pág. 2).

#### §5. EL CONTROLADOR DE LA MÁQUINA

Conociendo los comportamientos  $B_P$  y  $B_S$  podemos determinar la conexión  $\times_C$  y el comportamiento  $B_C$ . De hecho,  $\times_C$  se obtiene de forma inmediata por inspección de la Fig. 2 (pág. 4) ya que  $\mathbb{Y}_C = \mathbb{U}_P$  se construye con las señales de accionamiento/reposición de las electroválvulas (YA1, YA2,...,YG2) y  $\mathbb{U}'_C = \mathbb{Y}_P$  (cf. Fig. 3(b), pág. 4) se construye con las marcas de los cilindros (a0, a1,...,g1) y los sensores (CP1, CP2 y CP3).

Para completar el esquema de la Fig. 2 (pág. 4) es necesario introducir un nuevo «sistema», en este caso humano: el OPERADOR H. El par (H, C) se puede tratar formalmente como el par (C, P) y considerar que el OPERADOR actúa como controlador del



automatismo. Así, podemos hablar de la conexión  $H \times_{\mathbb{H}} C$  donde, para determinar qué variables forman el conjunto  $\mathbb{H}$  tenemos que acudir nuevamente al comportamiento de la **ESPECIFICACIÓN**.

La pantalla HMI de la Fig. 4 muestra una posible interfaz de comunicación entre el **OPERADOR** y el **CONTROLADOR**. Los indicadores luminosos que acompañan a cada pulsador o conmutador pueden ayudarnos a entender los modos de funcionamiento de la máquina automatizada y completar la lista de modos de funcionamiento que se solicita en el apartado anterior así como las variables que forman parte de la conexión  $\times_{\mathbb{H}}$ .

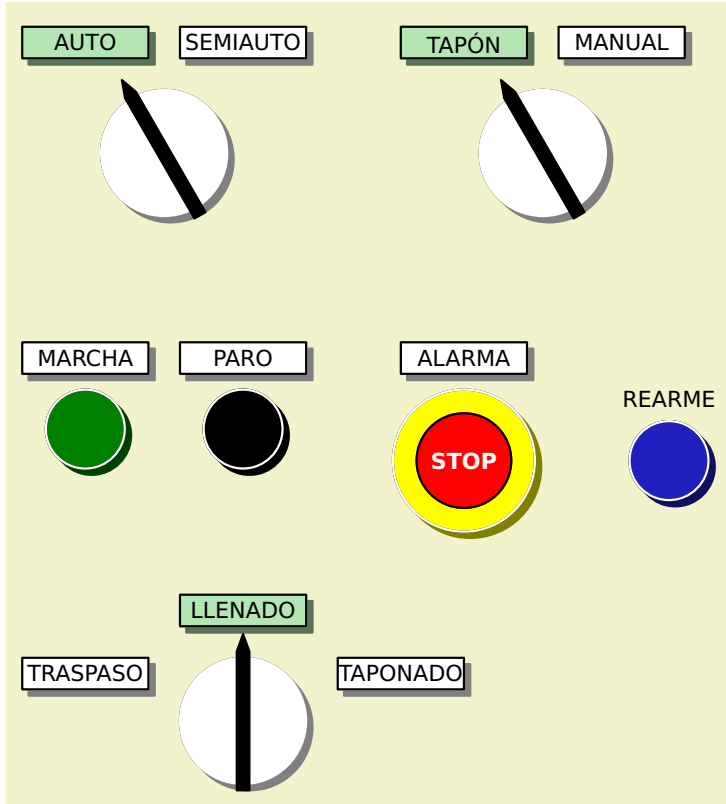


FIG. 4: Consola de comunicación OPERADOR-CONTROLADOR.

Volviendo al problema de obtener el **CONTROLADOR**, podemos argumentar formalmente como sigue: supongamos definida la «clase de los sistemas» a la que llamaremos  $\mathcal{S}$  y que la conexión de sistemas es una operación binaria  $\mathcal{S} \times \mathcal{S} \rightarrow \mathcal{S}$ <sup>3</sup> que cumple las siguientes propiedades:

1. **ASOCIATIVA.**  $S \times (S' \times S'') = (S \times S') \times S''$  para todo  $S, S', S'' \in \mathcal{S}$  y, por lo tanto, la expresión  $S \times S' \times S''$  (llamada «conexión en cascada») no es ambigua.
2. **ELEMENTO IDENTIDAD.** En  $\mathcal{S}$  existe un sistema de nombre 1 que actúa como elemento identidad, de forma que para todo sistema  $S \in \mathcal{S}$  se cumple

$$S \times 1 = 1 \times S = S.$$

Para definir 1 basta con imponer  $\mathbb{X}_1 = \emptyset$ ,  $\mathbb{Y}_1 = \mathbb{U}_1$  e  $y_1 = u_1$ , i.e., el sistema carece de espacio de estados y copia la entrada en la salida.

<sup>3</sup>El conjunto  $\mathbb{C}$  donde toman valor las variables de la conexión se considera implícitamente definido en cada caso y no se detalla aquí.

3. **EXISTENCIA DE INVERSO.**<sup>4</sup> Para cada sistema  $S \in \mathcal{S}$  existe un inverso que representaremos con  $S^{-1}$  y que cumple

$$S \times S^{-1} = S^{-1} \times S = 1.$$

Entonces, toda ecuación con la forma  $X \times A = B$  tiene solución en  $\mathcal{S}$ . En efecto,

$$\begin{aligned} X \times A &= B \\ (X \times A) \times A^{-1} &= B \times A^{-1} \text{ (propiedad 3)} \\ X \times (A \times A^{-1}) &= B \times A^{-1} \text{ (propiedad 1)} \\ X \times 1 &= B \times A^{-1} \text{ (propiedad 3)} \\ X &= B \times A^{-1} \text{ (propiedad 2)}. \end{aligned}$$

Esta es la forma «clásica» de resolver problemas de control para sistemas continuos LTI.<sup>5</sup> Recordemos que estos sistemas quedan perfectamente caracterizados por su «función de transferencia» y que para trabajar con ellos es suficiente la teoría de funciones analíticas [Ogata, 1998]. Así, p. ej., para el esquema clásico de control con realimentación negativa que se muestra en la Fig. 5 se cumple que

$$\begin{aligned} Y(s) &= U(s)P(s) \\ &= E(s)C(s)P(s) \\ &= [R(s) - Y(s)]C(s)P(s) \Rightarrow \frac{Y(s)}{R(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)}. \end{aligned}$$

Así, conocida la función de transferencia de la **ESPECIFICACIÓN**  $S(s) = Y(s)/R(s)$ , podemos calcular la función de transferencia del **CONTROLADOR**

$$C(s) = \frac{S(s)}{[1 - S(s)]P(s)} = S(s)[1 - S(s)]^{-1}P^{-1}(s).$$

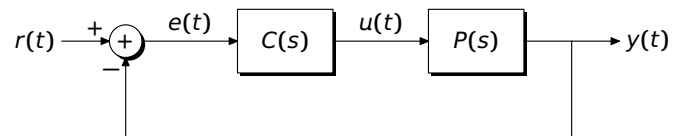


FIG. 5: Sistema de control LTI con realimentación negativa.

