Vol. 17, no. 1, pp. 81-92, enero-junio 2018
Revista UIS Ingenierías
Página de la revista: revistas.uis.edu.co/index.php/revistauisingenierias

Experimentación de controladores digitales clásicos en un sistema embebido aplicado en un proceso térmico

Experimentation of classic digital controllers in an embedded system applied in a thermal process

José Fuentes [1], Sergio Castro[2], Byron Medina[3], Francisco Moreno[4], Sergio Sepúlveda[5]

[1]Grupo de investigación y Desarrollo en Electrónica y Telecomunicaciones (GIDET), Departamento de electricidad y electrónica, Universidad Francisco de Paula Santander, Colombia. Email: josefernandofr@ufps.edu.co
[2]Grupo de investigación y Desarrollo en Electrónica y Telecomunicaciones (GIDET), Departamento de electricidad y electrónica, Universidad Francisco de Paula Santander, Colombia. Email: sergio.castroc@ufps.edu.co
[3]Grupo de investigación y Desarrollo en Electrónica y Telecomunicaciones (GIDET), Departamento de electricidad y electrónica, Universidad Francisco de Paula Santander, Colombia. Email: byronmedina@ufps.edu.co
[4] Grupo de investigación y Desarrollo en Electrónica y Telecomunicaciones (GIDET), Departamento de electricidad y electrónica, Universidad Francisco de Paula Santander, Colombia. Email: femgarcia@ufps.edu.co
[5]Grupo de investigación y Desarrollo en Electrónica y Telecomunicaciones (GIDET), Departamento de electricidad y electrónica, Universidad Francisco de Paula Santander, Colombia. Email: sergio.sepulveda@ufps.edu.co

RECIBIDO: Mayo 15, 2017. ACEPTADO: Agosto 11, 2017. VERSIÓN FINAL: Septiembre 28, 2017.

RESUMEN

Este proyecto de investigación muestra la implementación de controladores digitales clásicos dentro de un sistema embebido para controlar la temperatura de un sistema térmico. El sistema embebido seleccionado fue el computador de placa reducida Raspberry Pi 2 B+ junto con una tarjeta Arduino UNO. En la etapa de medición se desarrolló el hardware para transmitir los datos de la temperatura desde la termorresistencia PT-100 hacia la Raspberry Pi. Para la etapa de implementación computacional se desarrolló un algoritmo para los diferentes controladores y con base en este, se programaron los controladores en Python 2.7. Para el actuador se diseñó una etapa de potencia que convierte el voltaje ac en un voltaje dc, a partir de una señal PWM. Este voltaje dc se utiliza para alimentar una resistencia térmica que es la que se encarga de ejecutar la acción de control del sistema. Con la implementación de los tipos de controladores se realizó un análisis comparativo destacando las características más relevantes del sistema para cada tipo de controlador y permitiendo evaluar también el rendimiento de la Raspberry Pi. Con base en los resultados obtenidos en la ejecución del proyecto, se concluye la viabilidad para controlar sistemas térmicos bajo diferentes tipos de controladores embebidos en la Raspberry Pi y con la posibilidad de desarrollar estrategias de control inteligente para futuros proyectos, a través de herramientas de software y hardware libre.

PALABRAS CLAVE: Arduino UNO; controlador PID; raspberry Pi; sistema térmico.

ABSTRACT

This research project shows the implementation of classic digital controllers in an embedded system to control the temperature of a thermal system. The embedded system selected was in the computer board Raspberry Pi 2 B+ together with an ONE Arduino board. In the measurement stage the hardware was developed for to transmit the temperature data from the PT-100 thermometer resistance towards the Raspberry PI. For the implementation of the computational stage was developed an algorithm for the different controllers and based on this the controllers were programmed in Python 2.7. For the actuator a power stage was designed that converts the

ISSN Impreso: 1657 - 4583, En Línea: 2145 - 8456

ac voltage into a voltage dc, from a PWM signal. This dc voltage is used to feed a thermal resistance that is responsible for executing the control action of the system. With the implementation of the types of controllers, a comparative analysis was realized highlighting the most relevant characteristics for each type of controller and allowing to also evaluate the performance of the Raspberry Pi. Based on the results obtained in the execution of the project, it concluded the feasibility to control thermal systems under different types of controllers embedded in the Raspberry Pi and with the possibility of developing intelligent control strategies for futures projects, through the use of software tools and free hardware.

KEYWORDS: ONE arduino; PID controller; raspberry Pi; thermal system.

1. INTRODUCCIÓN

El rápido crecimiento de nuevas tecnologías en la creación de circuitos integrados, ha permitido la fabricación de nuevos dispositivos de una gran capacidad electrónica, mucho más pequeños y de mayor capacidad de cómputo en comparación con algunos años atrás. Un ejemplo de esto es el computador de placa reducida (SBC por sus siglas en inglés) conocido como Raspberry Pi, el cual presenta las características de un computador normal, pero de menor capacidad de hardware en comparación a equipos de cómputo comerciales, pero que son lo suficientemente justas como para ejecutar un sistema operativo y aplicaciones creadas por el usuario. Esta capacidad de cómputo junto con el puerto de entradas y salidas de propósito general (GPIO) perfila a la Raspberry Pi como un potente dispositivo capaz de interconectarse con otro hardware a través de protocolos como el I2C, Serial, entre otros que maneja. Por otro lado, la necesidad de aplicar control a distintos tipos de procesos, hace que la Raspberry Pi pueda considerarse como una solución alternativa, a pesar de que su uso no estuviese destinado para tal fin, a dispositivos que comúnmente son utilizados en este ámbito. Además, el hecho de que su software sea de código abierto, da valor agregado a la idea de implementar controladores embebidos en ella.

