# SISTEMA DE CONTROL DE UNA PLANTA EMBEBIDA EN FPGA EMPLEANDO HARDWARE-IN-THE-LOOP

# CONTROL SYSTEM OF A PLANT EMBEDDED IN FPGA USING HARDWARE IN THE LOOP


OSCAR IVÁN CALDAS FLAUTERO

*Ing. MCT Universidad Militar Nueva Granada, Investigador, davinci@unimilitar.edu.co*

SEBASTIÁN JIMÉNEZ GÓMEZ

*Ing. MCT Universidad Militar Nueva Granada, Investigador, davinci@unimilitar.edu.co*

EDILBERTO MEJÍA RUDA

*Ing. MCT Universidad Militar Nueva Granada, Investigador, davinci@unimilitar.edu.co*

OSCAR FERNANDO AVILÉS SÁNCHEZ

*Ph.D. Universidad Militar Nueva Granada, Director Programa Ing. Mecatrónica, oscar.aviles@unimilitar.edu.co*

DARÍO AMAYA HURTADO

*Ph.D. Universidad Militar Nueva Granada, Líder Grupo de Investigación GAV, dario.amaya@unimilitar.edu.co*



Recibido para revisar Enero 18 de 2013, aceptado Abril 26 de 2013, versión final Mayo 6 de 2013



**RESUMEN:** Los sistemas de prototipado rápido son ampliamente usados en el desarrollo de productos en diferentes industrias, ya que permite configurar prototipos iniciales, probarlos bajo condiciones reales y optimizarlos con efectividad en tiempo y costo, sin mayor desarrollo de hardware. Con el concepto de Hardware-in-the-loop (HIL), se pueden emular condiciones de operación de un sistema real, llevándolo a un modelo matemático embebido en un sistema electrónico. Este trabajo muestra la implementación de una planta térmica en una FPGA que emula el comportamiento real y permite evaluar los alcances y beneficios de la técnica HIL. Por otro lado, se diseñó un controlador tipo PID en el software LabView®, que reguló la dinámica de la planta embebida usando comunicación serial. Se hizo una comparación entre el experimento y una simulación en Matlab® Simulink para validar el desempeño del sistema con los parámetros de diseño.



**PALABRAS CLAVE:** HIL; Sistema de control; FPGA.



**ABSTRACT:** Rapid prototyping systems are widely used in product development in the different industries. They provide the ability to set up the initial prototypes, test them under real conditions, and optimize them time- and cost-effectively, without extensive hardware development. Following the Hardware-in-the-loop concept, the operation conditions of a real system can be emulated by means of a mathematical model, embedded in an electronic device. This paper shows a thermal plant implementation in a FPGA, which emulates the real behavior and allows to assess the scope and benefits of HIL technique. On the other hand, a PID controller was designed in LabView® to regulate the dynamics of the embedded plant by serial communication. A comparison was made to validate the system performance based on the design parameters.



**KEYWORDS:** Control system; Hardware-in-the-Loop; FPGA.


## 1. INTRODUCCIÓN

La tecnología *Hardware-in-the-loop* (HIL) se ha convertido en una parte integral del proceso de desarrollo electrónico para fabricantes y proveedores de productos, que emula en tiempo real la planta o el sistema de control, junto con elementos reales como sensores y actuadores (frecuentemente usados por su dificultad de modelado [1]), optimizando costos, duración y seguridad en las pruebas [2]. La simulación HIL es, por lo tanto, un método empleado en el ciclo de desarrollo de productos, en el cuál uno o más componentes reales interactúan con otros simulados en tiempo real, mediante modelos dinámicos. El principal propósito de esta simulación, es mejorar la calidad del producto, sobreponerse a la complejidad de los sistemas

y reducir los costos del desarrollo. Este último objetivo se alcanza realizando las pruebas de funcionamiento y diagnóstico en el laboratorio HIL, en lugar de hacerlas con experimentos en bancos de prueba. El resultado es la reducción considerable de costosos prototipos y del tiempo requerido en el banco de pruebas [3].

Recientemente, se ha hecho uso de las simulaciones HIL para poner a prueba desarrollos en diversas áreas tecnológicas. En [4] se destaca la reducción en costos y tiempo por a la facilidad para identificar errores de implementación y probar condiciones de falla de forma segura, en sistemas de generación distribuida (DG), cuyos prototipos de alto costo y consumo energético. En [5] también se emplea la técnica HIL para simular de forma segura y controlada el comportamiento de un motor a gas durante su arranque.

Por otro lado, en [6] se utiliza HIL para un sistema de comunicación móvil, alcanzando una simulación en tiempo real de alta precisión, bajo costo y mínimo riesgo, que además permite sintonizar diferentes características (recepción, velocidad, interferencia, etc) con total repetibilidad y controlabilidad.

