Supervisión de un sistema de control de nivel con sistemas expertos

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

MEMORIA

Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual
de Terrassa

Màster Universitari en Enginyeria Industrial en Sistemas Automàtics i Electrònica Industrial

Autores:
Christian Tigse
Adrià Ventura
Óscar Bautista

29 de mayo de 2018

Contenido

Índice de figuras

Índice de tablas

1. Introducción

1.1. Problema a resolver

El trabajo se basa en crear un supervisor de un sistema de control de nivel, por lo tanto, supervisa el controlador, no el proceso. Para crear este supervisor, se crea un sistema experto (rama de la inteligencia artificial) mediante el entorno de programación CLIPS.

El proceso (figura 1) que es controlado a través de este sistema de control consta de:

- Dos depósitos de líquido interconectados.
- Una bomba que permite la circulación de agua desde el depósito inferior al superior.
- Tres válvulas manuales (Válvula 1, Válvula 2 y Válvula 3).
- Un sensor de nivel (LT).
- Un sensor de caudal superior (LV).
- Un controlador de nivel (LC).

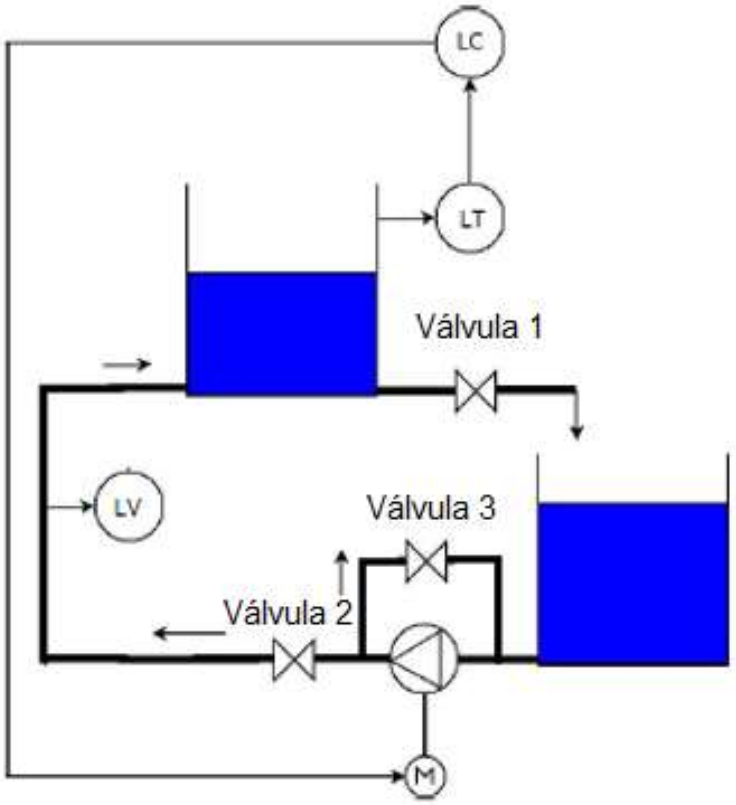

Figura 1. Esquema del proceso de estudio (1).

La circulación de agua en el sistema (figura 1) se realiza, desde el tanque inferior al superior, a través de la bomba y las tres válvulas. Se envía agua desde un depósito hasta el otro hasta que el sensor LT mide el mismo valor que la consigna. El controlador LC, a través de la señal de LT y la consigna introducida, da a la bomba la tensión necesaria para incrementar, reducir o mantener el nivel de líquido en el depósito superior con el objetivo de reducir el error entre el valor del

sensor LT y el valor deseado. La rapidez del sistema dependerá del estado de las válvulas y del diseño del controlador[1].

El supervisor, que será un sistema experto, determinará el estado del proceso a partir de los valores de los sensores y los actuadores registrados. Entonces, el supervisor no corregirá nada, únicamente analizará, determinará el estado del sistema y propondrá acciones correctivas.

Para entender mejor el esquema de la figura 1 y el sistema que se utilizará en este trabajo, a continuación, se muestra una imagen (figura 2) de la estación del laboratorio.

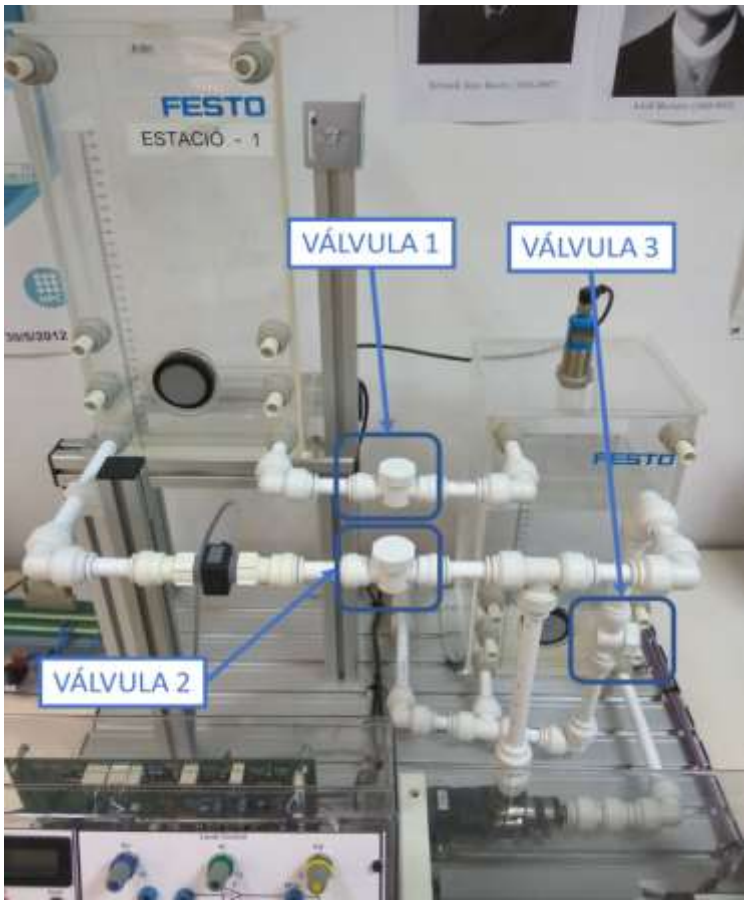

Figura 2. Imagen I in situ de la estación de laboratorio.

Los componentes del sistema (figura 2 y figura 3) de estudio que se utilizarán son:

- Válvula 1.
- Válvula 2.
- Válvula 3.
- Sensor de caudal.
- Sensor de nivel.
- Bomba.
- Panel analógico.
- Tarjeta de adquisición y envío de señales.
- PC.

Figura 3. Imagen II in situ de la estación de laboratorio.

1.2. Objetivos

El objetivo principal de este estudio es crear un supervisor, un sistema experto, de un sistema de control de nivel que sea capaz de analizar y determinar el estado del proceso (ver figura 2).

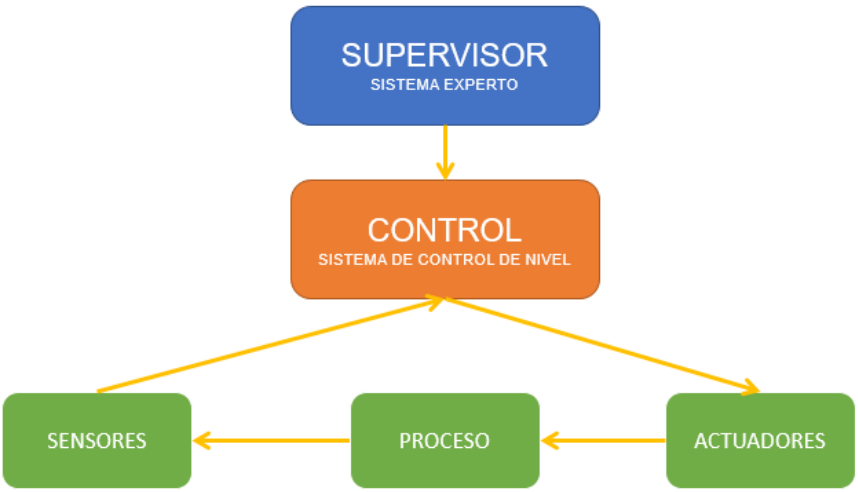

Figura 4. Esquema de supervisor (nivel superior), el sistema de control (nivel medio) y el proceso (nivel inferior).

Para ello, el primer objetivo de este trabajo será estudiar el concepto de supervisión de sistemas dinámicos y la teoría subyacente a los sistemas expertos.

El segundo será entender el funcionamiento del proceso y su sistema de control. Por último, se programará un sistema experto mediante CLIPS, un entorno de creación de sistemas expertos.

Considerando estos objetivos indicados en el párrafo anterior, se persigue dar los pasos adecuados para la creación de un sistema experto capaz de supervisar el sistema de control del proceso.

1.3. Planificación del trabajo

Se ha realizado una descomposición jerárquica del trabajo (WBS) para identificar los paquetes de trabajo (descomposición basándose en los diferentes procesos). Posteriormente, se ha definido la ventana temporal para cada actividad identificada dentro del límite de tiempo de entrega del trabajo como se indica en la Tabla 1.

Actividad	Fecha de inicio	Fecha de finalización
1. Análisis inicial del problema.	05/03/2018	09/03/2018
2. Planificación inicial.	05/03/2018	09/03/2018
3. Lectura y búsqueda de bibliografía.	06/03/2018	09/03/2018
4. Análisis del proceso y el sistema de control.	12/03/2018	16/03/2018
5. Aprendizaje de la programación en CLIPS.	12/03/2018	28/03/2018
6. Programación del sistema experto.	19/03/2018	06/04/2018
7. Comunicación entre sistema de control y sistema experto.	09/04/2018	20/04/2018
8. Modificación y mejora del sistema experto.	23/04/2018	06/05/2018
9. Redacción de la memoria.	07/05/2018	15/05/2018
10. Exposición oral.	23/05/2018	28/05/2018

Tabla 1. Planificación temporal.

