UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Ingeniería de Sistemas y Automática

Diseño e implementación de controladores avanzados en PLCs industriales

Trabajo Fin de Máster

Máster Universitario en Automática e Informática Industrial

AUTOR/A: Cazalla Moral, Juan Alberto

Tutor/a: Simarro Fernández, Raúl

Cotutor/a: Sanchis Saez, Javier

CURSO ACADÉMICO: 2022/2023

DEPARTAMENTO DE INGENIERÍA
DE SISTEMAS Y AUTOMÁTICA

UNIVERSIDAD POLITÉCNICA DE VALENCIA

Dpto. de Ingeniería de Sistemas y Automática

Diseño e implementación de controladores avanzados en PLCs Industriales

Trabajo de fin de Máster

Máster Universitario en Automática e Informática Industrial

AUTOR/A: Cazalla Moral, Juan Alberto
TUTOR/A: Raúl Simarro Fernández
COTUTOR/A: Javier Sanchis Sáez
CURSO ACADÉMICO:2022/2023

RESUMEN

El presente trabajo fin de máster tiene como objetivo la definición de una metodología para la implementación de algoritmos de control avanzados en PLC's industriales que no se encuentran en las librerías estándar propias de los autómatas. Se hace uso de herramientas como Simulink PLC Coder, que permite la generación de código en texto estructurado a partir de modelos de simulación, y la generación del código necesario para la creación de las funciones con el control diseñado que se ejecutarán en autómatas.

La prueba de esta metodología se realizará sobre el autómata PLCnext Control AXC F 2152. Como plataforma de pruebas se utilizará un proceso térmico y un motor de corriente continua.

ABSTRACT

This project pursues the definition of a methodology for the implementation of advanced control algorithms in PLC' that are not defined in the standard libraries of automata. Tools such as Simulink PLC Coder are used, which allows the generation of structured text code from simulation models, and the generation of the necessary code for the creation of the functions with the designed control that will be executed in automata.

The test of this methodology will be carried out on the PLCnext Control AXC F 2152 automaton. A thermal process and a dc motor will be used as a test platform.

RESUM

El present projecte té com a objectiu la definició d'una metodologia per a la implementació d'algorismes de control avançats en *PLC's industrials que no es troben en les llibreries estàndard pròpies dels autòmats. Es fa ús d'eines com *Simulink *PLC *Coder, que permet la generació de codi en text estructurat a partir de models de simulació, i la generació del codi necessari per a la creació de les funcions amb el control dissenyat que s'executaran en autòmats.

La prova d'aquesta metodologia es realitzarà sobre l'autòmat *PLCnext Control *AXC F 2152. Com a plataforma de proves s'utilitzarà un procés tèrmic i un motor de corrent continu.

ÍNDICE GENERAL

1. INTRODUCCIÓN

Este capítulo recoge las motivaciones que llevaron al planteamiento y realización del trabajo fin de máster, así como los objetivos definidos para su cumplimiento.

1.1. Motivación

El Control Predictivo basado en Modelo (MPC) emergió como una estrategia de controla altamente efectiva y está experimentando un crecimiento significativo. Este algoritmo destaca por su capacidad para predecir y optimizar el comportamiento futuro de un sistema de control, lo que permite un control preciso y eficiente.

En paralelo, el uso de redes neuronales en control está teniendo un gran desarrollo. Estas redes tienen la capacidad de aprender patrones complejos a partir de datos y adaptarse a las condiciones cambiantes del proceso. Su flexibilidad y capacidad para modelar sistemas no lineales las convierten en una opción prometedora para el control de sistemas industriales.

A pesar de las ventajas evidentes del este tipo de algoritmos de control, su implementación conlleva desafíos significativos. Uno de los problemas más notables es la complejidad inherente en la configuración y ajuste de estos controladores respecto a los clásicos algoritmos de control PID. Esta complejidad se ve aumentada por la falta de librerías específicas en los PLCs que simplifiquen la programación y configuración de estos controladores, en contraste con las facilidades encontradas a la hora de usar controladores PID. Esto hace que la implementación de estos controladores avanzados sea una tarea más compleja y requiera un mayor esfuerzo de ajuste, lo que puede resultar en un proceso de desarrollo más lento y propenso a errores, frenando su desarrollo e implementación en la industria.

1.2. Objetivos

El objetivo fundamental de este trabajo fin de máster es establecer una metodología que habilite la implementación directa de controladores predictivos y

redes neuronales en PLC's industriales, eliminando la necesidad de sistemas auxiliares para alojar dichos algoritmos. Para la integración de estos algoritmos en el entorno de programación del PLC, se harán uso de Simulink PLC Coder, que permite la generación de código en texto estructurado a partir de modelos de simulación. Esta metodología será probada sobre un proceso térmico y un proceso mecánico.

Para alcanzar este propósito se han establecido las siguientes metas a realizar:

- Identificación de sistemas. Se procederá a identificar los sistemas seleccionados como bancos de pruebas para los controladores a ser implementados. La obtención de un modelo de planta preciso es esencial para la formulación de los controladores.
- Desarrollo y simulación de controladores. Los algoritmos de control serán diseñados y sometidos a simulaciones utilizando el modelo de planta previamente identificado. Los controladores serán programados en MATLAB.
- Establecer metodología específica para cada controlador. Se establecerá una metodología de trabajo particular para cada tipo de controlador. Dado que las herramientas disponibles pueden presentar limitaciones, se considerará la adaptación de enfoques específicos para optimizar la implementación de cada tipo de control.
- Implementación y revisión de resultados. Los controladores serán implementados en PLC y sometidos a pruebas en tiempo real en condiciones operativas reales.

1.3. Estructura de la memoria

A continuación, se detallan los capítulos que componen la memoria del trabajo fin de máster:

- En el capítulo 1 se realiza una introducción en la cual se expone los motivos que llevaron al desarrollo del mismo, así como los objetivos marcados.

- En el capítulo 2 se realiza un análisis sobre el control predictivo y el uso de redes neuronales multicapa. Esta revisión literaria permitirá establecer el contexto necesario para el desarrollo del trabajo fin de máster.
- En el capítulo 3 se describen los sistemas utilizados como bancos de pruebas y el entorno de software usado para el diseño, simulación e implementación de los algoritmos de control.
- En el capítulo 4 desarrolla paso a paso la metodología seguida para el diseño e implementación de los controladores en el PLC.
- En el capítulo 5 se muestran los resultados de algoritmos de control aplicados a los bancos de pruebas.
- En el capítulo 6 se presenta las conclusiones obtenidas durante la realización del trabajo fin de máster, los problemas afrontados y se comenta algunas sugerencias de mejora para el mismo.
- En el capítulo 7 se muestra la bibliografía consultada durante el estudio de este trabajo fin de máster.

2. ANTECEDENTES

2.1. Control predictivo

El control predictivo, más conocido como MPC por sus siglas en inglés, (Model Predictive Control) se desarrolló a finales de los setenta y ha tenido un desarrollo considerable desde entonces. El termino control predictivo hace referencia a un conjunto de métodos de control que hacen uso explícito de un modelo del proceso para obtener la señal de control minimizando una función objetivo [1].

La metodología de los controladores pertenecientes al tipo MPC se caracteriza por seguir la siguiente estrategia, representada en la Figura 1:

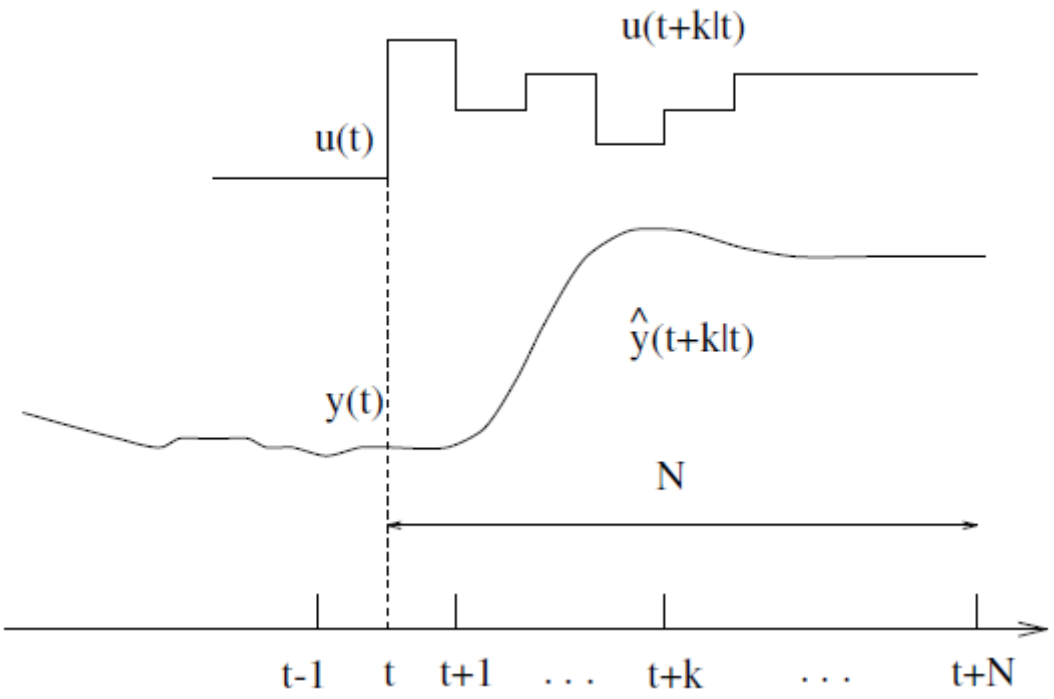

Figura 1. Estrategia MPC [1]

1. Las salidas futuras para un horizonte de predicción N se predicen cada instante utilizando el modelo del proceso. Las predicciones de la salida $y(t+k \mid t),\ k = 1 \ldots N$ dependen de los valores conocidos en el instante t y de las señales futuras de control $u(t+\ k \mid t), k = 0 \ldots N-1$, que han de ser calculadas. El modelo del proceso juega, en consecuencia, un papel decisivo en el controlador, ya que debe ser capaz de capturar la

dinámica del proceso para predecir de forma precisa la evolución del sistema.

2. Las señales de control futuras se calculan minimizando una función objetivo para mantener la salida del proceso lo más cercana posible de la trayectoria de referencia. Este criterio toma normalmente la forma de una función cuadrática del error entre la salida predicha y la trayectoria de referencia. La solución explícita se puede obtener cuando el criterio es cuadrático y el modelo lineal; en caso contrario, se ha de utilizar un método numérico para buscar la solución.
3. La acción de control $u(t \mid t)$ se envía al proceso mientras que el resto de las señales calculadas no se utilizan, ya que en el instante de muestreo siguiente $y(t+1)$ es ya conocida y los pasos anteriores se repiten con este nuevo valor. Esta estrategia se conoce como horizonte deslizante, y permite realizar un control en bucle cerrado que tiene en cuenta las posibles perturbaciones que puedan aparecer o, incluso, la no linealidad del modelo.

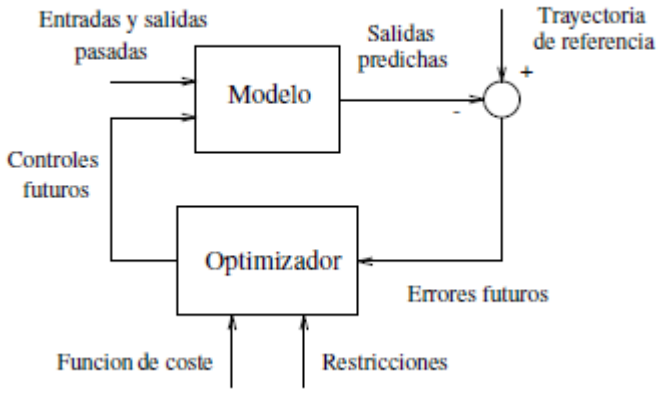

Figura 2. Estructura básica del MPC [1]

El control predictivo presenta un conjunto de ventajas sobre otros métodos, entre las más destacadas las siguientes:

- Se puede utilizar para controlar una gran variedad de procesos, desde procesos muy simples hasta procesos con dinámicas complejas.
- Su carácter predictivo lo hace compensar intrínsecamente los tiempos muertos.

- Introduce un control anticipativo (feed forward) compensando las perturbaciones medibles.
- La ley de control resultante es fácilmente implementable
- Es muy útil cuando se conocen las referencias futuras, como ocurre en el caso de robótica o procesos por lotes.
- Permite tratar las restricciones de una forma conceptualmente muy simple durante la fase de diseño.

Sin embargo, también presenta algunas desventajas. La principal es que su implementación, aunque no es compleja, resulta más difícil que la de los controladores PID. También cabe destacar que la carga computacional puede ser alta dependiendo de la dificultad de la dinámica del proceso.

2.1.1. Dynamic Matrix Control(DMC)

El control predictivo DMC es el primer algoritmo MPC desarrollado a principios de la década de 1980. También es probablemente el algoritmo de la familia de los controladores predictivos más usado, debido a que se basa en la respuesta de un modelo lineal ante una entrada escalón para predecir el comportamiento futuro del sistema [2].

La respuesta ante una entrada escalón de un sistema se puede describir mediante la siguiente ecuación:

$$y(t) = \sum_{i=1}^{\infty} g_i \Delta u(t-i) \qquad \textbf{(2.1)}$$

Donde $y(t)$ es la respuesta de la planta, g_i son los coeficientes de la respuesta ante una entrada escalón y Δu son los incrementos de la acción de control.

El modelo completo de predicción, siendo $n(t)$ las perturbaciones del sistema, viene dado por:

$$(t+k \mid t) = \sum_{i=1}^{k} g_i \Delta u\,(t\ +k-i) + \sum_{i=k+1}^{\infty} g_i \Delta u\,(t\ +k-i) + n(t+k) \qquad \textbf{(2.2)}$$

Considerando las perturbaciones constantes y siendo $y_m(t)$ la salida medida del sistema:

$$n(t+k) = n(t) = y_m(t) - y(t) = y_{m(t)} - \sum_{i=k+1}^{\infty} g_i \Delta u(t-i) \quad \textbf{(2.3)}$$

Así, la ecuación 2.2 puede escribirse como:

$$y(t+k \mid t) = \sum_{i=1}^{k} g_i \Delta u\,(t\ + k - i) + \sum_{i=k+1}^{\infty} g_i \Delta u\,(t\ + k - i) + y_m(t) - \sum_{i=k+1}^{\infty} g_i \Delta u(t-i) \quad \textbf{(2.4)}$$

Y se puede abreviar como:

$$y(t+k \mid t) = \sum_{i=1}^{k} g_i \Delta u(t+k-i) + f(t+k) \quad \textbf{(2.5)}$$

Siendo $f(t+k)$ la respuesta libre del sistema, que no depende de las acciones de control futuras. Teniendo en cuenta que, si el proceso es estable, los coeficientes de la respuesta escalón tenderán a ser constantes tras N períodos de muestreo, se puede definir la respuesta libre como:

$$f(t+k) = y_m(t) + \sum_{i=1}^{N} (g_{k+i} - g_i) \Delta u(t-i) \quad \textbf{(2.6)}$$

Aplicando las ecuaciones anteriores para un horizonte de predicción P y un horizonte de control C se obtiene:

$$y(t+P \mid t) = \sum_{i=P-C+1}^{P} g_i \Delta u\,(t\ + P - i) + f(t+P) \quad \textbf{(2.7)}$$

Definiendo la matriz del sistema como:

$$G = \begin{pmatrix} g_1 & 0 & 0 & 0 \\ g_2 & g_1 & 0 & 0 \\ g_C & g_{C-1} & \cdots & g_1 \\ \vdots & \vdots & \ddots & \vdots \\ g_P & g_{P-1} & \cdots & g_{P-C+1} \end{pmatrix} \quad \textbf{(2.8)}$$

Y usando formulación matricial, se puede escribir:

$$y = G\Delta u + f \quad \textbf{(2.9)}$$

Siendo y un vector de dimensión P que contiene las predicciones futuras del sistema en un horizonte de predicción P, Δu un vector de dimensión C que contiene los incrementos de la acción de control y f el vector de respuesta libre. Esta expresión relaciona las salidas futuras con los incrementos de control y se usa para calcular las acciones necesarias para obtener un comportamiento específico.

El control DMC busca encontrar un incremento de control que minimice una determinada función de coste [3]:

$$J = \sum_{i=1}^{P} \alpha_i (w(t+i) - (y(t+i \mid t))^2 + \sum_{j=1}^{C} \lambda_j \left(\Delta u(t+j-1 \mid k)\right)^2 \quad \textbf{(2.10)}$$

Siendo w el vector de referencias futuras. Si desarrollamos la ecuación usando formulación matricial se llegaría a la siguiente ecuación:

$$J = [W - Y]^T \alpha [W - Y] + \Delta u^T \lambda \Delta u \quad \textbf{(2.11)}$$

Si en esta expresión se define $E = W - F$, que representaría la discrepancia del vector de referencias futuras W respecto de toda la información conocida antes del instante t, resulta en la fórmula del índice de coste como:

$$J = [E - G\Delta u]^T \alpha [E - G\Delta u] + \Delta u^T \lambda \Delta u \quad \textbf{(2.12)}$$

Para resolver el problema, se agrupan los dos términos del índice J en uno solo, definiendo un residuo r de la siguiente forma:

$$r = \begin{bmatrix} \sqrt{\alpha} E \\ 0 \end{bmatrix} - \begin{bmatrix} \sqrt{\alpha} G \\ \sqrt{\lambda} \end{bmatrix} \Delta u = E' - G' \Delta u \quad \textbf{(2.13)}$$

Minimizando se obtiene:

$$J = r^T r = (E' - G'\Delta u)^T (E' - G'\Delta u) \quad \textbf{(2.14)}$$

$$\Delta u = (G'^T G')^{-1} G'^T E' = \left(\begin{bmatrix} \sqrt{\alpha} G \\ \sqrt{\lambda} \end{bmatrix}^T \begin{bmatrix} \sqrt{\alpha} G \\ \sqrt{\lambda} \end{bmatrix} \right)^{-1} \begin{bmatrix} \sqrt{\alpha} G \\ \sqrt{\lambda} \end{bmatrix} \begin{bmatrix} \sqrt{\alpha} E \\ 0 \end{bmatrix} = (G^T \alpha G + \lambda)^{-1} G^T \alpha E \quad \textbf{(2.15)}$$

Siendo $(G^T \alpha G + \lambda)^{-1} G^T \alpha$ la parte conocida e invariante en el cálculo (siempre será lo mismo independientemente del tiempo)

2.2. Redes neuronales

No existe una definición general de red neuronal artificial, existiendo multitud de ellas dependiendo del texto o artículo consultado. Pero en todas ellas aparece el componente de simulación del comportamiento biológico. Las operaciones a realizar se distribuyen en una serie de elementos básicos que, por analogía biológica, se conocen como neuronas. Estos elementos están interconectados entre sí mediante una serie de conexiones que se conocen como pesos sinápticos. Estos pesos varían con el tiempo mediante un proceso que se conoce como aprendizaje. Así pues, podemos definir el aprendizaje de una red como el proceso por el cual modifica las conexiones entre neuronas, pesos sinápticos, para realizar la tarea deseada [4].

En todo modelo artificial de neurona se tienen cuatro elementos básicos:

1. Un conjunto de conexiones, pesos sinápticos, que determinan el comportamiento de la neurona. Estas conexiones pueden ser excitadoras (signo positivo) o inhibidoras (signo negativo).
2. Un sumador que se encarga de sumar todas las entradas multiplicadas por los respectivos pesos sinápticos.
3. Una función de activación que determina la salida de la neurona en función de sus entradas ponderadas.
4. Un umbral exterior que determina el umbral por encima del cual la neurona se activa.

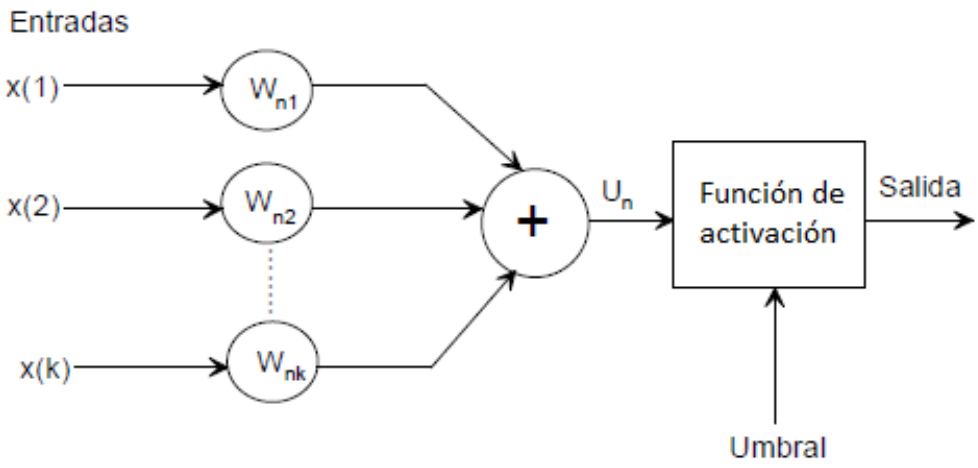

Figura 3. Esquema de un modelo neuronal [4]

Los elementos comentados anteriormente se pueden conectar entre sí para dar lugar a estructuras neuronales que pueden clasificarse de diferentes formas según el criterio usado. Así se tendría:

- Según el número de capas:
  - Redes neuronales monocapas: Se corresponde a la estructura más sencilla, ya que tiene una capa de entrada, que no se cuenta al no realizar ningún cálculo, y una capa de salida donde se realizan los cálculos.
  - Redes neuronales multicapa: Existen un conjunto de capas intermedias entre la entrada y la salida (capas ocultas).