Aunque la Raspberry Pi se lanzó oficialmente al mercado a principios del 2012 [1] y su uso está destinado principlamente a incentivar la enseñanza a cualquier persona que quiera aprender a programar [2], esto no ha sido impedimento para desarrollar aplicaciones que necesitan de gran capacidad de cálculo o aplicaciones que necesiten la interacción con hardware externo. Por ejemplo, en Colombia se han desarrollado propuestas como el desarrollo de un Control Adaptativo para optimizar una intersección semafórica en un sistema embebido [3], en donde se embebió el controlador adaptativo en la Raspberry Pi modelo B+ y se llevaron a cabo simulaciones de cómo se comporta el algoritmo en los sistemas de regulación vial, en [17] se utilizó la Raspberry Pi como sistema embebido de acuerdo a su capacidad de comunicación y de control, con la finalidad de implementar un sistema de distribución residencial, el cual permite controlar el flujo de energía entre diferentes fuentes de energía y las cargas de una vivienda. A nivel internacional también existen propuestas como la desarrollada por A. Gosh [4] en India, en la cual se evalúa la capacidad de desarrollar e implementar sistemas de control de bajo costo en la Raspberry Pi; básicamente es un laboratorio educativo en el área de control, [5] en Eslovaquia desarrollaron un laboratorio de bajo costo, cuyos experimentos se enfocaron en controlar remotamente tres procesos diferentes, a través de un servidor implementado en la Raspberry Pi; en el proyecto RapiBaBot [6] desarrollado en los Estados Unidos, se controla la estabilidad para un sistema de péndulo invertido mediante la Raspberry Pi y su puerto GPIO, que junto con un acelerómetro digital, se logra controlar la estabilidad de una estructura mecánica bajo el principio del péndulo invertido; y finalmente en Ecuador se desarrolló un proyecto [7] en el cual modelan una planta y se utiliza Octave junto con Python y C++ para evaluar el rendimiento del software en el control de velocidad para un motor DC aplicando estrategias de control analógico clásico.

En este trabajo se propone la comparación de controladores digitales clásicos, en este caso, controladores P, PI y PID, embebidos en la Raspberry Pi modelo 2 B+ para el control de temperatura de una planta térmica. Se utilizó una tarjeta Arduino UNO como conversor analógico digital (ADC) y como conversor digital analógico (DAC), que se comunica con la Raspberry Pi mediante el protocolo I2C; y una pantalla táctil HDMI para visualizar la interfaz gráfica del controlador. La eficiencia del sistema se evaluó mediante la comparativa de los tres tipos de controlador implementados y del rendimiento del procesador de la Raspberry Pi.

El artículo está organizado de la siguiente manera: el marco teórico utilizado para abordar el problema; la narrativa del proceso en la cual se explica el banco experimental sobre el cual se desarrolló el proyecto, el modelamiento de la planta térmica y el diseño y la implementación de los controladores digitales embebidos; posteriormente en la sección de resultados se compara el comportamiento del sistema de acuerdo a

cada tipo de controlador y destacando las características más relevantes de cada uno de ellos y el rendimiento del procesador de la Raspberry Pi; y finalmente las conclusiones del trabajo.

2. MARCO TEÓRICO

En los procesos industriales, más del 97 % de los controladores reguladores son del tipo PID [8]. Su popularidad se debe a la simplicidad del ajuste de sus parámetros y al hecho de estar disponible en casi todos los equipos de control de la industria. Un controlador PID calcula inicialmente el "error", entre su variable controlada (medida) y su valor deseado (setpoint) y en función de este error genera una señal de control para eliminar el desvío probable [9]. El algoritmo PID usa el "error" en tres módulos distintos para producir su salida o variable manipulada. Los nombres proporcional (P), integral (I) o derivativo (D), son usados para definir los principales controles usados en la práctica:

- Controlador proporcional (P)
- Controlador proporcional integral (PI)
- Controlador proporcional integral derivativo (PID)

2.1. Controlador P

La salida de este controlador es proporcional al error $e(t)$. El factor de proporcionalidad es llamado ganancia del controlador [9]. Este controlador se define matemáticamente como:

$$u_P(t) = K_P \cdot e(t) \tag{1}$$

En el control proporcional existe una relación lineal entre el valor de la variable controlada y la posición del elemento final de control [10]. La ecuación (1) es la expresión para cualquier controlador P continuo, pero si se expresa en términos discretos dicha ecuación se llega a la expresión de un controlador P digital:

$$u_P(k) = K_P \cdot e(k) \tag{2}$$

En donde $K_P = K_C$ es la ganancia proporcional del controlador.