Otros autores también emplean la simulación HIL para facilitar las pruebas de prototipos en múltiples condiciones. En [7] se emplea la simulación HIL de un tren motriz para un vehículo eléctrico, debido a la facilidad de comprobar su conveniencia para diferentes escenarios de uso y mejorarlos de forma particular, sin incurrir en las costosas adaptaciones de un desarrollo real. Así mismo, en [8] se desarrolló un método para reducir los tiempos de sintonización de los controladores de un Vehículo aéreo no tripulado (UAV), simulado con HIL para aplicarle fácilmente diferentes condiciones de ruido que se presentan en vuelo. De hecho, por esas razones el uso de HIL es muy popular en otras aplicaciones aeroespaciales, incluyendo aplicaciones para sistemas robóticos de rastreo, que requieren simular las diversas condiciones aerodinámicas de sus objetivos [9], de forma similar al control de orientación para toberas pivotantes en la industria espacial [10].

Por otro lado, hay aplicaciones en las que la comunicación con periféricos utiliza recursos máquina elevados, disminuyendo la capacidad computacional disponible para procesar los modelos. En esos casos, tiene sentido mover los puertos a un dispositivo más conveniente y las FPGA (del inglés *Field Programmable Gate Array*) son ideales para estas tareas, ya que su arquitectura de hardware permite una ejecución rápida y concurrente [11]. Adicionalmente, para aplicaciones con una dinámica rápida, como los conversores DC-DC, se requieren frecuencias de PWM (Modulación de ancho de pulso) bastante altas, que pueden ser superiores a 20kHz, además de corrientes y voltajes que pueden ser representados de forma realista sólo mediante simulaciones basadas en FPGA [12].

En este caso se realiza una simulación HIL con propósitos académicos, que así como en [13], se justifica como un procedimiento para comprender los alcances y las limitaciones de la técnica, antes de emplearla en aplicaciones con dinámicas rápidas y sistemas complejos. Particularmente, se presenta la implementación del modelo de un sistema lineal de primer orden en una FPGA de Altera®, controlado por un regulador PID de estructura fija, diseñado e implementado en la herramienta LabView®, en donde se visualiza la evolución de las variables del sistema en el tiempo y se manipulan los parámetros de diseño de la planta embebida como del controlador. Así mismo, se describe la manera en que se diseñaron dichos módulos. La comunicación entre LabView y la FPGA se realiza por medio de comunicación serial, cumpliendo con los requerimientos del protocolo de comunicación RS232 [14] [15].

En las siguientes secciones, se describe por etapas el diseño e implementación del protocolo de comunicación y las ecuaciones en diferencias de la planta y el controlador, establecidas en la FPGA y en LabVIEW®, respectivamente. Al final, se muestran los resultados obtenidos y se plantean las conclusiones correspondientes.

## 2. HERRAMIENTAS Y MÉTODOS

En la Figura 1 se puede ver la arquitectura general del sistema propuesto, en esta sección se va a entrar en detalle de la metodología de diseño de cada etapa, como es la implementación de la planta embebida en la FPGA y el controlador desarrollado en LabView®, como también su integración y comunicación.

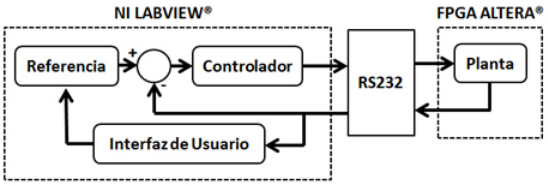


**Figura 1.** Diagrama de bloques sistema de control.

En los dos dispositivos, FPGA y el PC, se implementa el módulo de comunicación serial y un sistema dinámico invariante en el tiempo. Inicialmente, se hace una descripción de los programas desarrollados para la FPGA y se continua con el programa y la interfaz de usuario implementados en LabView®.

### 2.1. Sistema de primer orden en FPGA

Para la construcción del modelo de la planta y el protocolo que permite la comunicación con la Interfaz Gráfica de Usuario (GUI, por sus siglas en inglés), se utilizó la FPGA Altera® *Cyclone IV* [16]. Mediante el lenguaje de programación VHDL, se implementaron dos programas con funcionamiento concurrente (simultáneo) [17]. El primero es el algoritmo para introducir la ecuación en diferencias de un sistema de primer orden, con el cual se representa el modelo de la planta. El segundo, es el módulo para la comunicación serial, que hace uso del protocolo RS232. A continuación se presenta cada una de las etapas mencionadas.

#### 2.1.1. Modelo de la Planta

Se propone una planta de primer orden, que según [18] puede representar lo que físicamente sería un circuito RC o un sistema térmico. Por lo tanto, se consideró un sistema de control térmico sin retardo, donde la señal manipulada está en función a un voltaje de alimentación a una resistencia térmica y la variable controlada es la señal de temperatura como salida del sistema, cuyo modelo dinámico está definido bajo la función de transferencia (1):

$$G_p(s) = \frac{K}{\tau s + 1} \tag{1}$$

En donde es la ganancia en lazo directo y es el tiempo de crecimiento del sistema de primer orden.