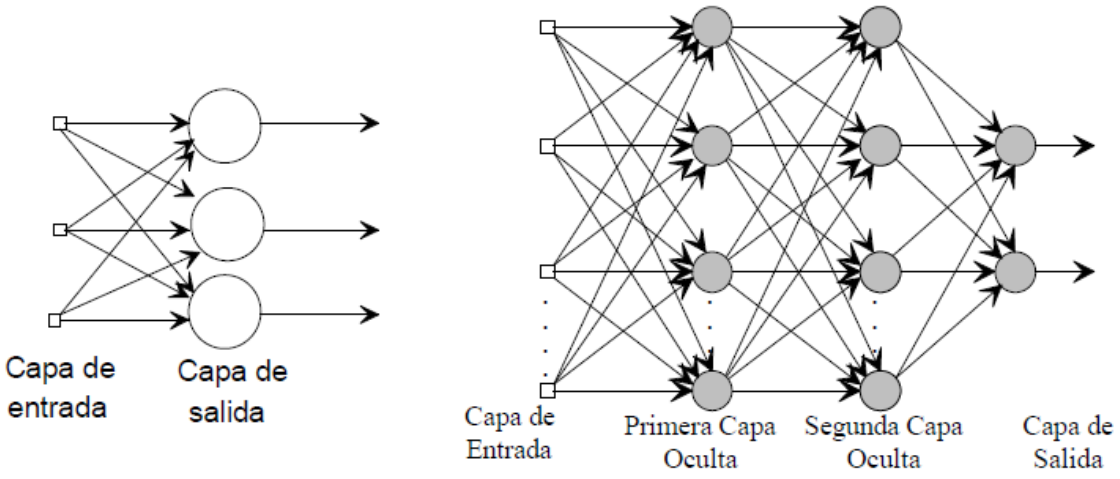

Figura 4. A la izquierda, el esquema de una red neuronal monocapa. A la derecha, el esquema de una red neuronal multicapa [4]

- Según el tipo de conexiones:
  - Redes neuronales no recurrentes: En este tipo de red la propagación de las señales se produce en un único sentido, no existiendo la posibilidad de realimentación. Esta estructura no tiene memoria.
  - Redes neuronales recurrentes: Esta red se caracteriza por la existencia de lazos de realimentación, siendo estos lazos entre neuronas de diferentes capas, de la misma capa o entre una misma neurona. Esta estructura recurrente la hace especialmente adecuada para estudiar la dinámica de sistemas no lineales.

En una red neuronal es necesario definir un algoritmo de aprendizaje. Los métodos de aprendizaje se pueden dividir en las siguientes categorías:

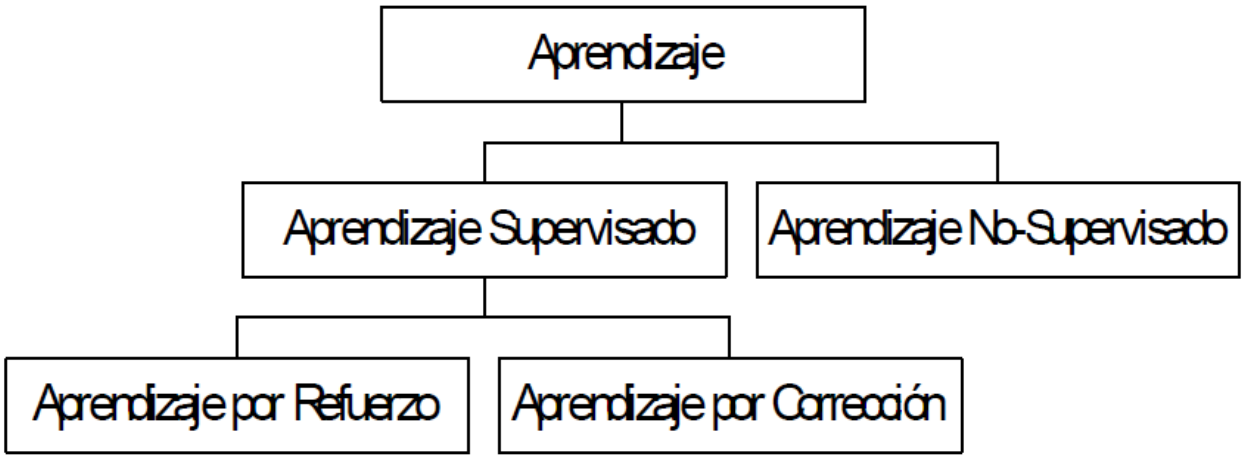

Figura 5. Métodos de aprendizaje de una red neuronal [4]

La primera división se hace entre algoritmos supervisados o no supervisados. En los métodos no supervisados no se conoce la señal deseada que debe dar como salida la red. La red en este caso se organiza mediante técnicas de agrupamiento o clustering.

El aprendizaje supervisado presenta a la red las salidas que debe proporcionar ante las señales que se le presentan. Posteriormente, se determina la diferencia entre la salida de la red y la señal deseada y los pesos se modifican de acuerdo al error cometido. Este aprendizaje tiene dos variantes: aprendizaje por refuerzo o aprendizaje por corrección. En el aprendizaje por refuerzo solo se conoce si la salida se corresponde o no con la deseada. En el aprendizaje por corrección se conoce la magnitud del error y esta determina la magnitud en el cambio del valor de los pesos sinápticos.

Las redes neuronales son muy útiles en problemas en los que se permite una cierta tolerancia a la imprecisión y donde la aplicación de reglas rápidas y robustas no es posible. Actualmente son usadas en campos de investigación muy diferentes entre sí, como en procesos de control, ayuda a la decisión clínica, ayuda a discapacidades físicas, modelización de mercados financieros, reconocimiento de patrones, etc.

2.2.1. Algoritmo backpropagation

El algoritmo de aprendizaje backpropagation es un algoritmo de descenso por gradiente que retropropaga el error desde la capa de salida hasta la capa de entrada, optimizando los valores de los pesos sinápticos mediante la minimización de una función de coste. El algoritmo se puede dividir en dos fases:

1. Propagación hacia delante: Se propagan las señales desde la capa de entrada hacia la de salida, obteniendo la salida de la red y el error al comparar esta salida con el valor deseado.
2. Propagación hacia atrás: En función de los errores cometidos en la capa de salida, el algoritmo optimiza los valores de los pesos sinápticos mediante la retropropagación del error desde la capa de salida a la de entrada.

Para la descripción de este algoritmo se obtendrá el algoritmo para el caso más sencillo de una red, estando formada por una capa de entrada, una capa oculta y una de salida.

En lo sucesivo $\{x_1, x_2, \dots, x_n\}$ serán las entradas de la red que formarán las capas de entrada, mientras que x_0 es la entrada que corresponderá al umbral. Por otra parte $\{w_{m1}, w_{m2}, \dots, w_{mn}\}$ serán los pesos sinápticos que conectan las entradas con la neurona m de la capa oculta y w_{m0} es el peso sináptico correspondiente al umbral. El potencial de activación de la neurona m vendrá dado por:

$$v_m(t) = \sum_{i=0}^{n} w_{mi} x_i(t) \qquad \textbf{(2.16)}$$

El índice t indica el número de iteración. Si se denota por φ la función de activación, la salida de la neurona viene dada por la siguiente expresión:

$$y_m(t) = \varphi_m(v_m(t)) \qquad \textbf{(2.17)}$$

El vector y_m, que tiene tantos elementos como neuronas en la capa oculta, definirá las salidas de la capa oculta y, por tanto, se corresponderá con las entradas a las neuronas de la capa de salida. Los pesos sinápticos que conectan las entradas m con la neurona p de la capa de salida se denotarán por $\{h_{p1}, h_{p2}, \dots, h_{pr}\}$,

donde r denota el número de neuronas en la capa oculta. El peso sináptico del umbral viene definido por h_{p0}. El potencial de activación de las neuronas de la capa de salida viene dado por:

$$z_p(t) = \sum_{j=0}^{r} h_{pj} y_j \quad \textbf{(2.18)}$$

Si ϕ es la función de activación, la salida de la neurona será:

$$o_p(t) = \phi_p(z_p(t)) \quad \textbf{(2.19)}$$

Denotando por d_p el valor de la salida deseada, el error de la red queda definido por:

$$e_p = d_p - o_p \quad \textbf{(2.20)}$$

En el proceso de retropropagación, los primeros pesos en actualizarse son los de la capa de salida. El proceso iterativo que actualiza los pesos sinápticos es la regla Delta:

$$\Delta\zeta(t) = -\alpha \frac{\partial J}{\partial \zeta(t)} \quad \textbf{(2.21)}$$

Siendo ζ el peso sináptico a actualizarse y α el factor de aprendizaje, un valor constante que ajusta la velocidad de aprendizaje de la red. La derivada parcial de la función de coste respecto a estos pesos puede expresarse, teniendo en cuenta la regla de la cadena, del siguiente modo:

$$\frac{\partial J}{\partial h_{pj}(t)} = \frac{\partial J}{\partial e_p(t)} \frac{\partial e_p(t)}{\partial o_p(t)} \frac{\partial o_p(t)}{\partial z_p(t)} \frac{\partial z_p(t)}{\partial h_{pj}(t)} \quad \textbf{(2.22)}$$

El resultado de la primera derivada, al usar el error cuadrático medio como función de coste, es igual a $2e_p(t)$. La segunda derivada es trivial a partir de la ecuación 2.20 y vale -1. La tercera derivada depende de la función de activación que se utilice. Derivando de la ecuación 2.18 se tiene:

$$\frac{\partial o_p(t)}{\partial z_p(t)} = \phi_p{}'(z_p(t)) \quad \textbf{(2.23)}$$

Por último, de la ecuación 2.17 se deduce:

$$\frac{\partial z_p(t)}{\partial h_{pj}(t)} = y_j(t) \quad \textbf{(2.24)}$$

Es decir, la actualización iterativa de los pesos de la capa de salida viene dada por:

$$\Delta h_{pj}(t) = 2\alpha e_p(t)\phi_p'\left(z_p(t)\right) y_j(t) \quad \textbf{(2.25)}$$

Para el peso sináptico del umbral, se particulariza del siguiente modo:

$$\Delta h_{pj}(t) = 2\alpha e_p(t)\phi_p'\left(z_p(t)\right) \quad \textbf{(2.26)}$$

Siguiendo la retropropagación, han de actualizarse los pesos de la capa oculta. La derivada parcial de la función de coste respecto a estos pesos se puede expresar con la regla de la cadena por:

$$\frac{\partial J}{\partial w_{mi}(t)} = \sum_p \frac{\partial J}{\partial e_p(t)} \frac{\partial e_p(t)}{\partial o_p(t)} \frac{\partial o_p(t)}{\partial z_p(t)} \frac{\partial z_p(t)}{\partial y_m(t)} \frac{\partial y_m(t)}{\partial v_m(t)} \frac{\partial v_m(t)}{\partial w_{mi}(t)} \quad \textbf{(2.27)}$$

Realizando las derivadas se obtiene que la actualización de los pesos se puede expresar como:

$$\Delta w_{mi}(t) = 2\alpha \sum_p e_p(t)\phi_p'\left(z_p(t)\right) h_{pm}(t)\varphi_m'\left(v_m(t)\right) x_i(t) \quad \textbf{(2.28)}$$

Para el umbral se particulariza de forma análoga a como se realizó con las neuronas de la capa de salida.

En el caso de que se tenga más de una capa oculta, el proceso se realiza de la misma forma. Se ha de seguir la retropropagación atravesando todas las capas hasta llegar a la de entrada.

3. HERRAMIENTAS

En este capítulo se describirán los elementos utilizados en la realización del presente trabajo fin de máster. Se hará mención de cada componente físico utilizado para la implementación y prueba de los algoritmos, así como el software utilizado para el diseño, programación e implementación de los controladores.

3.1. PLCnext Control AXC F 2152

Los dispositivos PLCnext Control son PLCs para el sistema abierto PLCnext Technology. Estos permiten la implementación de proyectos de automatización sin las limitaciones de sistemas propietarios. Además, permite la programación paralela y combinación de lenguajes de programación conocidos como IEC 61131-3, C/C++, C# o MATLAB y Simulink [5].

El controlador AXC F 2152 tiene un procesador Arm Cortex A9 de doble núcleo de 2 x 800 Mhz, puede realizar hasta 32 tareas simultáneas y puede soportar hasta 63 módulos de entrada y salida. El PLC disponible en el DISA está integrado en un kit de iniciación comercializado por Phoenix Contact como plataforma educativa. Este kit viene incorporado con:

- Controlador PLCnext control AXC F 2152
- Módulo de soporte AXL F BP SE4, con 4 ranuras para elementos Axioline (AXL) y una transmisión local de 100 Mbps.
- Módulo E/S AXL SE AI4 U 0-10, con de 4 entradas analógicas que trabajan en un rango de 0 a 10 voltios.
- Módulo E/S AXL SE DI16/1.
- Módulo E/S AXL SE DO16/1.

Además, el kit usado en el DISA, cuenta con un módulo E/S AXL F AI2 AO2 1H, un módulo de entradas y salidas analógicas en un rango de -10 a 10 voltios. Este módulo es el que será usado en este trabajo fin de máster para la lectura de sensores y el envío de la señal de acción de control.

Este PLC fue elegido entre los otros disponibles en el DISA por poder utilizar CODESYS ya que, como se explicará más adelante en este documento, es un entorno de programación compatible con una gran variedad de fabricantes, lo que facilita compatibilidad de la metodología desarrollada con otros autómatas.

Figura 6. PlCnext Control AXC F 2152

3.2. Feedback 33 – 033 servomotor system

Este producto es un entrenador utilizado en el ámbito educativo para la investigación de sistemas de control mediante el uso de un servomecanismo comprendido por un motor de corriente continua, una variedad de sensores y controladores tanto analógicos como digitales.

Este sistema es utilizado en este trabajo fin de máster como banco de pruebas para realizar el control de velocidad de un motor de corriente continua. La propia plataforma cuenta con un tacómetro de 2,5 volts/1000 rpm.

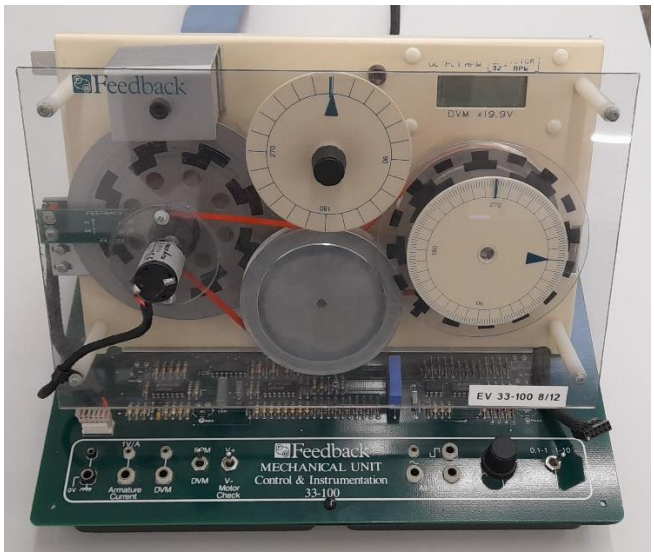

Figura 7. Feedback 33-033 servomotor system

3.3. Prototipo horno

Como segundo banco de pruebas, se ha decidido utilizar un prototipo de horno industrial disponible en el DISA. Este prototipo cuenta con una resistencia HSC 100 4R7 alimentada con un rango de voltaje de entre 0 y 21 voltios, debido a una limitación de la fuente de alimentación usada; un ventilador con distintas velocidades de funcionamiento, usado para poder aumentar la velocidad de enfriamiento del sistema; y un termopar perteneciente a la serie SEM104.

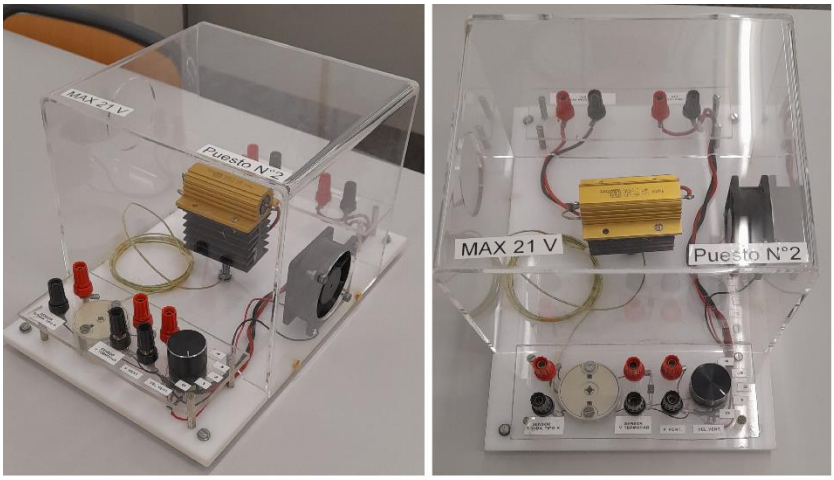

Figura 8. Prototipo de horno industrial

Los termopares son dispositivos no lineales, por lo que hay sido necesario linealizar la salida del sensor. Con un valor mínimo de medida de -50 ºC y un valor máximo de 250 ºC, y un voltaje de salida mínimo y máximo de 2 y 10 voltios,

respectivamente, la ecuación que determina la temperatura de salida del termopar es la siguiente:

$$Temp = \frac{(Tmax - Tmin) * (V - Vmin)}{(Vmax - Vmin)} + Tmin = \frac{300 * (V_m - 2)}{8} - 50 \quad \textbf{(3.1)}$$

Tanto este prototipo de horno, como el motor de corriente continua, fueron escogidos como banco de pruebas debido a que estos procesos (un sistema mecánico y un sistema térmico) son muy representativos de los procesos más utilizados en la industria.

3.4. MATLAB

Para el diseño y simulación de los algoritmos de control se ha utilizado MATLAB, un sistema de cálculo numérico que ofrece un entorno de desarrollo integrado fácil de manejar con un lenguaje de programación propio. Es un software que, debido a la facilidad de aprendizaje y utilización que ofrece, es muy usado en universidades y centros de investigación para cursos de introducción en matemáticas, ingeniería y ciencia.

Una de las razones por la que se ha escogido este software, aparte de la experiencia de trabajo obtenida a lo largo de la etapa universitaria, es que su elemento básico es un array que no requiere dimensionamiento. Esto permite formular soluciones a problemas técnicos computacionales que involucran representaciones matriciales en un tiempo mucho menor del que tomaría escribir un programa en un lenguaje escalar no interactivo como C. Esta propiedad hace que el lenguaje de programación sea ideal para el desarrollo de un algoritmo de control DMC.

3.5. Simulink PLC Coder

Simulink PLC Coder genera un programa compatible con el autómata en texto estructurado IEC 61131-3 a partir de modelos de Simulink, gráficas de Stateflow y funciones de MATLAB. El texto estructurado se genera en PLCopen XML y otros formatos de archivo soportados mediante entornos de desarrollo integrados, incluidos CODESYS, Rockwell Automation Studio 5000, Siemens TIA Portal y

Omron Sysmac Studio. En consecuencia, puede compilar e implementar aplicaciones en numerosos dispositivos PLC y PAC [6].

Es importante tener en cuenta que PLC Coder tiene ciertas limitaciones al convertir bloques de Simulink en código estructurado. Estas limitaciones pueden estar relacionadas tanto con las restricciones inherentes al lenguaje de texto estructurado como con las limitaciones específicas de la aplicación [7]. Para poder visualizar los bloques soportados por PLC Coder, al introducir “plclib” en la ventana de comandos se desplegará una ventana de Simulink con una librería. Esta librería está formada a su vez por dos librerías, Simulink y Stateflow. Cada librería contiene los bloques que pueden ser incluidos en un modelo para Simulink PLC Coder.

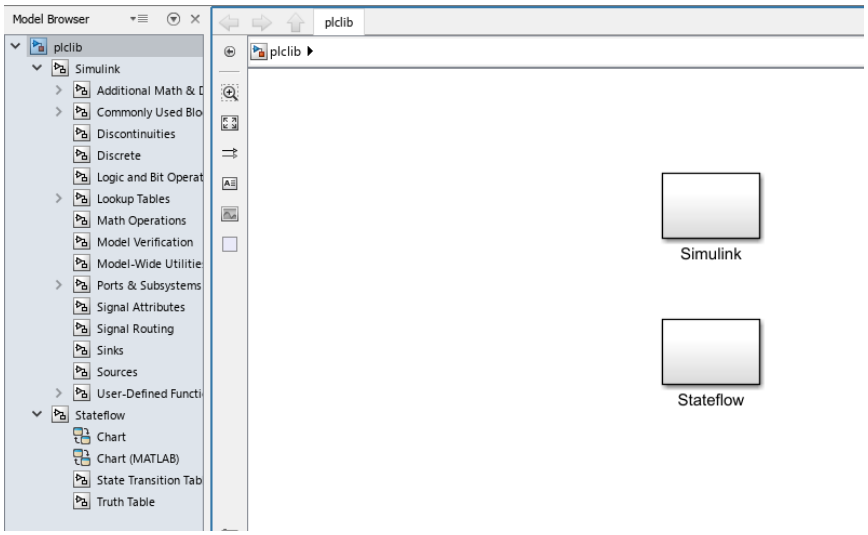

Figura 9. Librería de bloques compatibles con PLC Coder

3.6. CODESYS

CODESYS es un entorno de desarrollo para la programación de controladores conforme al estándar industrial internacional IEC 61131-3. El término CODESYS es un acrónimo de Sistema de Desarrollo de Controladores. Cuenta con los seis lenguajes de programación definidos en el estándar: IL (lista de instrucciones), ST (texto estructurado), LD (Diagrama Ladder), FBD (diagrama de bloques de función), SFC (Bloques de función secuenciales) y CFC (Continuos Function Chart).