Este planteamiento formal nos lleva a la siguiente solución del problema de control para sistemas de eventos discretos

$$\begin{aligned} H \times C \times P &= H \times S \\ (H^{-1} \times H) \times C \times (P \times P^{-1}) &= (H^{-1} \times H) \times S \times P^{-1} \\ C &= S \times_{\mathbb{P}} P^{-1}. \end{aligned}$$

Ahora bien, la propiedad número 3 del operador  $\times$  no se ha definido de forma constructiva, i.e., no se conoce un algoritmo general para calcular el inverso de un sistema de eventos discretos. A modo de ejemplo sobre la dificultad del problema

<sup>4</sup>Podríamos pensar que con estas propiedades el par  $(\mathcal{S}, \times)$  tiene estructura algebraica de grupo, sin embargo  $\mathcal{S}$  no es un conjunto sino una clase. Además, para completar la definición del operador  $\times$  es necesario especificar en cada caso el conjunto  $\mathbb{C}$  donde toman valor las variables de la conexión.

<sup>5</sup>Linear Time-Invariant.

y algunas de las soluciones parciales propuestas se puede consultar la monografía [Saeztzu et al. \[2013\]](#). Estamos pues en el punto de partida y tenemos que ensayar una solución de tipo heurístico<sup>6</sup> que se obtendrá con la metodología habitual para programar sistemas de eventos discretos:

1. Modelar el comportamiento del sistema con alguna herramienta formal: máquina de estados o red de Petri interpretada (RdPI). Si optamos por una RdPI se aconseja dibujar una red que modele los cambios entre los distintos modos de funcionamiento seleccionados en la guía GEMMA y una red por cada uno de estos modos (A1, A2,..., F6).
2. Implementar la red que modela los «cambios de modo» como un programa IEC 61131-3 y cada uno de los modos como un bloque de función que es invocado desde el programa anterior.

No hay que olvidar que en todos estos modelos las señales de salida del controlador se corresponden con señales de entrada del simulador de la máquina y viceversa, por lo que se aconseja declararlas en una sección de variables globales:

#### VAR\_GLOBAL

```
(* CONTROLADOR → PLANTA.*)
```

```
CP1 : BOOL;
```

```
CP2 : BOOL;
```

```
CP3 : BOOL;
```

```
YA1 : BOOL;
```

```
:
```

```
YG2 : BOOL;
```

```
(* PLANTA → CONTROLADOR.*)
```

```
ao : BOOL;
```

```
:
```

```
g1 : BOOL;
```

#### END\_VAR

También, teniendo en cuenta la información que se intercambia entre el OPERADOR y el CONTROLADOR (cf. [Fig. 4](#)), se puede declarar otra sección de variables globales que contenga estas señales:

#### VAR\_GLOBAL

```
MARCHA : BOOL;
```

```
PARO : BOOL;
```

```
AUTO : BOOL;
```

```
SEMIAUTO : BOOL;
```

```
:
```

```
STOP : BOOL;
```

```
REARME : BOOL;
```

#### END\_VAR

## §6. REALIZACIÓN DISTRIBUIDA

Por «realización distribuida» entendemos la implementación del sistema con dos PLCs conectados por una red de comunicaciones. En uno de los PLCs se ejecutará el simulador de

la PLANTA y en el otro el CONTROLADOR. La red será el canal de comunicación entre ellos y se utilizará para el intercambio de las variables de entrada/salida.

Para implementar esta solución con el entorno de desarrollo CODESYS se empleará el PLC WINNT V2.4. Este es un simulador de PLC con la funcionalidad de acceso a Ethernet y servidor HTML. En la [Fig. 6](#) se muestra la ventana de control de un PLC WINNT V2.4 ejecutándose en WINDOWS-XP y donde se ha cargado el proyecto `Simulador.pro`. La comunicación con el PLC se establece con el protocolo TCP/IP en la dirección IP 192.168.64.22 y el puerto 1200. Estos parámetros son locales a cada instalación y tienen que ser consultados con el administrador de la red.

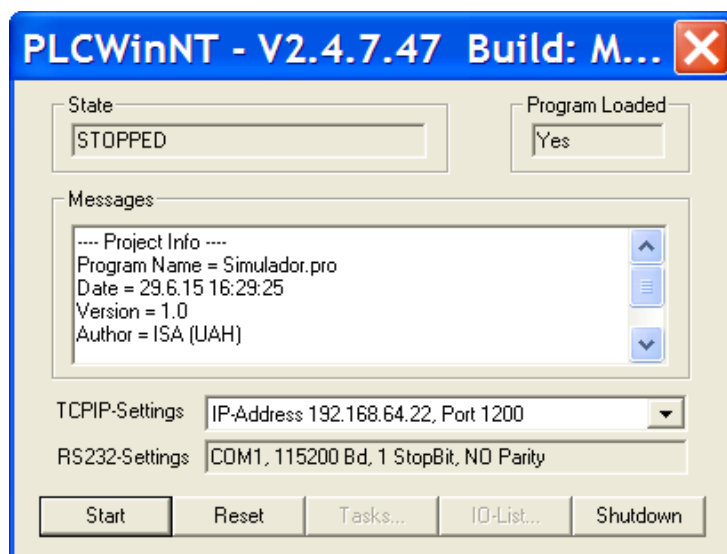


FIG. 6: PLC WINNT V2.4 donde se ejecutará el simulador de la PLANTA.

En la [Fig. 7](#) se muestra la ventana de control de otro PLC WINNT V2.4 ejecutándose en WINDOWS-7 y donde se ha cargado el proyecto `controlador.pro`. Este PLC está configurado con la dirección IP 192.168.64.15 y el mismo puerto 1200.

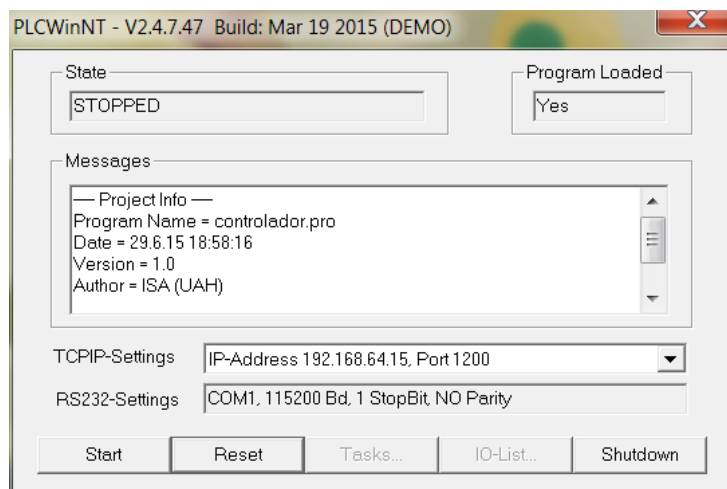


FIG. 7: PLC WINNT V2.4 donde se ejecutará el CONTROLADOR.