Primero se ha analizado el enunciado del trabajo y se ha hecho una planificación de los pasos a seguir para llegar al objetivo final de crear un supervisor mediante un sistema experto.

Seguidamente, se ha buscado bibliografía y leídos libros para profundizar en el concepto de sistemas de supervisión dinámicos y sistemas expertos dando paso al análisis del proceso y el sistema in situ. Paralelamente, se ha aprendido a programar en CLIPS, el entorno de programación que se ha utilizado para crear el sistema experto y se ha programado durante el aprendizaje el programa que se ha utilizado para supervisar el sistema de control del sistema.

Después de crear el primer programa, se ha establecido comunicación entre los datos obtenidos del sistema de control del proceso y el entorno de CLIPS permitiendo evaluar el primer resultado. Al observarse los diferentes aspectos a mejorar se ha pivotado hasta tener la versión definitiva.

Finalmente, se ha elaborado la memoria del proyecto y se ha preparado una exposición oral con finalidad didáctica para la clase.

2. La supervisión y los sistemas expertos

A continuación, se explicará la supervisión de los sistemas de producción y se hará una introducción a la teoría subyacente a los sistemas expertos.

2.1. Supervisión de sistemas de producción

Los sistemas de supervisión, monitorización y control de sistemas productivos surgen del incremento de la exigencia que se ha dado en cuanto a la calidad y la flexibilidad de los procesos industriales. Paralelamente, también ha crecido la exigencia en cuanto al rendimiento del proceso. En conjunto, estos hechos han obligado a informatizarse la industria con el fin de tener más datos y poder realizar un seguimiento de tareas. Un ejemplo actual son los SCADA que pueden monitorizar y controlar procesos a través de registrar datos de ellos y proporcionar una interacción entre el proceso y el usuario[2].

Estos sistemas de supervisión realizan la tarea, que se ha realizado desde siempre en la industria, que una persona hacía cuando vigilaba y decidía sobre la tarea según lo que observaba (información). Actualmente, el incremento de la complejidad de los sistemas productivos ha obligado a incorporar sistemas de supervisión[2].

Para crear un sistema de supervisión, el primer paso es centralizar y registrar los datos del proceso a supervisar. Una función que realizan los SCADA y otros programas de supervisión[2].

La supervisión tiene el objetivo de automatizar las tareas que realizan los operarios o responsables del proceso que detectan situaciones anómalas y toman decisiones en consecuencia[2].

Destacar que hoy en día los sistemas de supervisión existen, pero actúan junto a una intervención humana. Aún así, si se compara la automatización con la supervisión, esta última está por encima, jerárquicamente, ya que se encarga de que las tareas automatizadas tengan un correcto funcionamiento[2].

Un sistema de supervisión tiene que ofrecer las siguientes funciones:

- Registrar datos del proceso y detectar desviaciones.
- Analizar desviaciones e indicar la causa.
- Realizar informes de la situación y resolver las situaciones anómalas, en el caso de ser posible.
- Tomar medidas adecuadas para que no se vuelvan a dar las mismas situaciones.

Para que el supervisor actúe como tal, se le tiene que transmitir todo el conocimiento heurístico y teórico (reglas, condiciones, restricciones, etc.) para que pueda hacer una perfecta interacción con el proceso y su funcionamiento[2].

El funcionamiento que sigue un sistema de supervisión consta de tres etapas generales (ver figura 5):

1. Detectar fallos.
2. Diagnosticar fallos.
3. Reconfigurar el sistema → Proponer acciones sobre el sistema.

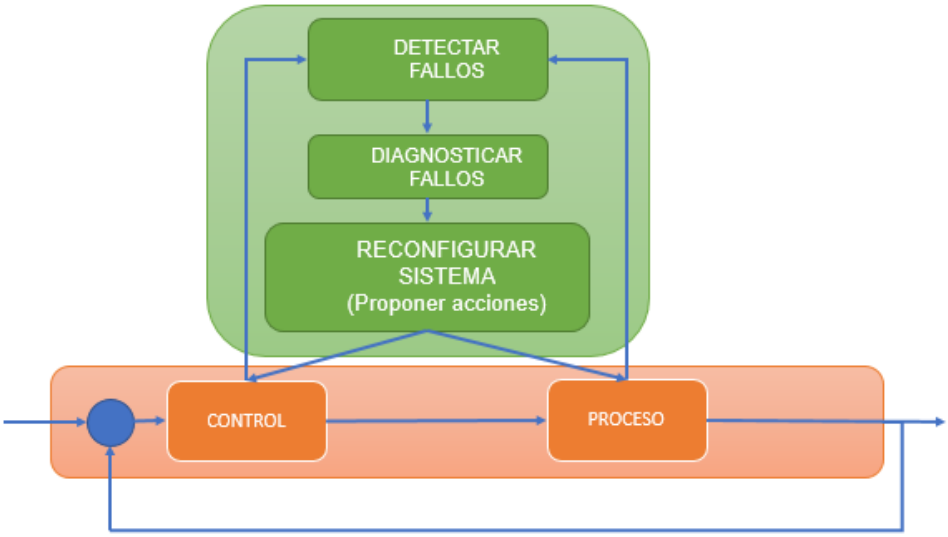

Figura 5. Etapas que sigue un sistema de supervisión (verde) (2).

Que el sistema realice las tres etapas o no nos permite diferenciar entre un entorno de supervisión y un sistema de monitorización (figura 6), siendo este último el que no propone acciones sobre el sistema[2].

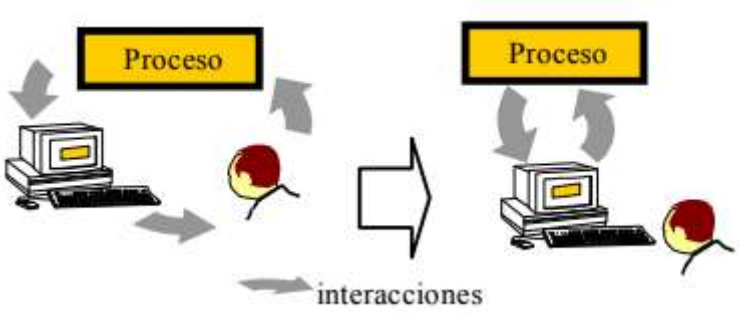

Figura 6. Diferencia entre un sistema de monitorización (izquierda) y un sistema de supervisión (derecha) (2).

Tener en cuenta que en muchos casos se considera un sistema de supervisión cuando realmente participa un sistema de monitorización y un operario. Por lo que, un sistema de supervisión es capaz de hacer lo mismo que un sistema de monitorización, pero añadiendo la posibilidad de proponer acciones sobre el sistema[2].

El sistema de supervisión, para poder realizar las tres etapas mencionadas en la página anterior, necesita tener un registro de datos centralizado. Para detectar los

fallos y proponer acciones, necesita que se le transmita el modelo teórico del proceso para detectar situaciones anómalas y conocer sus posibles caminos para llegar al comportamiento deseado. Además, en la reconfiguración del sistema, se utilizan métodos estadísticos y técnicas de Inteligencia artificial que permiten al sistema de supervisión utilizar conocimiento y experiencias transmitidas de forma automática junto al entorno de monitorización[2].

Por lo tanto, el sistema experto que se crea en este trabajo, realmente, es un sistema de monitorización ya que cumple dos de tres etapas. Detecta y diagnostica fallos a partir de los datos centralizados obtenidos del proceso, pero no propone acciones para obtener el funcionamiento deseado del sistema definido con la consigna. Aun así, el operador o el encargado del proceso ve disminuida la carga de trabajo correspondiente a la vigilancia constante y las tareas repetitivas (redacción de informes, análisis de datos, etc.). Además, la implementación de estos sistemas también permite una adaptación más eficaz del personal al proceso industrial asociado[2].

En el caso de estudio, la detección de fallos se realiza sabiendo que supervisamos el sistema de control por lo que un mal funcionamiento de este será indicado como un fallo. El siguiente paso será diagnosticar los fallos, que, en este caso, por ejemplo, sería el tener un nivel de consigna mayor de 10 o inferior a 0 (nivel de líquido en el depósito superior). Finalmente, la reconfiguración, es decir, la propuesta de acciones, se indicaría al usuario el poner un valor de consigna entre 0 y 10[2].

2.2. Sistemas expertos.

2.2.1. Introducción

¿Qué es un sistema experto?

Los sistemas expertos son una rama de la inteligencia artificial que utilizan el conocimiento de un especialista humano en algún área para resolver problemas por sí solos en esa área. Es decir, el sistema experto emula la habilidad de un experto de tomar decisiones. A pesar de que el resolver problemas de carácter general puede ser una meta deseada, los sistemas expertos son eficaces en dominios de problemas restringidos[3,4].

El concepto básico de un sistema experto es que el usuario le aporta los hechos y la información y recibe consejos e información a cambio. En el interior del sistema experto hay, básicamente, dos componentes: la base de conocimiento y el mecanismo de inferencia. La base de conocimiento contiene toda la información necesaria que el mecanismo de inferencia necesita para sacar conclusiones a través de los hechos aportados. En la figura 7 se puede entender de forma esquemática este concepto[3,4].