CODESYS es conocido por su compatibilidad con una gran variedad de controladores y hardware de diferentes fabricantes. Más de 250 fabricantes de

dispositivos de diferentes sectores industriales ofrecen sus dispositivos con la interfaz de programación CODESYS, siendo la herramienta de desarrollo basada en IEC 61131-3 más extendida en Europa.

4. METODOLOGÍA

En este capítulo se va a describir de forma precisa la metodología creada durante este trabajo fin de máster para la implementación de los algoritmos de control avanzados en un PLC. El flujo de trabajo seguido se muestra en la siguiente figura:

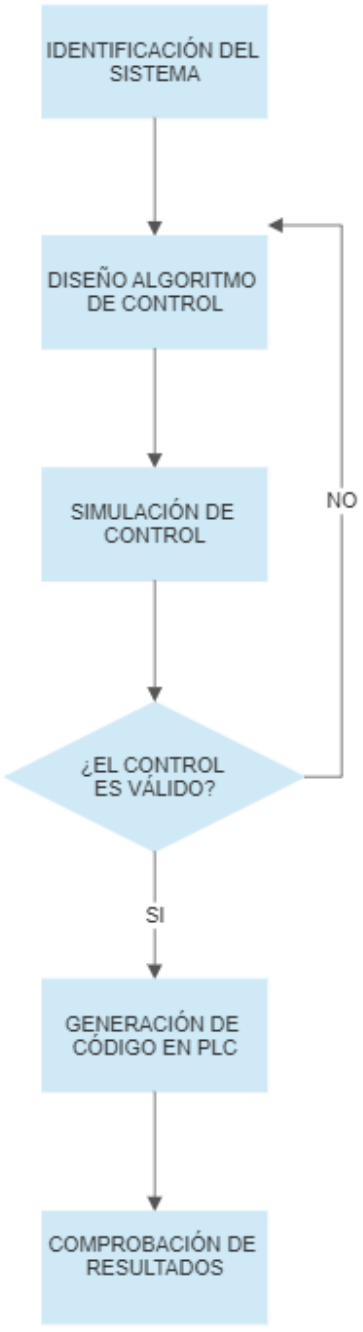

Figura 10. Flujo de trabajo durante el trabajo fin de máster

El primer paso a seguir es la identificación de los sistemas que componen nuestro banco de pruebas para obtener un modelo de planta. Este modelo será usado tanto para la simulación como para el desarrollo de los algoritmos de control, por lo tanto es un paso crítico en el resultado del control.

Es el segundo paso se enfoca en el diseño de los algoritmos de control. En esta fase, más que elegir el algoritmo más adecuado para cada planta, ya que no es el objetivo, se debe escoger el método de desarrollo del controlador que permita su implementación en el PLC.

El tercer paso consiste en la simulación de los controladores. Este proceso es necesario para comprobar que los controladores diseñados funcionarán correctamente al implementarlos en las plataformas de pruebas reales. En caso de que no sean válidos, se deberá revisar y rediseñar el algoritmo de control.

El cuarto paso es la implementación de los algoritmos en el PLC, una vez probados su funcionamiento en simulación. Se utilizará Simulink PLC Coder para la generación de texto estructurado a través del modelo creado en Simulink.

Por último, se compararán los resultados obtenidos con los controladores diseñados con un controlador PID diseñado para cada sistema. Este controlador PID permitirá establecer una referencia en cuanto al rendimiento de los controladores creados.

4.1. Identificación y modelado de planta

El método escogido para la identificación de los sistemas usados como banco de pruebas es la identificación mediante una respuesta ante entrada escalón. Este método consiste en aplicar sobre un sistema en equilibrio una entrada en escalón, y observar la respuesta que produce en el sistema. Posteriormente se analiza esta respuesta para obtener el modelo de planta del sistema.

Para actuar sobre el proceso y obtener la salida del sistema se ha utilizado una tarjeta de adquisición de datos NI USB-6001. Este dispositivo cuenta con 8 entradas

y 2 salidas analógicas con una resolución de 14 bits y un rango de operación de ± 10 V.

El rango de trabajo del actuador del motor DC se encuentra en ± 10 V y es un sistema rápido, por lo que se ha optado por introducir varios escalones positivos y negativos. En el caso del horno, el rango de operación del actuador se encuentra en el intervalo de 0 a 7 V, y debido a su elevado tiempo de establecimiento se ha utilizado un único escalón positivo y un escalón negativo. El tiempo de muestreo utilizado para el motor ha sido de 0.02 s, mientras que para el horno ha sido de 1 s. La respuesta de los sistemas se muestra en la siguiente figura:

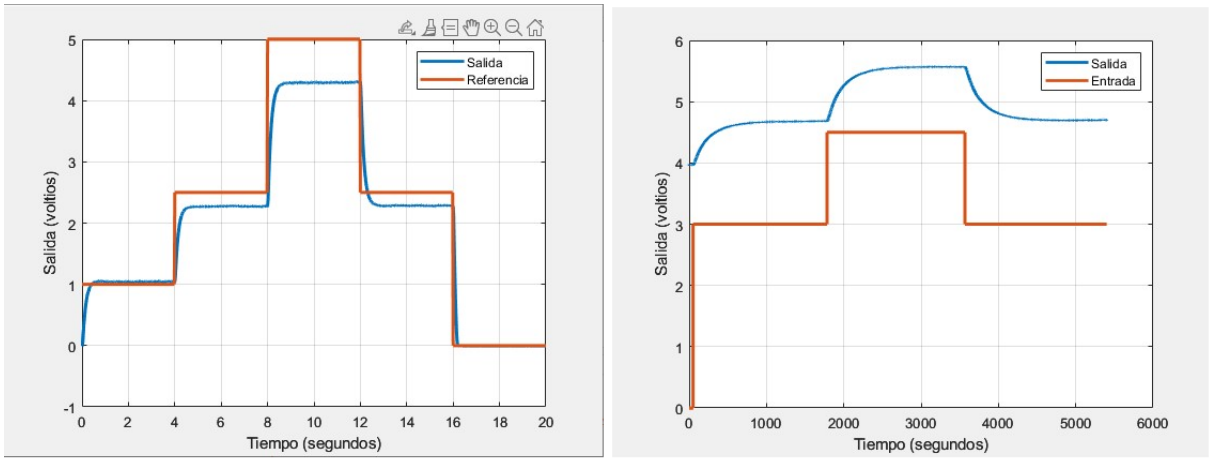

Figura 11. Respuesta de los sistemas ante entrada escalón.

Debido a que ninguna de las respuestas presenta sobreoscilación, se puede afirmar que ambos sistemas son de primer orden. La función de trasferencia que representa la respuesta de un sistema de primer orden es la siguiente:

$$G(s) = \frac{K}{1 + \tau * s} \quad \textbf{(4.1)}$$

Si se representa en el dominio del tiempo:

$$y(t) = K(1 - e^{-\frac{1}{\tau}t}) \quad \textbf{(4.2)}$$

Siendo K la ganancia estática del sistema y τ la constante de tiempo. Para obtener el modelo de planta se deben calcular el valor de estos parámetros.

La constante de tiempo representa el tiempo que tarda el sistema en llegar a un 63.2 % del valor final. La ganancia estática del sistema es equivalente a la diferencia

de la salida del sistema dividido por la diferente entra el estado inicial y la entrada escalón introducida.

Se ha obtenido el valor de cada sistema mediante la media de cada escalón. Para el motor de corriente continua, se tienen los siguientes valores:

Tabla 1. Identificación motor corriente continua

Escalón	τ (seg)	Δy (V)	Δu (V)	K
0-1	0.23	0.90	1	0.9
1-2.5	0.21	2.07	2.5	0.828
2.5-5	0.20	4.06	5	0.812
5-2.5	0.23	2.08	2.5	0.832
Media	0.2175			0.843

Por lo tanto, se tiene que la función de transferencia del modelo de motor de corriente continua es:

$$G_{motor}(s) = \frac{0.843}{0.2175 * s + 1} \quad \textbf{(4.3)}$$

Para el horno, se han recogido los siguientes valores de la identificación:

Tabla 2. Identificación horno

Escalón	τ (seg)	Δy (V)	Δu (V)	K
3-4.5	200	0.8863	1.5	0.5909
4.5-3	200	0.8831	1.5	0.5887
Media	200			0.5898

La función de transferencia del horno vienen dada por:

$$G_{Horno}(s) = \frac{0.5898}{200 * s + 1} \quad \textbf{(4.4)}$$

Una vez obtenido el modelo de planta de los sistemas, se está en posición de iniciar el diseño de los controladores. Este proceso se ha visto condicionado al hecho de que el algoritmo de control debía ser compatible con su implementación en un PLC. A continuación se describirá el proceso que se ha realizado hasta la obtención de los algoritmos de control compatibles con el controlador industrial.

4.2. Algoritmo de control DMC

El primer paso realizado fue estudiar las implementaciones de controladores DMC que fueran compatibles con Simulink PLC Coder, por lo que se realizó una búsqueda de diseños de controladores DMC en Simulink.

Durante el desarrollo del presente máster se cursó una asignatura que incluía el diseño de controladores predictivos en MATLAB [3], por lo tanto la primera fuente a la que consultar fue la implementación realizada durante las prácticas de laboratorio. Se escogió un modelo que había sido usado para mostrar la influencia de los diferentes parámetros de ajuste de un controlador DMC, por lo que su funcionamiento estaba más que probado.

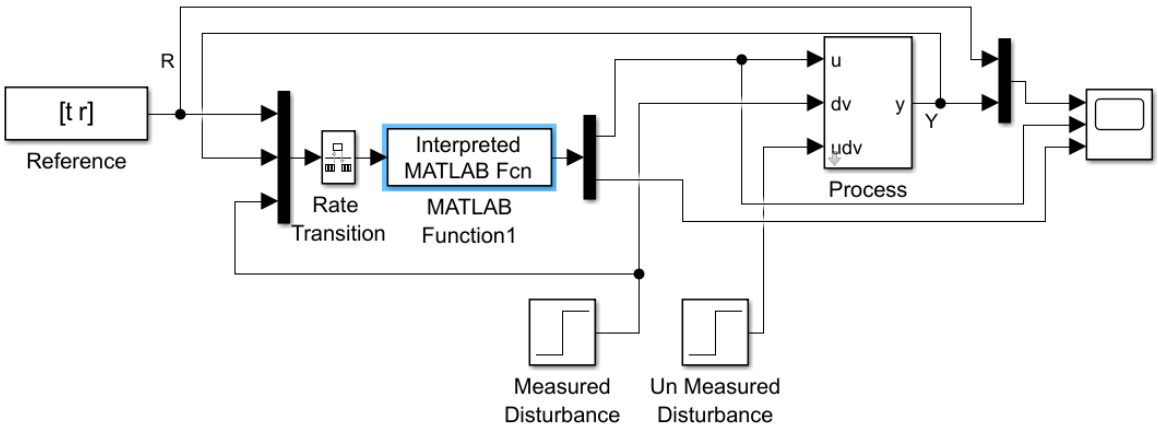

Figura 12. Esquema en Simulink de un control DMC

El controlador DMC estaba implementado mediante un bloque S-function. Este bloque permite añadir a un modelo de Simulink algoritmos en lenguaje MATLAB desarrollados en una función S.

Al consultar en la biblioteca mencionada en la sección 3 de esta memoria, se encontró que este bloque de Simulink no está soportado por PLC Coder, por lo tanto se tuvo que buscar otra forma de implementar el algoritmo de control.

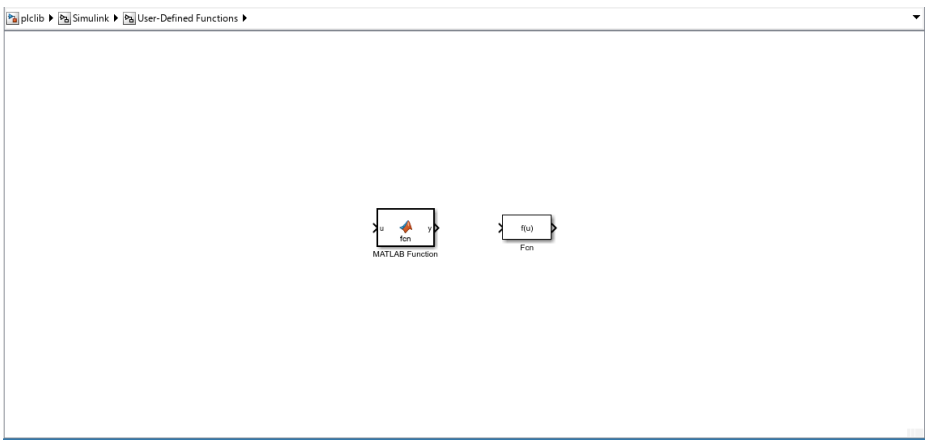

Figura 13. Busqueda de S-Function en la biblioteca plclib entre los bloques de funciones definidas por el usuario

La segunda fuente a consultar fue la página web oficial de MATLAB. Mathworks proporciona una gran cantidad de recursos de aprendizaje, incluyendo tutoriales, documentación ejemplos de código y videos instructivos para ayudar a los usuarios a dominar sus productos, teniendo un énfasis particular en MATLAB y Simulink.

En esta web se encontró un tutorial de algoritmos de control MPC que incluye un ejemplo de un control de un intercambiador de temperatura usando un controlador DMC en Simulink [8].

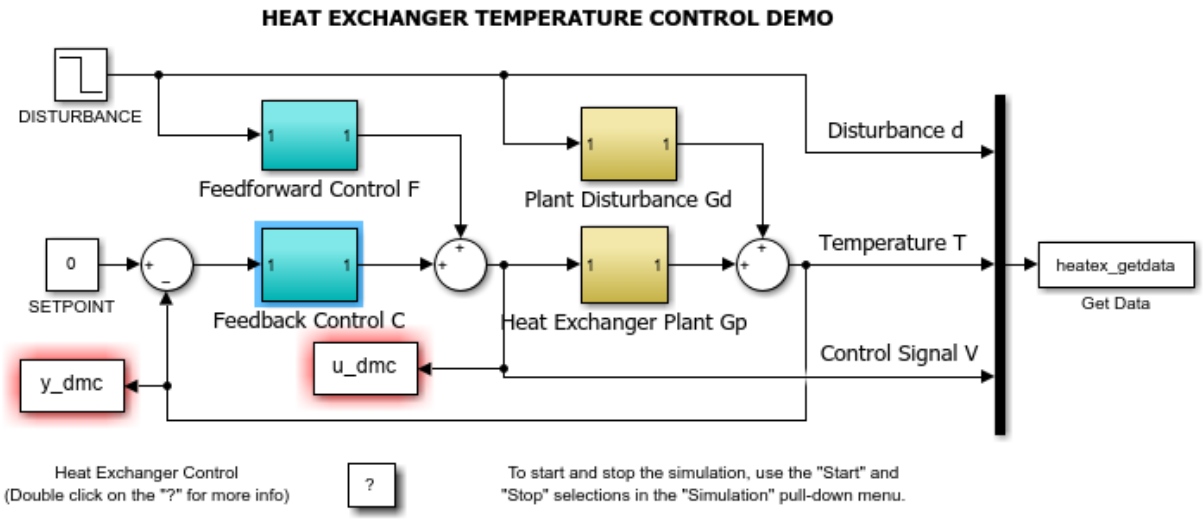

Figura 14. Ejemplo de implementación DMC de Mathworks [8]

Al abrir el bloque Feedback Control C, se encontró de nuevo que se había utilizado el bloque S-Function para llamar a una función programada en código.

Tras varias búsquedas similares, se encontró que el usar el bloque S-Function era el método más común para implementar este tipo de controlador, ya sea por su facilidad o por la influencia de la propia página web de MATLAB. Viendo esto, se decidió utilizar el bloque MATLAB Function.

Este bloque se diferencia del anterior en que, en vez de hacer una llamada a una función ya programada, permite programar dentro de Simulink en el lenguaje de programación MATLAB. Este bloque también presenta algunas limitaciones y requiere algunos ajustes para su correcto funcionamiento, los cuales se describirán más adelante.

4.2.1. Programación y simulación en MATLAB

El siguiente paso consiste en, ahora sí, diseñar el algoritmo de control. En vez de programar el algoritmo en el bloque de Simulink mencionado anteriormente, se ha optado primero por usar el entorno de programación de MATLAB. Se ha creado un archivo .m donde se incluye el diseño del controlador y una posterior simulación para comprobar su funcionamiento.

Lo primero a realizar para diseñar un controlador DMC es determinar, dada la dinámica del proceso, cuál deber ser el orden del modelo que permita recoger la respuesta dinámica del proceso de forma adecuada. Para ello, se puede simular la respuesta del sistema ante una entrada de tipo escalón unitario y medir su respuesta en los instantes siguientes.

Para el motor de corriente continua, se obtuvo que el sistema se estabiliza al llegar a 93 periodos de muestreo. En el caso del prototipo del horno, la salida del modelo se estabiliza tras 1254 periodos de muestreo.

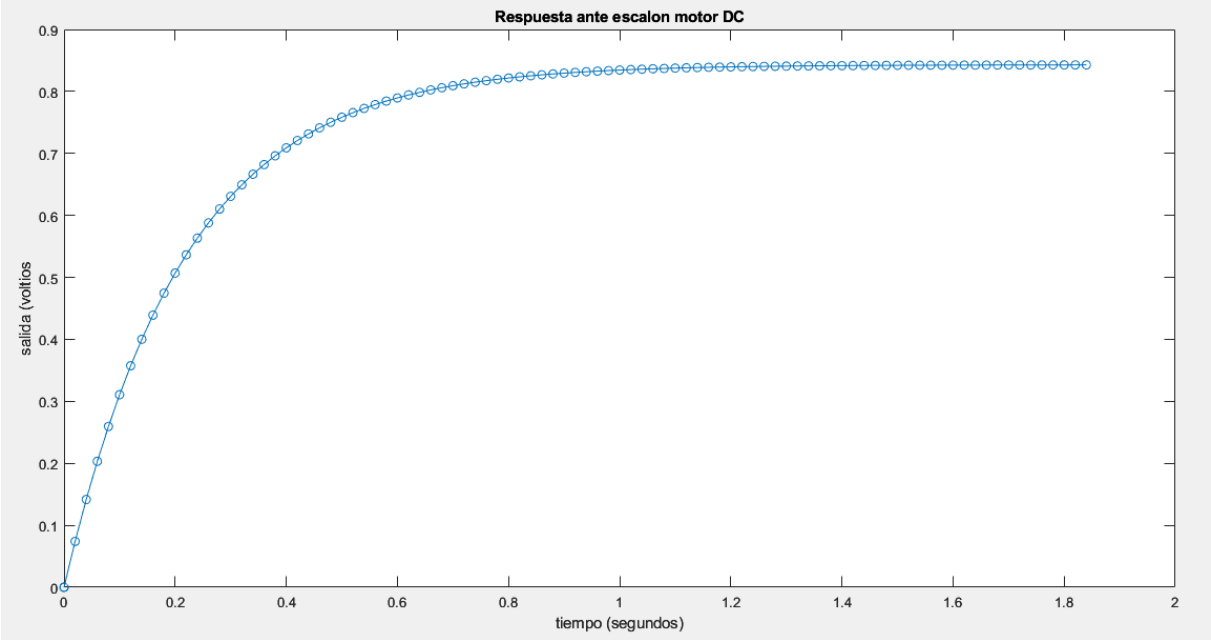

Figura 15. Respuesta ante escalón motor corriente continua (tiempo de muestreo = 0.02 segundos)

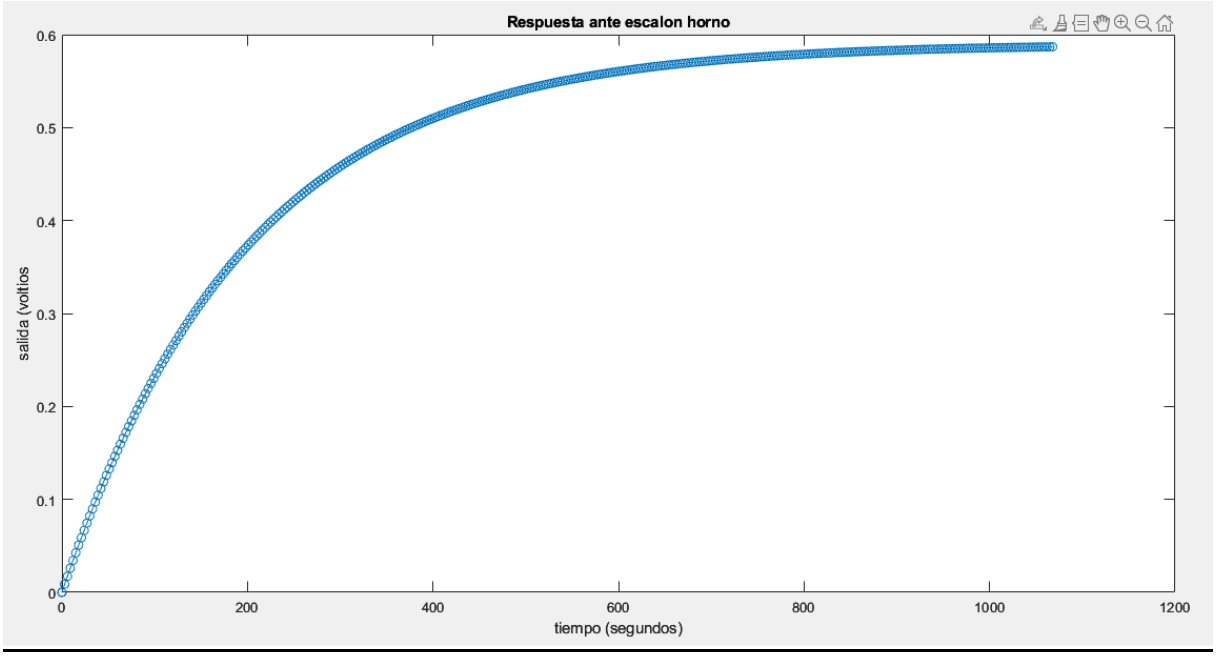

Figura 16. . Respuesta ante escalón horno (tiempo de muestreo = 1 segundo)

Seguidamente, han de elegirse los parámetros para el algoritmo de control. El horizonte de muestreo p debe ser igual al orden del modelo obtenido con la respuesta a la entrada de escalón unitaria. El horizonte de control c, al ser un parámetro que aumenta en gran medida la complejidad computacional del controlador, y al ser ambos sistemas a controlar de primer orden, se ha asignado un valor conservador, de un 20 % del horizonte de predicción. Los valores de ponderación de errores α y del esfuerzo de control λ han sido de 4 y 1 para el motor y de 50 y 1 para el horno, respectivamente, buscando un mejor seguimiento del error a costa de menor agresividad del controlador.