Para comunicar el CONTROLADOR con la PLANTA se emplearán las «variables de red». Este es un servicio de comunicaciones implementado en los simuladores PLC WINNT V2.4 (cf. [Fig. 8](#)),

<sup>6</sup> **heurística.** (Del gr. εὐρίσκειν *heurískein* ‘hallar’, ‘inventar’, y *-tiko*) • f. [...] ||4. En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc. [DRAE].

pertenece a la capa de aplicación de la jerarquía OSI y permite que varios PLCs conectados a una misma red local compartan variables. Este servicio se construye sobre el protocolo UDP de las redes IP. En el ANEXO C (pág. 15) se muestra el apartado 6.2 del manual de CODESYS [CODESYS, 2010] donde se explica la forma de declarar bloques con «variables de red». Como parte del entrenamiento de la competencia CG3, el alumno tendrá que trabajar con esta fuente de información para ver la forma de incorporar este tipo de variables al programa y realizar una implementación distribuida.

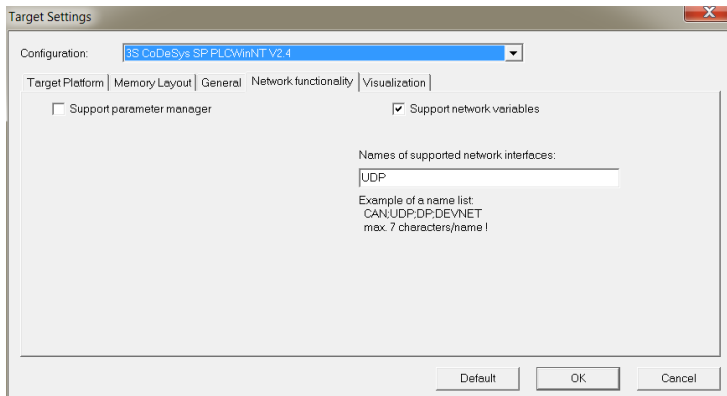


FIG. 8: Para utilizar el servicio de «variables de red» hay que habilitarlo en el simulador PLC WINNT V2.4.

Hay que reseñar que los bloques de variables globales mencionados en el apartado anterior tienen que ser redefinidos y declarados como «variables de red». A modo de ejemplo, se muestran a continuación dos posibles bloques declarados en el CONTROLADOR junto con los parámetros de cada uno de ellos (Fig. 9 y Fig. 10)

```
(* Bloque: Controlador_Planta *)
VAR_GLOBAL
(* SALIDAS del CONTROLADOR →
  ENTRADAS a la PLANTA.*)
(* Actuadores.*)
YA1, YA2,
YB1, YB2,
YC1, YC2,
YD1,
YE1, YE2,
YF1,
YG1, YG2 : BOOL;
END_VAR
```

```
(* Bloque: Planta_Controlador *)
VAR_GLOBAL
(* SALIDAS de la PLANTA →
  ENTRADAS al CONTROLADOR.*)
(* Actuadores.*)
ao, a1,
bo, b1,
co, c1,
eo, e1,
go, g1 : BOOL;
(* Presencia de botellas.*)
CP1, CP2, CP3 : BOOL;
END_VAR
```

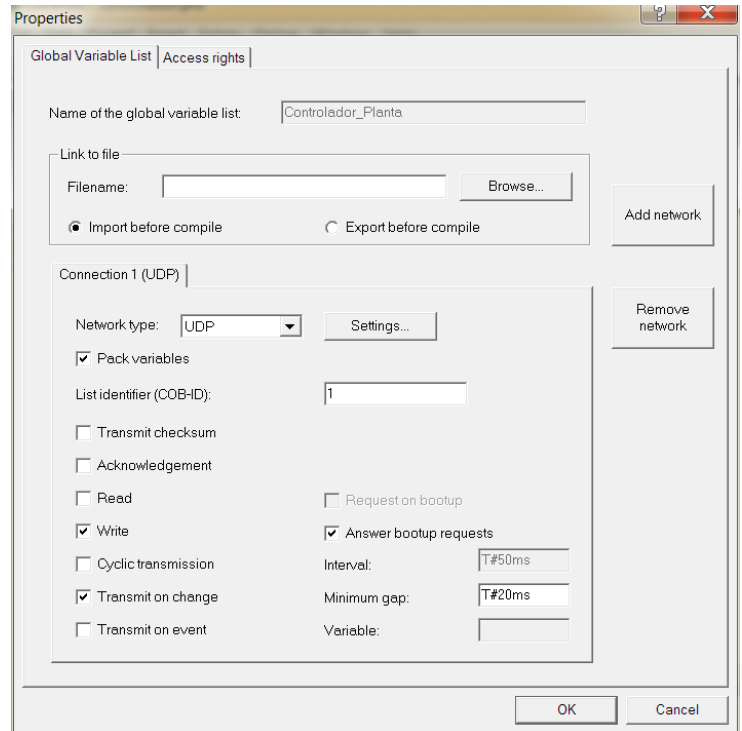


FIG. 9: Propiedades como «variables de red» del bloque de variables globales Controlador\_Planta definidas en el CONTROLADOR.

En la PLANTA habrá que hacer declaraciones parecidas *mutatis mutandis*.

## §7. DOCUMENTACIÓN

Al finalizar el trabajo se entregarán los archivos generados con la herramienta CODESYS que contienen la solución<sup>7</sup> al problema de automatización planteado. La solución constará del archivo *embotelladora#.pro*<sup>8</sup> en el caso de implementar una solución centralizada; y de los archivos *Simulador#.pro* con el simulador (sustituye a la PLANTA) de la máquina embotelladora y *controlador#.pro* con el CONTROLADOR, en el caso de implementar la solución distribuida. Cada una de las «unidades organizativas de programa» que aparezcan en los archivos anteriores tendrá una comentario de cabecera donde figure la información siguiente:

**Nombre del POU.**

**Descripción:** breve descripción de su funcionalidad.

**Programado por:** nombre del programador/es y fecha de finalización.

**Revisado por:** nombre del revisor/es y fecha de la revisión (si la hubiere).<sup>9</sup>

Se entregará además un *dossier* o carpeta con la documentación generada<sup>10</sup> durante el proceso de elaboración del trabajo y que incluirá, al menos, la siguiente documentación:

1. Modelos de la PLANTA, de la ESPECIFICACIÓN y del CONTROLADOR:<sup>11</sup>

<sup>7</sup>Entrena la parte de síntesis de la competencia CG1.

<sup>8</sup>El símbolo «#» se sustituirá por el nro. asignado al grupo de trabajo.