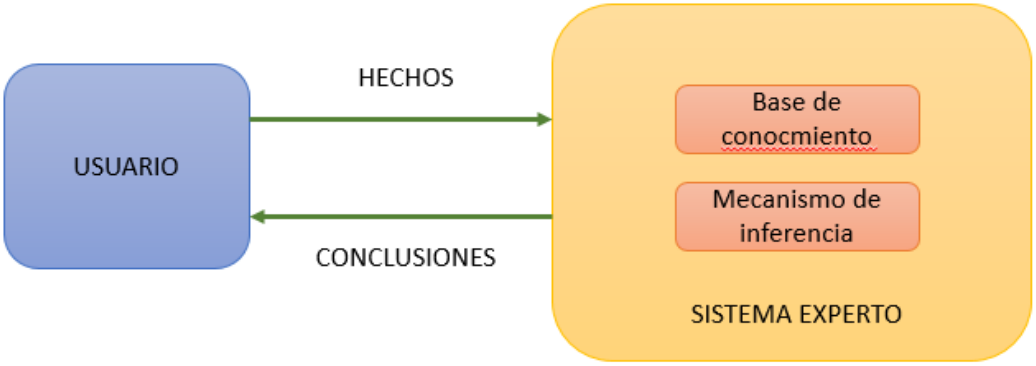

Figura 7. Concepto básico de un sistema experto (3).

En el dominio de conocimiento del sistema experto, este hace inferencias ("razona") igual que hace un humano. A partir de unos hechos se saca una conclusión[3,4].

Este conocimiento puede representarse de diferentes maneras, encapsulado en reglas y/o objetos. Las reglas pueden ser del tipo SI…ENTONCES. Por ejemplo, "SI la luz es roja ENTONCES deténgase". En el caso de darse el hecho de que la luz sea roja, entonces el sistema experto sacará como conclusión detenerse. Pero, todo este conocimiento debe sacarse de un especialista a través de entrevistas y después ser codificado en el sistema experto mediante entornos especializados como es CLIPS (entorno que utilizaremos en este trabajo)[3,4].

Los sistemas expertos ofrecen una solución diferente a los programas convencionales. En vez de solucionar problemas mediante algoritmos, propio de los programas convencionales, dependen de inferencias para llegar a la solución razonable[3,4].

Por lo tanto, la tipología de conocimiento utilizado para programar sistemas expertos es mucho más apropiado que sea un conocimiento superficial, basado en la empírica y la heurística, que no en un conocimiento profundo, donde aparecen las estructuras básicas, las funciones y el comportamiento de los objetos. Para facilitar la comprensión de ello, es mucho más fácil programar un sistema experto con la finalidad de que recete un medicamento (conclusión) por tener un dolor específico (hecho) que no programar todo el conocimiento bioquímico del cuerpo humano[3,4].

Mientras que un algoritmo es una solución garantizada para un problema, la heurística o conocimiento empírico no garantiza la solución a un problema, pero puede ayudar en la solución a través del conocimiento extraído de la experiencia[3,4].

Otro problema asociado a esta técnica de la IA es que su eficacia está limitada al área del dominio, y no se puede utilizar su conocimiento para hacer analogías sobre otras áreas o situaciones[3,4].

El quid de la cuestión trata en saber diferenciar cuando utilizar la programación convencional y cuando la programación de sistemas expertos[3,4].

Características de un sistema experto

Un sistema experto se diseña para que tenga las siguientes características[3,4]:

- Capacidad de responder con un grado de competencia igual o superior al especialista.
- El tiempo de respuesta tiene que ser el adecuado según la situación.
- Comprensible. Tiene que ser capaz de explicar los pasos que ha dado a través del razonamiento que ha seguido.
- Flexibilidad para añadir, modificar o eliminar conocimiento.

Desarrollo de los sistemas expertos

Una de las raíces de las cuales provienen los sistemas expertos es la ciencia cognitiva, el área de procesamiento de la información humana. El entender la cognición, como piensa la gente, es fundamental para hacer que los ordenadores emulen este comportamiento. El mecanismo de inferencia, visto en la figura 7, sería el procesador cognitivo humano. El procesador cognitivo de un humano trata de encontrar que reglas se satisfacen, dado un estímulo, para sacar unas conclusiones determinadas[3,4].

El modelo utilizado por las personas para resolver problemas donde actúa la memoria a largo plazo, que son las reglas, la memoria a corto plazo (memoria activa) y el procesador cognitivo (mecanismo de inferencia) es el fundamento de los sistemas expertos utilizados actualmente. La memoria a corto plazo es el almacenamiento temporal de información para solucionar el problema[3,4].

Aplicaciones de los sistemas expertos

Para poner un ejemplo real, se menciona el sistema XCON de Digital Equipment Corporation (DEC). Este es un sistema experto que se utiliza para configurar los sistemas de cómputo. Es decir, cuando un cliente hace un pedido de un sistema de cómputo grande (software, hardware, interconexiones, cableado, documentación, …), este sistema experto determina todas las partes y configura el pedido correctamente. Esto lo que ofrece es una reducción significativa del tiempo de configuración de los pedidos y una reducción de las equivocaciones en cuanto a componentes[3,4].

Dominios apropiados para los sistemas expertos

Antes de programar un sistema experto es aconsejable hacerse consideraciones acerca de si el dominio del problema es apropiado para esta técnica de inteligencia artificial. Que sea apropiado o no depende de diferentes factores[3,4]:

- Si el problema se puede solucionar mediante programación convencional no es un problema apropiado para sistemas expertos. Estos últimos son apropiados en situaciones en que una solución algorítmica no es eficiente.
- El dominio del sistema experto tiene que estar bien definido, es decir, que se espera de él teniendo en cuenta que, cuanto más amplio sea el dominio, más complejo será el sistema experto.

- ¿Es necesario un sistema experto? Un sistema experto se hace para después ser utilizado y también ser aceptado, dado que es una técnica menos conocida que la programación convencional.
- Tiene que haber por lo menos un especialista humano sobre el área interesada que esté dispuesta a cooperar. Es importante este punto ya que el paso del conocimiento del especialista al sistema experto conlleva un tiempo crítico, un paso que suele ser el cuello de botella del proceso de creación de un sistema experto.
- A parte de haber un especialista humano, ¿es capaz de transmitir su conocimiento?
- El conocimiento apropiado para resolver el problema tiene que ser heurístico e incierto.

2.2.2. Tipos de sistemas expertos

A parte del sistema experto explicado anteriormente, basado en reglas establecidas con anterioridad, hay otros dos tipos de sistemas expertos[3,4]:

1. Basados en la heurística.
2. Basados en casos (Case Based Reasoning, CBR).
3. Basados en redes bayesianas.

Para los sistemas expertos basados en reglas, su estructura tiene un mecanismo de inferencia y la base de conocimiento para a partir de unos hechos aportar unas conclusiones[3,4].

Para el segundo tipo de sistema experto se puede decir que deriva del primero. Esta tipología se da cuando se quiere reutilizar el conocimiento de un sistema experto para otro problema. Para poder reutilizar el conocimiento se tiene que ajustar y/o modificar tanto como sea necesario siempre y cuando esto requiera un tiempo y dificultad menor a crear un sistema experto desde cero. En este tipo de sistemas expertos también se utiliza el conocimiento de un especialista humano[3,4].

El tercer tipo se le suma los fundamentos probabilísticos al conocimiento del especialista humano. Esto permite solucionar problemas más complejos[3,4].

En este trabajo se creará un sistema experto basado en las reglas para supervisar el sistema de control del proceso[3,4].

3. Resolución del problema

3.1 Estado del sistema.

Antes de la toma de datos es necesario identificar los casos que se obtienen al ir variando las válvulas (abrir o cerrar), es decir, los efectos que causa sobre el sistema. Para realizar las pruebas se utilizó Simulink y la tarjeta de adquisición y envío de señales del laboratorio.

Caso	Válvula 1	Válvula 2	Válvula 3	Efecto
1	Cerrada	Abierta	Cerrada	Nivel óptimo de llenado.
2	Abierta	Abierta	Cerrada	Llega al nivel superior pero tarda más tiempo ya que por la válvula 1 retorna el agua.
3	Cerrada	Abierta	Abierta	Llega al nivel superior pero tarda más tiempo ya que por la válvula 3 retorna el agua.
4	Abierta	Abierta	Abierta	No llega nunca al nivel superior.
5	Cerrada	Cerrada	Cerrada	No llega nunca al nivel superior, recalentamiento en la bomba y no se vacía el depósito superior.
6	Abierta	Cerrada	Cerrada	No llega nunca al nivel superior, recalentamiento en la bomba y se vacía el depósito superior.
7	Cerrado	Cerrada	Abierta	No llega nunca al nivel superior, el agua recircula por el depósito inferior y no se vacía el depósito superior.
8	Abierta	Cerrada	Abierta	No llega nunca al nivel superior, el agua recircula por el depósito inferior y se vacía el depósito superior.

Tabla 2. Efectos del sistema en los diferentes casos.

Al observar la tabla 2, el color verde indica que el sistema cumple con su objetivo que es llenar el depósito superior mediante una consigna establecida. De lo contrario el color rojo indica que el sistema no cumple con su objetivo, y además hay casos en los que la bomba se sobrecalentaría debido a que el agua no puede circular.

La Figura 8 muestra uno de los casos descritos anteriormente, en donde el sistema funcionara de manera óptima puesto que el agua podrá subir sin restricción alguna por la válvula 2 en cuanto se le dé una consigna al sistema. La imagen izquierda muestra el depósito superior vacío y el de la derecha muestra el depósito superior lleno.