2.2. Controlador PI

Este controlador genera su salida proporcional a la integral del error.

$$u_{PI}(t) = K_P \cdot e(t) + K_P \left(\frac{1}{\tau_i}\right) \int e(t)\, dt \tag{3}$$

Las características más relevantes del controlador PI: es que para un error en escalón (pulso), la acción integral será la integral del escalón que es una rampa. La acción integral aumenta o disminuye la salida del controlador indefinidamente hasta que el error se elimine [9]. La ecuación (3) corresponde al controlador PI continuo, de dicha expresión se puede expresar en términos discretos el término integral para hallar una ecuación de un controlador I digital:

$$u_I(k) = u_I(k-1) + K_I \cdot e(k) \tag{4}$$

Donde $K_I = K_P \cdot (T/\tau_i)$, siendo K_P la ganancia proporcional, T el tiempo de muestreo del controlador digital y τ_i el tiempo integral, y el término $u_I(k-1)$ se refiere al resultado del valor del cálculo de la acción de control anterior. Con las ecuaciones (2) y (4) se puede definir la ecuación para un controlador PI digital:

$$u_{PI}(k) = u_P(k) + u_I(k) \tag{5}$$

2.3. Controlador PID

Este controlador genera su salida proporcional al error, a la integral del error y a la derivada del error.

$$u_{PID}(t) = K_P e(t) + K_P \left(\frac{1}{\tau_i}\right) \int e(t)\, dt + K_P \tau_D \frac{d[e(t)]}{dt} \tag{6}$$

La acción derivativa tiene como finalidad "anticipar hacia dónde va el proceso", mediante la observación de la rapidez en el cambio del error [10]. La ecuación (6) corresponde a la expresión de un controlador PID continuo, de dicha expresión se puede expresar en términos discretos el término derivativo para hallar una expresión de un controlador D digital:

$$u_D = K_D[e(k) - e(k-1)] \tag{7}$$

Donde $K_D = K_P(\tau_D/T)$, siendo K_P la ganancia proporcional, T el tiempo de muestreo del controlador digital y τ_D el tiempo derivativo, y el término $e(k-1)$ que se refiere al error anterior. Con las ecuaciones (2), (4) y (7) se puede definir la ecuación para un controlador digital:

$$u_{PID}(k) = u_P(k) + u_I(k) + u_D(k) \tag{8}$$

3. NARRATIVA DEL PROCESO

Los pasos mencionados a continuación describen el banco experimental que se utilizó para la realización del proyecto, el modelamiento de la planta a partir de datos

experimentales, y el diseño e implementación de los controladores clásicos embebidos en la Raspberry Pi.

3.1. Banco Experimental

El propósito del controlador embebido es el de controlar cualquier sistema, pero en el caso de desarrollo de este proyecto, se implementó el controlador con el fin de ejecutar la acción de control sobre un sistema térmico, Figura 1. el cual es un horno a pequeña escala, que tiene por elemento calefactor una resistencia térmica tipo cartucho para un voltaje nominal de $110V_{AC}$ y una potencia eléctrica de $100W$; para medir la temperatura dentro del horno, se utilizó un termorresistor PT-100, el cual va conectado a un transmisor de temperatura de referencia Sitrans TH-100.

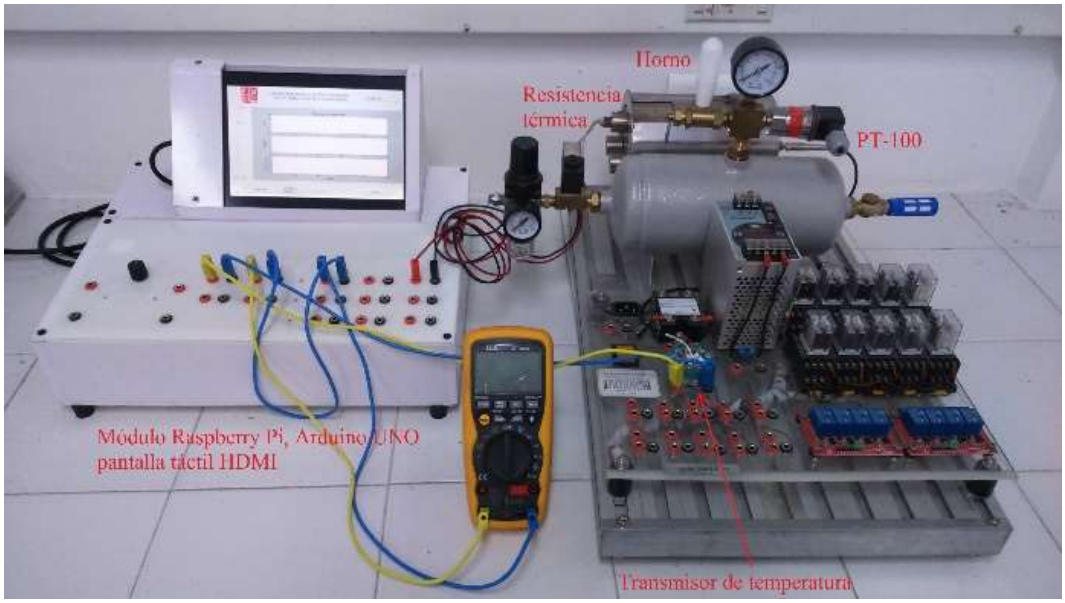

Figura 1. Módulo de control y planta térmica. **Fuente.** Elaboración propia.

El transmisor está programado para medir un rango de temperatura entre $0℃ - 150℃$, y para entregar una señal de corriente proporcional a la temperatura de $4mA - 20mA$. La señal de corriente generada por el transmisor es convertida en una señal de voltaje dc de $1V_{DC} - 5V_{DC}$ por un potenciómetro de precisión que se calibra en un valor de 250Ω. Esta señal de voltaje es leída por la Raspberry Pi mediante un canal analógico del Arduino UNO; la Raspberry Pi y el Arduino UNO se comunican por el protocolo I2C, siendo la Raspberry Pi el maestro y el Arduino UNO el esclavo.