<sup>9</sup>El objetivo de las revisiones es entrenar la competencia CT5.

<sup>10</sup>Competencia CB9.

<sup>11</sup>Parte de análisis de la competencia CG1.



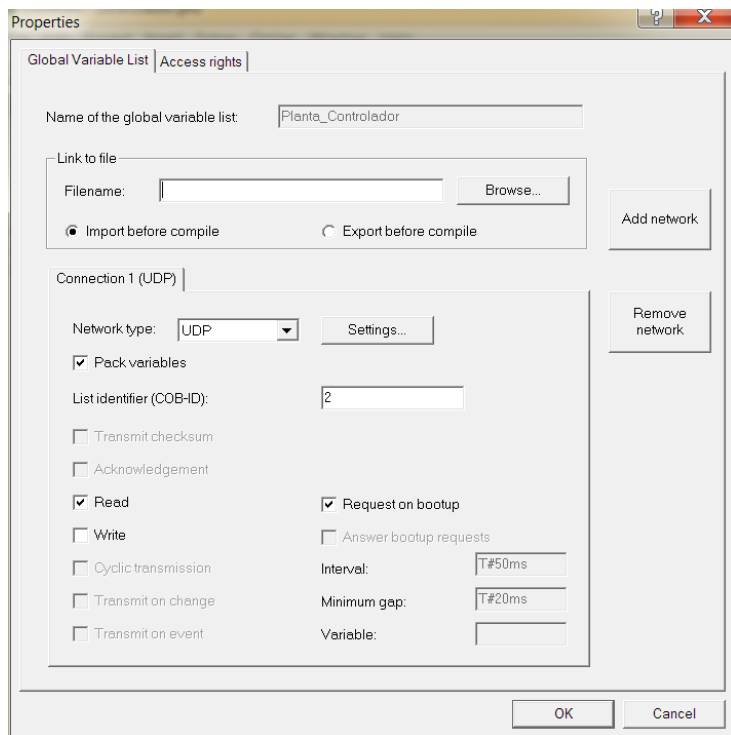


FIG. 10: Propiedades como «variables de red» del bloque de variables globales `Planta_Controlador` definidas en el `CONTROLADOR`.

- (a) Modelo estructural: está formado por los planos que describen la `PLANTA` y el `CONTROLADOR`. Como estas máquinas no se construyen físicamente, los planos de las mismas serán sustituidos por las pantallas HMI generadas, que servirán para conocer los elementos que las componen y la relación que hay entre ellos. Las figuras 1 (pág. 3), 2 (pág. 4) y 4 (pág. 6) pueden servir de ejemplo para ilustrar el tipo de modelo al que hace referencia este apartado.
  - (b) Modelo funcional: lista numerada de especificaciones que describen el comportamiento manifiesto de la `PLANTA` ( $B_P$ ) y de la `ESPECIFICACIÓN` ( $B_S = B_C \times B_P$ ).
  - (c) Modelo procesal: diagramas (grafos, redes de Petri, guía GEMMA,...) que describen el comportamiento dinámico de la `PLANTA` y del `CONTROLADOR` y que han ayudado para implementar los programas de la solución.
2. Manual de instalación. Lista numerada con los equipos necesarios y los pasos a seguir para instalar la solución generada.
  3. Plan de pruebas. Lista numerada con los pasos a seguir para probar las especificaciones funcionales. Esta lista se redactará a partir del modelo funcional y permitirá que un evaluador externo valide la solución entregada, i.e., compruebe que la solución cumple con las especificaciones.
  4. Actas de las reuniones. Se celebrarán al menos tres reuniones:
    - (a) Inicial: servirá para constituir el grupo y también se puede aprovechar para dividir el trabajo en tareas, elaborar un cronograma o diagrama de Gantt con

las precedencias entre ellas y repartirlas entre los miembros.<sup>12</sup>

- (b) Intermedia: servirá para analizar las incidencias y problemas surgidos, controlar el estado de las tareas, controlar los riesgos (p.ej.: ¿cómo gestionar los retrasos?, causas y responsables de los mismos), introducir cambios en la planificación y en la distribución del trabajo debidos a las problemas surgidos y adoptar acuerdos.<sup>13</sup>
- (c) Final: análisis del trabajo realizado y cierre del proyecto. Contendrá la lista de especificaciones que no se han podido implementar y servirá para delimitar responsabilidades (quién no ha cumplido con su parte del trabajo y por qué).<sup>14</sup>

Cada reunión será presidida por el miembro de mayor edad y actuará de secretario el miembro de menor edad. El secretario será el encargado de redactar y custodiar las actas.<sup>15</sup> En el acta de cada reunión figurará:

- (a) Lugar, fecha y hora de inicio.
- (b) Nombre de los asistentes y ausentes, con la justificación de las ausencias.
- (c) Puntos tratados, intervenciones y acuerdos tomados en cada punto.
- (d) Ruegos y preguntas.
- (e) Hora de fin de la reunión
- (f) En cada página del acta también figurará el número de la página y el total de páginas.

El *dossier* puede incluir opcionalmente los siguientes puntos que no son evaluables y que sirven como realimentación para la mejora de esta parte de la asignatura.

#### 1. Conclusiones de tipo didáctico:

- (a) Dificultades encontradas: ¿ha encontrado alguna dificultad para entender o implementar alguna parte del enunciado de la práctica?, ¿cuál?, ¿cómo ha afrontado/superado esa dificultad?
- (b) Resultados del aprendizaje: ¿ha encontrado esta práctica útil o no para adquirir/reforzar alguna de las competencias planteadas en la asignatura? ¿cuáles?
- (c) Recomendaciones para la mejora de esta parte de la asignatura (p.ej.: simplificar la práctica, suprimirla, cambiar el peso en la calificación final, cambiar la forma de evaluarla,...).
- (d) Impresión global.

#### 2. Futuros trabajos. Ampliación de esta práctica u otros temas sobre los que podría versar.

<sup>12</sup>Competencias CG2 y CT3.

<sup>13</sup>CG4.

<sup>14</sup>CT2 y CT4.

<sup>15</sup>La redacción de estas actas también ayuda a entrenar la competencia CT5 ya que se pueden emplear en la mejora de la calidad del proceso.