En [19] se propone un experimento en el que se le aplica una entrada escalón de 2,5V y el sistema varía de la temperatura ambiente de 20,5°C a un valor final de 47,2°C, entonces la constante de proporcionalidad *K* está dada como la pendiente de la relación Y(s)/U(s), relación de sensibilidad del sensor utilizado durante el experimento (2):

$$K = \frac{\Delta T}{\Delta V} = \frac{47{,}2 - 20{,}5}{2{,}5 - 0} = 10{,}68 \left[\frac{^\circ C}{V}\right] \tag{2}$$

La constante de tiempo τ para un sistema de primer orden se define como el tiempo que se tarda el sistema en alcanzar el 63,2% del valor final (equivalente a 37,37°C), que corresponde a 2.4 horas, obteniéndose los valores para (1):

$$G_p(s) = \frac{10{,}68}{2{,}4s + 1} \tag{3}$$

Se simuló el sistema de primer orden en lazo abierto, con el entorno de simulación visual Simulink de Matlab®, El diagrama de bloques del sistema se muestra en la Figura 2, cuya respuesta ante una entrada escalón se ve en la Figura 3.

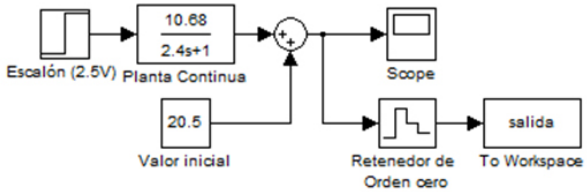


**Figura 2.** Simulación de sistema en lazo abierto

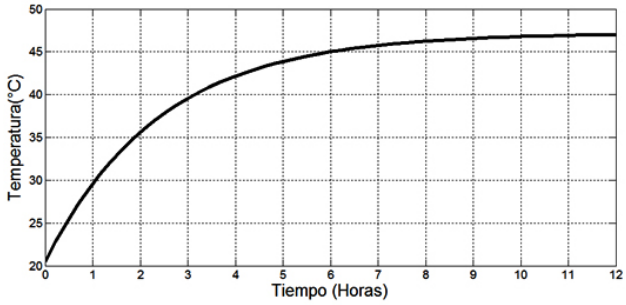


**Figura 3.** Salida de la planta en lazo abierto.

La planta se discretizó usando un modelo de retenedor de orden cero, teniendo en cuenta un periodo de muestreo de 0,24s. El tiempo fue escogido con base en el teorema de muestreo de Nyquist-Shannon, que

aunque indica que el tiempo de muestreo de un sistema debe ser al menos la mitad del tiempo de respuesta, se acostumbra a tomar muestras con periodos 10 veces menores, para que haya integridad en la señal [20]. En la Figura 4 se muestra la respuesta del sistema al escalón y se compara con la del sistema discretizado, con diferentes periodos de muestreo: igual al tiempo de respuesta, a la mitad y 10 veces menor.

En la ecuación (4) se muestra la planta discretizada.

$$G_{pd}(z) = \frac{1{,}016}{z - 0{,}9048} \tag{4}$$

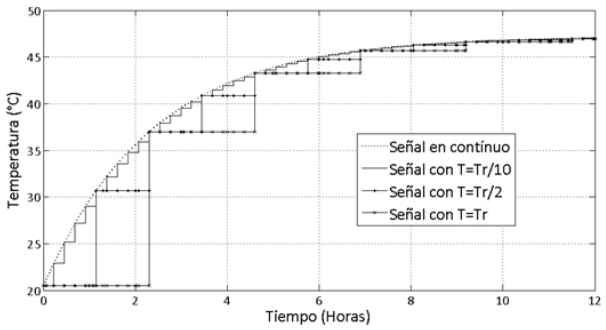


**Figura 4.** Salida de la planta continua y discreta con periodos de muestreo $T_r$, $T_r$/2 y $T_r$/10.

Una vez se tiene la función de transferencia discreta de la planta (4), se obtiene la salida a partir de la ecuación en diferencias (5), para ser implementada en la FPGA.

$$Y(k) = 1.016 \cdot u(k-1) + 0.9048 \cdot y(k-1) \tag{5}$$

Para implementar la ecuación en diferencias en la FPGA, se siguió el procedimiento indicado en el diagrama de flujo de la Figura 5, definiendo los procesos y las variables según [21]. En la línea 2 del pseudocódigo embebido en la FPGA, que se muestra a continuación, se observa la ecuación en diferencias del modelo, magnificada en una relación 100000:1. Una vez se van a enviar los datos, se procede a retornarlos a la escala 1:1, con el objetivo de obtener números enteros, antes de enviarlos por el puerto serial. Esto debido a que la FPGA *Cyclone IV,* en forma nativa, sólo trabaja los flotantes bajo arquitectura de punto fijo [16].