En la Raspberry Pi se encuentra embebido el controlador, programado en Python versión 2.7, junto con la interfaz gráfica, en la cual se puede observar el comportamiento del sistema de control, esto se logra a través de tres gráficas, las cuales son el error del sistema, la acción de control y la temperatura medida. Una vez que la Raspberry recibe el valor de la temperatura, el programa en Python se encarga de calcular la acción de control de acuerdo al tipo de controlador que se esté utilizando, para luego enviar la acción de control al Arduino UNO, el cual genera una señal de PWM proporcional a dicha acción de control. El PWM opera a una frecuencia de $490Hz$, el ciclo de trabajo útil varía entre el $0\% - 73.3\%$, esto con el fin de no exceder los límites de voltaje para la resistencia térmica. Esta señal PMW es inyectada a un convertidor AC-DC, el cual consta de una etapa de rectificación, una etapa de filtrado y la etapa de regulación. La rectificación se hace mediante un puente rectificador de diodos; el filtrado se hace mediante un condensador electrolítico; y la regulación se realiza con un Mosfet de potencia con referencia IRF630, el cual es disparado por la señal PWM generada por el Arduino UNO; esta etapa hace que el ciclo de trabajo útil de la señal PWM genere un voltaje dc proporcional entre $0V_{DC} - 120V_{DC}$.

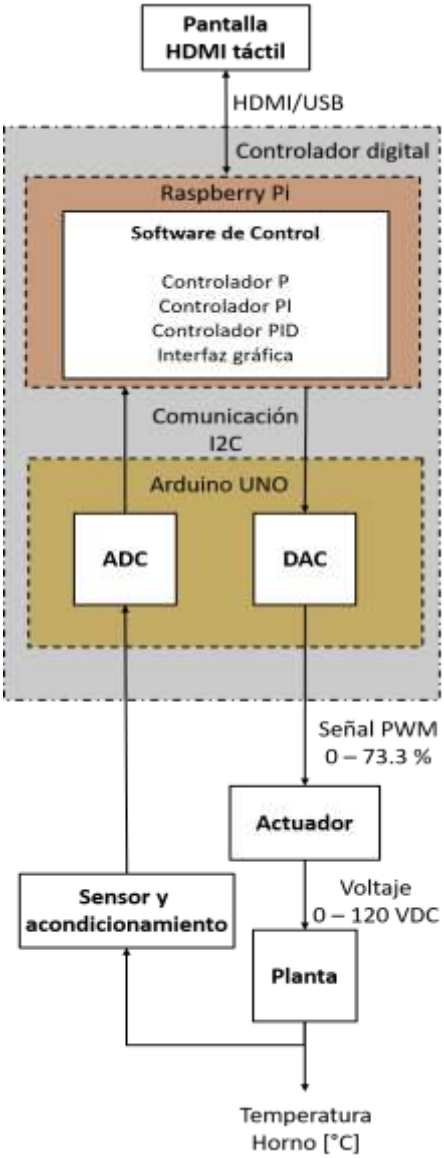

Figura 2. Digrama de bloques del sistema de control digital. **Fuente.** Elaboración propia.

Este voltaje dc se utiliza para alimentar la resistencia térmica del horno. A través de la pantalla HDMI se selecciona la temperatura de referencia, y se supervisa el estado del controlador en tiempo real. En la Figura 2 se puede observar el diagrama de bloques del sistema de control desarrollado en el proyecto.

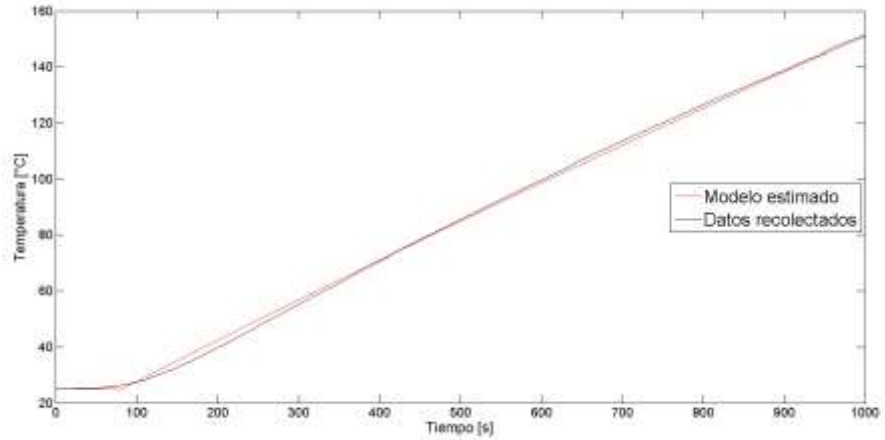

Figura 3. Datos recolectados y estimación del modelo de la planta. **Fuente.** Elaboración propia.

Con el fin de mantener la simplicidad, el sistema se modela como una planta de primer orden con retardo (POR) [10]. La función de transferencia para dicho modelo está dada por:

$$G(s) = \frac{Ke^{-t_d s}}{1 + \tau s} \tag{9}$$

En donde K es la ganancia de la planta, τ es la constante de tiempo y t_d es el tiempo muerto.

Con los datos recolectados y el *System Identification*, se obtiene una función de transferencia para una planta de sistema de primer orden con retardo, con las siguientes características:

$$G(s) = \frac{4.6736e^{-81.403s}}{1 + 5649.1s} \tag{10}$$

En dónde sí se compara la ecuación (10) con la ecuación (9) se tiene que la ganancia de la planta es 4.6736 la cual indica cuánto cambia la variable de la salida por una unidad de cambio en la función de la variable de entrada [14]. La constante de tiempo es $5649.1\ s$ la cual guarda relación con la velocidad de respuesta del proceso [14]. El tiempo muerto es $81.403\ s$, el cual indica el tiempo que se demora en cambiar la variable controlada ante un cambio en la entrada del actuador.