## §8. EVALUACIÓN

La evaluación de la práctica se basará en las pruebas realizadas sobre la PLANTA y la ESPECIFICACIÓN=CONTROLADOR×PLANTA, el tipo de implementación (centralizada/distribuida) y la documentación entregada; de acuerdo con el siguiente baremo:

1. Funcionamiento de la PLANTA (hasta 0,8p):
  - (a) Activación individual de válvulas (algunas 0,03p, casi todas 0,08p)
  - (b) Movimiento de válvulas distribuidoras (algunas 0,03p, casi todas 0,08p), cilindros y finales de carrera (algunos 0,03p, casi todos 0,08p) y cinta transportadora junto con las botellas que hay sobre ella (0,08p).
  - (c) Activar/desactivar botellas en la cinta: algunas (0,03p), casi todas (0,08p).
  - (d) Llenado de botellas (0,08p).
  - (e) Colocación de tapones (0,08p).
  - (f) Ajuste dinámico de los parámetros  $T_s$  (0,04p) y  $T_R$  (0,04p).
  - (g) Ajuste de la velocidad de avance de la cinta en función del número de botellas y su carga (0,08p).
  - (h) Detección de fallos y avisos (algunos 0,03p, casi todos 0,08p).
  - (i) Otros (0,08p).
2. Funcionamiento de la ESPECIFICACIÓN=CONTROLADOR×PLANTA (hasta 0,7p):
  - (a) Prueba de cada puesto individual: traspaso (0,07p), llenado (0,07p), taponado (0,14p).
  - (b) Producción normal (0,14p).
  - (c) Parada controlada con vaciado de la cinta (0,14p).
  - (d) Producción a pesar de los defectos: falta de tapones y taponado manual posterior (0,07p).
  - (e) Alarma/rearme (0,07p).
  - (f) Otros (0,07p).
3. Implementación distribuida (0,25p).
4. Documentación (hasta 0,25p):
  - (a) Programas claros y comentados: algunos (0p), bastantes (0,3p), casi todos (0,6p).
  - (b) Modelos: estructural (0,02p), funcional (0,03p) y procesal (0,04p).
  - (c) Manual de instalación: 0,02p.
  - (d) Plan de pruebas: deficiente (0p), incompleto (0,0p), completo o casi exhaustivo (0,06p).
  - (e) Actas: solo la inicial y la intermedia (0,03p), al menos las tres obligatorias (0,06p), las tres obligatorias y otras adicionales (0,09p).

La evaluación del apartado 1 la realizará el profesor en presencia del equipo que ha desarrollado la práctica, al cual se le podrán pedir aclaraciones sobre el trabajo realizado así como que opere sobre el sistema para comprobar si cumple con

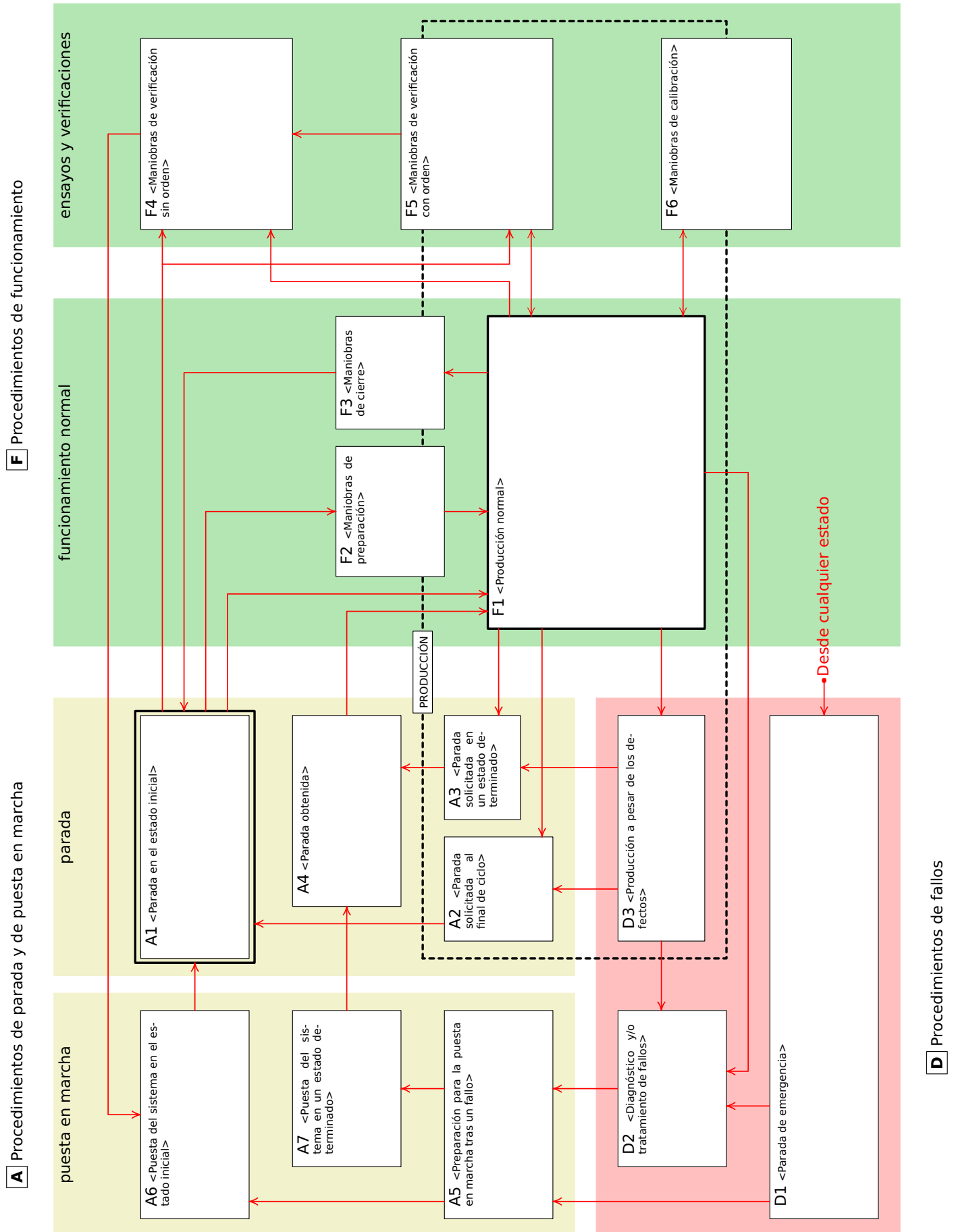
las especificaciones. La evaluación de los apartados 2 y 3 la realizará un equipo formado por dos o tres alumnos (coevaluadores) elegidos al azar entre los restantes grupos, con el asesoramiento del profesor y en presencia del equipo que ha desarrollado la práctica, al cual se le podrán pedir aclaraciones sobre el trabajo realizado así como que opere sobre el sistema para comprobar si cumple con las especificaciones. Este procedimiento de evaluación ayudará a entrenar la competencia CB9 así como comprobar que todos los miembros del grupo han colaborado en la elaboración del trabajo (competencia CT3). La evaluación del apartado 4 la realizará el profesor examinado la documentación entregada.

Los datos del baremo anterior se introducirán en las hojas de cálculo que se muestran en el ANEXO B (pág. 12). Las celdas sombreadas en verde se rellenarán con los datos del equipo que presenta el trabajo y de los coevaluadores. Las celdas sombreadas en naranja se marcarán con **X** cuando la práctica cumpla la especificación correspondiente. Si se marcan con **X** las celdas de los apartados 1.(i) o 2.(f) será necesario introducir, en las celdas sombreadas en azul, una pequeña descripción de la especificación verificada.