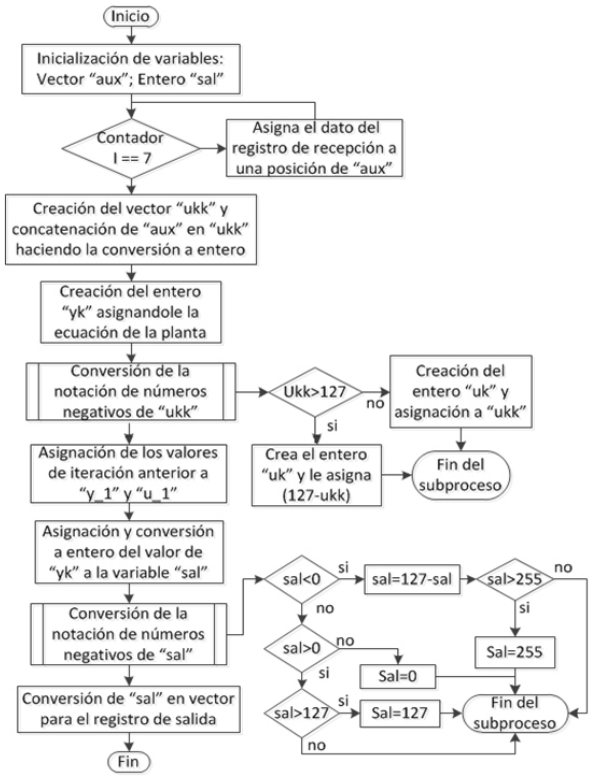


**Figura 5.** Diagrama de flujo para el algoritmo de la planta.

En la línea 1 del pseudocódigo se puede apreciar que siempre la señal de entrada a dicho proceso está determinado por la variable *RegIn*, la cual indica que el buffer de lectura serial se encuentra lleno y ha recibido un nuevo dato, este evento está gobernado directamente desde LabView® por el *time loop* cada 0,24s, tiempo de muestreo seleccionado.

```
1.  planta:process(RegIn)--Inicia Proceso
2.--Ecuación en diferencias Magnificada
     yk<=integer(101600)*u_1+integer(90480)*y_1;
     sal:=conv_integer(yk/1000000);
     RegOut<=conv_std_logic_vector(sal,8);
3.  end process planta; --Finaliza Proceso
```

El software utilizado para programar la FPGA es *Quartus* de Altera. Dentro de la jerarquía del algoritmo se tiene un proceso para la planta, el cual indica si hay un nuevo dato en el registro de recepción.

Previamente, se considera el uso consecutivo de sentencias *if* para la conversión de números enteros negativos (0-127) y positivos (128-255), dentro del rango estándar del RS232, que establece el envío de 8 bits, lo que implica un rango de valores entre 0 y 255.

En el algoritmo, “yk” y “y_1” corresponden a la salida de la ecuación en diferencias y su valor anterior, respectivamente. La entrada “uk” y su valor anterior “u_1”, se obtienen luego de identificar si el valor de “ukk” (el entero del registro de recepción) es negativo (debido al bit de signo de “dbIn”). Adicionalmente, se observan dos variables internas: la primera se denomina “aux” y se utiliza como registro temporal para la conversión de los datos del registro de recepción “debIn”, de tipo *std_logic* a tipo entero, por medio del uso secuencial de una sentencia *for loop* y una concatenación de sus posiciones; la otra variable es “sal”, que corresponde al resultado de la ecuación en diferencias “yk”, empleada para enviar su valor al registro de transmisión, ubicando los valores negativos de tal forma que se genere un bit de signo al convertirse al sistema binario.

### 2.1.2. Comunicación Serial

De acuerdo a la Figura 6, la arquitectura de la entidad UART (del inglés *Universal Asynchronous Receiver / Transmitter*) está diseñada para realizar las acciones de transmisión y recepción entre la FPGA y el sistema SCADA.

La transmisión y la recepción se diseñan mediante la lógica secuencial de las denominadas Máquinas de Estados Finitos tipo Mealy, es decir, un modelo de comportamiento que define un conjunto de estados intermedios entre entradas y salidas, definiendo las salidas según el estado actual y las entradas [21].

