ACTIVIDAD ADELANTO NOTA (CONTROL II) - INFORME 1 MIACON | MÓDULO 2: Identificación y Control de una Planta de Primer Orden [1]

Diego Andrés García Díaz - 2195533
diego2195533@correo.uis.edu.co
Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones
Universidad Industrial de Santander

Octubre 05, 2025

Resumen

En este informe se documenta el proceso de identificación y control de un sistema térmico de primer orden, conformado por un heater como actuador y un sensor DHT11 conectado a una ESP32 para la adquisición de datos. Se realizaron diferentes pruebas experimentales, procesamiento de datos en MATLAB, identificación de la planta mediante la System Identification Toolbox y validación en Simulink. Finalmente, se diseñaron y compararon estrategias de control ON/OFF, PWM, P, PD, PI y PID.

Palabras clave: Controladores, ESP32, Primer Orden, Matlab, Simulink, System Identification Toolbox.

1. Introducción

El control de procesos térmicos constituye una aplicación fundamental en ingeniería de control, ya que aparece en múltiples ámbitos industriales como hornos, intercambiadores de calor y dispositivos electrónicos. Estos procesos se caracterizan por poseer inercia térmica y pérdidas de energía al ambiente, lo que genera una dinámica lenta y con retardo. Un sistema de calentamiento se modela habitualmente como una planta de primer orden con retardo, lo cual lo hace ideal para ejercicios de identificación y diseño de controladores básicos y clásicos. Dicho modelo permite analizar el comportamiento transitorio mediante parámetros como la ganancia, la constante de tiempo y el tiempo muerto o retardo, que son esenciales para la sintonización de estrategias de control. Además, este tipo de sistemas ofrece un entorno didáctico seguro y representativo para comparar distintas metodologías de control (ON/OFF, PWM o PID), evaluando su desempeño en términos de estabilidad, error en estado estacionario y robustez frente a perturbaciones. [2] [3]

Este trabajo busca cerrar el ciclo completo: desde el montaje físico (heater + DHT11 + ESP32), pasando por la adquisición de datos y su procesamiento, hasta la identificación matemática del sistema y la validación de controladores en Simulink, comprendiendo como responde un sistema real ante señales de control ya que a diferencia de un sistema isntantáneo, una planta térmica no responde de forma inmediata, lo que hace crucial predecir su comportameinto con un modelo matemático. [1] [4]

2. Objetivos

2.1. Objetivo General

Entender el comportamiento dinámico de un sistema térmico (planta de primer orden) y aplicar herramientas de MATLAB/Simulink para su modelado y diseño de controladores.

2.2. Objetivos Específicos

1. Realizar el montaje experimental del sistema, usar un microcontrolador **ESP32** para adquisición de datos, control en tiempo real e integración con Matlab/Simulink para posterior identifiación del modelo de la plnata.
2. Implementar la estrategia de control **PWM**, para identificar la planta con el software Matlab/Simulink.
3. Aplicar la herramienta **System Identification Toolbox** para la identificación del modelo de un Sistema Térmico de Primer Orden.
4. Diseñar y probar controladores P/PD/PI/PID en el entorno de simulación de Matlab/Simulink.

3. Marco Teórico

3.1. Control PWM

El control por PWM (Pulse Width Modulation) es una forma de controlar la potencia entregada a un actuador como un heater. En esencia, no se trata de un simple encendido y apagado rápido, sino de una técnica que, al variar el ciclo de trabajo de una señal cuadrada, permite regular la potencia de forma equivalente a una señal analógica continua. Esta técnica es muy eficiente y es ampliamente utilizada en la industria para motores, iluminación y climatización. [5]

3.2. Control PID

El controlador PID (Proporcional–Integral–Derivativo) es una de las estrategias de control más empleadas en la industria debido a su simplicidad y efectividad para regular sistemas lineales y no lineales. Su acción de control se compone de tres términos:

- **Acción proporcional (P):** responde de manera inmediata al error actual, proporcionando una corrección proporcional a la diferencia entre el valor de referencia y la variable medida. Su efecto es reducir la magnitud del error, aunque no lo elimina completamente.
- **Acción integral (I):** acumula el error a lo largo del tiempo y ajusta la salida del controlador en consecuencia. Gracias a esta acción, es posible **eliminar el error en estado estacionario**, algo que los controladores P o PWM simples no logran.
- **Acción derivativa (D):** anticipa el comportamiento futuro del error al considerar su tasa de cambio. De este modo, mejora la estabilidad del sistema y reduce la tendencia a oscilar, aunque también lo hace más sensible al ruido de la señal de medición.

La ecuación general del controlador PID en el dominio del tiempo es:

$$u(t) = K_p e(t) + K_i \int e(t)\,dt + K_d \frac{de(t)}{dt} \tag{1}$$

donde $u(t)$ es la señal de control, $e(t)$ el error de seguimiento y K_p, K_i, K_d los parámetros de sintonización.

En el dominio de Laplace, el controlador PID puede expresarse como:

$$C(s) = K_p + \frac{K_i}{s} + K_d s \tag{2}$$

3.2.1. Forma discreta del PID

Cuando se implementa en un microcontrolador o computadora digital, el PID debe expresarse en forma discreta. Una representación común utilizando el método de diferencias hacia atrás es:

$$\begin{aligned} u[k] = u[k-1] &+ K_p\big(e[k] - e[k-1]\big) \\ &+ K_i T_s\, e[k] \\ &+ \frac{K_d}{T_s}\big(e[k] - 2e[k-1] + e[k-2]\big) \end{aligned} \tag{3}$$

donde:

- $u[k]$ es la salida de control en el instante k,
- $e[k]$ es el error en el instante k,
- T_s es el tiempo de muestreo.

Este esquema permite implementar el controlador PID en plataformas como Arduino, ESP32 o en simulaciones discretas en `Simulink`, garantizando un comportamiento similar al controlador continuo siempre que el tiempo de muestreo T_s sea suficientemente pequeño.

3.2.2. Consideraciones en procesos térmicos

En el caso de procesos térmicos, el PID es ampliamente utilizado porque compensa el error estacionario que dejan los controladores P o PWM. Sin embargo, debe considerarse el **retardo inherente de la planta**, ya que un ajuste inadecuado puede generar oscilaciones o inestabilidad. El método de ajuste de Ziegler–Nichols es de uso frecuente para encontrar un homogeneidad entre rapidez de respuesta, error estacionario y robustez.

3.3. Modelado de un Sistema de Primer Orden

La planta térmica se modela como un sistema de primer orden con retardo. Para identificarlo, se puede utilizar el método de Dos Puntos de Smith. Este método permite obtener los parámetros T (constante de tiempo) y L (tiempo muerto o retardo) a partir de la respuesta del sistema a un escalón. Con la gráfica obtenida, se miden los tiempos t1 y t2 y se calculan los parámetros de la planta. [6] [7]

$$G_p(s) = \frac{K_p}{T_{p1} \cdot s + 1} \cdot e^{-T_d \cdot s} \tag{4}$$

$$K = K_p = \text{Ganancia} \tag{5}$$

$$K = \frac{\Delta Y(\text{salida})}{\Delta U(\text{entrada})} \tag{6}$$

$$T = T_{p1} = \text{Constante de tiempo} \tag{7}$$

$$T = \frac{3}{2}(t_2 - t_1) \tag{8}$$

$$L = T_d = \text{Tiempo muerto (Delay)} \tag{9}$$

$$L = t_2 - T \tag{10}$$

$$t_1 = L + \frac{1}{3} \cdot T; \quad t_2 = L + T \tag{11}$$

3.4. Integración de Sensores Personalizados en Simulink

La integración de sensores y periféricos que requieren librerías de terceros, como el DHT11, se logra utilizando la aplicación **IO Device Builder**, una capacidad disponible desde la versión 2023b del Simulink Support Package for Arduino. Esta herramienta permite a los usuarios diseñar y crear un bloque personalizado en Simulink que interactúa directamente con la placa Arduino y/o ESP32. El proceso de integración comienza con dos prerrequisitos esenciales: la instalación del **Simulink Support Package for Arduino** y la obtención de los archivos de librería de terceros del sensor (ej. de GitHub), los cuales deben ser incluidos como fuente. Además, el modelo de Simulink debe ser configurado previamente para el hardware **Arduino Uno y/o ESP32**.