## REFERENCIAS

- ADEPA (1992). *GEMMA—Guide d'Étude des Modes de Marches et d'Arrêts*. Agence Nationale pour le Développement de la Production Automatisée, France.
- CODESYS (2010). *User Manual for PLC Programming with Co-DeSys 2.3*. 3S-Smart Software Solutions GmbH.
- Miyagi, P. E. and Riascos, L. A. M. (2006). Modeling and analysis of fault-tolerant systems for machining operations based on Petri nets. *Control Engineering Practice*, 14(4):397–408. <http://dx.doi.org/10.1016/j.conengprac.2005.02.002>.
- Ogata, K. (1998). *Ingeniería de control moderna*. Prentice Hall, 3<sup>a</sup> edition.
- Saeztzu, C., Silva, M., and van Schuppen, J. H. (2013). *Control of Discrete-Event Systems. Automata and Petri Net Perspectives*. Springer. <http://dx.doi.org/10.1007/978-1-4471-4276-8>.
- Tabuada, P. (2009). *Verification and Control of Hybrid Systems*. Springer. <http://dx.doi.org/10.1007/978-1-4419-0224-5>.
- Willems, J. C. (1991). Paradigms and Puzzles in the Theory of Dynamical Systems. *IEEE Transactions on Automatic Control*, 36(3):259–294. <http://dx.doi.org/10.1109/9.73561>.
- Willems, J. C. (2007). The Behavioral Approach to Open and Interconnected Systems. *IEEE Control Systems Magazine*, 27(6):46–99. <http://dx.doi.org/10.1109/MCS.2007.906923>.

§A. PLANTILLA DE LA GUÍA GEMMA [ADEPA, 1992]





**Técnicas de la Automatización (cód. 201987)**  
**Evaluación del Trabajo de Laboratorio (hoja 2/3)**

**2. Especificación =** Máximo: 0,70 Total:   
**Controlador X Planta**

2.(a)	Puestos individuales		Traspaso	Llenado	Taponado	
2.(b)	Producción normal					
2.(c)	Parada controlada con vaciado de cinta					
2.(d)	Producción a pesar de los defectos (falta de tapones y taponado manual posterior)					
2.(e)	Alarma/rearme					
Breve descripción de la especificación verificada.						
2.(f)	Otros (descripción)					

**3. Implementación distribuida** Máximo: 0,25 Total:



**Técnicas de la Automatización (cód. 201987)**  
**Evaluación del Trabajo de Laboratorio (hoja 3/3)**

**4. Documentación**

Máximo: 0,25

Total

--

Marque con x una sola de las casillas siguientes.

4.(a)	Programas claros y comentados		Algunos	Bastantes	Casi todos	

Marque con x las casillas de color naranja que proceda.

4.(b)	Modelos		Estructural	Funcional	Procesal	

4.(c)	Manual de Instalación					

Marque con x una sola de las casillas siguientes.

4.(d)	Plan de pruebas		Deficiente	Incompleto	Casi comp.	

Marque con x las casillas de color naranja que proceda.

4.(e)	Actas		Inicial e interm.	+ final	+ adicionales	

## §C. EXTRACTO DEL MANUAL DE CODESYS 2.3 [CODESYS, 2010]

Global Variables, Variable Configuration, Document Frame

### 6.2 Global Variables, Variable Configuration, Document Frame

#### Objects in 'Global Variables'

In the Object Organizer, you will find three objects in the **Resources** register card in the **Global Variables** folder (default names of the objects in parentheses).

Global Variables List (Global Variables)

Variables Configuration (Variable Configuration)

All variables defined in these objects are recognized throughout the project.



If the global variables folder is not opened (plus sign in front of the folder), you can open it with a double-click<Enter> in the line.

Select the corresponding object. The **'Object Open'** command opens a window with the previously defined global variables. The editor for this works the same way as the declaration editor.

#### Several Variables Lists

Global variables, global network variables (**VAR\_GLOBAL**), global network variables (**VAR\_GLOBAL**, target specific) and variable configurations (**VAR\_CONFIG**) must be defined in separate objects.

If you have declared a large number of global variables, and you would like to structure your global variables list better, then you can create further variables lists.

In the Object Organizer, select the **Global Variables** folder or one of the existing   objects with global variables. Then execute the **'Project' 'Object Add'** command. Give the object that appears in the dialog box a corresponding name. With this name an additional object will be created with the key word **VAR\_GLOBAL**. If you prefer an object a variable configuration, change the corresponding key word to **VAR\_CONFIG**.

#### 6.2.1 Global Variables

##### What are Global Variables

„Normal“ variables, constants or remanent variables that are known throughout the project can be declared as global variables, but also network variables that are also used for data exchange with other network subscribers.

**Please regard:** In a project you can define a local variable which has the same name like a global variable. In this case within a POU the locally defined variable will be used.

It is not allowed to name two global variables identically. For example you will get a compiler error, if you have defined a variable "var1" in the PLC Configuration as well as in a global variables list.

##### Network variables

**Note:** The use of network variables must be supported by the target system and must be activated in the target settings (category Network functionality).

By an automatic data exchange (compare this to the non-automatic data exchange via the Parameter Manager) it is possible to update the value of a network variable on several controller systems within a **CoDeSys** compatible controller network. This requires no controller-specific functions but the network subscribers must use identical declaration lists and matching transfer configurations for network variables in their projects. In order to make this possible it is recommended that the declaration not be entered manually in each controller application, but loaded from a separate file when creating the list. (see 'Create a global variables list').

## 6 - The Resources

**Create a Global Variable List**

To create a Global Variable List, open the register 'Resources' in the Object Organizer and select the entry 'Global Variables' or select an already existing list. Then choose the command 'Project' 'Object' 'Add' to open the dialog Global variable list.

This dialog can also be opened by the command 'Project' 'Object' 'Properties' which is available if an existing Global Variable List is marked in the object organizer. It shows the configuration of this list..


**Dialog to create a new Global Variable List**

**Name of the global variable list:** Insert a list name.

Link to file:

**Filename:** If you have an export file (\*.exp) or a DCF file, which contains the desired variables, you can set up a link to this file. To do this, insert the path of the file in the field **Filename** resp. press the button **Browse** to get the standard dialog 'Select text file'. DCF files are converted to ICE syntax when they are read in.

Activate option **Import before compile**, if you wish that the variable list will be read from the external file before each compilation of the project. Activate the option **Export before compile**, if you want the variable list to be written to the external file before each compilation of the project.

If you close the 'Global variable list' dialog with **OK**, the new object is created. Global variables lists can be recognized in the Object Organizer by the symbol . With the command 'Project' 'Object' 'Properties' you can re-open the 'Global variable list' configuration dialog for the entry marked in the Object Organizer.

## Global Variables, Variable Configuration, Document Frame

Configuration of network variables:

If the option 'Support network variables' is activated in the target settings, then the button <Add network> is available. Pressing this button the dialog gets extended and looks like shown in the picture. If the option is not activated, the button is not available.