El modelo de la función de transferencia presentado en la ecuación (10) es una aproximación de los datos recolectados y presentados en la Figura 3. Este modelo de la función de transferencia tiene una estimación del $96.81\ \%$ respecto a los datos experimentales recolectados.

3.2. Cálculo e implementación de los controladores

Para calcular los controladores P, PI y PID, se utilizó la herramienta *PID Tuner* [15] de Matlab, puesto que como se mencionó en la parte del modelamiento de la planta, el sistema es un sistema sin autorregulación [12], con lo cual, los métodos convencionales para sintonizar controladores PID como reguladores [16], no se pudieron aplicar. En este caso, los parámetros calculados para los controladores se muestran en la Tabla 1. Estos parámetros regulan la planta para operar en el rango de medición del sensor y garantizan la estabilidad del sistema.

Tabla 1. Ganancias para los controladores. **Fuente.** Elaboración propia

Controlador	K_P	K_I	K_D
P	7.5831	-	-
PI	7.7942	0.00087944	-
PID	8.1623	0.00096398	29.2527

De la Tabla 1 se puede observar que las ganancias proporcionales aumentan en los controladores PI y PID respecto a la ganancia del controlador P, esto con el fin de disminuir el error de estado estacionario [14], pero sin llegar a desestabilizar el sistema. En cuanto a la ganancia integral, su finalidad es eliminar la desviación presentada por la ganancia proporcional y que la respuesta del controlador sea más rápida [14], con lo cual, se espera que el controlador PID responda de manera más rápida que el controlador PI, puesto que la ganancia integral del controlador PID es mayor. En el caso de la acción derivativa, se utiliza para dar al controlador la capacidad de anticipar hacia dónde va el proceso [14], para este caso, una ganancia derivativa grande, hace que el controlador se anticipe mucho más rápido respecto a una ganancia derivativa menor.

Las ganancias de la Tabla 1, se utilizan en el cálculo de la acción de control presentado en las ecuaciones (2), (5) y (8) para controladores P, PI y PID respectivamente.
Para implementar el control en la Raspberry Pi, se tiene en cuenta el algoritmo descrito a continuación, en el cual se muestran las etapas que son ejecutadas secuencialmente por el programa desarrollado en Python:

• Leer la señal de voltaje correspondiente a un valor de temperatura, en la entrada analógica del Arduino UNO.

• Realizar el escalamiento y conversión del voltaje leído en su valor de temperatura, expresado en °C, y almacenar este valor en una variable del sistema.

• Calcular el error que hay entre la temperatura deseada (set point) y la temperatura que se está leyendo en el sensor. Este dato también se almacena en una variable del sistema.

• Se calcula la acción de control dependiendo del controlador que se desee implementar a partir de las ecuaciones (2), (5) u (8). El valor de la acción de control es almacenado en una variable del sistema.

3.3. Modelamiento de la planta

Para realizar el modelamiento de la planta, se hizo un ensayo experimental basado en el método de la curva de reacción del proceso [11], el cual consistió en aplicar una señal escalón a la planta y registrar la dinámica de la temperatura y posteriormente realizar la curva de reacción del proceso, como se observa en la Figura 3. Como la temperatura excede el rango de operación del sensor antes de estabilizarse, el proceso no se puede considerar como auto-regulado [12], con lo cual, no se puede aplicar el método completo de la curva de reacción [11], por lo tanto, para realizar el modelamiento de la planta, se utilizó la herramienta *System Identification* [13] de Matlab versión 2010, junto con los datos recolectados experimentalmente.

• Si la acción de control es negativa se debe aplicar un limitador con el fin de evitar que el actuador opere en condiciones no deseadas. Si la acción de control es mucho mayor al límite fijado para el actuador, se debe saturar la señal al máximo para evitar que el actuador opere en condiciones no deseadas.

• Se actualiza la variable que contiene el error anterior si se está utilizando un controlador PID. Se actualiza la variable que contiene la acción de control integral anterior para controladores PI y PID. Si se utiliza un controlador P no es necesario actualizar ninguna de las dos variables mencionadas anteriormente.

• Finalmente, se envía la acción de control al actuador mediante la salida por PWM de la tarjeta Arduino UNO.

En la Figura 4 se puede observar el diagrama de flujo del algoritmo para ejecutar el controlador.

Cabe resaltar, que este algoritmo es ejecutado cada T, el cual se había definido como periodo de muestreo. Este periodo de muestreo está implícito en las ganancias de los controladores como se puede observar en las ecuaciones (4) y (7). En el proyecto se utilizó un periodo de muestreo de $1\ s$, el cual es suficiente para calcular y realizar la acción de control del sistema térmico.