Una vez definidos los parámetros, se pasa a la programación del algoritmo. Esta programación se puede dividir en dos pasos:

1. Cálculo de ganancia de la acción de control. Como se explicó en el segundo capítulo de este documento, exceptuando el error, el resto de parámetros son conocidos e invariantes en el tiempo. Esto quiere decir que este cálculo, si se saca al error de la ecuación, puede realizarse solo una vez y durante la inicialización del control.
2. Cálculo de la acción de control y actualización de acciones pasadas. Durante cada periodo de muestreo se realiza el cálculo de las respuesta libre del sistema y de la acción de control. La nueva acción de control es

guardada para ser utilizada en los siguientes p periodos de muestreo para el cálculo de la respuesta libre.

A continuación se muestra el resultado de la simulación de los controladores diseñados para el motor de corriente continua y para el horno, respectivamente.

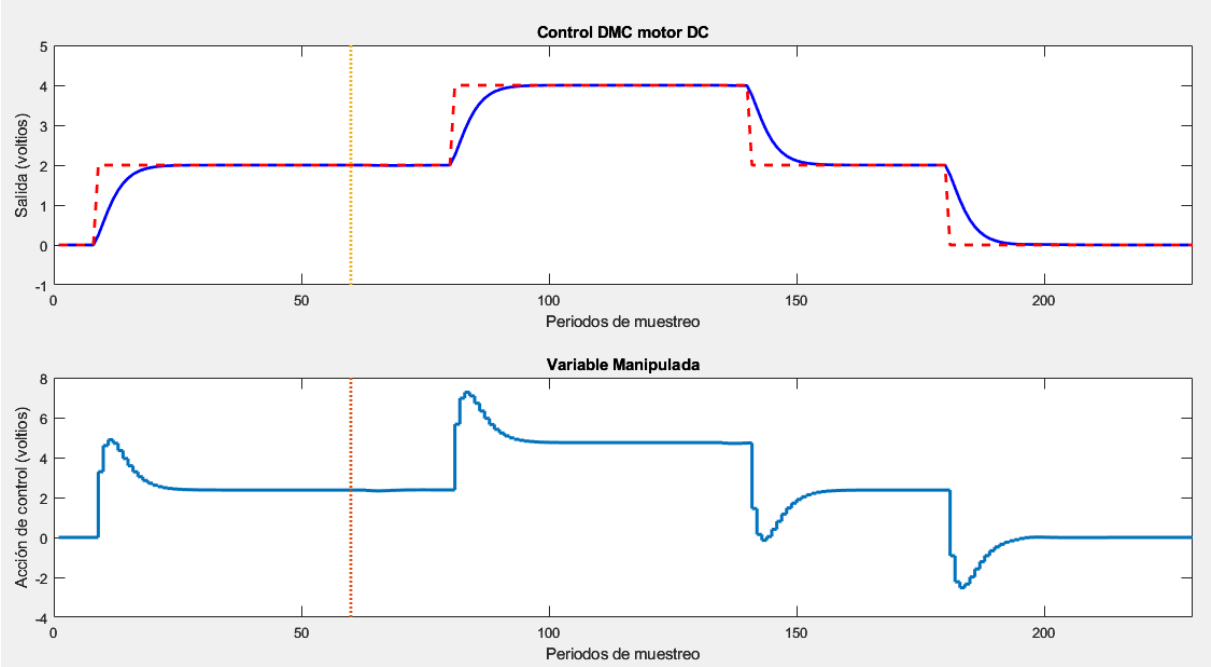

Figura 17. Simulación del control DMC aplicado al modelo del motor DC.

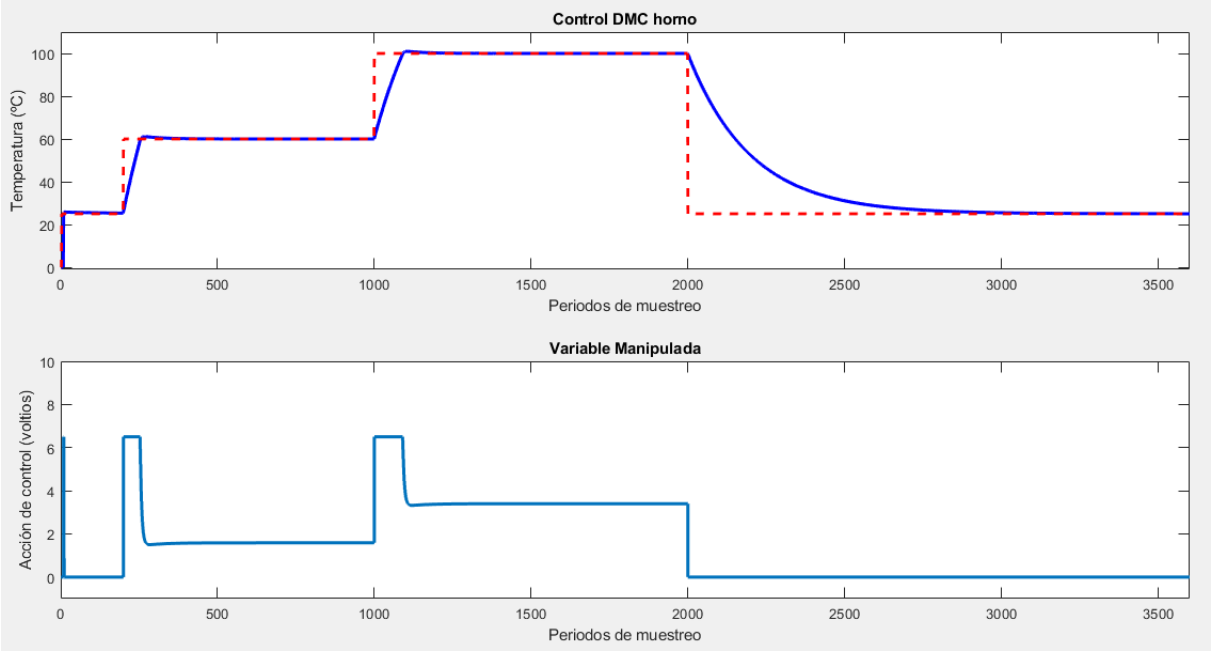

Figura 18. Simulación del control DMC aplicado al modelo del horno.

4.2.2. Modelado y simulación en Simulink

Ahora que tenemos la certeza de que el controlador DMC programado funciona correctamente, se pasa a modelar el sistema en Simulink. La siguiente figura muestra el esquema utilizado para la simulación:

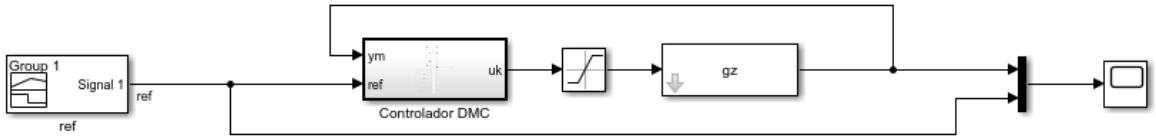

Figura 19. Esquema Simulink del control DMC

El controlador DMC se ha ubicado en un subsistema debido a que es necesario para la posterior utilización de Simulink PLC Coder, además de que aporta una mejor visualización y organización del modelo. Los bloques utilizados para el controlador son los siguientes:

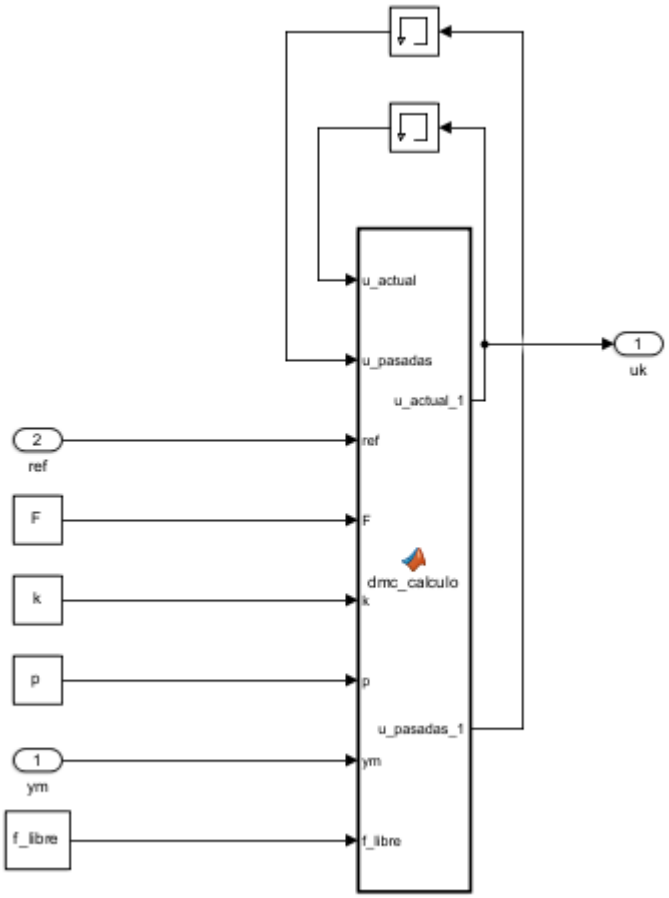

Figura 20. Bloques usados para el controlador DMC

Como se comentó anteriormente en este documento, para la implementación del controlador se utiliza el bloque “MATLAB Function”. Este bloque contiene la programación que debe ejecutarse en cada periodo de muestreo. Para los parámetros de entrada que contienen arrays que van a ser escritos durante la ejecución del bloque, hay que indicar previamente el tamaño de estos arrays. Para ello, en “MODELING” hay que activar las opciones “Symbols Pane” y “Property Inspector”. Una vez han aparecido estos paneles a la derecha, en Symbols se selecciona la variable y en Property Inspector se escribe el tamaño del array.

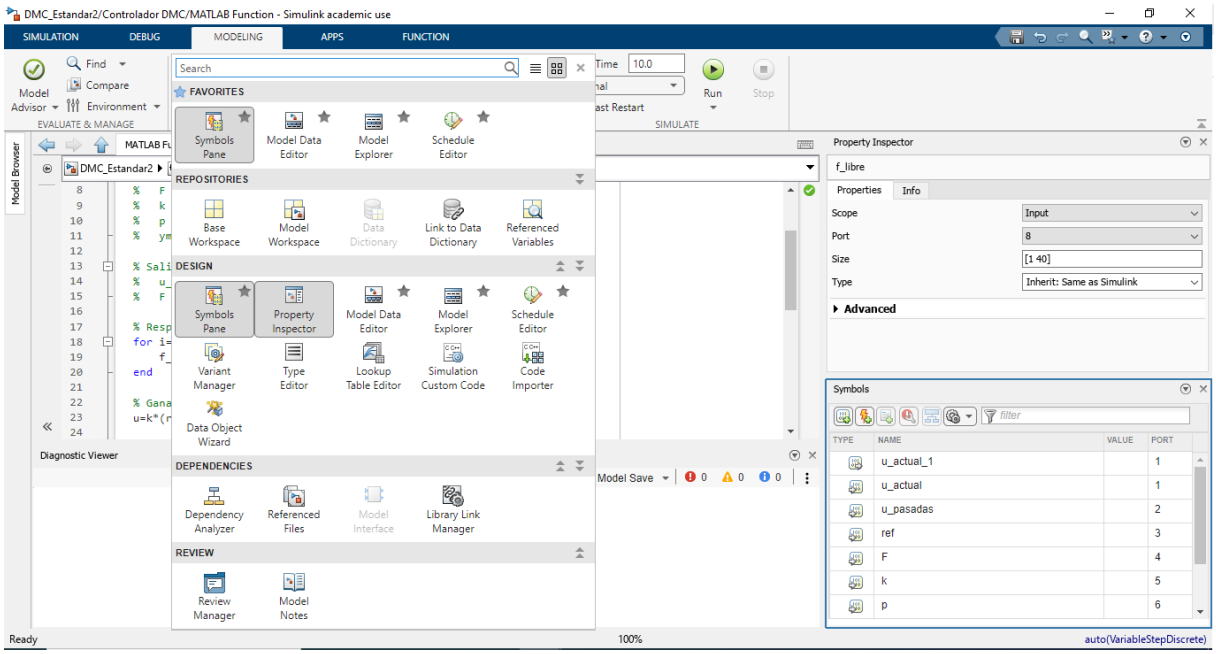

Figura 21. Ajuste de tamaño de arrays en el bloque MATLAB Function

Los bloques “Constant” se usan tanto para los parámetros invariantes en el tiempo como para los vectores y matrices que se ejecutan en el bloque de función, ya que el bloque exige que el tamaño de los arrays sea definido previamente y que este tamaño no pueda ser modificado. Para poder almacenar las acciones de control pasadas se utiliza el bloque “Memory”. Este bloque retiene el valor que recibe de entrada y retrasa su salida durante un periodo de muestreo.

Una vez realizada la configuración, se está en disposición de simular el controlador. El resultado para el motor y el horno, respectivamente, es el siguiente:

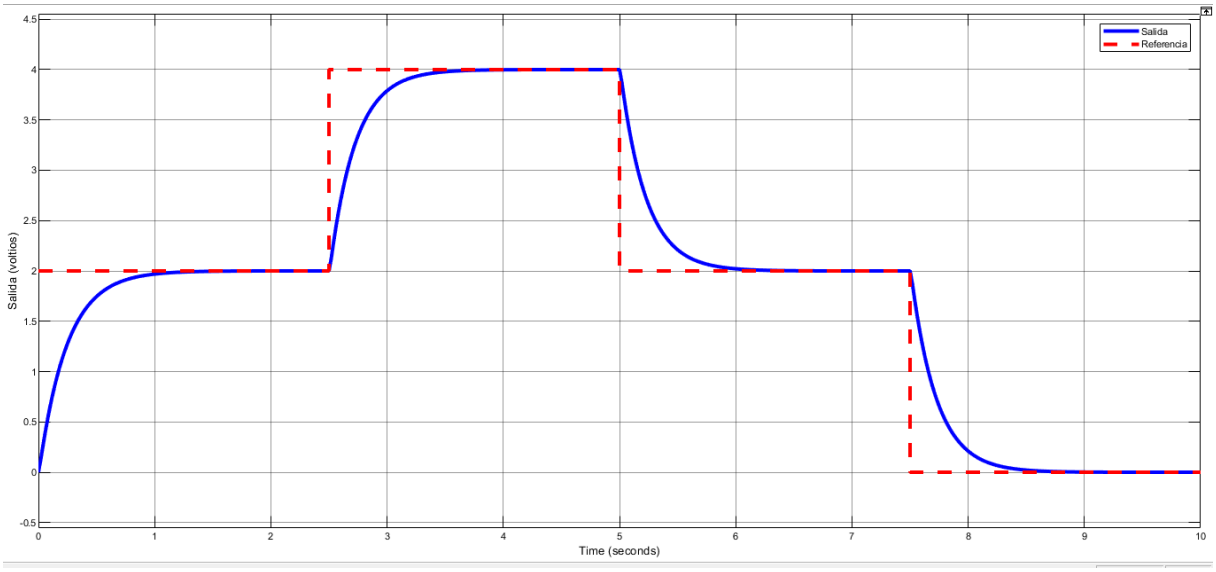

Figura 22. Simulación control DMC motor

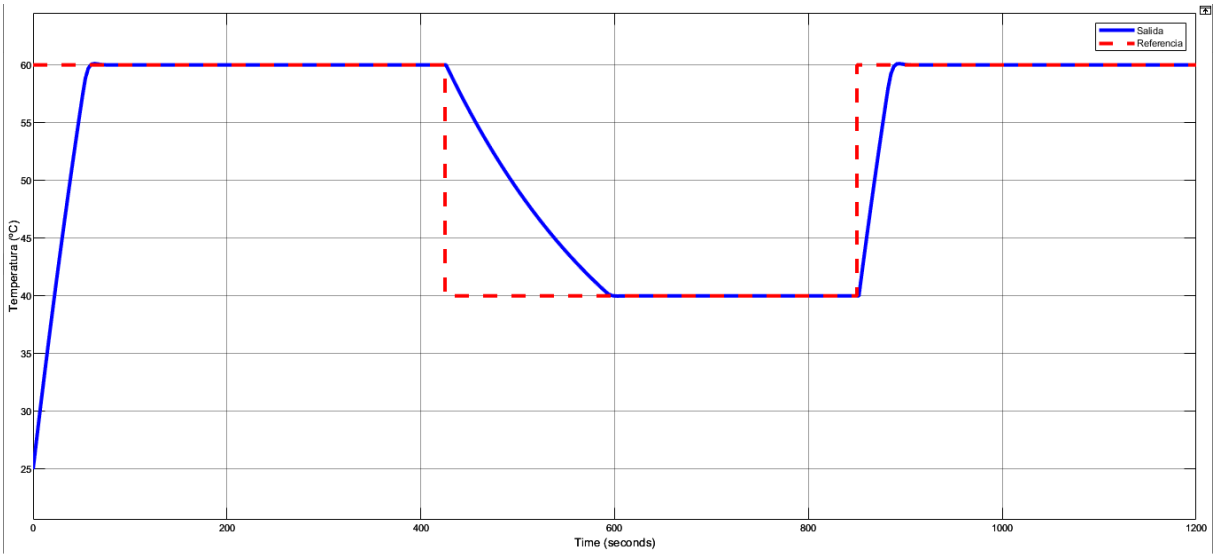

Figura 23. Simulación control DMC horno

En el control del motor DC, se ha obtenido un tiempo de establecimiento medio de 0.854 segundos sin sobreoscilación. En el control del horno, el tiempo de establecimiento ha sido de 84.16 segundos, y tampoco se han producido sobreimpulsos. En ninguno de los controladores, en régimen permanente, se presenta error de posición u oscilaciones.

4.3. Control mediante red neuronal

De la misma forma que en el controlador predictivo, la primera decisión fue revisar la implementación de redes neuronales como controladores usualmente realizada. Se encontró que Simulink cuenta con una librería llamada “Deep Learning

Toolbox" y esta, a su vez, una sub-librería llamada "Control Systems" en la que se encuentran bloques de Simulink para la utilización de controladores basados en redes neuronales.

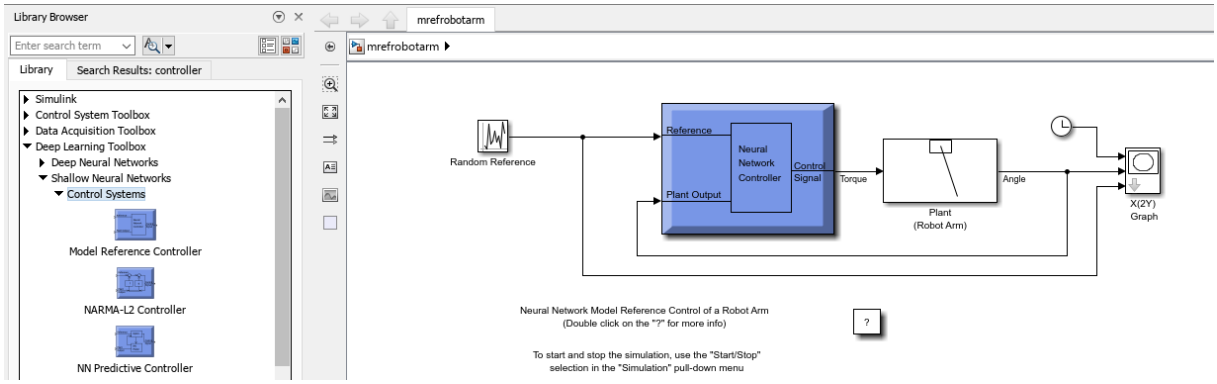

Figura 24. Modelo de ejemplo en Simulink "mrefrobotarm". A la izquierda se muestra la librería Deep Neural Networks.

Estos bloques de controladores resultaron ser incompatibles con Simulink PLC Coder debido a que no está habilitada la opción de "treat as atomic unit", necesaria para generar código en texto estructurado.