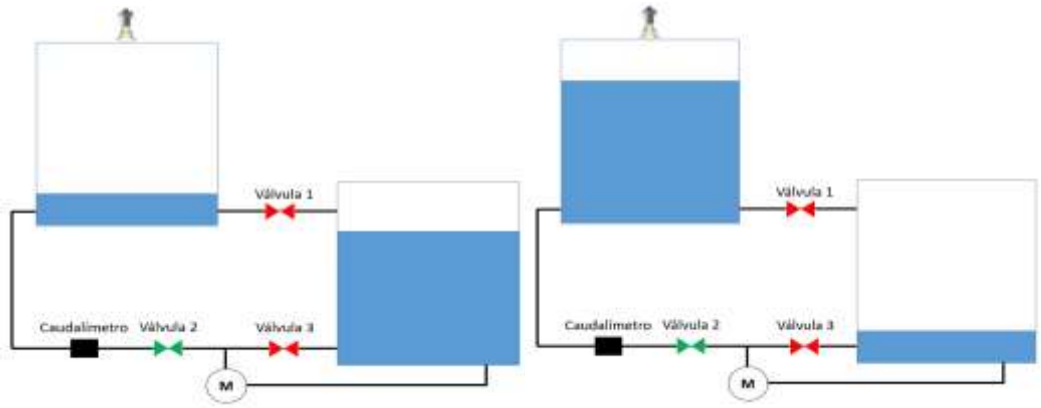

Figura 8. Efectos del sistema en los diferentes casos

3.2 Diagrama de estados.

Una vez se conoce a detalle el comportamiento del sistema, se procede a realizar el diagrama de estados, el cual servirá de base para el desarrollo del sistema experto.

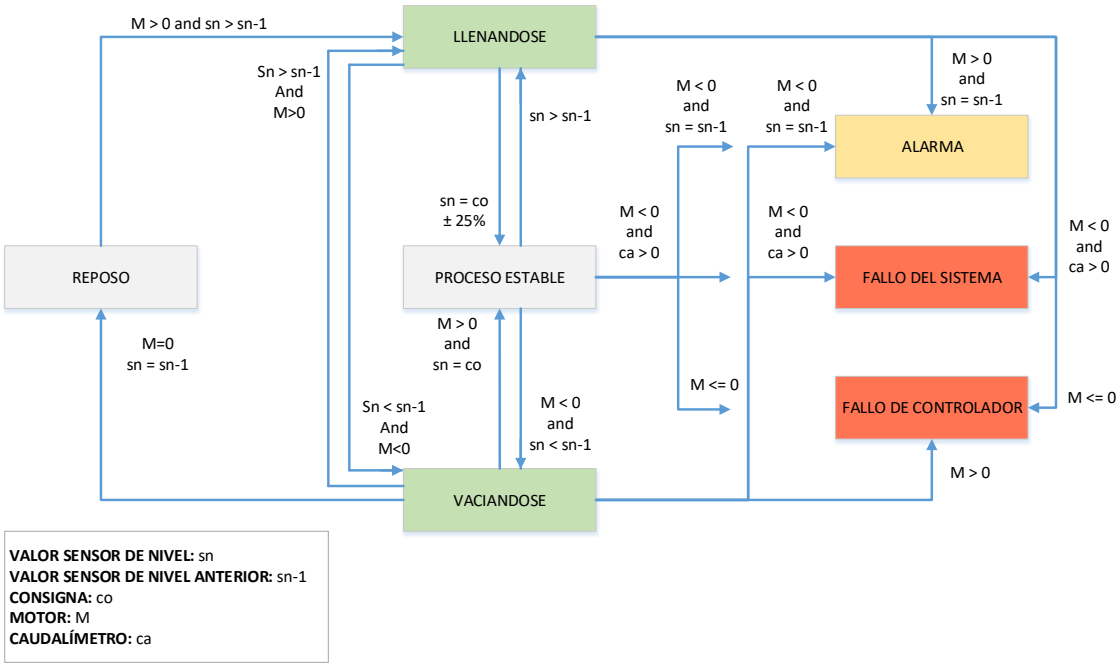

Figura 9. Diagrama de estados

En el diagrama de estados de la figura 9 se puede identificar 7 estados diferentes, los cuales pasaran de uno a otro a medida que se manipule las válvulas 1, 2 y 3 y se establezca valores de consigna. A continuación se describe cada uno de los estados que intervienen en el sistema:

"**REPOSO**", se da cuando el motor está apagado (M≤0), es decir, no se ha establecido una consigna y por lo tanto el valor del sensor de nivel actual coincide con el valor de sensor anterior (sn=0 = sn-1=0).

"**LLENANDOSE**", este estado se da cuando se establece una consigna y por lo tanto el motor se enciende (M>0) y el valor del sensor de nivel es mayor al valor del sensor anterior (sn>sn-1).

"**PROCESO ESTABLE**", una vez que el sistema ha llegado al nivel de consigna establecido (sn=c0), se considera que el proceso está estable cuando se encuentre en un rango de ± 25% de la consigna. Se ha considerado este rango puesto que a un inicio el proceso requiere tiempo para estabilizarse.

"**VACIANDOSE**", este estado se da cuando el motor se encuentra apagado (M≤0) y a su vez el valor del sensor de nivel actual es menor que el valor del sensor de nivel anterior (sn<sn-1).

Como se puede ver en el diagrama de estados, si el nivel de agua continua bajando va a llegar nuevamente al estado de reposo.

Pero también puede darse otros casos considerados bidireccionales, por ejemplo cuando está en la fase de llenado y de improvisto se establece un nivel de consigna inferior entonces pasara al estado "vaciándose" o viceversa, todo depende del nivel de consigna que se establezca. Los casos de "llenado", "proceso estable" y "vaciándose" también cuentan con esta característica.

"ALARMA", las alarmas aparecen cuando por algún motivo falla un sensor y muestran lecturas erróneas, en el diagrama de estados se tiene tres posibles casos de alarma que proviene de los estados "llenándose", "proceso estable" y "vaciándose". Por ejemplo si tomamos el estado "llenándose", este entrara a "alarma" cuando aún de estar la bomba encendida el valor del sensor de nivel actual sea igual al sensor de nivel anterior (sn = sn-1), es decir, simula un estado de reposo.

"FALLO DEL SISTEMA", aparece cuando hay datos totalmente contradictorios y se manifiestan para los tres estados "llenándose", "proceso estable" y "vaciándose". Por ejemplo si el nivel de agua está bajando es lógico pensar que el motor debe estar apagado y el caudalímetro en cero puesto que no hay agua que circule, en este caso como se observa en el diagrama de estados hay falla del sistema porque los valores que se muestra son motor apagado (M≤0) y caudalímetro en funcionamiento (ca>0).

"FALLO DEL CONTROLADOR", este fallo se produce cuando se detecta una actividad inusual o contradictoria del motor o del proceso, por ejemplo que estuviera en estado vaciándose y el motor estuviera activado.

3.3 Obtención de datos.

Para la obtención de datos se utiliza Simulink conjuntamente con la tarjeta de adquisición y envío de señales del laboratorio. El grafico correspondiente a la conexión se muestra en la figura 10.

Esta tarjeta lleva un terminal de bornes (figura izquierda) que permiten conectar fácilmente el computador con el panel frontal de gobierno del proceso para adquirir / enviar señales al proceso. Los bornes OUTPUT PC corresponden al envío de tensión desde del computador y habrá conectarlos (+ rojo y - negro) a la entrada INPUT PUMP del panel frontal. Los bornes INPUT NIVEL y CAUDAL servirán para recibir las señales provenientes de los sensores de nivel y caudal, respectivamente, y que lleguen al computador.

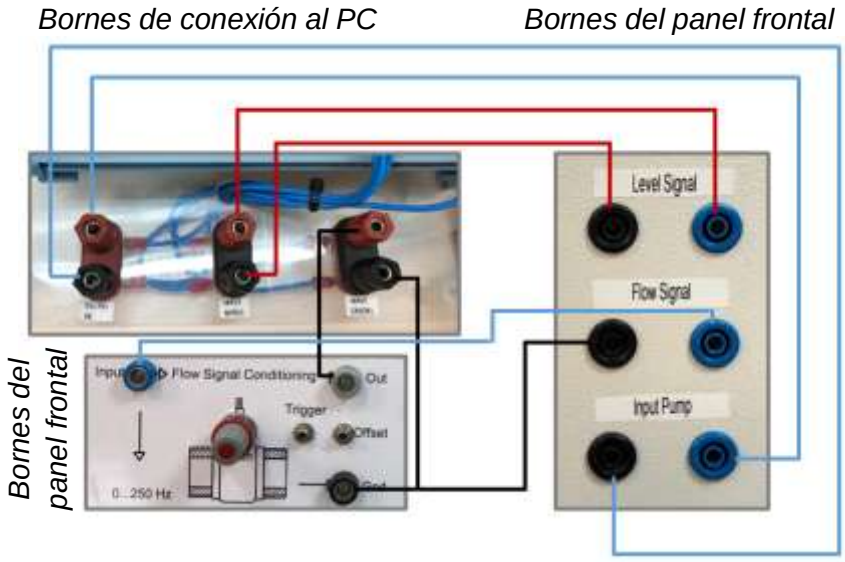

Figura 10. Diagrama de conexión

La figura de la derecha corresponde a los bornes del panel frontal los cuales tienen las siguientes funciones:

Level Signal: El voltaje que da el sensor de nivel y que es proporcional al nivel del líquido del tanque a controlar. Proporciona de 0 a 10 Voltios para un rango de entre 0 y 30 centímetros.
Flow Signal: Onda cuadrada que da el sensor de caudal. Su frecuencia es proporcional al caudal volumétrico impulsado por la motobomba. En la parte inferior izquierda de la figura 10 se muestra los bornes correspondientes al caudalímetro que permite la comunicación para la toma de datos correspondiente ya que este da una señal de salida en función de una frecuencia y con el panel frontal convertimos esta frecuencia a una tensión.

Input Pump: Señal de excitación de la motobomba. En realidad se trata de una tensión entre -10 V y 10 V que se introduce a la entrada del driver, que es quien alimenta la motobomba, proporcionándole la potencia eléctrica necesaria para poder impulsar el agua.
La conexión física del sistema de llenado se muestra a continuación:

Figura 11. Conexión del sistema físico para la toma de datos.

3.4 Diagrama de bloques del proceso

El proceso de laboratorio se puede representar de forma esquemática mediante un diagrama de bloques en el que aparece cada una de las subpartes en que se divide.