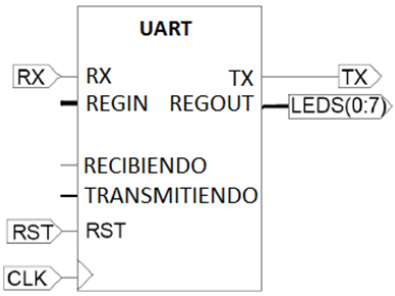


**Figura 6.** Arquitectura de la entidad de comunicación serial.

Según [22], el uso de procesos en el lenguaje VHDL permite el funcionamiento concurrente de varias tareas, por lo que se diseñan e implementan dos Máquinas de estados, cada una mediante dos procesos que trabajan en forma concurrente para cumplir con tres tareas como se presenta en la Figura 7.

El primero de esos procesos contiene la lógica combinacional que calcula el estado siguiente y la lógica secuencial que define las salidas, ambas dependiendo de las entradas y del estado presente; el segundo realiza la transición de estados, es decir, actualiza el estado presente con lo establecido como “estado siguiente” por el primer proceso.

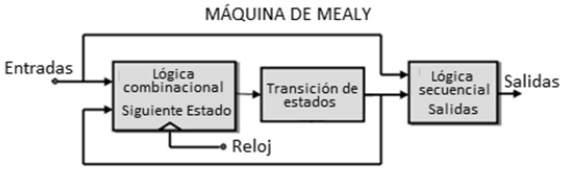


**Figura 7.** Máquina de estados de Mealy

Las Máquinas de Estado para transmisión y recepción se encuentran enlazadas con dos bits de control de flujo “transmitiendo” y “recibiendo”, mutuamente excluyentes, que indican la acción en proceso. Además, cada máquina cuenta con un registro de 8 bits que corresponde al resultado de la conversión serial/paralelo de los datos recibidos o por transmitir, que con propósitos prácticos se referencia a un puerto digital, para así mostrar su valor en los visualizadores de 7 segmentos incluidos en el kit de desarrollo de la FPGA *Cyclone II*. Cabe aclarar que cada una de las dos acciones está directamente relacionada con uno de los dos pines de comunicación asociados al puerto de la FPGA (“Rx” y “Tx”).

En la Figura 8 se observa la Máquina de Estados de transmisión. Esta máquina consta de 3 estados, uno de inicio, que espera por la orden de transmisión, otro en donde se espera por la conversión paralelo-serial y otro para transmitir los siguientes datos por el puerto: bit de inicio, byte de dato, bit de paridad y bit de parada.

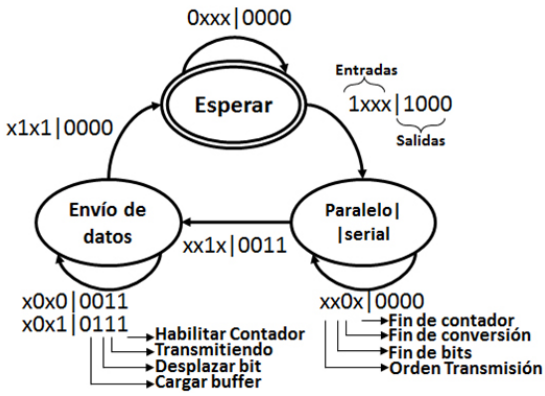


**Figura 8.** Máquina de estados de Transmisión

La Figura 9 muestra la Máquina de Estados de recepción. Aquí se observan 4 estados, uno que espera por la orden de recepción, un estado que da el retardo de sincronización, uno para la recepción del bit de inicio, del byte de dato y del bit de paridad, y un último estado para verificar la recepción del bit de parada.

En adición a los cuatro procesos que determinan el funcionamiento de las dos máquinas de estado finitas, la comunicación contiene un proceso encargado de hacer la división de frecuencia para asignar la velocidad de transferencia, otro como contador, que da la espera entre cada bit de datos (según la velocidad de transferencia seleccionada) y un último para realizar la conversión serial-paralelo y paralelo-serial.

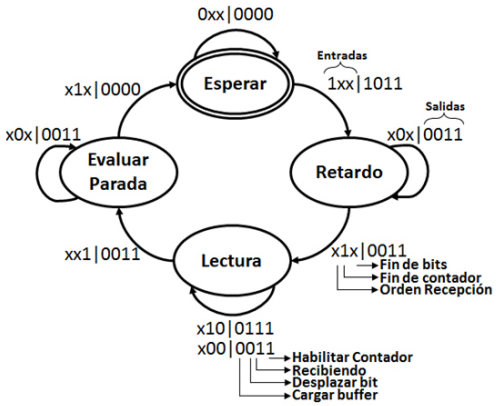


**Figura 9.** Máquina de estados de Recepción

## 2.2. Controlador e interfaz de usuario implementado en sistema SCADA

Una vez desarrollada la etapa de implementación de la planta embebida y la comunicación en la FPGA, los cuales se ejecutan de manera concurrente, con variables tipo señal, que transfieren la información bidireccionalmente entre procesos. La comunicación en LabView® se hace mediante la sentencia *While-true*, que una vez inicializa las variables, espera por los eventos necesarios para dar transición. El proceso de control se realiza mediante una estructura *Time loop,* con un periodo de muestreo del sistema de control (0,24s). A continuación se describirán los diagramas de programación mencionados.