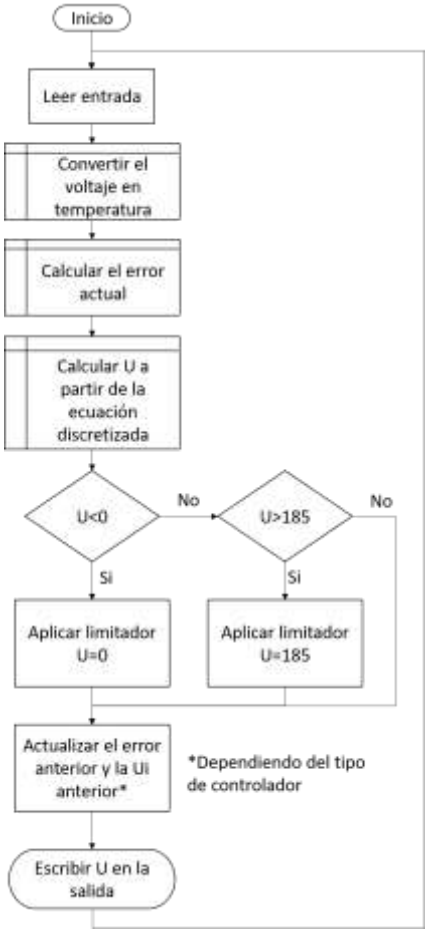

Figura 4. Diagrama de flujo del algoritmo de control. **Fuente.** Elaboración propia.

Paralelamente, se ejecuta dentro del sistema embebido otro algoritmo, el cual utiliza los datos que se recolectan en el algoritmo del controlador, y los utiliza para graficar en tiempo real la evolución de la variable del error, de la acción de control y de la temperatura dentro del horno. Este algoritmo se ejecuta cada segundo permitiendo

realizar la supervisión del sistema térmico en tiempo real desde la pantalla HDMI.

Los dos algoritmos mencionados anteriormente, se ejecutan desde un programa principal, el cual se desarrolló en Python 2.7, junto con las bibliotecas smbus, time, RPi.GPIO y matplotlib.

4. RESULTADOS

Para evaluar el desempeño de la Raspberry Pi se realizaron pruebas, implementado los tres tipos de controladores y se utilizó una temperatura de referencia de $100\,°C$ para posteriormente comparar los resultados obtenidos de cada controlador. Se observaron las características de cada controlador tanto en la temperatura, el tiempo de establecimiento y la acción de control entregada al actuador. También se evaluó el rendimiento del sistema embebido en cuanto al tiempo de ejecución del algoritmo y el uso de la capacidad de procesamiento de la CPU al ejecutar el programa.

4.1. Temperatura

Para comparar la temperatura de los tres tipos de controlador, se estableció como parámetro de referencia una temperatura de 100℃. En la Figura 5 se puede observar la comparación de la temperatura de salida de acuerdo a cada tipo de controlador.

De la Figura 5 se dedujo el tiempo muerto, el tiempo de asentamiento, la constante de tiempo y la temperatura final del sistema. En la Tabla 2 se presentan estos parámetros para cada tipo de controlador. El parámetro τ representa la constante de tiempo, t_d representa el tiempo de retardo, t_a representa el tiempo de asentamiento para un valor del 2 % de la variable controlada y T_F representa la temperatura final en la cual el sistema se estabilizó.

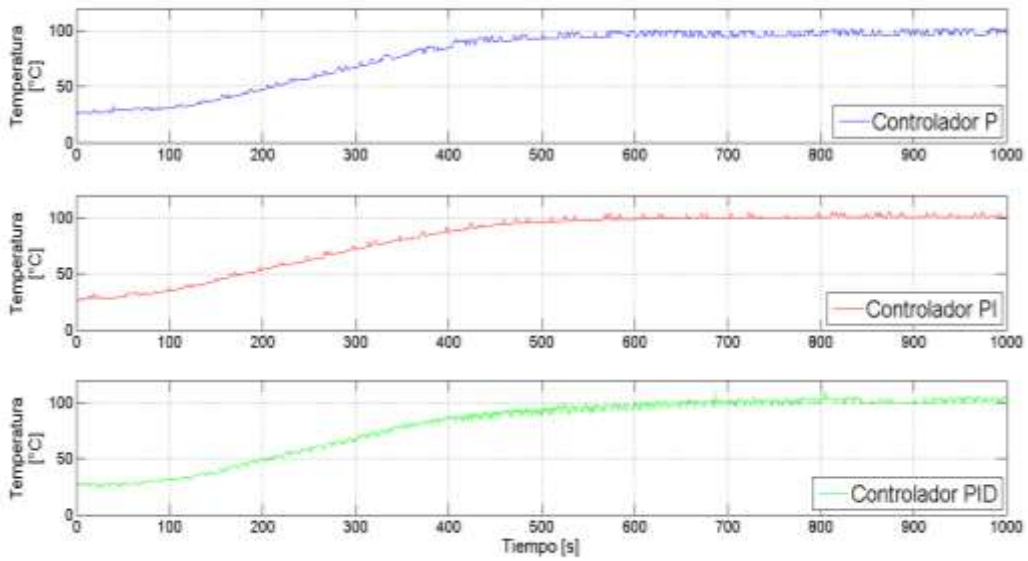

Figura 5. Temperatura del proceso térmico para los tres tipos de controladores. **Fuente.** Elaboración propia.

Tabla 2. Parámetros del sistema para cada controlador implementado.

Tipo de controlador	Parámetro			
	τ	t_d	t_a	T_F
P	173.2 s	70 s	600 s	97.5 °C
PI	180.6 s	70 s	534 s	100 °C
PID	188 s	70 s	589 s	102 °C

Fuente. Elaboración propia.

4.2. Acción de control

Para comparar la acción de control ejecutada por cada controlador, se estableció como parámetro de referencia una temperatura de 100℃. En la Figura 6 se puede observar la acción de control expresada en porcentaje de acuerdo a cada tipo de controlador.