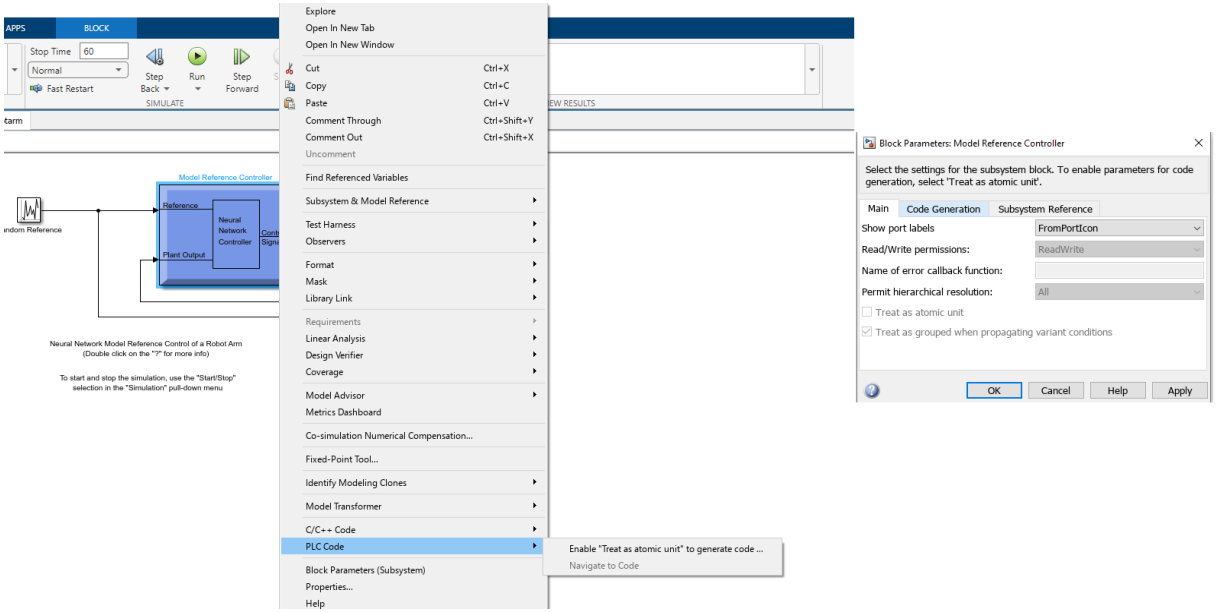

Figura 25. "treat as atomic unit" no disponible para los bloques de controladores basados en redes neuronales.

Debido a esto, se decidió que para la implementación del controlador por red neuronal, se utilizaría de nuevo el bloque MATLAB function. En el bloque de Simulink se programaría el algoritmo de propagación utilizando los pesos obtenidos durante una anterior fase de entrenamiento de la red diseñada.

4.3.1. Entrenamiento de red y simulación en MATLAB

La red neuronal diseñada es una red multicapa con tres entradas, diez neuronas en la capa oculta y una neurona de salida. La primera entrada corresponde al error del sistema, es decir, la diferencia entre la señal deseada y la actual. La segunda entrada corresponde a la derivada del error, y la tercera entrada a la integral de error, obteniendo una estructura prácticamente idéntica a la de un controlador PID. El número de neuronas en la capa oculta se determinó experimentalmente. Se fue aumentando el número de neuronas a medida que se aumentaba la precisión conseguida por la red. A partir de 10 neuronas, no se conseguía un aumento significativo de la precisión, por lo que se optó por no seguir aumentando el número para no incrementar el coste computacional.

El entrenamiento de la red se realiza mediante el algoritmo backpropagation. Como datos de entrenamiento se ha utilizado una señal de referencia con escalones de diferentes amplitudes y valor de establecimiento. En cada iteración, la red calcula una acción de control y ésta se aplica al modelo de planta, obteniendo la salida del sistema. Esta salida se compara con la referencia, obteniendo el error de la red, que será utilizado para calcular el valor de los nuevos pesos. El cálculo de estos pesos se realiza en cada iteración, y al final de cada época de entrenamiento, se calcula si el error total de la red es inferior a la precisión requerida.

Al igual que en el desarrollo del control predictivo, se ha optado por realizar el diseño en el entorno de programación de MATLAB, y después aplicar el algoritmo diseñado en Simulink. Para ello, se ha creado un archivo .m para el entrenamiento de la red neuronal y otro para la simulación de red entrenada. Los resultados de la simulación de la red neuronal para el motor y el horno se muestran en las siguientes figuras:

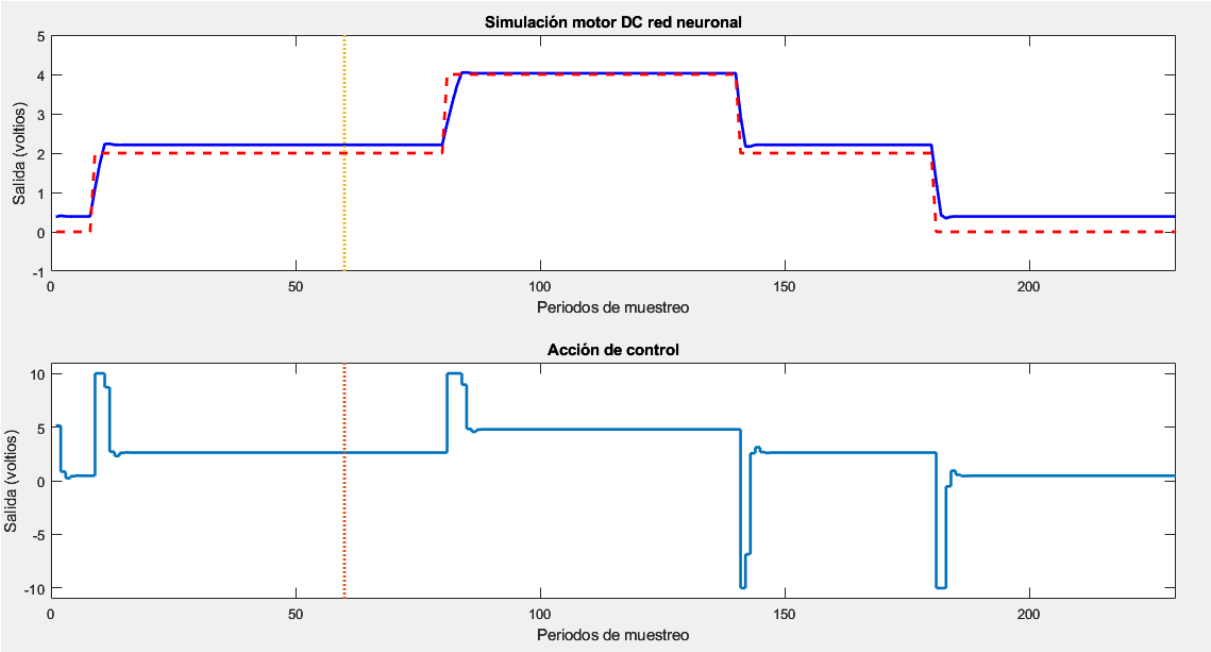

Figura 26. Simulación del control por red neuronal aplicado al motor DC.

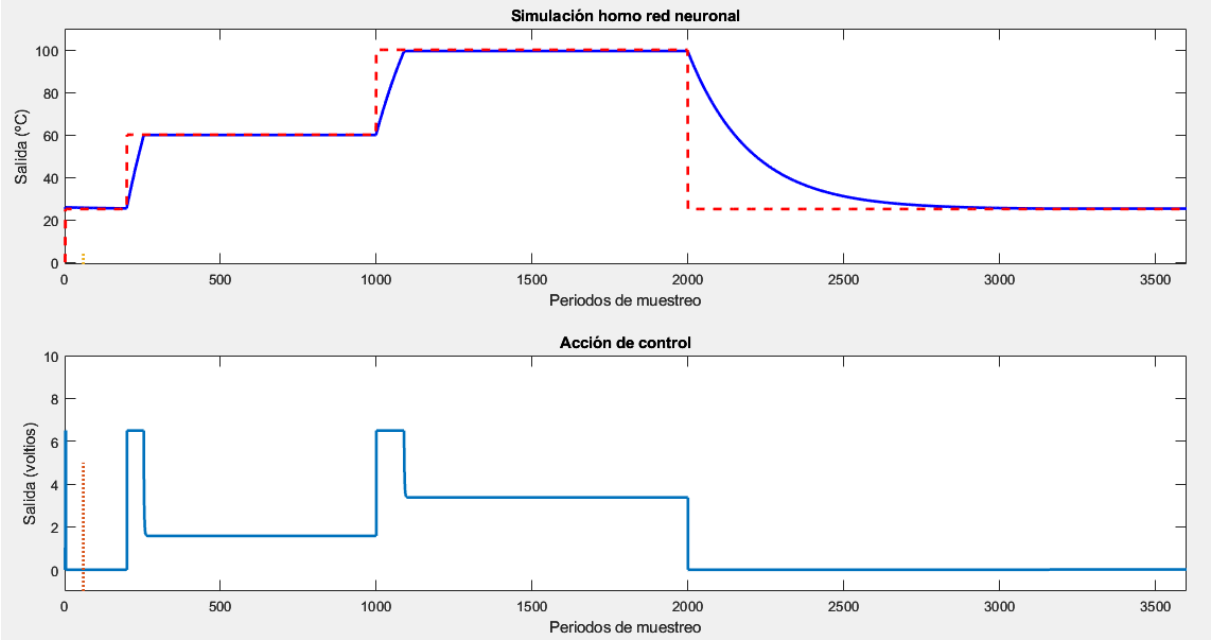

Figura 27. Simulación del control por red neuronal aplicado al horno

4.3.2. Modelado y simulación en Simulink

De forma análoga a como se ha procedido en el control predictivo, el modelado en Simulink de este controlador contiene los mismos bloques y estructura. Por ello, para conocer como se ha configurado el modelo, consultar la sección 4.2.2 de este documento. A continuación se muestra el subsistema creado para la red neuronal:

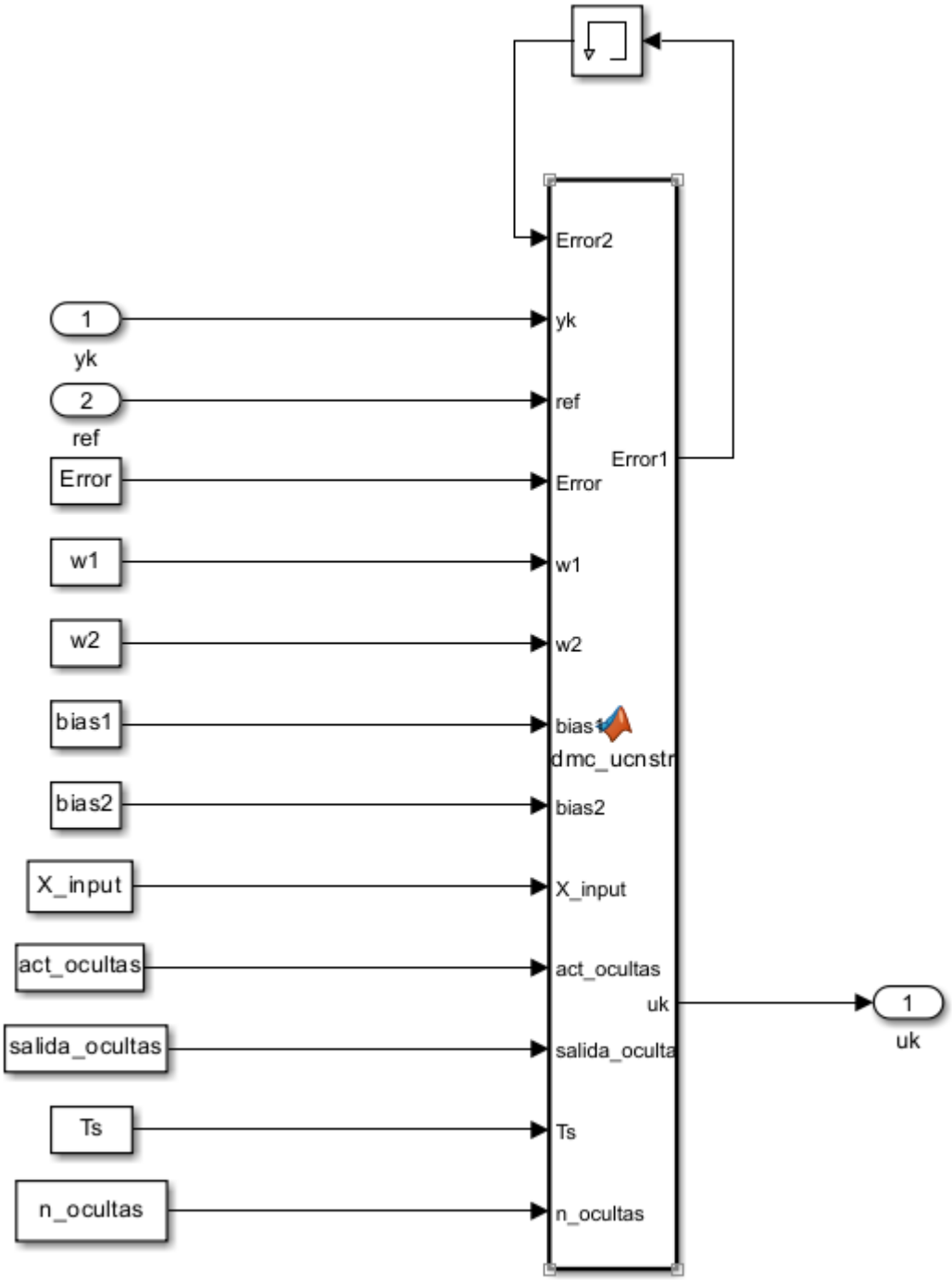

Figura 28. Bloques usados para el control por red neuronal en Simulink

El resultado de las simulaciones para el motor de corriente continua y para el horno son las siguientes:

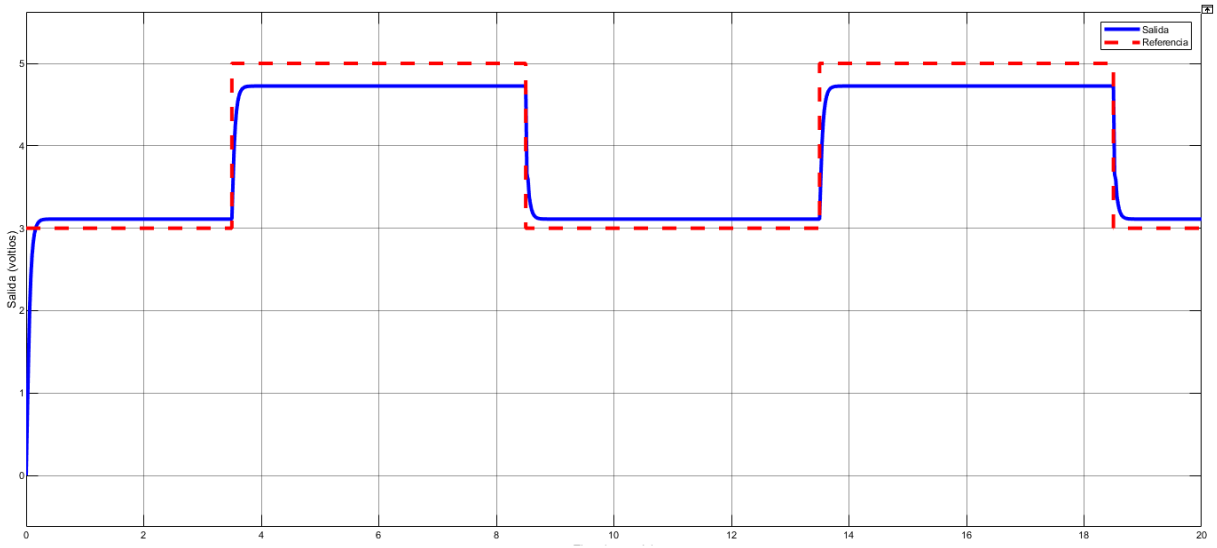

Figura 29. Simulación control por red neuronal motor DC

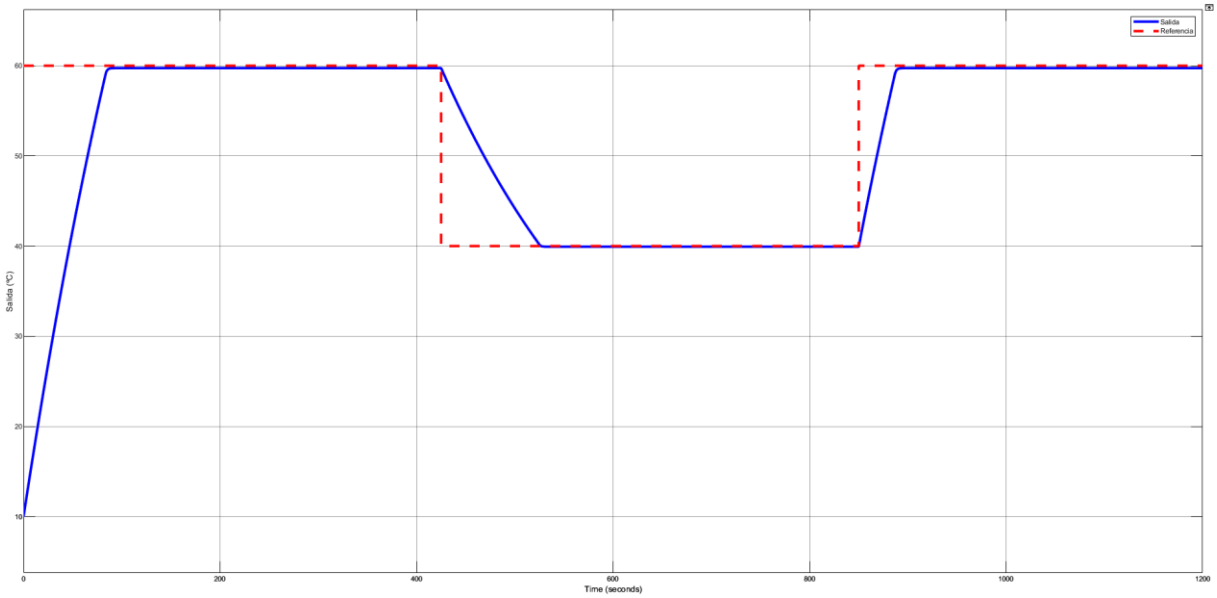

Figura 30. Simulación control por red neuronal horno

El controlador del motor tiene un tiempo de establecimiento de 0.161 segundos, pero esta velocidad se debe a que se produce un error en régimen permanente del 4 % en la referencia de amplitud tres y del 10 % en la referencia de amplitud cinco. Esto puede deberse a que en el entrenamiento se ha minimizado el error por medio de una respuesta rápida del seguimiento de la señal y no tanto por una disminución del error estacionario.

El control del horno tiene un tiempo de establecimiento de 84 segundos, igual al obtenido en la simulación del control DMC. La respuesta no presenta oscilaciones, pero si un error de posición del 0.45 %.

4.4. Generación de código mediante Simulink PLC Coder

Para poder generar código en texto estructurado mediante Simulink PLC Coder, primero debemos englobar todos los bloques a convertir en un único bloque o subsistema. Una vez hecho esto, en este subsistema hay que habilitar la opción “treat as atomic unit”, tal como se muestra en la Figura 25 pero, en este caso, la opción debe estar habilitada.

Una vez, habilitada la opción, ya es posible generar el código en texto estructurado, pero primero hay que seleccionar el entorno de programación en el que se va a utilizar el código generado.