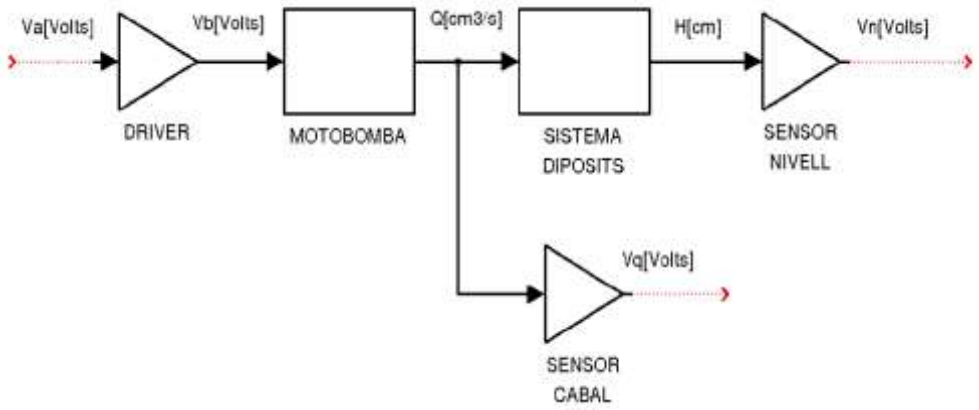

Figura 12. Diagrama de bloques del proceso

En la figura 12 aparece un elemento llamado DRIVER que recibe la tensión de alimentación; es el elemento (dispositivo electrónico) que permite dar potencia a la bomba para que ésta pueda impulsar el agua. La motobomba recibe la potencia eléctrica del DRIVER y la transforma en un caudal volumétrico impulsado a través de la tubería desde el depósito inferior al superior, cuyo nivel evoluciona dinámicamente resultando en un nivel de entre 0 y 30 centímetros. El caudal que impulsa la motobomba se mide a través de un sensor en forma de turbina que transforma los cm3 / seg que pasan en una onda cuadrada que tiene más frecuencia cuanto más caudal circule a través de ella. El depósito superior dispone de un sensor de nivel de líquido que funciona a través de ultrasonidos y da un valor de tensión eléctrica de entre 0 y 10 Voltios que es aproximadamente proporcional a la altura del líquido en el tanque.

El agua llega al depósito superior por el fondo del tanque de forma que cuanto más lleno de líquido se encuentre el tanque superior, más le costará a la bomba impulsar el agua de un lugar a otro.

3.5 Modelo de simulación del proceso de laboratorio

La siguiente figura muestra el diagrama de bloques en SIMULINK que permita simular el modelo (determinado en el apartado anterior) del depósito con el voltaje de la bomba como entrada y el nivel del depósito superior como salida.

Para evaluar el sistema se utiliza un regulador PI con una Kp = 5 y Ki = 1, el cual presenta una buena respuesta y sincronización con el sistema físico. El proceso inicia cuando se designa una consigna en Matlab y esta a su vez envía la señal a la bomba para que se encienda y tome los valores de las otras variables.

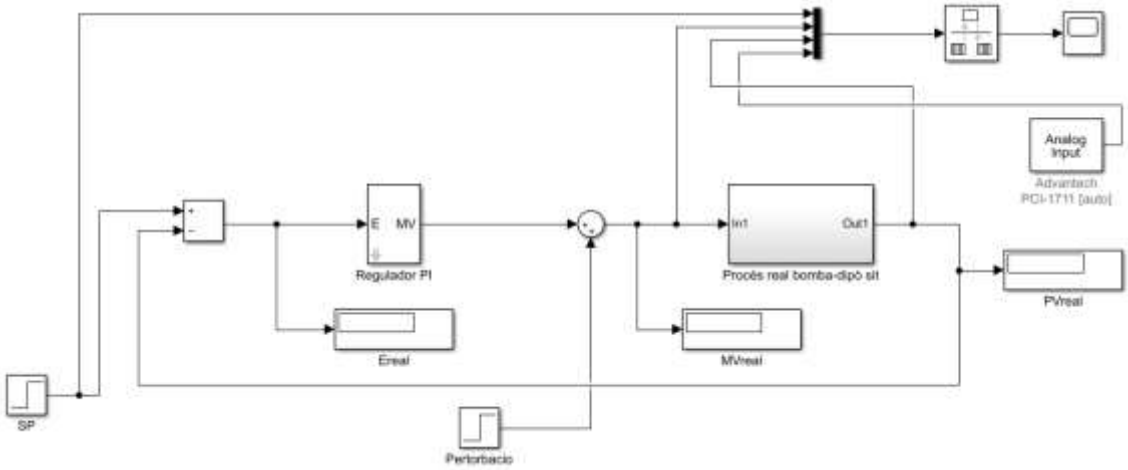

Figura 13. Diagrama de bloques en SIMULINK para la toma de datos.

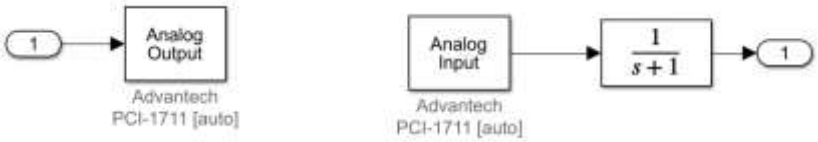

Figura 14. Conexiones con el sistema físico

Un aspecto muy importante que hay que destacar en la atapa de adquisición es que los datos que se obtienen del proceso deben guardarse en un archivo .xls de excel para posteriormente convertirlo a un archivo .txt que es el que se utilizara para la programación en CLIPS.

3.6 Introducción a CLIPS

CLIPS es un entorno de programación diseñado para escribir aplicaciones llamadas sistemas expertos. CLIPS es el acrónimo de C Lenguaje Integrated Production System.

Un programa escrito en CLIPS consta de un conjunto de **reglas** y una base de **hechos** aunque puede tener también una serie de **objetos**. De forma que un programa o sistema experto basado en reglas escrito en CLIPS es un programa dirigido por los datos (hechos y objetos), es decir, utiliza encadenamiento hacia adelante donde las reglas son activadas por los datos y se ejecutan de izquierda a derecha (primero antecedente y luego consecuente).

Características:

- CLIPS proporciona un entorno para el desarrollo de sistemas expertos.
- Incluye tanto un lenguaje de programación, como una herramienta para el desarrollo de sistemas expertos.

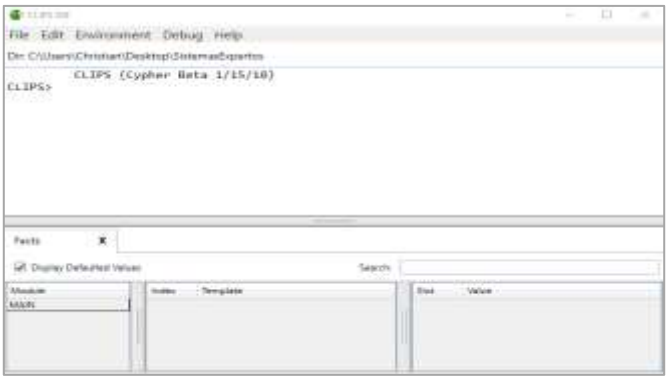

Figura 15. Entorno de programación en CLIPS

Creación y ejecución de un programa en CLIPS

Para realizar un programa en CLIPS es necesario seguir la nomenclatura que contiene este lenguaje de programación. Por lo tanto, para el desarrollo de un sistema experto se toma en consideración los siguientes enunciados:

- **Estructura de una plantilla**
  Consisten en una serie de campos (slots) que almacenarán datos relativos a un hecho determinado. El modelo del deftemplate es el siguiente:
  (deftemplate <*nombre*> (slot <*nombre del campo*>))
- **Definición de reglas**
  Las reglas (rule) describen el comportamiento del programa en función de la información existente. El modelo de regla es la siguiente:
  (defrule <nombre-regla> ((Condición) => (Acción))
- **Definición de hechos**
  La información que CLIPS utiliza para conocer el estado del sistema se maneja mediante hechos. La instrucción (facts) muestra todos los hechos que se han introducido con su número de identificador.
  El formato de la dirección de un hecho es:
  <Fact-x> donde x es el identificador del hecho.
  La acción de añadir hechos es:
  (assert (<atributo> <valor>))
- **Variables**
  En CLIPS, una variable es una zona de memoria que se utiliza para almacenar valores. Su sintaxis es la siguiente:
  ?<nombre> para variables locales
  ?*<nombre>* para variables globales

- **Instrucciones más comunes.**

  Run: permite la ejecución del programa

  Reset: permite limpiar la memoria de trabajo.

  Test: esta instrucción permite añadir condicionantes en la programación.

  Bind: se utiliza para asignar un valor a una variable.

  Refresh: activa y ejecuta la instrucción especificada.

3.7 Descripción del código

Para poder determinar mejor el estado en que se encuentra el proceso y el controlador se ha elaborado la siguiente tabla:

Variables					Estado		
co	**M**	**ca**	**sn**	**sn-1**	**sn>co**	**sn<co**	**sn=co**
0	0	0	0	0	parado		
0	0	0	0	1	vaciándose		
0	0	0	1	0	fallo controlador		
0	0	0	1	1	válvula 1 cerrada		
0	0	1	0	0	fallo controlador		
0	0	1	0	1			
0	0	1	1	0			
0	0	1	1	1			
0	1	0	0	0			
0	1	0	0	1			
0	1	0	1	0			
0	1	0	1	1			
0	1	1	0	0			
0	1	1	0	1			
0	1	1	1	0			
0	1	1	1	1			
1	0	0	0	0	válvula 1 cerrada	fallo controlador	estable
1	0	0	0	1	vaciándose		estable
1	0	0	1	0	fallo sistema		
1	0	0	1	1	válvula 1 cerrada	fallo controlador	estable
1	0	1	0	0	fallo sistema		
1	0	1	0	1			
1	0	1	1	0			
1	0	1	1	1			
1	1	0	0	0	fallo controlador	3 abierta//2 cerrada//falta de agua	estable
1	1	0	0	1			
1	1	0	1	0		llenándose	
1	1	0	1	1		3 abierta//2 cerrada//falta de agua	
1	1	1	0	0		1 abierta	
1	1	1	0	1			
1	1	1	1	0		llenándose	
1	1	1	1	1		1 abierta	

Tabla 3: Estados equivalentes para cada situación.