Una vez iniciada la aplicación, el usuario define las especificaciones del bloque. Esto incluye establecer un nombre (ej. DHT11) y descripción, configurar parámetros ajustables (ej. 'dataDelay') que pueden ser modificados durante la ejecución, y declarar las **salidas del sensor**, como Temperatura y Humedad. El proceso de generación resulta en la creación de un *system object* y un archivo C++. El paso más crítico es la modificación manual del archivo C++, donde se especifica el pin de conexión del sensor (ej. pin digital del Arduino y/o ESP32), se inicializa la comunicación serial y el sensor en la función 'setup', y se implementa el código en la función 'step' para *leer continuamente los datos y asignarlos a los puertos de salida* del bloque. Finalmente, el bloque personalizado puede ser insertado en el modelo de Simulink para visualizar los datos en tiempo real. [8]

4. Metodología

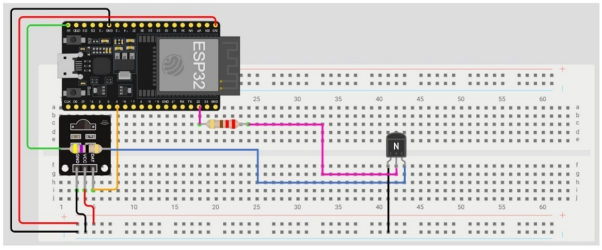

Fig. 1: Circuito inicial, con conexión erronea del transistor

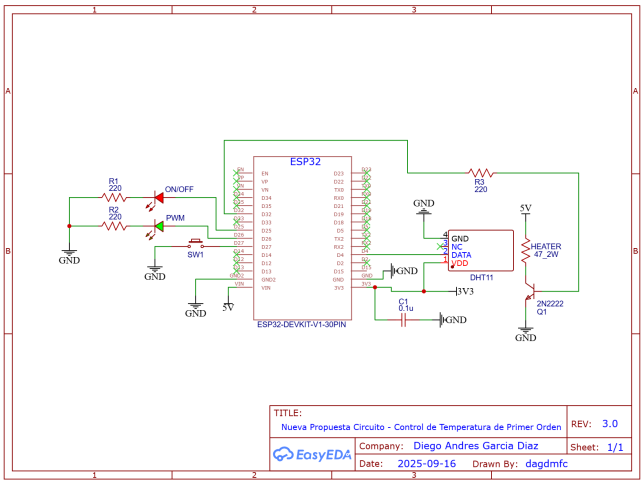

Fig. 2: Nueva propuesta, Diego Garcia

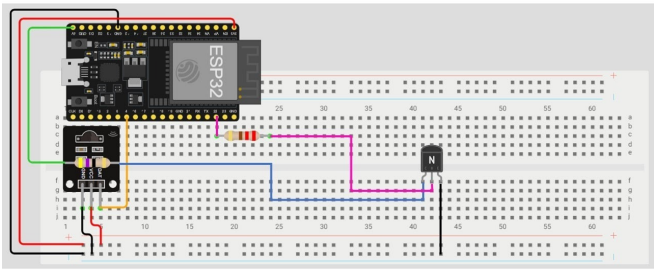

Fig. 3: Circuito modificado luego de validación de la nueva propuesta

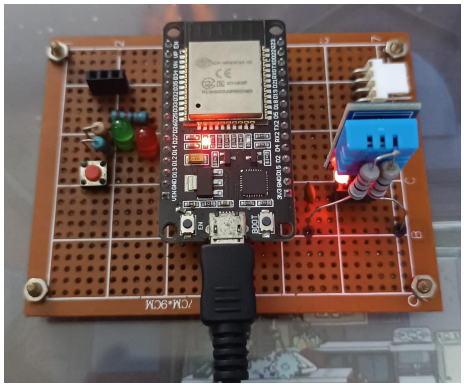

Fig. 4: Montaje experimental

4.1. Reconocimiento de la Planta para obtención y procesamiento de datos en MATLAB/SIMULINK