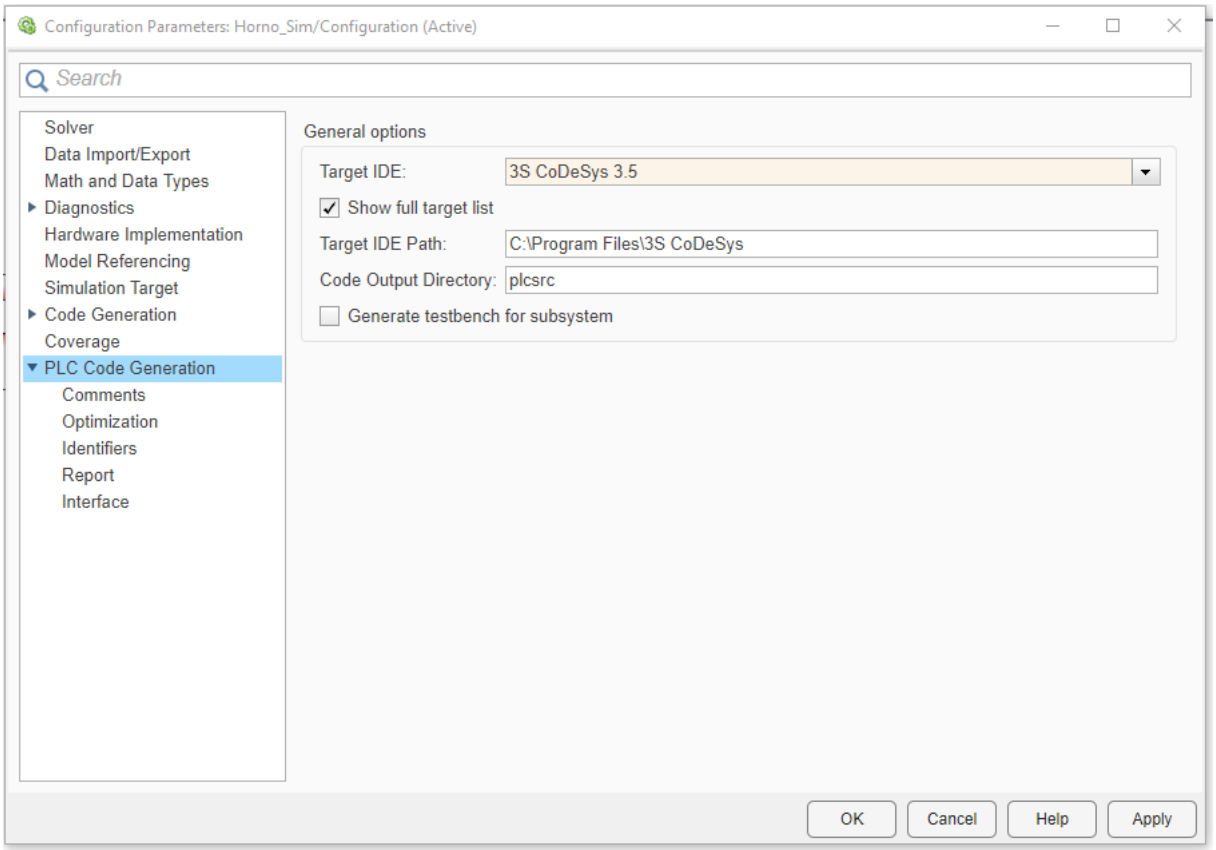

Figura 31. Selección de IDE objetivo en Simulink PLC Coder

Una vez hecho esto, se selecciona la opción “Generate Code for Subsystem”. Al hacer esto, se generará un archivo .st y con el código en formato de texto estructurado y organizado para ser implementado en un PLC como un bloque de función (FB).

```
FUNCTION_BLOCK Red
VAR_INPUT
    ssMethodType: SINT;
    yk: LREAL;
    ref: LREAL;
END_VAR
VAR_OUTPUT
    uk: LREAL;
END_VAR
VAR
    Memory_PreviousInput: LREAL;
    Constant8_Value: ARRAY [0..9] OF LREAL := 0.150591934939313706,0.18414815213144736,0.98001686430339585,
        1.0458123459904709,0.61498862695542589,0.078409364062224721,0.31357161808737516,0.39824384894373338,
        0.85481419789605961,0.066311422822905772;
    Constant2_Value: ARRAY [0..9] OF LREAL := 0.81124786463765552,0.75073944200697873,0.54750897962096146,
        1.1517206141415293,0.56105585553110062,0.38522972481643173,1.0571914127317363,0.70184087047404276,
        0.58910646669418787,1.0548655980411654;
    Constant1_Value: ARRAY [0..29] OF LREAL := 0.817303220653433,0.86869470536350968,0.084435845510910323,
        0.39978264909889649,0.25987040285065421,0.80006848022430754,0.43141382746354462,0.91064759442952292,
        0.18184702830285249,1.9719982896381554,0.14553898038471702,0.13606855870866374,0.8692922076400893,
        0.57970458736557018,0.54986020183633 2,0.14495479822372681,0.85303111772189366,0.622055131485066,
        0.35095238089227088,0.89281899361243422,0.40180803375194174,0.075966691690841914,0.23991615355365803,
        0.12331893483516554,0.18390778828241672,0.23995252566490277,0.41726706908436 95,0.049654430325742127,
        0.90271610991528106,2.463197835965135;
END_VAR
VAR_TEMP
    act_ocultas: ARRAY [0..9] OF LREAL;
    y: ARRAY [0..9] OF LREAL;
    i: DINT;
    tmp: LREAL;
    Error_idx_0: LREAL;
    X_input_idx_1: LREAL;
    X_input_idx_2: LREAL;
END_VAR
CASE ssMethodType OF
    SS_INITIALIZE:
        (* SystemInitialize for Atomic SubSystem: '<Root>/Red Multicapa' *)
        (* InitializeConditions for Memory: '<S1>/Memory' *)
        Memory_PreviousInput := 0.0;
        (* End of SystemInitialize for SubSystem: '<Root>/Red Multicapa' *)
    SS_STEP:
        (* Outputs for Atomic SubSystem: '<Root>/Red Multicapa' *)
        (* MATLAB Function: '<S1>/MATLAB Function' incorporates:
         *  Constant: '<S1>/Constant1'
         *  Constant: '<S1>/Constant8'
         *  Memory: '<S1>/Memory' *)
        FOR i := 0 TO 9 DO
            act_ocultas[i] := 0.0;
            y[i] := 0.0;
        END_FOR;
        END_FOR;
        (* MATLAB Function 'Red Multicapa/MATLAB Function': '<S2>:1' *)
        (* '<S2>:1:3' X_input = X_input'; *)
        (* '<S2>:1:4' w2 = w2'; *)
        (* '<S2>:1:6' Error(2) = Error2; *)
        (* '<S2>:1:7' Error(1) = ref - yk; *)
        Error_idx_0 := ref - yk;
        (* '<S2>:1:8' Error1 = Error(1); *)
        (* Entradas de red Adaline *)
        (* '<S2>:1:11' X_input(1) = Error(1); *)
        (* Entrada del error *)
        (* '<S2>:1:12' X_input(2) = (Error(1) - Error(2))/Ts; *)
        X_input_idx_1 := Error_idx_0 - Memory_PreviousInput;
        (* Entrada derivativa *)
        (* '<S2>:1:13' X_input(3) = ((Error(1) + Error(2))/2) * Ts; *)
        X_input_idx_2 := (Error_idx_0 + Memory_PreviousInput) / 2.0;
        (* Entrada integral *)
        (* PROPAGACIÓN HACIA DELANTE *)
        (* Propagación capa oculta *)
        (* '<S2>:1:17' for i=1:n_ocultas *)
        FOR i := 0 TO 9 DO
            (* Calculo del potencial de activación *)
            (* '<S2>:1:19' act_ocultas(i) = X_input*w1(i,:)' + bias1(i); *)
            act_ocultas[i] := ((((Constant1_Value[i + 10] * X_input_idx_1) +
            (* Cálculo de salida función sigmoidal *)
            (* salida_ocultas(i) = sigmoide(act_ocultas(i)); *)
            (* Cálculo de slaida funcion purelin *)
            (* '<S2>:1:23' salida_ocultas(i) = act_ocultas(i); *)
            y[i] := act_ocultas[i];
        END_FOR;
        (* Update for Memory: '<S1>/Memory' incorporates:
         *  MATLAB Function: '<S1>/MATLAB Function' *)
        (* Propagación capa de salida *)
        (* Cálculo del potencial de activación *)
        (* '<S2>:1:27' act_salida = w2*salida_ocultas + bias2; *)
        (* '<S2>:1:28' salida_net = act_salida; *)
        (* Salida de la red *)
        (* '<S2>:1:31' uk = salida_net; *)
        Memory_PreviousInput := Error_idx_0;
        (* MATLAB Function: '<S1>/MATLAB Function' incorporates:
         *  Constant: '<S1>/Constant2' *)
        tmp := 0.0;
        FOR i := 0 TO 9 DO
            tmp := (Constant2_Value[i] * y[i]) + tmp;
        END_FOR;
        (* Outport: '<Root>/uk' incorporates:
         *  Constant: '<S1>/Constant9'
         *  MATLAB Function: '<S1>/MATLAB Function' *)
        uk := tmp + 0.1325694231627311 8;
        (* End of Outputs for SubSystem: '<Root>/Red Multicapa' *)
END_CASE;
```

Figura 32. Código generado por Simulink PLC Coder

4.5. Programación control en PLC

El primer paso que se debe realizar al crear un proyecto en CODESYS es seleccionar el controlador lógico junto a los módulos que van a utilizarse. En este caso, se ha introducido el hardware descrito en la sección 3.1 de este documento.

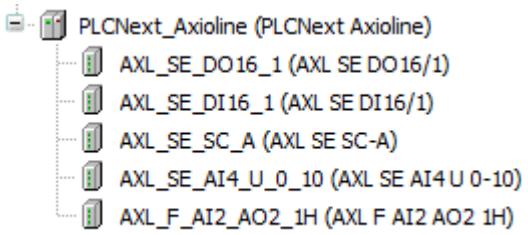

Figura 33. Dispositivos hardware incluidos en CODESYS

A continuación, ha de configurarse el módulo de entradas y salidas analógicas usado para la lectura de la señal de salida del sistema y para el envío de la señal de acción de control. En el caso del motor, el rango de medición ha de ser de entre -10 y 10 V; en el caso del horno de 0 a 10 V.

Para la lectura y el envío de la acción de control se tuvo que realizar un escalado del mismo. Para ello, se registraron los valores de saturación del módulo

de entrada y de salida tanto del rango de medida positivo como negativo. Los valores obtenidos fueron los siguientes:

Tabla 3. Escalado entrada analógica

Voltaje (V)	Lectura del sensor
-10	-30000
-5	-15000
0.12	30 - 40
-0	65535
-5	50535
-10	35535

Por lo tanto, se dedujo que el escalado del rango positivo era el siguiente:

$$y = x * 3000 \quad \textbf{(4.5)}$$

Y para el rango negativo era:

$$y = x * 3000 + 65535 \quad \textbf{(4.6)}$$

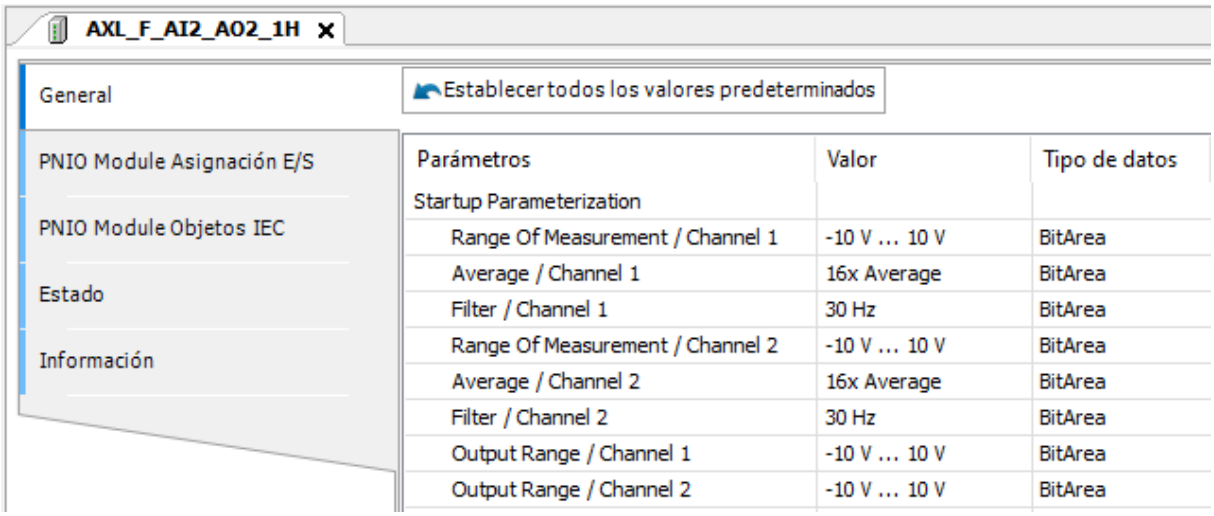

Figura 34. Ajuste módulo de entrada y salida analógica

Además de realizar el escalado, hay que asignar las direcciones de memoria de entrada y de salida usadas para la lectura de las entradas analógicas y el envío de las salidas analógicas.

Figura 35. Asignación de direcciones de memoria

El siguiente paso es añadir el programa principal y FBs utilizadas, incluyendo las generadas con Simunk PLC Coder. El programa principal realiza la lectura de entrada, el cálculo del error y el envío de la acción de control. Para el cálculo de la acción de control realiza una llamada a las FB generadas, excepto un controlador PID utilizado como referencia para comparar el resultado de los controladores avanzados. Este controlador se programará en el programa principal, debido a su simplicidad.

Para la llamada a las FBs se requiere el uso de una instancia de la función de bloque. Esta instancia debe ser declarada junto a las variables del programa y su declaración se realiza de la misma forma que cualquier variable, solo que en vez de asignarlo un tipo de dato se le asigna el nombre de la FB a la que hace referencia. En el programa, para utilizar la FB, primero se debe realizar la escritura de los valores de entrada; después se realiza la llamada a la FB; y por último se lee la salida del bloque.

```
InstanciaDMC.ref := referencia; // Escritura de entrada
InstanciaDMC.ym := y;
InstanciaDMC(); // Ejecución del bloque
u := InstanciaDMC.uk; // Lectura de la salida
```

Figura 36. Llamada de a una FB mediante la instancia "InstanciaDMC"

Para que el programa principal se ejecute de forma cíclica, hay que incluir una tarea en el proyecto. Además, el tiempo de ciclo debe ser el mismo que el tiempo de muestreo utilizado durante el diseño de los controladores. Normalmente, para el uso de controladores se utiliza un ciclo de interrupción, pero en este programa solo se realiza el control de un sistema, por lo que no es necesario.

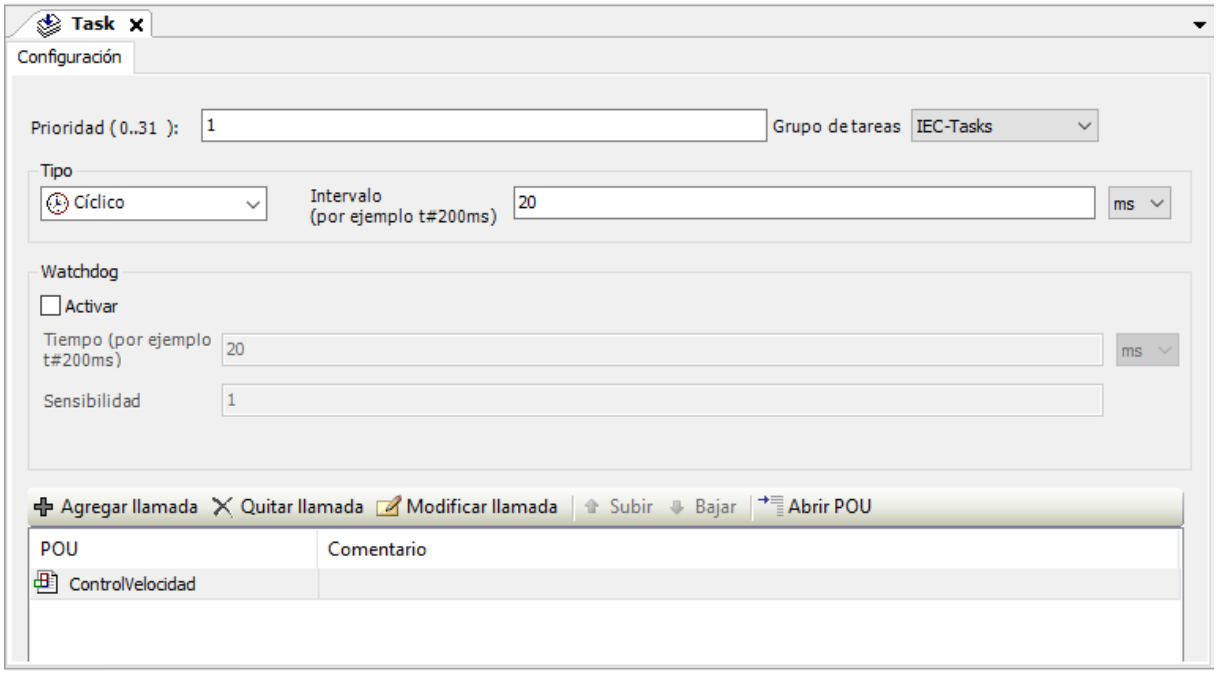

Figura 37. Configuración de la tarea del proyecto

Por último, se ha añadido una interfaz de usuario gráfica. Esta interfaz incluye una “trace” para la visualización del control en directo durante la ejecución del programa. También incluye unos botones de opción que permiten seleccionar el tipo de controlador a utilizar, y un botón de inicio y parada del control del sistema.

Figura 38. Interfaz gráfica de usuario

Las variables globales utilizadas se muestran a continuación:

```
1   VAR_GLOBAL
2       AI_1: REAL;
3       AI_2: REAL;
4       AO_1: REAL;
5       AO_2: REAL;
6       TipoControl: INT;
7       SS_DMC_ST: SINT := 0;
8       SS_RED: SINT := 0;
9       Inicio_Control : BOOL := FALSE;
10      END_VAR
```

Figura 39. Variables globales

Las primeras variables de tipo real se han utilizado para almacenar los valores de entrada y salida del módulo E/S analógico. La variable “TipoControl” es utilizada en el programa para asignar el tipo de controlador a utilizar. Su escritura se realiza en la interfaz gráfica mostrada en la Figura 38. Las variables “SS_DMC_ST” y “SS_RED” se utilizan para identificar si ha de inicializarse o no las variables de los controladores. Por último, la variable “Inicio_Control” se escribe en la interfaz gráfica y se usa para iniciar o parar el control sobre el proceso.

5. RESULTADOS OBTENIDOS

En esta sección se van a mostrar los resultados de los controladores avanzados diseñados en MATLAB y Simulink y exportados a un controlador lógico. Para tener una referencia se ha utilizado un controlador PID diseñado mediante el método de asignación de polos. Primero se comparará la respuesta de cada controlador entre la simulación realizada en Simulink y el resultado obtenido en la práctica en el PLC. Después, se comparan el resultado experimental de los tres controladores.

Durante el control experimental, se realizará una conexión online con el PLC para poder seleccionar el controlador a utilizar y visualizar los datos de salida y acción de control.

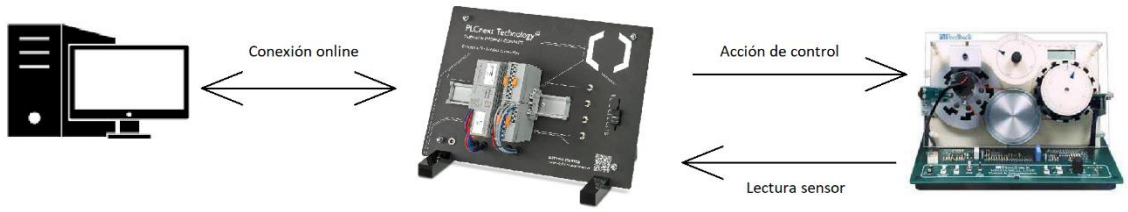

Figura 40. Esquema de comunicación en proceso experimental.

Los bancos de pruebas serán sometidos a dos escalones positivos y dos negativos, de un rango de amplitud de 3 y 5 V en para el motor; y de 40 y 60 ºC en el caso del horno.

CODESYS permite la exportación de los datos recogidos en una trace en un archivo .txt, así que para poder visualizar estos resultados se ha realizado una gráfica de dispersión usando Excel.

5.1. Motor DC

5.1.1. Controlador DMC

En la simulación mostrada en la Figura 22, se ha obtenido un tiempo de establecimiento medio de 0.854 segundos sin sobreoscilación. Durante el control experimental en el PLC se han obtenido los siguientes resultados:

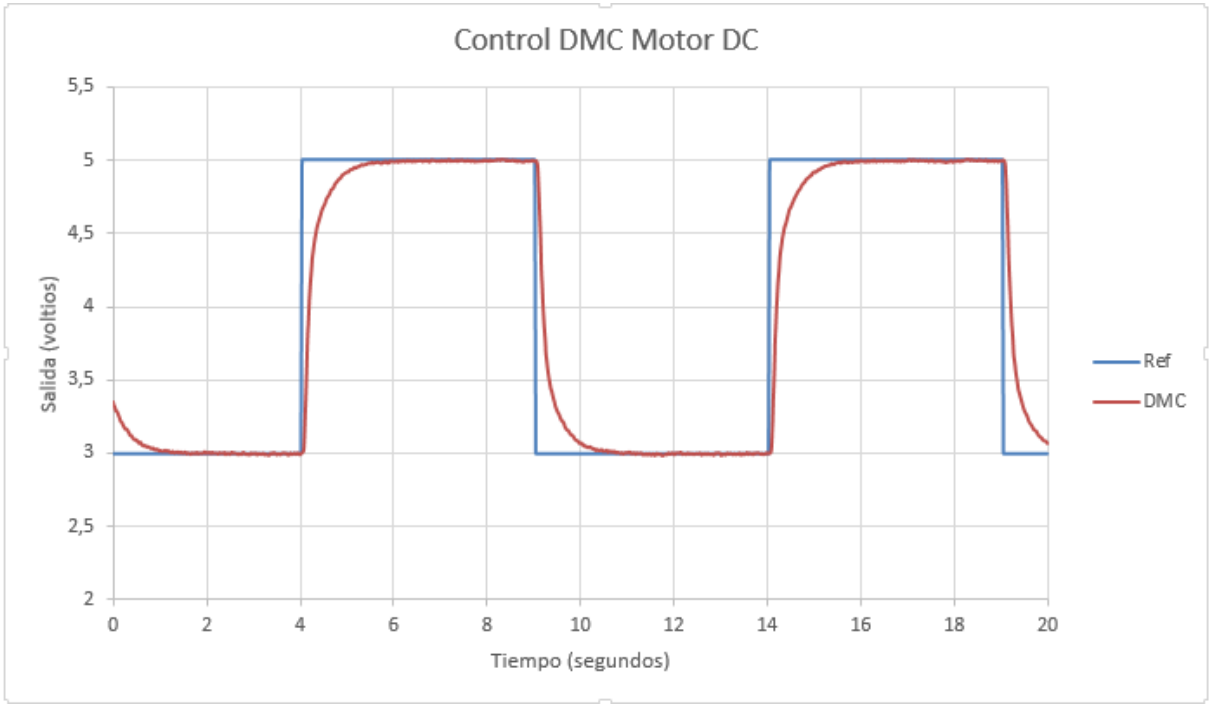

Figura 41. Resultado experimental controlador DMC motor DC

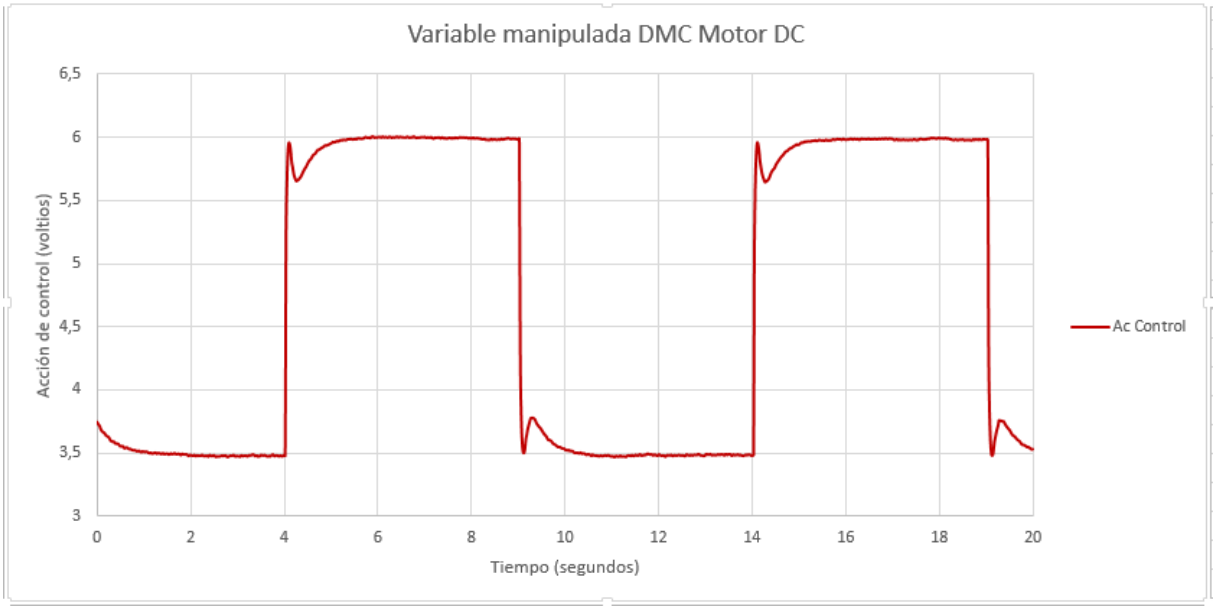

Figura 42. Acción de control controlador DMC motor DC

Al igual que en la simulación, no se ha producido sobreoscilamiento en el seguimiento de la referencia experimental. El tiempo de establecimiento medio ha sido de 1.099 segundos, aumentando ligeramente el de simulación. Además, no se presenta error en régimen permanente.