En esta tabla se presenta todos los distintos estados en los que se puede encontrar el proceso según los datos adquiridos por los distintos sensores y actuadores que lo conforman.

Para simplificarlo se ha considerado que cada variable con dos posibles estados distintos, 0(o sin lectura) i 1 (o un valor cualquiera), de tal forma que con las cinco variables: co(consigna), M(motor), ca(caudal), sn(sensor de nivel actual) y sn-

1(sensor de nivel anterior), tenemos un total de 32 combinaciones, pero además hay una regla que nos define 3 estados distintos para una misma combinación: cuando la lectura del sensor de nivel disminuye, aumenta o se mantiene respeto a la lectura anterior. Esto conlleva que el total de combinaciones posibles sea de 96.

A partir de esta tabla se ha programado la supervisión del controlador para saber si en cada situación dada el controlador funciona correctamente o no.

Primeramente se ha creado el fichero “Paso 1.txt” que contiene:

(load "D:\\drive\\MUESAEI\\Trabajo Inteligencia Artificial\\Reglas.clp")

(open "D:\\drive\\MUESAEI\\Trabajo Inteligencia Artificial\\Laboratorio\\tablareducida.txt" data)

(batch "D:\\drive\\MUESAEI\\Trabajo Inteligencia Artificial\\reiteracion.bat")

Este fichero lo cargamos en el programa CLIPS para ejecutar la inicialización. Primeramente cargamos el fichero Reglas.clp con el comando “load” en este fichero se han definido previamente todas las reglas y estados para cada situación y que comentaremos más adelante. Seguidamente abrimos el fichero tablareducida.txt con la comanda “open” este fichero es de tipo data y contiene todos los datos registrados durante un periodo de tiempo determinado. Finalmente hacemos la primera iteración mediante el comando “batch” en este fichero se encuentran todas los comandos para cargar cada variable y evaluar cada estado.

-Reglas.clp:

Definimos las distintas plantillas:

```
(deftemplate estado
        (slot tipo))
(deftemplate co         ;consigna
        (slot valor))
(deftemplate M          ;motor
        (slot lectura))
(deftemplate ca         ;caudal
        (slot lectura))
(deftemplate sn         ;sensor de nivel actual
        (slot valor))
(deftemplate sn-1               ;sensor de nivel anterior
```

```
(slot valor))
```

Definimos las distintas reglas:

```
(defrule parado
    (sn(valor ?vsn))
    (sn-1(valor ?vsn-1))
    (M(lectura ?vM))
    (co(valor ?vco))
    (ca(lectura ?vca))
    (test (= ?vco 0))
    (test (<= ?vca 0))
    (test (< ?vsn 0.6))
    (test (<= ?vM 0))
    (test (= ?vsn ?vsn-1))
    =>
    (assert(estado(tipo parado)))
    (printout t "Controlador en modo reposo" crlf)
    (printout t "Esperando consigna" crlf))
```

En estado “Parado” el controlador no está actuando ya que aún lo se le ha dado una consigna superior a cero.

```
(defrule sensor-superior-consigna
    (sn(valor ?vsn))
    (sn-1(valor ?vsn-1))
    (M(lectura ?vM))
    (co(valor ?vco))
    (ca(lectura ?vca))
    (test (<= ?vca 0))
    (test (<= ?vM 0))
    (test (< ?vsn ?vsn-1))
    (test (> ?vsn (* ?vco 1.25)))
    =>
    (assert(estado(tipo vaciandose)))
    (printout t "EL Controlador esta trabajando correctamente" crlf)
    (printout t "El proceso esta en transición" crlf)
    (printout t "El nivel del agua esta bajando" crlf))
```

En estado “vaciándose” el controlador no está actuando activamente en la señal del motor ya que el nivel es superior a la consigna y el segundo depósito se está vaciando.

Siempre que hacemos alguna comparación del sensor de nivel con la consigna, aplicamos un factor de seguridad en la consigna de ±25% de la consigna, esto es para dar un margen de activación del estado “estable” que se comenta posteriormente. De este modo y como el controlador no funciona a la perfección (el sensor de nivel no llega directamente a la consigna sino que va oscilando debido a les imperfecciones físicas del proceso).

```
(defrule sensor-igual-consigna
        (sn(valor ?vsn))
        (sn-1(valor ?vsn-1))
        (M(lectura ?vM))
        (co(valor ?vco))
        (ca(lectura ?vca))
        (test (> ?vco 0))
        (test (>= ?vca 0))
        (test (<= ?vM 0))
        (test (< ?vsn ?vsn-1))
        (test (and(>= ?vsn (* ?vco 0.75)) (<= ?vsn (* ?vco 1.25))))
        =>
        (assert(estado(tipo estable)))
        (printout t "EL Controlador esta trabajando correctamente" crlf)
        (printout t "El proceso esta estable" crlf))
```

El estado “estable” se produce cuando sensor de nivel llega a la consigna y se mantiene en esa posición a lo largo del tiempo (oscilando entre ±25% de la consigna).

```
(defrule sensor-inferior-consigna
        (sn(valor ?vsn))
        (sn-1(valor ?vsn-1))
        (M(lectura ?vM))
        (co(valor ?vco))
        (ca(lectura ?vca))
        (test (<= ?vca 0))
        (test (<= ?vM 0))
        (test (< ?vsn ?vsn-1))
        (test (< ?vsn (* ?vco 0.75)))
```

```
=>
(assert(estado(tipo error)))
(printout t "Fallo del controlador" crlf))
```

El “fallo de controlador se produce” cuando el sistema experto detecta que el controlador no está actuando como debería, por ejemplo en este caso no se detecta actividad del motor aunque el nivel está por debajo de la consigna.

No se especifica porqué se ha producido el fallo en concreto ya que por cada caso puede fallar por motivos distintos. De esta forma se pueden agrupar los diferentes fallos de controlador con unas pocas reglas, para así no definir una regla distinta para cada caso particular del proceso (96).

```
(defrule fallo-controlador
    (sn(valor ?vsn))
    (sn-1(valor ?vsn-1))
    (M(lectura ?vM))
    (co(valor ?vco))
    (ca(lectura ?vca))
    (test (<= ?vca 0))
    (test (<= ?vM 0))
    (test (> ?vsn ?vsn-1))
    (test (= ?vco 0))
    =>
    (assert(estado(tipo error)))
    (printout t "Fallo controlador" crlf))
```

En este caso el fallo es debido porque se detecta una subida de nivel sin aplicarle ninguna consigna ni detectar actividad del motor.

Cabe decir que en este caso se podría cumplir que lo que esté fallando no se el controlador sino el sistema (el sensor de nivel). Pero se ha considerado que es más probable que falle el controlador y no el sistema ya que por esto se hace la supervisión del mismo.

```
(defrule sensor-superior-consigna1
    (sn(valor ?vsn))
    (sn-1(valor ?vsn-1))
    (M(lectura ?vM))
    (co(valor ?vco))
    (ca(lectura ?vca))
    (test (<= ?vca 0))
    (test (<= ?vM 0))
```

```
(test (= ?vsn ?vsn-1))
(test (> ?vsn (* ?vco 1.25)))
=>
(assert(estado(tipo alarma)))
(printout t "Controlador en modo reposo" crlf)
(printout t "Alarma: valvula 1 cerrada" crlf)
(printout t "Abrir valvula 1" crlf))
```

El estado de “alarma” se produce cuando el proceso está fallando (no el controlador) eso ocurre cuando hay alguna de las válvulas en un estado que no debería por lo que el proceso deja de actuar correctamente. Por ejemplo el sensor está por encima de la consigna pero el nivel de agua no está bajando, por lo que se asume que la válvula 1 está cerrada.

```
(defrule fallo-controlador1
    (M(lectura ?vM))
    (sn(valor ?vsn))
    (sn-1(valor ?vsn-1))
    (ca(lectura ?vca))
    (co(valor ?vco))
    (test (= ?vco 0))
    (test (or (and (<= ?vM 0) (> ?vca 0)) (> ?vM 0)))
    =>
    (assert(estado(tipo error)))
    (printout t "Fallo del controlador" crlf))
```

Como se ha comentado anteriormente en la definición de esta regla se han agrupado distintas situaciones de “fallo de controlador” mediante el operador “or.”

```
(defrule sensor-inferior-consigna1
    (sn(valor ?vsn))
    (sn-1(valor ?vsn-1))
    (M(lectura ?vM))
    (co(valor ?vco))
    (ca(lectura ?vca))
    (test (> ?vco 0))
    (test (<= ?vca 0))
    (test (<= ?vM 0))
    (test (= ?vsn ?vsn-1))
    (test (< ?vsn (* ?vco 0.75)))
```

```
        =>
        (assert(estado(tipo error)))
        (printout t "Fallo controlador" crlf))

(defrule sensor-igual-consigna1
        (sn(valor ?vsn))
        (sn-1(valor ?vsn-1))
        (M(lectura ?vM))
        (co(valor ?vco))
        (ca(lectura ?vca))
        (test (> ?vco 0))
        (test (<= ?vca 0))
        (test (<= ?vM 0))
        (test (= ?vsn ?vsn-1))
        (test (and(>= ?vsn (* ?vco 0.75)) (<= ?vsn (* ?vco 1.25))))
        =>
        (assert(estado(tipo estable)))
        (printout t "EL Controlador está trabajando correctamente" crlf)
        (printout t "El proceso está estable" crlf))

(defrule fallo-sistema
        (sn(valor ?vsn))
        (sn-1(valor ?vsn-1))
        (M(lectura ?vM))
        (co(valor ?vco))
        (ca(lectura ?vca))
        (test (> ?vco 0))
        (test (<= ?vca 0))
        (test (<= ?vM 0))
        (test (> ?vsn ?vsn-1))
        =>
        (assert(estado(tipo error)))
        (printout t "Fallo del sistema" crlf)
        (printout t "Compruebe el estado de los sensores y actuadores" crlf))
```