Se usó el montaje experimental de la Figura 4, además se siguieron las instriccuiones de la página web desarrollada por MIACON [1] con el fin de obtener el modelo que representa el sistema y lograr hacer el respectivo disñeo de los controladores. Se realizó una simulación en **Simulink** a partir de una configuración de diagramas de bloques ejerciendo un control PWM con $K = 2{,}55$ (ver Figura 5). El resultado de la simulación se puede observar en la Figura 6.

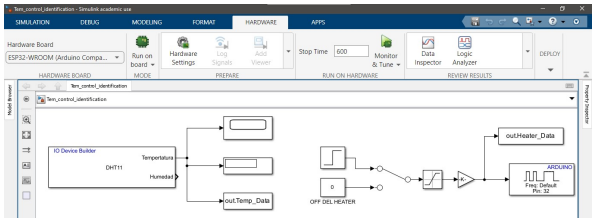

Fig. 5: Obtención de datos con Simulink

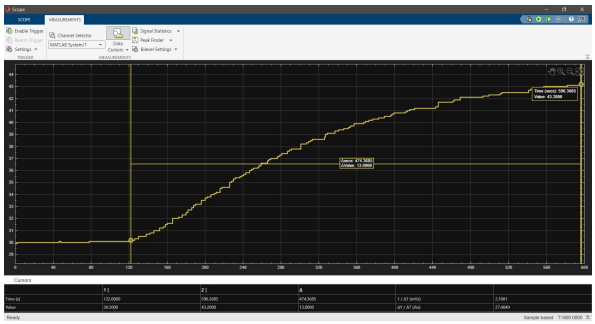

Fig. 6: Simulación de 600 $[s]$ para obtención y tratamiento de datos

5. Identificación del Sistema

Con los datos preprocesados, se abrió el **System Identification Toolbox** en MATLAB (ver Figura 7). Se importaron los datos (***Heater_Data.signals.values*** como *Input* y y como *Output*), estableciendo el tiempo de muestreo en 1 segundo. El modelo se estimó utilizando la opción Process Models, seleccionando la opción con un polo y retardo. El ajuste del modelo fue mayor al 80 %, lo que validó su confiabilidad. Los parámetros obtenidos son los que se observan en la ecuación (4) y el resultado del modelo obtenido se observa en la ecuación (12). [9]

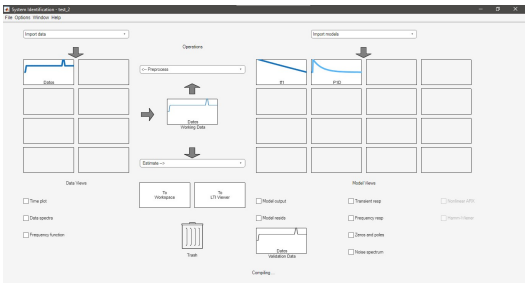

Fig. 7: Uso del **System Identification Toolbox** de Matlab

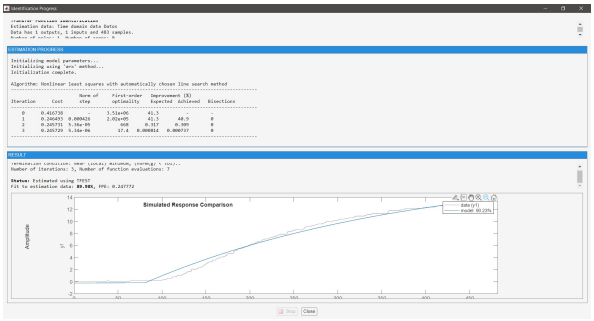

Fig. 8: Identificación de la planta $P(s)$ en Matlab.