5.1.2. Controlador red neuronal

En simulación este controlador ha obtenido el tiempo de establecimiento más rápido, de 0.161 segundos de media. Sin embargo, esta velocidad puede deberse a que se produce un error en régimen permanente, siendo del 4 % en la referencia de amplitud tres y del 10% en la referencia de amplitud cinco.

En el proceso experimental en el PLC se ha obtenido el siguiente resultado:

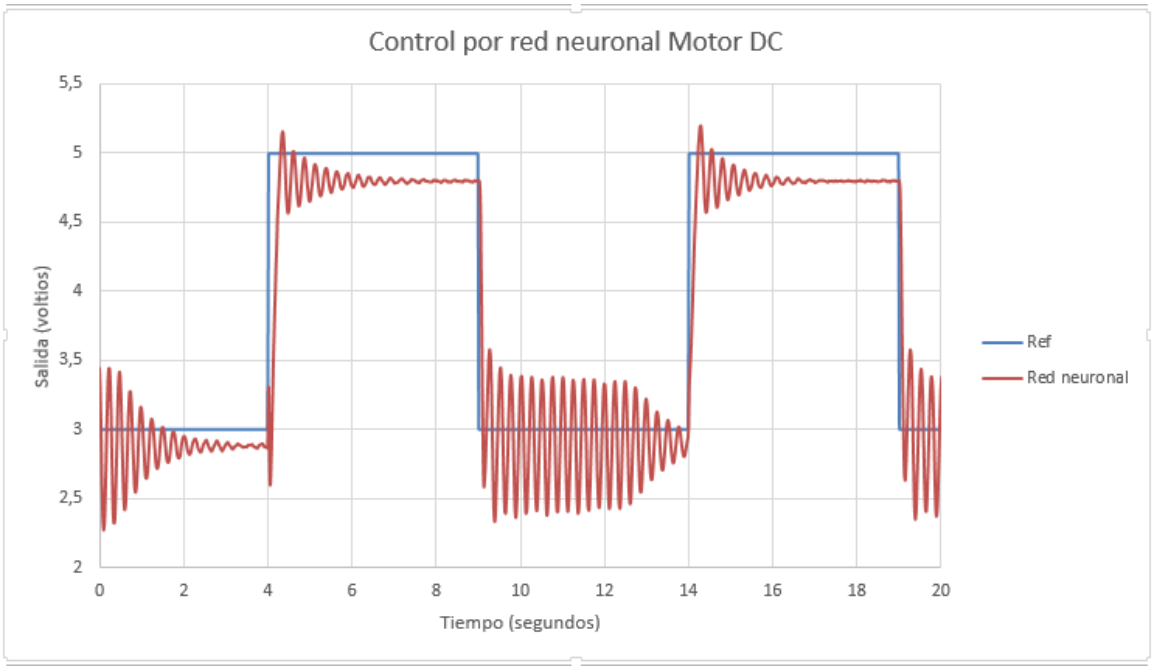

Figura 43. Resultado experimental red neuronal motor DC

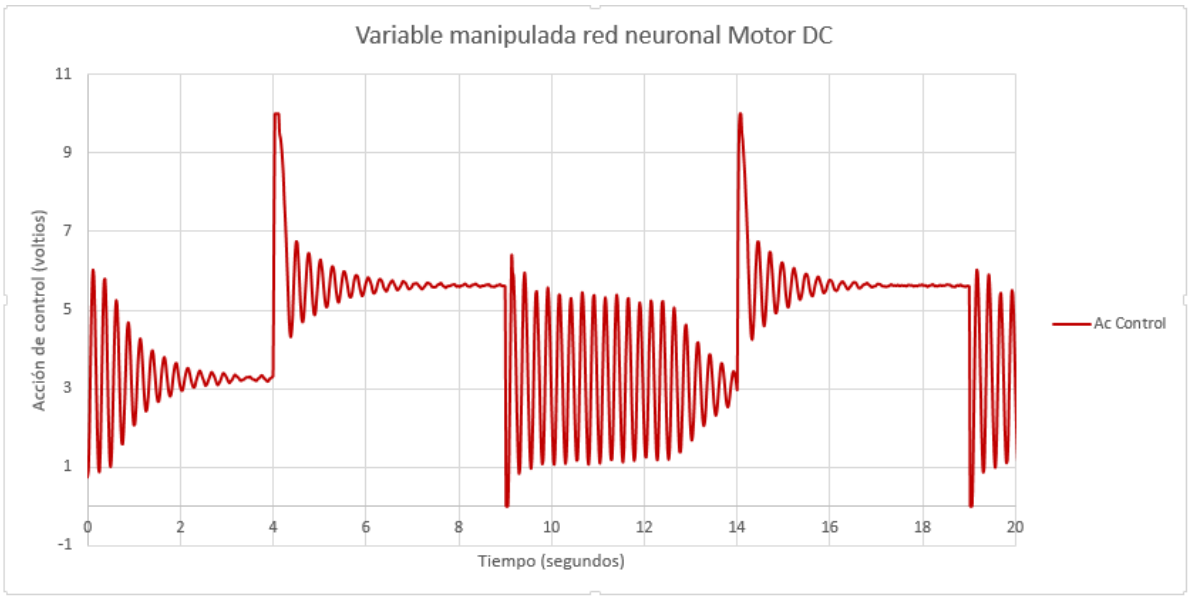

Figura 44. Acción de control controlador red neuronal motor DC

Con este controlador se tienen una amplitud de oscilaciones que llegan a 1.3 voltios de amplitud en el caso de la referencia del valor 3, y una amplitud de 0.7 en el la referencia de valor 5. Sin embargo, estas oscilaciones se van reduciendo en régimen permanente llegando a estabilizarse. Cuando se estabiliza en el escalón de unidad 5, el error de referencia es del 4 %.

Este resultado puede deberse a que se haya producido “overfitting” o sobreajuste durante el entrenamiento de la red. El algoritmo de control se ha alineado demasiado con los cambios de referencia, obteniendo una respuesta agresiva del control. Esto puede haber sido causado por un excesivo número de neuronas en la capa oculta o una mala elección en los datos de entrenamiento para este caso.

5.1.3. Comparación global

Por último, se va a comparar el resultado experimental obtenido por los controladores avanzados con un controlador PID discreto obtenido mediante el método de asignación de polos. Los polos se han posicionado en 0.85 priorizando la estabilidad de la respuesta del sistema.

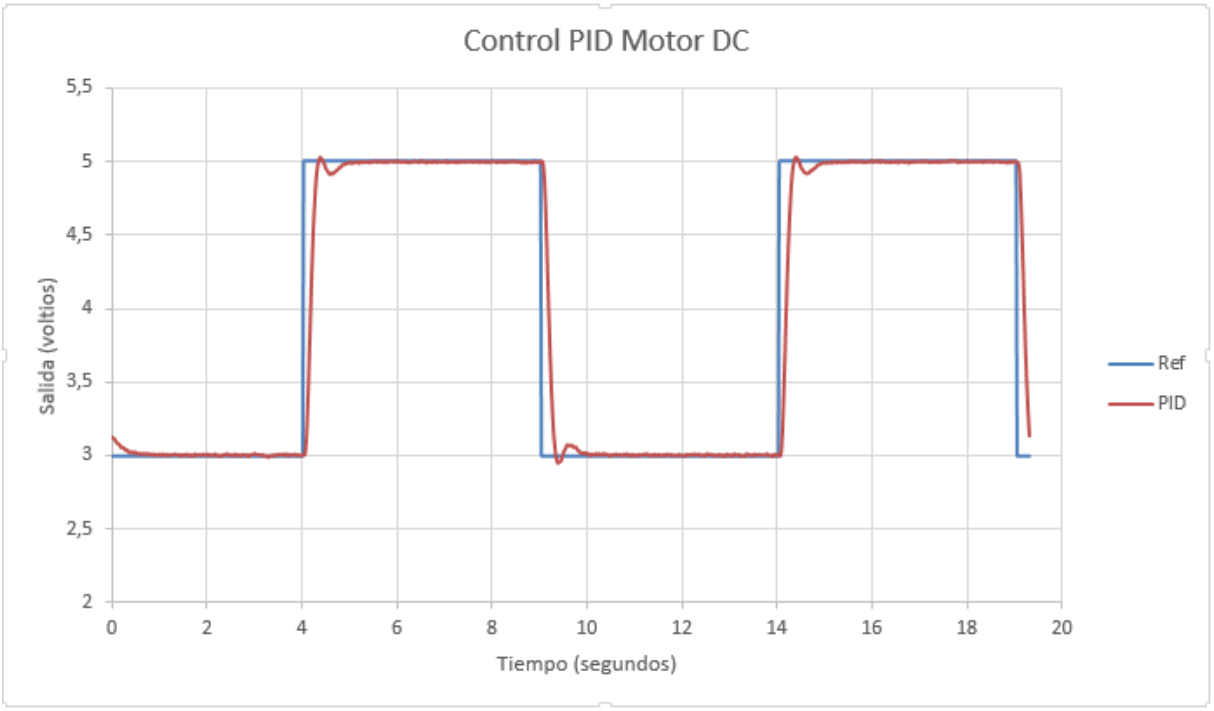

Figura 45. Respuesta experimental controlador PID motor DC

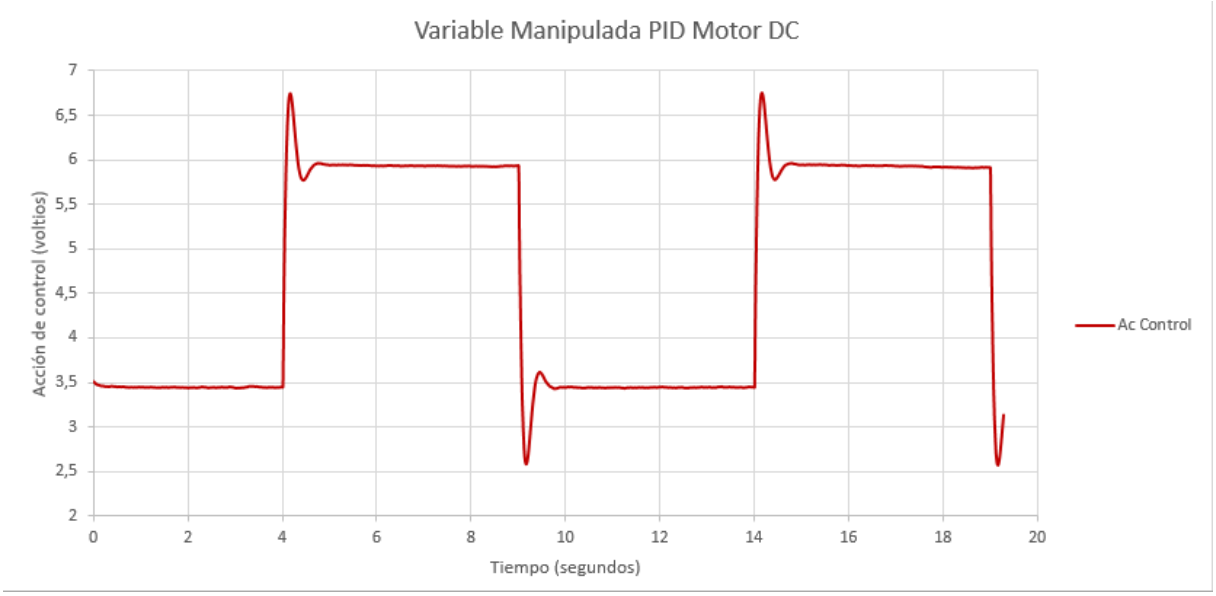

Figura 46. Acción de control PID motor DC

A continuación, se muestra en la Figura 47 la respuesta de los tres tipos de controladores.

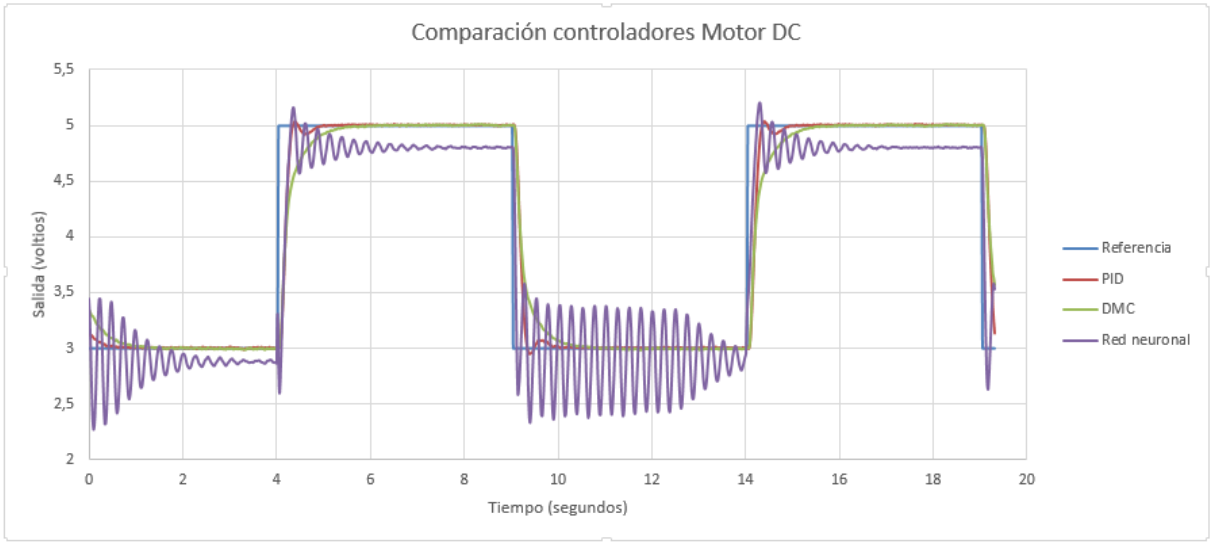

Figura 47. Comparación respuesta experimental motor DC

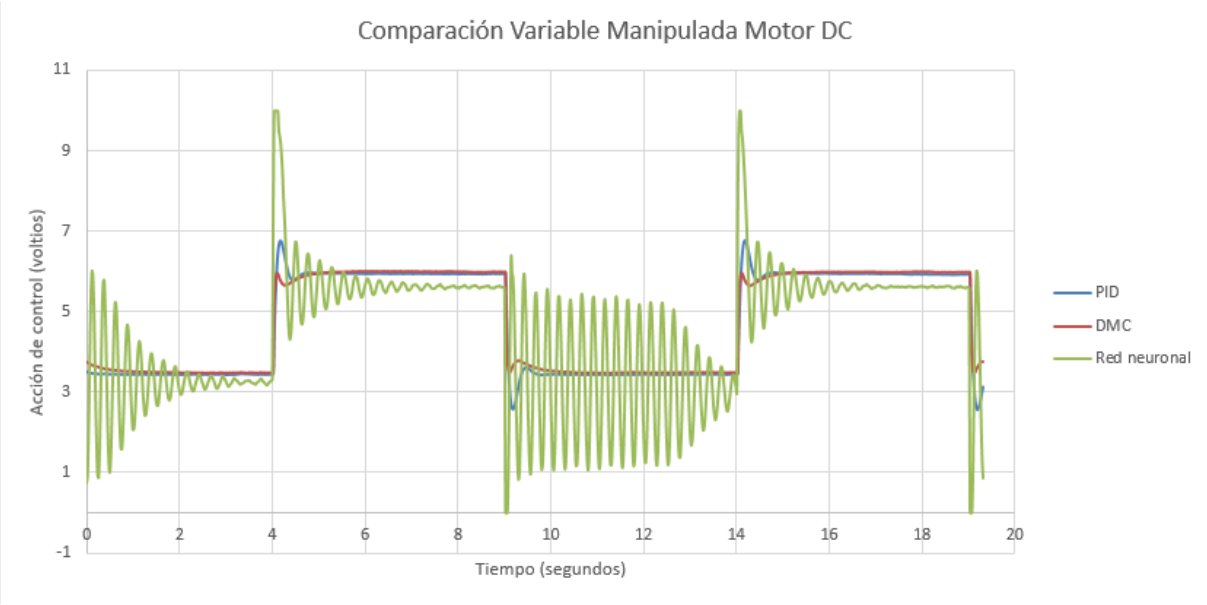

Figura 48. Comparación acción de control experimental motor DC

En el caso del motor DC, el controlador PID ha sido el que ha conseguido un tiempo de establecimiento menor, pero el controlador DMC no tiene sobreoscilación. En el error en régimen permanente, no hay diferencias significativas entre ambos controladores.