**Connection <n> (<Network type>):** In the lower part of the dialog you can create configuration sets for up to four network connections, each on a separate tab. A configuration set defines the parameters of the data exchange for the particular variables list within the network. In order for the exchange in the network to work as intended, the same variable list must be compatibly configured to match in the other network subscribers.

If no configuration is yet present, you will get in the case of a UDP network a single tabulator sheet with the inscription '**Connection 1 (UDP)**'. Each time the 'Add network' button is pressed again, you get up to four more sheets inscribed with serial numbers after „Connection“.

**Network type:** Choose the desired type from the list. The list is defined by the target system entries. For example, „CAN“ as an abbreviation for a CAN network, or „UDP“ for a UDP transmission system, might be selectable.

**Settings:** This button opens the dialog **Settings for <networktype>** with the following configuration parameters:

UDP:

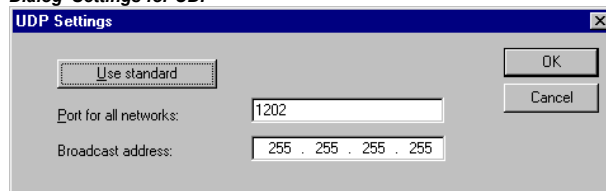
**Use standard** If this button is pressed, Port 1202 will be defined for the data exchange with the other network participants. The Broadcast/Multicast address will be set to "255 . 255 . 255 . 255", which means, that the data exchange will be done with all participants in the network.

**Port:** Enter here a desired port number to overwrite the standard setting (see above, Use standard). Make sure that the other nodes in the network define the same port! If you have more than one UDP connection defined in the project then the port number will be automatically modified in all configuration sets according to the input you make here.

**Broadcast/Multicast address:** Enter here an address resp. the address range of a sub-network, if you want to overwrite the standard setting (e.g. "197 . 200 . 100 . 255", if you want to communicate with all nodes with IP-addresses 197 . 200 . 100 . x).

Regard for Win32 systems, that the Broadcast addresses must match the subnet mask of the TCP/IP configuration of the PC !

*Dialog 'Settings for UDP*



CAN:

**Controller Index:** Index of the CAN Controller, by which the variables should be transferred.

The following options can be activated or deactivated in configuring the transmission behaviour of the variables:

**Pack variables:** The variables are assembled for transfer into packets (telegrams) whose size depends on the network. If the option is deactivated, a packet is set up for each variable.

**List identifier (COB-ID):** Identification number of the first packet, in which the variables will be sent. (default = 1). Further packets will be numbered in ascendant order. Note: A COB-ID **must be unique** within all network variables lists used in the project. Using the same ID in multiple list configurations will cause a compile error with compiler versions >= 2.3.7.0.

## 6 - The Ressources

It depends on the target system, whether the network variables of the list can be defined to be 'readable' and 'writing' or exclusively one of both. To set this property activate the respective options 'Read' and 'Write':

**Read:** The variables in the list will be read; if the option is deactivated, further variables sent over the net will be ignored. The following option can be activated in addition:

**Request at Bootup:** If the local node is a "reading" node (Option 'Read' activated), then as soon as it gets re-booted the actual variable values will be requested from all writing nodes and will be sent by those, independently of any other transmit conditions (time, event), which normally trigger the communication. Precondition: In the configuration of the writing nodes the option 'Answer Bootup requests' must be activated ! (see below).

**Write:** The variables will be written, the following options can be set additionally:

**Include Checksum:** A checksum will be added to each packet which is sent. The checksum will be checked by the receiver to make sure that the variable definitions of sender and receiver are identical. A packet with a non-matching checksum will not be accepted and – if this is configured ('Use acknowledge transfer', see below) – will be acknowledged negatively.


**Use acknowledged transfer:** (no function in case of CAN) Each message will be acknowledged by the receiver. As soon as the sender does not get at least one acknowledgement within a cycle, an error message will be produced, which in case of an UDP-network will be written to the diagnosis structure provided by NetVarUdp\_LIB\_V23.lib.

**Answer Bootup requests:** If the local node is a "writing" node (Option 'Write' activated), then each request of a reading node which is sent by it at bootup (Option Request on Bootup, see above), will be answered. That means that the actual variable values will be transmitted even if none of the other defined transmission triggers (time or event) would force this at this moment.

**Transmit each cycle:** Variables are written within the intervals specified after **Interval**. (time notation e.g. T#70ms).

**Transmit on change:** Variables are written only when their values change; an entry after Minimum can, however, set a minimum time lapse between transfers.

**Transmit on event:** The variables of the list will be written as soon as the variable inserted at **Variable** gets TRUE.

Global Network variables lists are marked by the symbol  in the Object Organizer.

<b>Note:</b>	If a network global variable is used on one or more <b>tasks</b> , the following applies to the time component of the transfer: When each task is called it is tested to determine which parameters apply to the transfer of the variable value (configuration in the 'Global variables list' dialog). The variable value will be transferred or not, depending on whether the specified time interval has passed. At each transfer the time interval counter for this variable is reset to zero.
--------------	---

Sending is always undertaken from the run-time system of the controller affected. Thus no control-specific functions have to be provided for the data exchange.

### Editing Global Variable and Network Variable Lists

The editor for global variables works similar to the declaration editor. But note that you cannot edit in this editor an list, which is an image of an linked external variable list ! External variable lists only can be edited externally and they will be read at each opening and compiling of the project.

Syntax:

```
VAR_GLOBAL
(* Variables declarations *)
END_VAR
```

Network variables only can be used, if allowed by the target system. They are also defined in this syntax.

Example of a network variables list which was created by linking of an export file \*.exp and which got the name NETWORKVARIABLES\_UDP:



## Global Variables, Variable Configuration, Document Frame

Example of a network variables list, which has been created by loading an export file \*.exp and which was named Network\_Vars\_UDP:

```

0001 VAR_GLOBAL CONSTANT
0002 MAX_NetVarItems_UDP : INT := 0;
0003 MAX_NetVarPDO_Rx_UDP : INT := 0;
0004 MAX_NetVarPDO_Tx_UDP : INT := 0;
0005 MAX_NetVarOD_UDP : INT := 0;
0006 END_VAR
0007 VAR_GLOBAL
0008 pNetVarItems_UDP : ARRAY[0..MAX_NetVarItems_UDP] OF NetVarDataItem_UDP;
0009 pNetVarPDO_Rx_UDP : ARRAY[0..MAX_NetVarPDO_Rx_UDP] OF NetVarPDO_Rx_UDP;
0010 pNetVarPDO_Tx_UDP : ARRAY[0..MAX_NetVarPDO_Tx_UDP] OF NetVarPDO_Tx_UDP;
0011 pNetVarOD_UDP : ARRAY[0..MAX_NetVarOD_UDP] OF NetVarSDO_UDP;
0012 END_VAR

```

### Editing Remanent Global Variables Lists

If they are supported by the runtime system, remanent variables may be processed. There are two types of remanent global variables (see also Chapter 5.2.1, Remanent Variables !):

**Retain variables** remain unchanged after an uncontrolled shutdown of the runtime system (off/on) or an 'Online' 'Reset' in CoDeSys. **Persistent variables** remain unchanged only after a download.