6. Resultados y Validación

6.1. Modelo de la Planta

Se obtuvo la función de transferencia de primer orden con retardo que describe el comportamiento del sistema (ver ecucación (12).

$$G_P(s) = P(s) = \frac{0{,}11864}{170{,}65 \cdot s + 1} \cdot e^{-30 \cdot s} \qquad (12)$$

$$G_p(s) = P(s) = \text{Actuador} + \text{Planta} + \text{Sensor} \qquad (13)$$

6.2. Validación y resultados usando Simulink

Una vez obtenida la función de transferencia que modela el sistema o planta (ver ecuación (12)), se realizó la respectiva validación en Simulink, se puede observar en la Figura 9. También se hizo uso de método de los dos puntos de Smith para generar la respectiva función de transferencia que se puede ver en la ecuación (14).

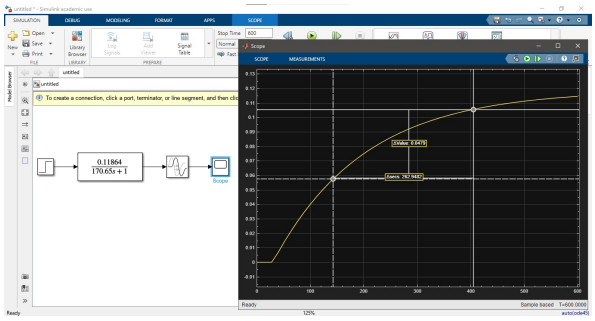

Fig. 9: Validación en Simulink del modelo identificado

$$G_p(s) = \frac{0{,}1144}{158{,}7423 \cdot s + 1} \cdot e^{-31{,}7485 \cdot s} \tag{14}$$

$$t_1 = L + \frac{1}{3} \cdot T = 84{,}6626\,[s]; \quad t_2 = L + T = 190{,}4908\,[s] \tag{15}$$

$$\text{Ganancia} \rightarrow K = \frac{\Delta Y(\text{salida})}{\Delta U(\text{entrada})} = 0{,}1144 \tag{16}$$

$$\text{Constante de tiempo} \rightarrow T = \frac{3}{2}(t_2 - t_1) = 158{,}7423\,[s] \tag{17}$$

$$\text{Retardo} \rightarrow L = t_2 - T = 31{,}7485\,[s] \tag{18}$$

Finalmente, con el modelo ya identificado (ver ecuación (12)) el siguiente paso es el diseño de los diferentes controladores, como referencia se puede tener el diagrama de blques de la Figura 10.

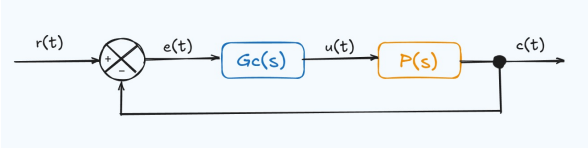

Fig. 10: Próximos pasos para el diseño de los controladores

7. Recomendaciones

1. Probar frecuencias PWM en el rango 500–2000 Hz para observar diferencias en la respuesta térmica.
2. Considerar sensores más rápidos o con mejor resolución (ej. DHT22 o termistor con ADC).
3. Explorar controladores P/PD/PI/PID para eliminar error estacionario.
4. Sobre el diseño de la página (Google Sites), falta aclarar un poco más ciertas secciones, específicando un poco más ciertos pasos con el fin de aclarar más el proceso a realizar. Opcionalmente se puede agregar un vídeo explicativo.