5.2. Horno

5.2.1. Controlador DMC

En la simulación mostrada en la Figura 23 se ha obtenido un tiempo de establecimiento de 84.16 segundos sin sobreoscilación. Para estos valores solo se ha tenido en cuenta los escalones positivos, debido a que la limitación en la acción de control impide acelerar el enfriamiento del horno. El resultado obtenido experimentalmente es el siguiente:

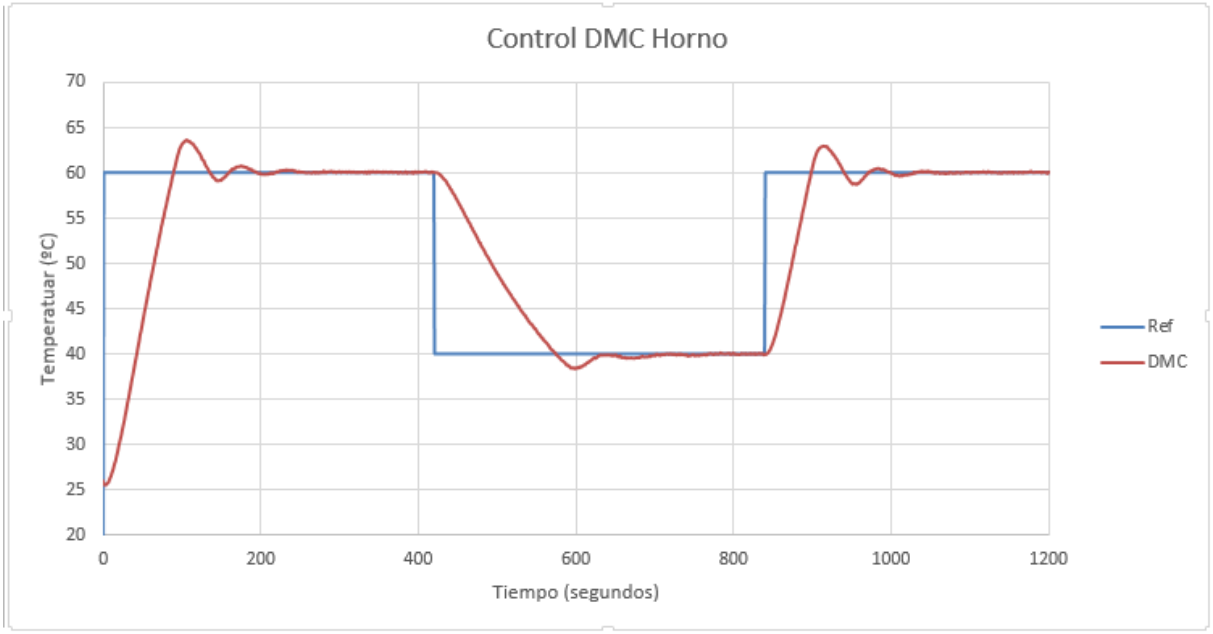

Figura 49. Respuesta experimental controlador DMC horno

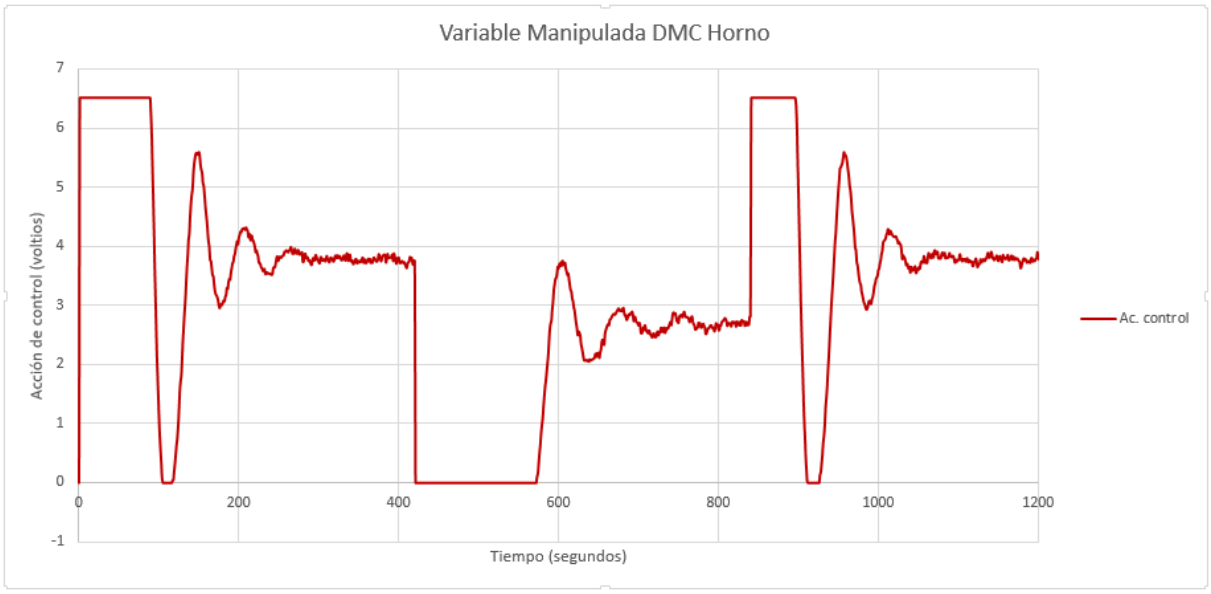

Figura 50. Acción de control experimental DMC horno

El tiempo de establecimiento ha pasado a ser de 128 segundos, debido a que el sistema presenta sobreoscilaciones, llegando a ser la más alta del 7.83 %. Las oscilaciones tienden a estabilizarse y no se presenta error en régimen permanente.

5.2.2. Controlador red neuronal

En la simulación, visible en la Figura 30, el tiempo de establecimiento se ha mantenido en 84 segundos. El seguimiento no presenta sobreoscilación, pero tiene un error en régimen permanente de 0.45 %.

En el control experimental se han producido sobreoscilaciones del 4 % y la acción de control es oscilante durante el régimen permanente, además de que el error de posición ha pasado a ser de un 1.6 %. El resultado se puede ver en las siguientes figuras:

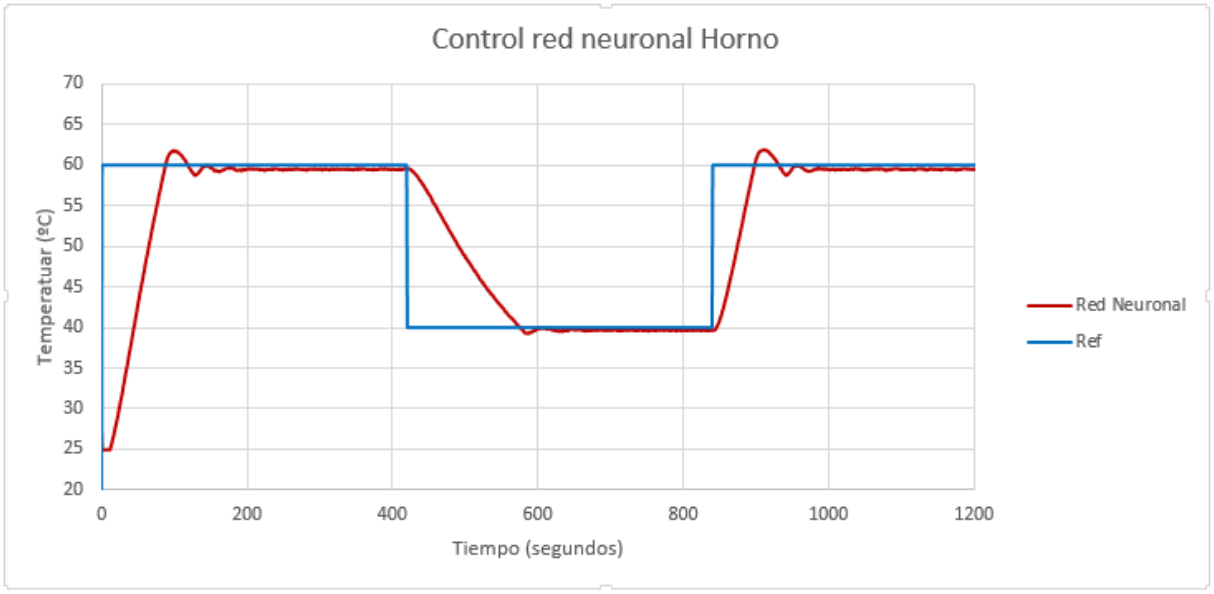

Figura 51. Respuesta experimental red neuronal horno

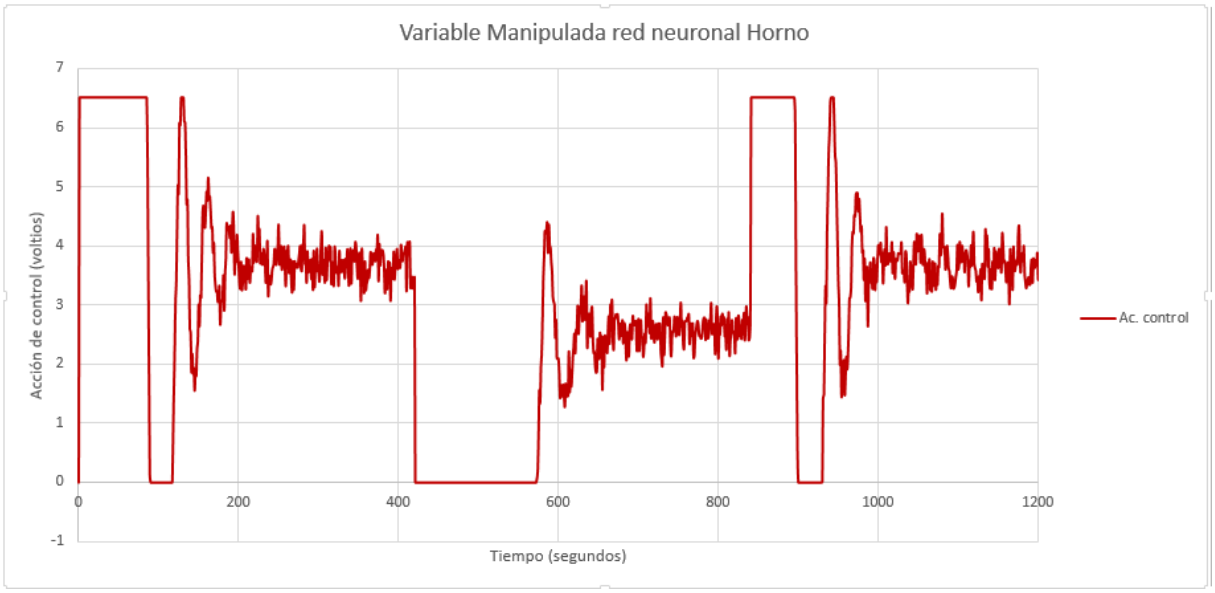

Figura 52. Acción de control experimental red neuronal horno

5.2.3. Comparación global

Como se ha realizado anteriormente para el motor, se van a comparar los resultados experimentales con los obtenidos por un PID discreto cuyos polos han sido asignados en 0.6. Su resultado ha sido el siguiente:

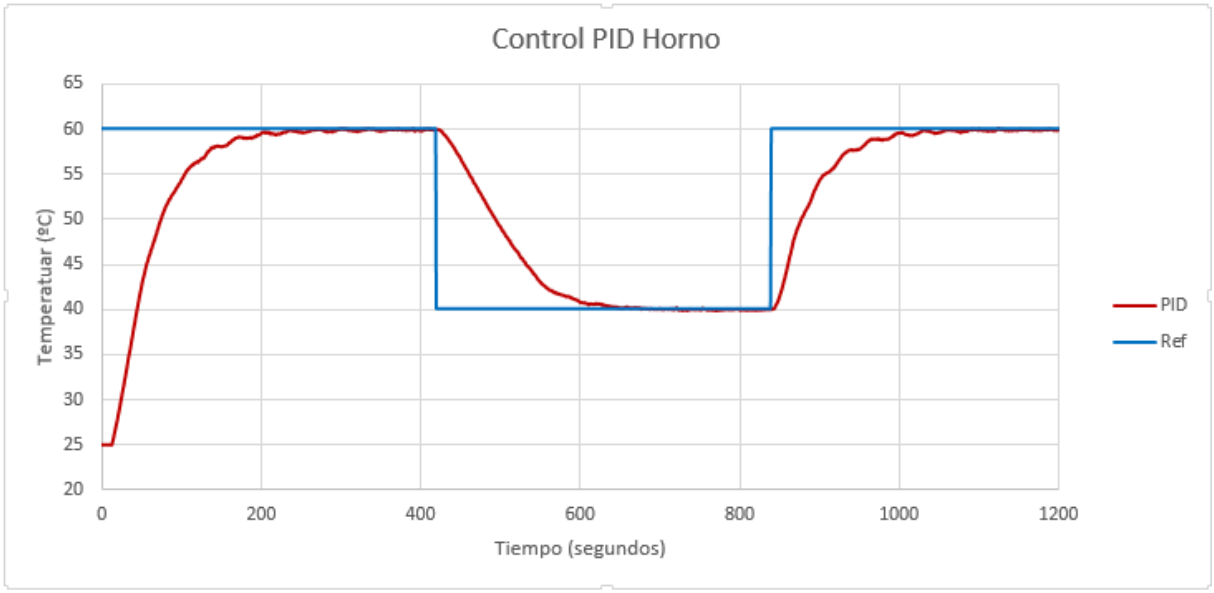

Figura 53. Resultado experimental controlador PID horno

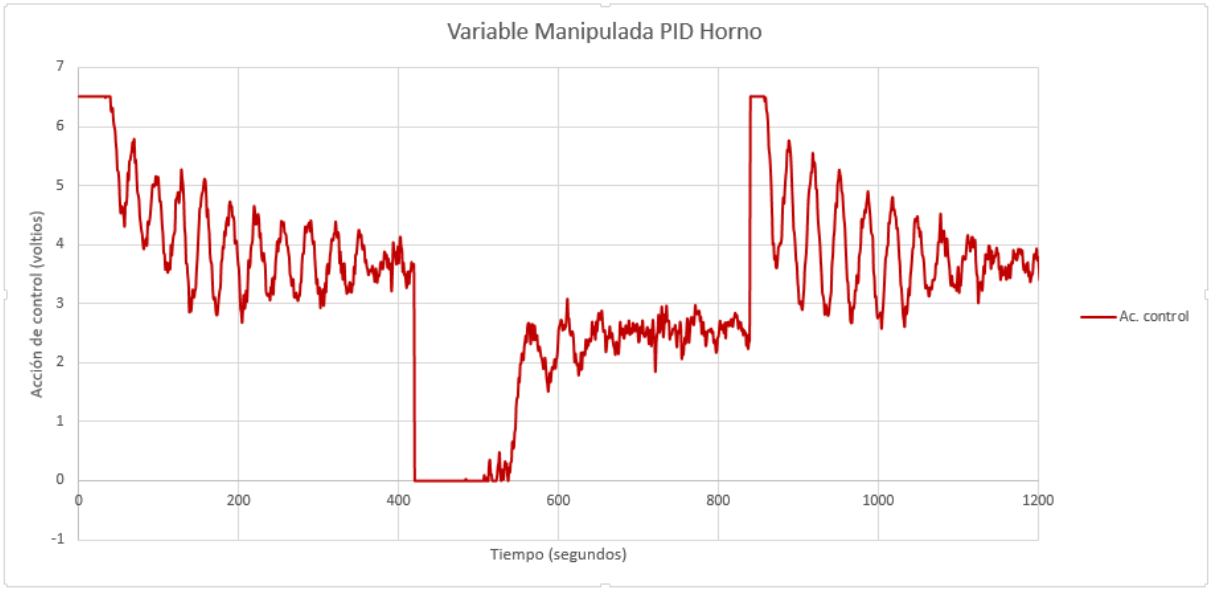

Figura 54. Acción de control PID horno

Este controlador ha tenido un tiempo de establecimiento de 209 segundos. Aunque esta respuesta haya sido lenta no se pudo aumentar la cercanía de los polos a cero, debido a que la respuesta se hacía cada vez más oscilante.

A continuación, se muestra la comparativa entre los tres controladores:

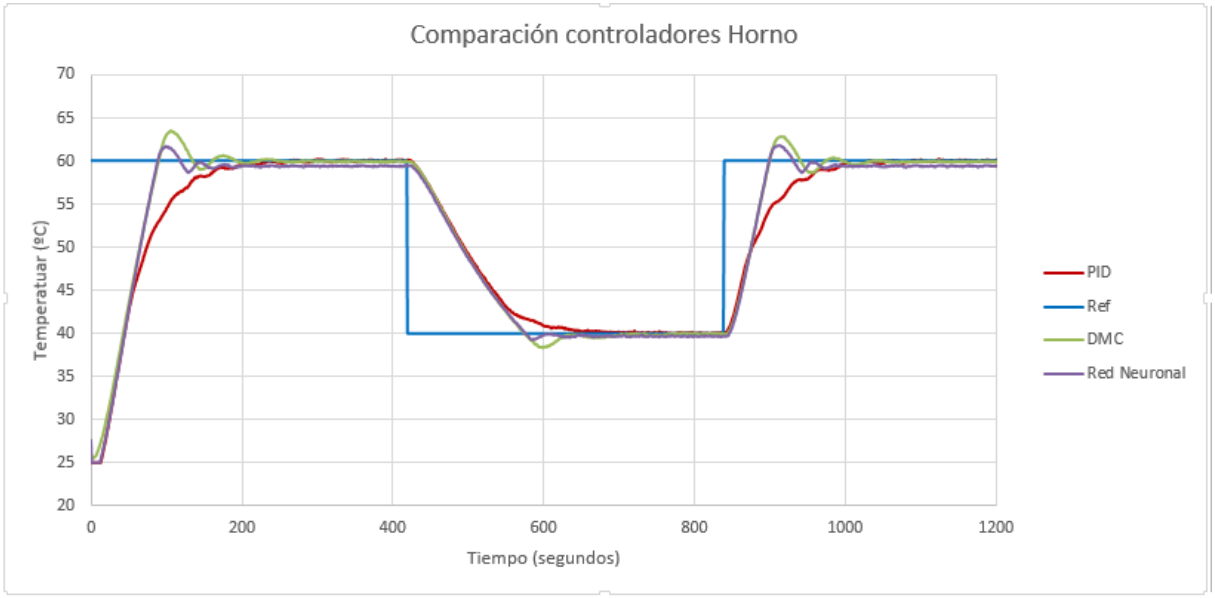

Figura 55. Comparación respuesta experimental horno

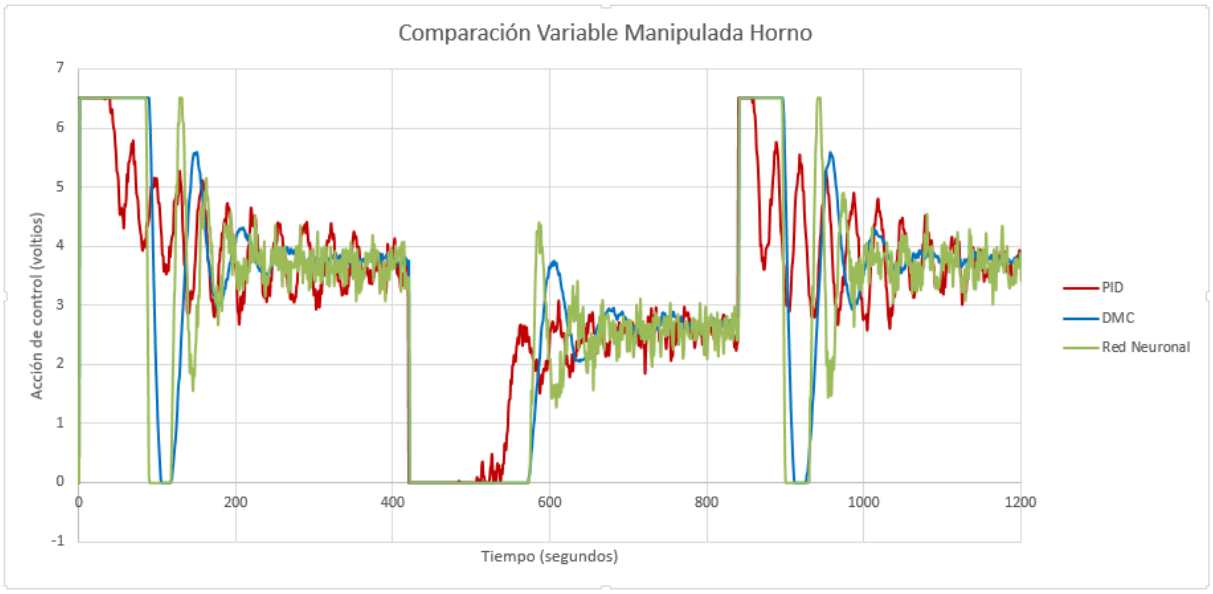

Figura 56. Comparación acción de control experimental horno

A diferencia del control del motor, los controladores avanzados han obtenido respuestas similares, siendo ambos más rápidos que el PID. Aunque el control por red neuronal se estabilice antes y tenga menos oscilaciones, presenta un error en régimen permanente que el DMC no tiene.

.

6. CONCLUSIONES Y TRABAJOS FUTUROS

A continuación se van a enumerar las conclusiones obtenidas durante el desarrollo del trabajo fin de máster así como propuestas para mejorar el trabajo realizado en futuros proyectos.

Mediante el trabajo realizado se concluye que se ha conseguido desarrollar una metodología que permite la implementación de algoritmos de control avanzados en un PLC real, obteniendo pequeñas diferencias entre simulación y control real, excepto en el caso del control por red neuronal del motor. Además, se ha utilizado el mismo método en ambos controladores, por lo que seguramente aplicable a otro tipo de algoritmos de control.

Durante el desarrollo del control DMC, se encontró el problema de no poder aplicar restricciones duras debido a que Simulink PLC Coder no es compatible con la toolbox de programación cuadrática de MATLAB. Sería interesante averiguar cómo poder solventar esta problemática.

La red neuronal utilizada tiene una estructura similar a un controlador PID. Existen otras redes neuronales, como las redes NARX (nonlinear autoregressive network with exogenous inputs) o NARMA-L2 (nonlinear autoregression moving average) que podrían probar a ser implementadas en un PLC usando el método planteado.

7. BIBLIOGRAFÍA

[1] Eduardo F. Camacho y Carlos Bordons. “Control predictivo: pasado, presente y futuro”, *Revista Iberoamericana de Automática e Informática Industrial,* Vol. 1, N. 3, Octubre 2004.

[2] Manzanera C. (2017): *Reglas de Diseño para la Sintonía de Controladores Predictivos Dynamic Matrix Control (DMC),* Tesis doctoral, Universidad Politécnica de Cartagena, Cartagena.

[3] Martínez Iranzo, M.A. (2021). *Apuntes de clase de la asignatura control predictive e inteligente*, Universidad Politécnica de Valencia.

[4] Serrano, Antonio J.; Soria, Emilio; Martín; José D. (2009). *Redes neuronales artificiales,* Programa 3er ciclo (doctorado), Universidad de Valencia.

[5] phoenixcontact.com (2023). *PLC Next technology.* [online] Available at: https://www.phoenixcontact.com/es-es/industrias/plcnext-technology

[6] es.mathworks.com (2023). Simulink PLC Coder. [online] Available at: https://es.mathworks.com/products/simulink-plc-coder.html

[7] es.mathworks.com (2023). Simulink PLC Coder limitations. [online] Available at: https://es.mathworks.com/help/plccoder/ug/structured-text-code-generation-limitations.html

[8] es.mathworks.com (2023). DMC Simulink Block and Example. [online] Available at: https://es.mathworks.com/matlabcentral/fileexchange/25412-mpc-tutorial-iv-dmc-simulink-block-and-example