Persistent variables are not automatically also Retain variables !

Remanent variables are additionally assigned the keyword **RETAIN** or **PERSISTENT** or both.

See chapter 5.2.1, Remanent Variables, for further information.

Network variables are also defined in this syntax.

Syntax:

```

VAR GLOBAL RETAIN
(*_Variables declarations *)
END_VAR

VAR GLOBAL PERSISTENT
(*_Variables declarations *)
END_VAR

```

For the combination of retain and persistent properties both keywords are used:

```

VAR_GLOBAL RETAIN PERSISTENT OF VAR_GLOBAL PERSISTENT RETAIN

```

Network variables (target specific) are also defined using this syntax.

### Global Constants

Global constants additionally get the keyword **CONSTANT**.

Syntax:

```

VAR GLOBAL CONSTANT
(*_Variables declarations *)
END_VAR

```

## 6.2.2 Variable Configuration

In function blocks it is possible to specify addresses for inputs and outputs that are not completely defined, if you put the variable definitions between the key words **VAR** and **END\_VAR**. Addresses not completely defined are identified with an asterisk.

**Example:**

```

FUNCTION_BLOCK locio
VAR
  loci AT %I*: BOOL := TRUE;

```

## 6 - The Resources

```

    loco AT %Q*: BOOL;
END_VAR

```

Here two local I/O-variables are defined, a local-In (%I\*) and a local-Out (%Q\*).

If you want to configure local I/Os for variables configuration in the Object Organizer in the **Resources** register card, the object **Variable\_Configuration** will generally be available. The object then can be renamed and other objects can be created for the variables configuration.

The editor for variables configuration works like the declaration editor.

Variables for local I/O-configurations must be located between the key words **VAR\_CONFIG** and **END\_VAR**.

The name of such a variable consists of a complete instance path through which the individual POU's and instance names are separated from one another by periods. The declaration must contain an address whose class (input/output) corresponds to that of the incompletely specified address (%I\*, %Q\*) in the function block. Also the data type must agree with the declaration in the function block.

Configuration variables, whose instance path is invalid because the instance does not exist, are also denoted as errors. On the other hand, an error is also reported if no configuration exists for an instance variable. In order to receive a list of all necessary configuration variables, the "All Instance Paths" menu item in the 'Insert' menu can be used.

**Example for a Variable Configuration**

Assume that the following definition for a function block is given in a program:

```

PROGRAM PLC_PRG
VAR
Hugo: locio;
Otto: locio;
END_VAR

```

Then a corrected variable configuration would look this way:

```

VAR_CONFIG
PLC_PRG.Hugo.locio AT %IX1.0 : BOOL;
PLC_PRG.Hugo.loco AT %QX0.0 : BOOL;
PLC_PRG.Otto.locio AT %IX1.0 : BOOL;
PLC_PRG.Otto.loco AT %QX0.3 : BOOL;
END_VAR

```

**'Insert' 'All Instance Paths'**

With this command a **VAR\_CONFIG - END\_VAR**-block is generated that contains all of the instance paths available in the project. Declarations already on hand do not need to be reinserted in order to contain addresses already in existence. This menu item can be found in the window for configuration of variables if the project is compiled ('Project' 'Rebuild All').

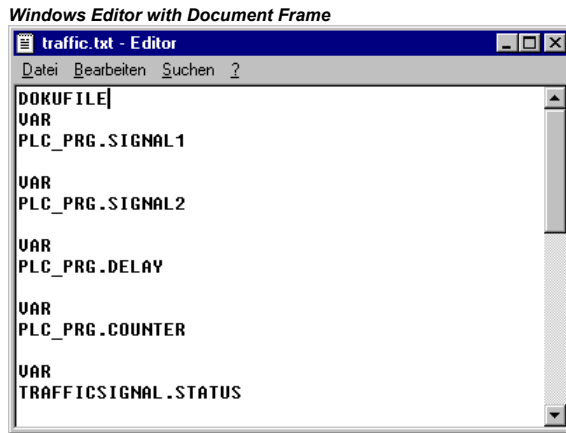
**6.2.3 Document Frame**

If a project is to receive multiple documentations, perhaps with German and English comments or if you want to document several similar projects that use the same variable names, then you can save yourself a lot of work by creating a docuframe with the 'Extras' 'Make Docuframe File' command.

The created file can be loaded into a desired text editor and can be edited. The file begins with the **DOCUFILE** line. Then a listing of the project variables follows in an arrangement that assigns three lines to each variable: a **VAR** line that shows when a new variable comes; next, a line with the name of the variable; and, finally, an empty line. You can now replace this line by using a comment to the variable. You can simply delete any variables that you are unable to document. If you want, you can create several document frames for your project.

In order to use a document frame, give the 'Extras' 'Link Docu File' command. Now if you document the entire project, or print parts of your project, then in the program text, there will be an insertion of the comment produced in the docuframe into all of the variables. This comment only appears in the printout!

## Alarm Configuration

**'Extras' 'Make Docuframe File'**

Use this command to create a document frame. The command is at your disposal, whenever you select an object from the global variables.

A dialog box will open for saving files under a new name. In the field for the **name file**, the \*.txt extension has already been entered. Select a desired name. Now a text file has been created in which all the variables of your project are listed.

**'Extras' 'Link Docu File'**

With this command you can select a document frame.

The dialog box for opening files is opened. Choose the desired document frame and press **OK**. Now if you document the entire project, or print parts of your project, then in the program text there will be an insertion of the comment produced in the docuframe into all of the variables. This comment only appears in the printout!

To create a document frame, use the 'Extras' 'Make Docuframe File' command.

## 6.3 Alarm Configuration

### 6.3.1 Overview

The alarm system integrated in CoDeSys allows detecting critical process states, recording them and visualizing them for the user with the aid of a visualization element. The alarm handling can be done in CoDeSys or alternatively in the PLC. For alarm handling in the PLC please see the target settings category 'Visualization'.

**If supported by the target system**, the dialogs for the 'Alarm configuration' are available in the 'Resources' tab.

Here you define **Alarm classes** and **Alarm groups**. An alarm class serves for the typing of an alarm, that means it assigns certain parameters to the alarm. An alarm group serves for the concrete configuration of one or several alarms (which get assigned a certain class and further parameters) for the use in the project. Thus a class is useful for structuring the available alarms. The different alarm groups are defined by the user by inserting appropriate entries below the header '**System**' in the configuration tree.

For the **visualization** of alarms the element 'Alarm table' is available in the CoDeSys visualization. Using this table the user can watch and acknowledge alarms.