### 2.2.1. Comunicación serial

Se implementó una comunicación serial con LabView®, que trabaja con una Máquina de Estados Finita tipo Mealy, que evalúa tanto el estado actual como el evento para la transición.

La configuración de la comunicación se realiza mediante el paquete de librerías *VISA*, específicamente con las funciones *VISA Write* y *VISA Read*, que permiten la comunicación con cualquier tipo de instrumento de forma serial.

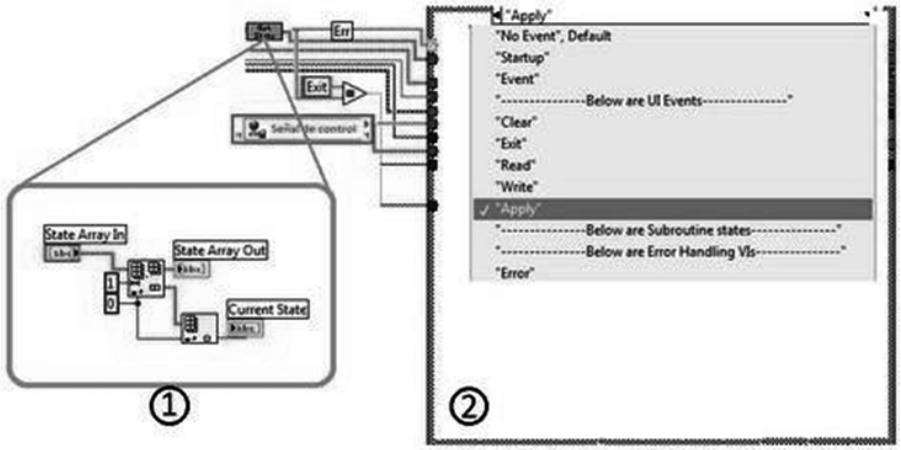


**Figura 10.** Etapa de comunicación serial en LabView® por máquinas de estado.

El estado de aplicación de parámetros ("Aplicar") es el que tiene mayor prioridad, porque si no se ha ejecutado previamente, no puede realizarse ningún tipo de tarea. Este estado solicita la configuración de ciertos parámetros: nombre del puerto, frecuencia de transmisión, cantidad de bits, existencia del bit de paridad y parámetros de control de flujo.

Inmediatamente después de la aplicación de parámetros, la Máquina de Estados queda en un estado de espera (*timeout*), especialmente por el evento *Read*. Para realizar los cambios de eventos, se utiliza una estructura *Case Structure*, que de acuerdo a cambios en las variables de entrada, producen otro en la salida.

En la Figura 10, el bloque 1 corresponde a la visualización de la estructura interna del subVI de la máquina de estados finitos, donde se verifica el evento de entrada y el estado presente. El bloque 2 corresponde a una *case structure* donde se realiza la tarea específica de lectura, escritura o configuración serial, según el estado presente de la máquina de estados.

## Controlador

El controlador es diseñado en Matlab® por un proceso de autosintonía de un PID con estructura fija, para obtener las constantes del controlador (6), que garantizan estabilidad y tiempo de establecimiento de 1,5s. El diagrama de bloques implementado en LabView®, se observa en la Figura 11.

$$K_p = 0{,}2933\,;K_i = 0{,}5679\,;\ K_d = 0 \qquad (6)$$

En la Figura 11 se discrimina cada sección que contiene la etapa de control en LabView®. Con el bloque 1 se habilita la transmisión de señal de control hacia la FPGA; en el bloque 2, *equation block,* se agrega la ecuación en diferencias del controlador PID, ecuación que fue discretizada utilizando el método de aproximación de atraso de Euler y el tiempo de muestreo T=0,24s, obteniendo (7) [23].

Posteriormente, se despejó la ecuación en diferencias para la señal de control, que se muestra en (8).

$$G_c(z) = \frac{U(z)}{E(z)} = K_p + K_i\frac{zT}{z-1} + K_d\frac{z-1}{zT} \qquad (7)$$

$$u(k) = \left(K_p + K_iT + \frac{K_d}{T}\right)e(k) - \left(2\frac{K_d}{T} + K_p\right)e(k-1) + \frac{K_d}{T}e(k-2) + u(k-1) \qquad (8)$$

Los datos que corresponden a valores pasados de cada una de las variables utilizadas en la ecuación en diferencias, se toman de sucesivos *shift register*, que son registros que se van actualizando en cada iteración del *Time Loop*. El bloque 3 corresponde al cambio de valores de la señal de control a valores entre 0 y 255, que serán interpretados por la FPGA, y finalmente en el bloque 4 se habilita o no la acción de control y la transmisión de datos.