8. Conclusiones

- En el circuito de la Figura 2 se propuso la incorporación de LEDs como indicadores del modo de control activo y un pulsador para permitir la conmutación entre dichos modos. Esta implementación se planteó con el objetivo de desarrollar un código en Arduino IDE que integre los controladores diseñados, facilitando su validación en un entorno alternativo a Matlab y Simulink. Cabe resaltar que esta propuesta es opcional, dado que la guía y el proyecto MIACON están orientados principalmente al uso de Matlab y Simulink, donde se propone el circuito de la Figura 3, el cual fue el implementado en las diferentes pruebas que se realizaron.
- La configuración en emisor común del transistor 2N2222 resulta la adecuada para el control del heater (ver Figuras 2 y 3), ya que garantiza un camino de corriente correcto desde la fuente de alimentación, a través de la carga, hasta tierra. De esta forma, el transistor puede funcionar como interruptor controlado por la señal PWM proveniente de la ESP32, permitiendo regular el calentamiento del sistema. En contraste, una conexión con el colector directamente a tierra no establece el flujo adecuado y, por tanto, impide el funcionamiento esperado del control térmico (ver Figura 1). Finalmente, en la Figura 4 se observa montaje experimental para las pruebas con Matlab/Simulink.
- La temperatura del sistema aumenta de manera gradual ante un cambio de potencia, siguiendo una dinámica exponencial definida por la constante de tiempo térmica. El comportamiento no es instantáneo debido a la inercia térmica y a los procesos

de transferencia de calor entre el heater, el aire y el sensor. Su evolución puede predecirse mediante un modelo matemático de primer orden con retardo (4), que permite estimar parámetros como ganancia, constante de tiempo y tiempo muerto o retardo.

- Validar el sensor antes de diseñar el controlador es indispensable, ya que garantiza que las mediciones sean confiables y representen con precisión el estado del sistema; de lo contrario, el controlador operaría con información errónea, comprometiendo el desempeño y la estabilidad del lazo de control.
- La planta térmica se comporta como un sistema de primer orden con retardo, lo cual es típico en sistemas con inercia.
- El PWM es una forma de control eficiente que ofrece una alternativa superior al control ON-OFF.
- La Toolbox de Matlab y el método de Dos Puntos de Smith permiten una simplificación efectiva de la planta para facilitar el diseño de controladores.
- La identificación en MATLAB/Simulink proporciona un modelo válido (ajuste >80 %).
- Finalmente, se identificó el modelo de primer orden y se obtuvo su correspondiente función de transferencia para el sistema de control de temperatura, aplicando para ello el método de los Dos Puntos de Smith.

Referencias

[1] MIACON, "Módulo 2: Identificación y control de una planta de primer orden," 2025. [Online]. Available: https://sites.google.com/view/miacon-proyectodegrado/gu%C3%ADas-de-laboratorio/m%C3%B3dulo-2-identificaci%C3%B3n-y-control-de-una-planta-de-primer-orden

[2] S. A. C. Giraldo, "Modelo matemático de un sistema térmico." [Online]. Available: https://controlautomaticoeducacion.com/analisis-de-sistemas/modelo-matematico-de-un-sistema-termico/

[3] S. C. Giraldo, "Retardo o tiempo muerto de un sistema." [Online]. Available: https://controlautomaticoeducacion.com/analisis-de-sistemas/retardo-o-tiempo-muerto-de-un-sistema/

[4] U. del País Vasco, "Régimen transitorio." [Online]. Available: https://ocw.ehu.eus/file.php/83/cap6_html/capitulo-6.html#s60

[5] L. M. Engineers, "Esp32 basics: Generating a pwm signal on the esp32." [Online]. Available: https://lastminuteengineers.com/esp32-pwm-tutorial/

[6] J. J. C. Zagarra, "Método de curva de reacción o de dos puntos de smith," 2023. [Online]. Available: https://www.youtube.com/watch?v=_6ywp9ryB-Q

[7] L. F. Rodriguez, "Sintonización pid: Método de smith y corripio," 2020. [Online]. Available: https://www.youtube.com/watch?v=8sQChijpKgA

[8] MATLAB, "How to build custom sensor blocks for arduino in simulink," 2024. [Online]. Available: https://www.youtube.com/watch?v=ZdgMAcMhQ-Q

[9] C. de Ayuda de MATLAB, "System identification toolbox." [Online]. Available: https://la.mathworks.com/help/ident/getting-started-1.html