El estado “fallo sistema” se produce cuando hay alguna variable de no coincide con el estado del proceso ya sea debido al fallo de un sensor o del actuador. Por ejemplo en esta regla definida se produce el fallo del sistema cuando se le asigna una consigna al proceso y el nivel de agua está subiendo (por lo que nos encontraríamos en el estado de llenándose), pero aun así en motor se mantiene a cero por lo que en teoría no está activado y no puede estar enviado agua al depósito superior, por lo que se está produciendo algún fallo de lectura de variables. En este caso podría ser debido por el motor o por el sensor de nivel.

```
(defrule fallo-sistema1
        (sn(valor ?vsn))
        (sn-1(valor ?vsn-1))
        (M(lectura ?vM))
        (co(valor ?vco))
        (ca(lectura ?vca))
        (test (> ?vco 0))
        (test (> ?vca 0))
        (test (<= ?vM 0))
        =>
        (assert(estado(tipo error)))
        (printout t "Fallo del sistema" crlf)
        (printout t "Compruebe el estado de los sensores y actuadores" crlf))

(defrule fallo-controlador2
        (sn(valor ?vsn))
        (sn-1(valor ?vsn-1))
        (M(lectura ?vM))
        (co(valor ?vco))
        (ca(lectura ?vca))
        (test (> ?vco 0))
        (test (> ?vM 0))
        (test (> ?vsn (* ?vco 1.25)))
        =>
        (assert(estado(tipo error)))
        (printout t "Fallo del controlador" crlf))

(defrule sensor-igual-consigna2
        (sn(valor ?vsn))
```

```
    (sn-1(valor ?vsn-1))
    (M(lectura ?vM))
    (co(valor ?vco))
    (ca(lectura ?vca))
    (test (> ?vco 0))
    (test (> ?vM 0))
    (test (and(>= ?vsn (* ?vco 0.75)) (<= ?vsn (* ?vco 1.25))))
    =>
    (assert(estado(tipo estable)))
    (printout t "EL Controlador esta trabajando correctamente" crlf)
    (printout t "El proceso esta estable" crlf))

(defrule sensor-inferior-consigna2
    (sn(valor ?vsn))
    (sn-1(valor ?vsn-1))
    (M(lectura ?vM))
    (co(valor ?vco))
    (ca(lectura ?vca))
    (test (> ?vco 0))
    (test (> ?vM 0))
    (test (<= ?vca 0))
    (test (< ?vsn (* ?vco 0.75)))
    (test (or (= ?vsn ?vsn-1) (< ?vsn ?vsn-1)))
    =>
    (assert(estado(tipo alarma)))
    (printout t "EL Controlador esta trabajando correctamente" crlf)
    (printout t "Alarma: valvula 3 abierta i/o valvula 2 cerrada" crlf)
    (printout t "Cerrar la  valvula 3, abrir la valvula 2" crlf)
    (printout t "La consigna podría ser demasiado alta, el depósito 1 esta vacío" crlf))
```

Hay un estado particular del proceso que se cumple con esta regla, se produce cuando se le asigna una consigna muy elevada al controlador (ya que no está limitada a ningún rango) entonces este va llenando el depósito superior hasta que el deposito inferior se queda sin agua, en ese momento el motor empieza a aspirar aire y el nivel de agua deja de subir, por lo que se produce este estado de alarma.

```
(defrule sensor-inferior-consigna3
    (sn(valor ?vsn))
```

```
(sn-1(valor ?vsn-1))
(M(lectura ?vM))
(co(valor ?vco))
(ca(lectura ?vca))
(test (> ?vco 0))
(test (> ?vM 0))
(test (> ?vca 0))
(test (< ?vsn (* ?vco 0.75)))
(test (or (= ?vsn ?vsn-1) (< ?vsn ?vsn-1)))
=>
(assert(estado(tipo alarma)))
(printout t "EL Controlador esta trabajando correctamente" crlf)
(printout t "Alarma: valvula 1 abierta" crlf)
(printout t "Cerrar valvula 1" crlf))
```

Este estado de alarma se cumple por consignas elevadas. Para consignas bajas si la válvula 1 se encuentra abierta el nivel de agua va subiendo (más despacio que si estuviera cerrada) por eso se considera que el sistema está funcionando correctamente ya que no se puede detectar en qué estado se encuentra la válvula 1. Solo se detecta en el momento en que en nivel de agua es suficientemente alto como para que salga la misma cantidad de agua de la que entra debido a la presión que hace el agua.

```
(defrule sensor-inferior-consigna4
    (sn(valor ?vsn))
    (sn-1(valor ?vsn-1))
    (M(lectura ?vM))
    (co(valor ?vco))
    (ca(lectura ?vca))
    (test (> ?vco 0))
    (test (> ?vM 0))
    (test (< ?vsn (* ?vco 0.75)))
    (test (> ?vsn ?vsn-1))
    =>
    (assert(estado(tipo llenándose)))
    (printout t "EL Controlador esta trabajando correctamente" crlf)
    (printout t "El proceso esta en transición" crlf)
    (printout t "El nivel del agua esta subiendo" crlf))
```

Finalmente el estado "llenándose" se produce cuando el motor está activo y el sensor de nivel va aumentando acercándose a la consigna que se ha establecido.

Para entender mejor como se desarrolla el programa en clips tenemos que entender como lee los datos. En el fichero de texto tablareducida.txt los datos se presentan de la siguiente forma:

2.00	10.00	0.00	0.00	0.00
2	10.00	9.36	0.33	0.00
2	10.00	9.28	0.42	0.33
2	10.00	9.33	0.50	0.42
2	10.00	9.28	0.61	0.50
2	10.00	9.29	0.71	0.61
2	10.00	8.73	0.82	0.71
2	10.00	9.28	0.93	0.82
2	10.00	9.23	1.04	0.93
2	10.00	9.26	1.15	1.04
2	10.00	9.15	1.26	1.15
2	10.00	9.14	1.37	1.26
2	10.00	9.12	1.48	1.37
2	10.00	9.08	1.59	1.48
2	9.88	9.11	1.73	1.59
2	9.90	9.03	1.80	1.73
2	9.62	9.01	1.91	1.80
2	9.13	7.29	2.02	1.91

Tabla 4: representación de una muestra de datos del fichero tablareducida.txt

Los datos siempre se presentan con el mismo orden para saber en todo momento a cual equivale cada uno. Para este caso el orden es el siguiente:

Consigna (co), voltaje del motor (M) sensor de caudal (ca), sensor de nivel (sn) sensor de nivel anterior (sn-1). En el caso del sensor de nivel anterior, la primera lectura se asume que es cero ya que no se dispone de un registro anterior del sensor de nivel, en algún caso esto puede llevar a asumir que se el proceso se encuentra en un falso estado por este motivo la primera lectura siempre se menospreciará para no dar falsos errores.

Además cabe destacar que al momento de activar el proceso es cuando se resetean las lecturas de los sensores simultáneamente con la adquisición de datos como se puede comprobar en la figura anterior, hasta la segunda iteración el sensor de nivel no se estabiliza.

Finalmente hay que comentar que por limitaciones físicas del proceso el depósito superior nunca se va a vaciar por completo sino que su lectura mínima oscila entre el 0.5 ±20%, es por este motivo que cuando el sensor baja de 0.6 el proceso ya se considera en estado reposo. De un modo similar actúa el sensor de caudal ya que su funcionamiento óptimo con la válvula 3 cerrada es cuando el voltaje del motor (M) es superior a 6.5, para los otros caso el caudal es tan bajo que no se puede realizar una lectura precisa, por lo que el sensor dará un valor de cero, de igual modo ocurre con la válvula 3 abierta y aunque el motor trabaje a máxima potencia el caudal que pasa por el sensor de caudal no es suficiente. Debido a

este hecho se ha tenido que considerar que aunque el sensor de caudal sea cero tanto el controlador como el sistema pueden estar funcionando correctamente.

En el documento reiteración.bat encontramos el siguiente código:

(reset)

(bind ?vco (read data))

(bind ?vM (read data))

(bind ?vca (read data))

(bind ?vsn (read data))

(bind ?vsn-1 (read data))

Aquí leemos una fila del fichero tablareducida.txt y asignamos el valor a una variable.

Al abrir un fichero .txt por defecto CLIPS siempre lee la primera fila

(assert(co (valor ?vco)))

(assert(M (lectura ?vM)))

(assert(ca (lectura ?vca)))

(assert(sn (valor ?vsn)))

(assert(sn-1 (valor ?vsn-1)))

Asignamos a cada variable su nombre correspondiente.

(refresh parado)

(refresh sensor-superior-consigna)

(refresh sensor-igual-consigna)

(refresh sensor-inferior-consigna)

(refresh fallo-controlador)

(refresh sensor-superior-consigna1)

(refresh fallo-controlador1)

(refresh sensor-inferior-consigna1)

(refresh sensor-igual-consigna1)

(refresh fallo-sistema)

(refresh fallo-sistema1)

(refresh fallo-controlador2)

(refresh sensor-igual-consigna2)

(refresh sensor-inferior-consigna2)

(refresh sensor-inferior-consigna3)

(refresh sensor-inferior-consigna4)

Actualizamos todos los estados del archivo reglas.clp

(run)

Ejecutamos el código para que imprima los resultados correspondientes.