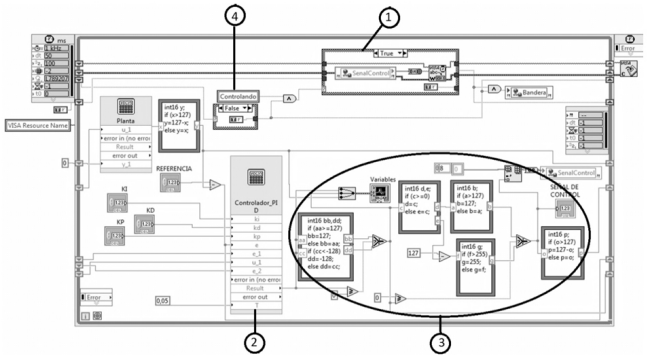


**Figura 11.** Diagrama de control y simulación.

## 3. RESULTADO

Se diseñó e implementó un controlador PID discreto para la planta de primer orden en LabView®, cuya señal de salida se debió saturar para trabajar dentro del rango de 0 a 255 bits, definida por el ancho de transmisión establecido por el protocolo de comunicación serial (1 Byte), en donde el bit más significativo se reservó para indicar el signo.

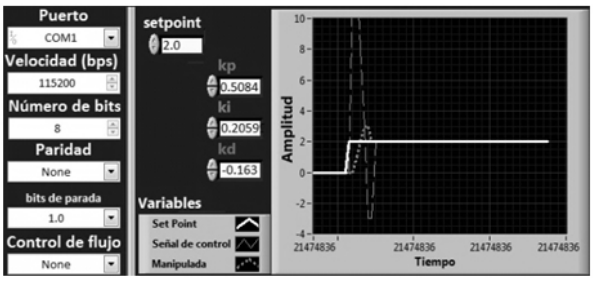


**Figura 12.** Experimento Lazo cerrado de control.

En la Figura 12 se observa el potencial de la integración propuesta, al permitir monitorear con LabView® todas las variables de control y la referencia, evaluando la evolución en el tiempo de la función de transferencia del sistema térmico embebido en la FPGA. Debido a que la FPGA no hace uso del punto flotante como una representación para los números reales, se hizo necesario magnificar considerablemente los coeficientes de la ecuación en diferencias para trabajarlos como números enteros, lo que al final significa facilitar los cálculos en la herramienta utilizada.

Utilizando la estrategia modular propuesta y por etapas, se implementó un sistema de control en lazo cerrado para manipular la dinámica de la planta embebida en la FPGA con el PID de estructura fija diseñado, con el que se cumplió con el parámetro de diseño , referente para la evaluación de desempeño del sistema. Debido a que es una planta de primer orden con realimentación unitaria, el error se hace igual a cero por la acción del integrador en el controlador propuesto.

Al realizar la magnificación de las señales de control y salida, se evidenció la limitación en la manipulación del punto flotante en la FPGA, cuando no se emplean librerías complementarias de la FPGA. En el sistema en lazo cerrado, esto se reflejó como falta de resolución, lo que conllevó a saturaciones de la señal de control por la magnificación de pequeñas variaciones. Además, se generaron valores elevados de error, produciendo también saturación en la señal de salida. Como consecuencia, las pruebas se debieron realizar con valores de referencia en órdenes bajos.

Teniendo en cuenta que la señal de referencia podría variar en bits de 0 a 255, se trabajaron valores entre 0 y 10 bits. Para el caso de la Figura 12, se asignó un valor de referencia de 2, y se observó que la señal de control estabiliza el sistema en 1,5 segundos, tal como se diseñó y se simuló en *Simulink* de Matlab® (Figura 13). Las diferencias con respecto a los datos de simulación hicieron evidentes en los esfuerzos de la señal de control, debido a lo expuesto anteriormente acerca de la magnificación necesaria de la señal de control y la señal de salida de la planta.