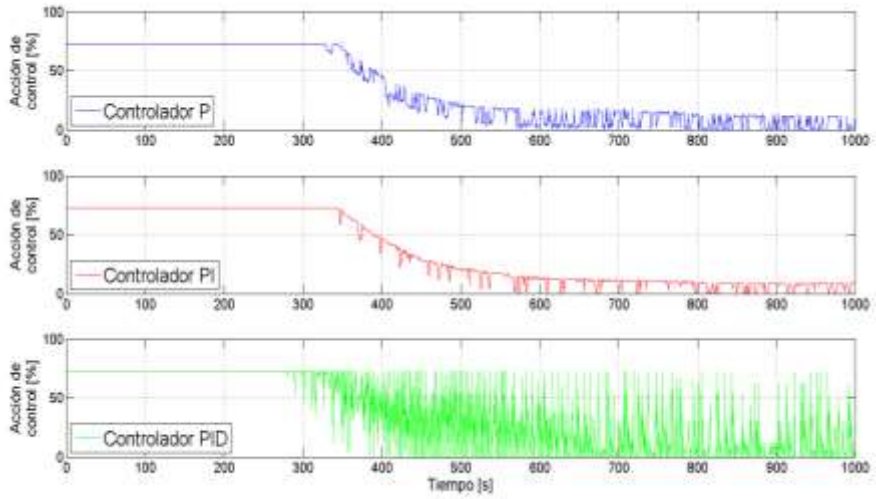

Figura 6. Acciones de control generadas por los tres tipos de controladores. **Fuente.** Elaboración propia.

De la Figura 6 se puede observar que las acciones de control calculadas por los controladores P y PI son atenuadas y presentan picos pequeños en comparación con la acción de control calculada por el controlador PID, esto se debe a que el término proporcional que se calcula solamente utiliza el error presente para calcular la acción de control; en cuanto al término integral, éste también utiliza el error actual y la acción de control integral calculada anteriormente para generar la nueva acción de control integral, estos factores hacen que las acciones de control calculadas por los controladores P y PI no presenten picos mayores al 20 %. Este fenómeno es debido al ruido presente en el conversor analógico digital del Arduino UNO. En cuanto a la acción de control calculada por el controlador PID, se puede observar que este genera picos de hasta el 100 % de la acción de control, esto se debe a que el término derivativo del controlador, utiliza el error actual y el error anterior para calcular la acción de control, eso, sumado a una alta ganancia derivativa, como la calculada en este controlador (Tabla 1), hace que el controlador amplifique el ruido generado por el conversor analógico digital del Arduino UNO.

4.3. Rendimiento del sistema embebido

Para evaluar el rendimiento del sistema embebido, primero se determinó cuánto tiempo tarda en ejecutar completamente el programa el algoritmo de control, desde que lee la señal del sensor hasta que le envía la señal a la planta. En la Tabla 3, se observan los tiempos de ejecución del algoritmo para cada tipo de controlador.

Tabla 3. Tiempo de ejecución del algoritmo de control para cada tipo de controlador.

Controlador	Tiempo de ejecución (ms)
P	2.550
PI	2.374
PID	2.585

Fuente. Elaboración propia.

Como se puede observar en la Tabla 3, los tiempos de ejecución del algoritmo de control son muy parecidos y están por el orden de los milisegundos.

Paralelamente, el programa principal, ejecuta otro algoritmo, el cual toma los datos almacenados por el algoritmo de control, y va graficando cada segundo las variables del sistema de control; este algoritmo encargado de graficar los datos en la interfaz gráfica, es ejecutado cada segundo y éste tarda en mostrar las gráficas unos 874 ms.

En cuanto al uso del procesador por parte de Python, cuando se ejecuta el programa principal, únicamente ejecutando el algoritmo de control, Python usa un 0.3 % de la CPU, mientras que ejecutar el programa completo, o sea graficando y ejecutando el algoritmo de control, el uso de la CPU asciende al 12 %.

La frecuencia de trabajo del procesador se deja por defecto a la que trae la Raspberry Pi 2 B+, la cual es de 900 MHz.

5. CONCLUSIONES

Se puede concluir que la mejor estrategia de control que se desarrolló en el sistema embebido fue la del controlador PI, ya que este controlador presentó un tiempo de asentamiento menor de la variable controlada, respecto a los otros dos controladores. Además, se controló con mayor precisión la temperatura con este controlador; el controlador P presenta un error de estado estacionario que es una desventaja si se pretende controlar con gran precisión la temperatura, y el controlador PID tiene una leve desviación debido al ruido generado en la etapa de conversión analógica digital.

Finalmente, en cuanto al tiempo de ejecución del algoritmo, el que presentó un mejor tiempo es el controlador PI; los controladores P y PID tienen un tiempo muy similar, aunque no hay una diferencia muy notable entre los tres tiempos calculados para cada controlador.

El comportamiento abrupto y los picos que se observan en la acción de control calculada por el controlador PID son debido al ruido inherente del conversor analógico digital del Arduino UNO; este efecto de ruido se puede mitigar un poco leyendo varias veces la entrada y promediando estas lecturas, aunque esto implica utilizar más tiempo en la ejecución del algoritmo de control. Este ruido es amplificado por la alta ganancia derivativa del controlador lo que representa estos picos agresivos en la ejecución de la acción de control, que pueden ocasionar daños en la planta. Si bien la lenta respuesta del proceso hace que este comportamiento inestable de la acción de control no afecte la variable controlada, sí es importante considerar utilizar un conversor analógico digital que tenga una mayor precisión en la conversión para mejorar el cálculo de la acción de control y evitar este comportamiento no deseado.