(batch "D:\\drive\\MUESAEI\\Trabajo Inteligencia Artificial\\reiteracion.bat")

Finalmente creamos un bucle infinito para que vuelva a ejecutar este mismo código hasta leer todo el fichero simulando de este modo un proceso continuo.

4. Resultados

Para realizar una correcta representación de los resultados obtenidos del programa se han graficado las distintas variables:

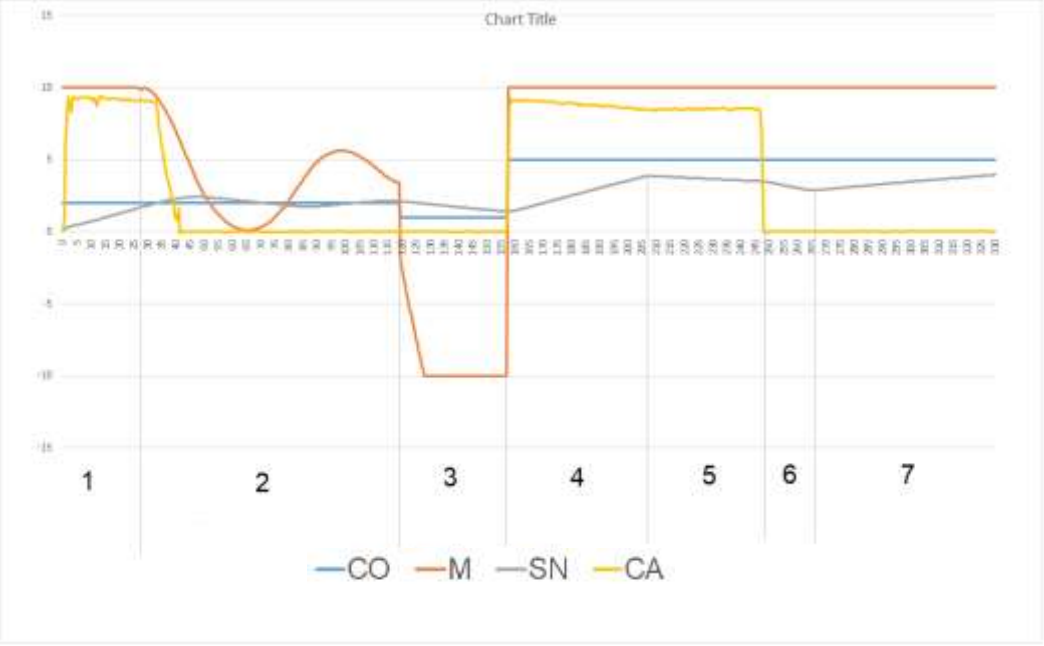

Figura 16. representación de las variables en distintos estados.

Esta gráfica representa la adquisición de las distintas variables de uno de los procesos reales del laboratorio durante un periodo de tiempo, en el cual se han pasado por distintos estados:

-Estado 1: llenándose.

-Estado 2: estable.

-Estado 3: vaciándose.

-Estado 4: llenándose.

-Estado 5: válvula 1 abierta.

-Estado 6: válvula 1 abierta y válvula 3 abierta.

-Estado 6: válvula 3 abierta.

El punto de partida es con la válvula 1 y 3 cerrada y la 2 abierta.

Al ejecutar el código en clips como se ha descrito anteriormente se visualiza lo siguiente:

CLIPS (Cypher Beta 1/15/18)

CLIPS> (batch "Paso 1.bat")

TRUE

CLIPS> (load "D:\\drive\\MUESAEI\\Trabajo Inteligencia Artificial\\Reglas.clp")

*%%%%%%**********

TRUE

CLIPS>

(open "D:\\drive\\MUESAEI\\Trabajo Inteligencia Artificial\\Laboratorio\\tablareducida.txt" data)

TRUE

CLIPS>

(batch "D:\\drive\\MUESAEI\\Trabajo Inteligencia Artificial\\reiteracion.bat")

TRUE

Se cargan los tres archivos uno por uno que ya se han detallado anteriormente, si se carga correctamente se muestra un TRUE en pantalla.

Como ya se ejecuta la iteración nos aparecen todos los resultados.

En cada iteración observamos:

CLIPS> (reset)

CLIPS> lectura de datos:

(bind ?vco (read data))

2

CLIPS> (bind ?vM (read data))

10.0

CLIPS> (bind ?vca (read data))

9.36

CLIPS> (bind ?vsn (read data))

0.33

CLIPS> (bind ?vsn-1 (read data))

0.0

CLIPS> asignación de las variables:

(assert(co (valor ?vco)))

<Fact-1>

CLIPS> (assert(M (lectura ?vM)))

<Fact-2>

CLIPS> (assert(ca (lectura ?vca)))

<Fact-3>

CLIPS> (assert(sn (valor ?vsn)))

<Fact-4>

CLIPS> (assert(sn-1 (valor ?vsn-1)))

<Fact-5>

CLIPS> actualización de las reglas:

(refresh parado)

CLIPS> (refresh sensor-superior-consigna)

CLIPS> (refresh sensor-igual-consigna)

CLIPS> (refresh sensor-inferior-consigna)

CLIPS> (refresh fallo-controlador)

CLIPS> (refresh sensor-superior-consigna1)

CLIPS> (refresh fallo-controlador1)

CLIPS> (refresh sensor-inferior-consigna1)

CLIPS> (refresh sensor-igual-consigna1)

CLIPS> (refresh fallo-sistema)

CLIPS> (refresh fallo-sistema1)

CLIPS> (refresh fallo-controlador2)

CLIPS> (refresh sensor-igual-consigna2)

CLIPS> (refresh sensor-inferior-consigna2)

CLIPS> (refresh sensor-inferior-consigna3)

CLIPS> (refresh sensor-inferior-consigna4)

CLIPS>

(run)

Finalmente se imprimen la información establecida para cada estado (se adjuntan los valores de las variables para confirmar el estado):

-Estado 1: llenándose.

2 10 9.36 0.33 0.0

EL Controlador está trabajando correctamente

El proceso está en transición

El nivel del agua está subiendo

-Estado 2: estable.

2 9.13 7.29 2.02 1.91

EL Controlador está trabajando correctamente

El proceso está estable

-Estado 3: vaciándose.

1 -10 -0.01 1.87 1.9

EL Controlador está trabajando correctamente

El proceso está en transición

El nivel del agua está bajando

-Estado 4: llenándose.

5 10.0 9.02 2.09 1.99

EL Controlador está trabajando correctamente

El proceso está en transición

El nivel del agua está subiendo

-Estado 5: válvula 1 abierta.

5 10.0 8.56 3.67 3.69

EL Controlador está trabajando correctamente

Alarma: válvula 1 abierta

Cerrar válvula 1

-Estado 6: válvula 1 abierta y válvula 3 abierta.

5 10.0 0.0 3.42 3.49

EL Controlador está trabajando correctamente

Alarma: válvula 3 abierta i/o válvula 2 cerrada

Cerrar la válvula 3, abrir la válvula 2

La consigna podría ser demasiado alta, el depósito 1 esta vacío

-Estado 7: válvula 3 abierta.

5 10.0 0.0 3.4 3.36

EL Controlador está trabajando correctamente

El proceso está en transición

El nivel del agua está subiendo

5. Conclusiones

Parte de la dificultad de este trabajo ha sido la programación en un lenguaje totalmente distinto a lo que se está acostumbrado como podrían ser C, Visual Basic, Python,… También hay que hace hincapié en el riguroso estudio del proceso y de sus distintas situaciones para cada estado tal y como se ha visto en la tabla 3 a partir de la cual se han desarrollado las distintas reglas para definir el sistema experto.

Referente a la comunicación con el sistema experto no se ha podido establecer un comunicación directa de Matlab con CLIPS debido a la incompatibilidad de este último. Aun así cabe la posibilidad que de algún modo igual que se ha utilizado un fichero de texto para pasar los datos de Matlab a CLIPS se pueda realizar una solución parecida de forma automática para conseguir una lectura a tiempo real.

Referente al funcionamiento del sistema experto podrían haber algunos aspectos de mejora. Por ejemplo en el caso anterior para testear su funcionamiento han ocurrido algunas incidencias, en el caso del estado 7 el sistema no es capaz de detectar que la válvula 3 está abierta ya que el sistema funciona correctamente, de igual modo se puede dar con las válvulas 1 i 2 o cuando se abren o cierran simultáneamente. Aun así cabe remarcar que la finalidad principal del sistema experto es la de supervisar el funcionamiento del controlador y no del proceso aunque se pueda conocer su estado a partir de este.

Finalmente cabe destacar que se ha cumplido el objetivo principal de este trabajo que consistía en desarrollar un sistema de supervisión para un controlador, y es que se podido obtener un sistema de supervisión básico de un proceso real y testear su correcto funcionamiento, permitiendo así a cualquier usuario ser capaz de entender que está ocurriendo o como está funcionando el proceso y aplicar las acciones indicadas por el sistema sin tener total conocimiento sobre el proceso controlado.

6. Bibliografía

1. Masip Álvarez, A. Manual del Laboratori de Control Industrial.
2. Colomer i Ribas, J. Sistemas de supervisión: Introducción a la monitorización y supervisión experta de procesos: métodos y herramientas. A: Colomer i Ribas, J. Introducción a la supervisión. Barcelona: Cetisa Boixareu, 2000. ISBN: 8493132713.
3. Giarratano, J; Riley, G. Sistemas expertos. Principios de programación. A: Giarratano, J; Riley, G. Introducción a los sistemas expertos. 3ª edición. México: International Thomson, cop.2001. ISBN: 9706860592.
4. Rich, Elaine. Inteligencia Artificial. Madrid: McGraw-Hill, DL 1994. ISBN:8448118588.
5. *CLIPS.* Volume I – The Basic Programming Guide.
6. Antonio Calvo Cuenca, Programación en lenguaje CLIPS, 2da edición, septiembre 2008