El uso del Arduino UNO como conversor analógico digital, aunque es una opción viable debido al bajo costo del dispositivo, se debe considerar con detenimiento, ya que la resolución con la cual se envían los datos leídos a la Raspberry Pi es de 8 bits, además existe ruido inherente a la conversión que genera error en la conversión; también, la máxima velocidad de transmisión del protocolo de comunicación I2C del Arduino UNO es de 100 kHz. Si se requiere mejorar en alguno de estos aspectos, es posible utilizar conversores analógicos digitales que tengan una mayor precisión y resolución de conversión, y que tengan mayor velocidad de transmisión, lo cual se vería reflejado en un aumento en la precisión de la lectura del sensor, mejora en el cálculo de la acción de control e inclusive disminuir la velocidad de ejecución del algoritmo de control, todo esto con el fin de lograr implementar estrategias de control para sistemas de respuesta rápida.

Lo más destacable de implementar estrategias de control clásico en el sistema embebido es la facilidad de reprogramar el controlador, además, de modificar los parámetros con el fin de poder visualizar como afectan estos cambios en la dinámica del sistema de control.

El buen desempeño de los controladores clásicos implementados para ejecutar el control del sistema térmico, junto con el buen rendimiento del procesador, deja la puerta abierta a la idea de implementar sistemas de control inteligente, como controladores de lógica difusa, algoritmos genéticos, etc.; inclusive, se puede pensar en implementar sistemas de múltiples entradas y múltiples salidas (MIMO) ya que es posible usar más de una entrada analógica y más de una salida digital por PWM del Arduino UNO, o si es el caso, pensar en utilizar otros dispositivos para realizar la conversión analógico digital con el fin de mejorar la robustez del hardware del sistema embebido.

El uso de tecnologías de código abierto como Python y el software de Arduino, y de hardware libre como la Raspberry Pi y el Arduino, hace posible pensar en implementar nuevas tecnologías que permitan desarrollar estrategias de control en sistemas industriales con la viabilidad de reducir costos por el uso de licencias de fabricantes, factores que hacen que se incrementen los costos de un proyecto.

6. REFERENCIAS

[1] "The Raspberry Pi computer goes on general sale BBC News", *BBC News*, 2012. [Online]. Available: http://www.bbc.com/news/technology-17190918. Mar, 2017

[2] "Raspberry Pi FAQs - Frequently Asked Questions", *Raspberry Pi*, 2017. [Online]. Available: https://www.raspberrypi.org/help/faqs/. Mar, 2017.

[3] J. Celis, C. Escobar, S. Sepúlveda, S. Castro, B. Medina, J. Ramírez, "Control adaptativo para optimizar una intersección semafórica basado en un sistema embebido", *Ingeniería y Ciencia*, vol. 12, no. 24, pp. 169-193, 2016.

[4] A. Ghosh, "Intelligent appliances controller using Raspberry Pi", *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2016.

[5] M. Kalúz, Ľ. Čirka, R. Valo and M. Fikar, "ArPi Lab: A Low-cost Remote Laboratory for Control Education", *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 9057-9062, 2014.

[6] C. Micklisch and H. ElAarag, "RapiBaBot: A solution to the inverted pendulum using a raspberry Pi and its GPIO", *IEEE SOUTHEASTCON 2014*, 2014.

[7] R. Ponguillo, C. Medina, "Using open source embedded hardware and software tools in automatic control from mathematical model", *Lecture Notes in Engineering and Computer Science*, vol. 2225, pp. 333-337, 2016.

[8] C. Yu, *Autotuning of PID controllers*, 1st ed. London: Springer, 2006, p. 3.

[9] O. Hernández, F. Moreno, J. Becerra, *Control de procesos térmicos*, 1st ed. Bogotá, D.C.: Universidad Francisco de Paula Santander, pp. 106-111.

[10] L. García Jaimes, *Control Digital. Teoría y Práctica.*, 2nd ed. Medellín: Politécnico Colombiano JIC, 2010, pp. 189-195.

[11] V. Alfaro, "Identificación de procesos sobreamortiguados utilizando técnicas de lazo abierto", *Revista Ingeniería*, vol. 11, no. 1-2, Ene-Dic, 2011.

[12] A. Roca, *Control automático de procesos industriales*, 1st ed. Ediciones Díaz de Santos, 2014, pp. 84-88.

[13] "System Identification Toolbox Documentation - MathWorks United Kingdom", *Mathworks.com*, 2017. [Online]. Available: https://www.mathworks.com/help/ident/. Apr- 2017.

[14] C. Smith and A. Corripio, *Control automático de procesos*, 1st ed. México, D.F.: Limusa, 2001.

[15] "Open PID Tuner for PID tuning - MATLAB pidTuner - MathWorks United Kingdom", *Mathworks.com*, 2017. [Online]. Available: https://www.mathworks.com/help/control/ref/pidtuner.html. Apr- 2017.

[16] V. Alfaro Ruíz, "Métodos de sintonización de controladores pid que operan como reguladores", *Revista Ingeniería*, vol. 12, no. 1-2, Ene-Dic, 2011.

[17] L. Rueda, J. Barrero, C. Duarte, "El conmutador inteligente de potencia y la sub-medición por circuito como herramientas para la gestión energética residencial," Rev. *UIS Ing.*, vol. 16, no. 1, pp. 35-46, 2017.