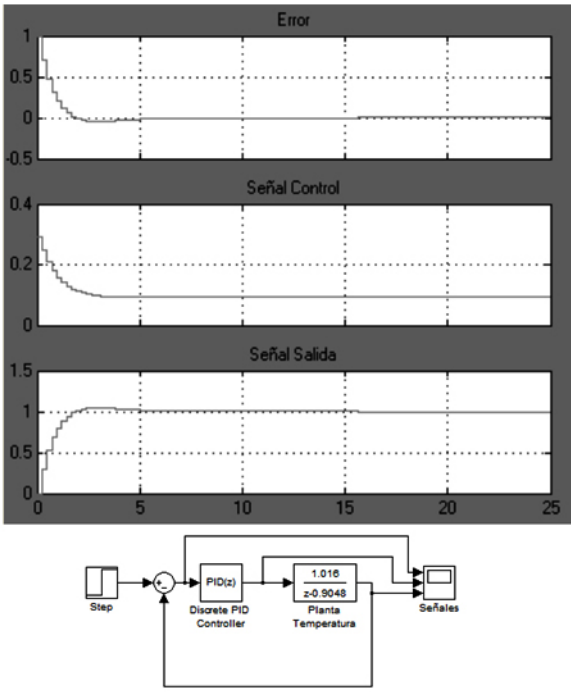


**Figura 13.** Diagrama y simulación en lazo cerrado.

Finalmente, fue posible realizar un experimento para acoplar las dos etapas desarrolladas (FPGA-LabView), empleando la comunicación serial como puente y sincronizador de los tiempos de muestreo para cada evento, que se solicitaban mutuamente la planta embebida el controlador.

## 4. CONCLUSIONES

Tener un sistema ya sea de control o simplemente una planta embebida en una FPGA, permite realizar un monitoreo de las señales en paralelo, por lo que se puede realizar seguimiento al sistema en tiempo real, algo que puede ser práctico en sistemas que lo requieran, ya que se puede definir un proceso independiente dentro de la arquitectura que se encargue de enviar dichas señales que son manipuladas en procesos concurrentes.

Con este trabajo se partió de lo particular para llegar a soluciones más generales, donde se detectaron potenciales dificultades de desarrollo de sistemas con la arquitectura propuesta, desde embeber la planta en la FPGA, hasta definir las velocidades máximas de muestreo, limitadas por la comunicación serial; lo que permitirá en trabajos futuros diseñar sistemas

con dinámicas más complejas, tiempos de respuesta menores y mayor cantidad de periféricos acoplados a la FPGA, donde se implementen sensores reales, aumentando la complejidad del sistema.

La utilización de HIL permite disminuir los ciclos en el desarrollo de un producto, ahorrando costos de manufactura en cuanto a materiales para fabricación de prototipos y protege la integridad de operarios en el desarrollo de pruebas. Además, permite detectar y diagnosticar fallas, para corregirlas de forma oportuna y evaluar protocolos de detección.

## REFERENCIAS

[1] Bacic, M., On hardware-in-the-loop simulation. 44th IEEE Conference on Decision and Contro. Sevilla, España, pp. 3194-3198, Diciembre 2005.

[2] Kohl, S. y Jegminat, D., How to do Hardware-in-the-Loop Simulation Right. SAE World Congress. Detroit, MI, USA, pp. 1-12, Abril 2005.

[3] Waltermann, P., Hardware-in-the-Loop: The Technology for Testing Electronic Controls in Automotive Engineering. 6th Paderborn Workshop Designing Mechatronic Systems. Paderborn, Alemania, pp. 1-20, Abril 2009.

[4] Mauri, M., Castelli, F. y Marchgiani, G., Hardware in the loop (HIL) test bench for small-scale distributed generation systems. IEEE International Symposium on Industrial Electronics. Cambridge, UK, pp. 2177-2182, Junio 2008.

[5] Todd, R. y Forsyth, A., HIL Emulation of All-Electric UAV Power Systems. IEEE Energy Conversion Congress and Exposition. San José, CA, USA, pp. 411-416, Septiembre 2009.

[6] Xiong, L., Zhong, Z., Ai, B. y Hong, Wei., Performance Evaluation for High-Speed Railway Mobile Communication on HIL Simulation platform. 4th IET International Conference on Wireless, Mobile & Multimedia Networks. Beijing, China, pp. 141-144, Septiembre 2011.

[7] Jeschke, S. y Koppers, M., HIL Simulation of electric vehicles in different usage scenarios. IEEE International Electric Vehicle Conference. Greenville, SC, USA, pp. 1-8, Marzo 2012.

[8] Stojcsics, D. and Molnar, A., Fixed-wing small-size UAV navigation methods with HIL simulation for AERObot autopilot. IEEE 9th International Symposium on Intelligent Systems and Informatics. Subotica, Serbia, pp. 241-245, Septiembre 2011.

[9] Tong, Z., Zhang, H., Ye, Z. y Han, J., RCP-based HIL Simulation and Control for 2-DOF Tracking Robot of Maneuvering Target. International Asia Conference on Informatics in Control, Automation and Robotics. Bangkok, Tailandia, pp. 121-125, Febrero 2009.

[10] Pedroni, J. y Cova, W., Implementación del control de orientación con realimentación de velocidad de una tobera pivotante de uso espacial en un ambiente de simulación Hardware-in-the-Loop. ARGENCON, Cordoba, Argentina, pp. 1-6, Junio 2012.

[11] Mertens, F. and Sander T., When Processor and FPGA Work Together, Elektronik Automotive, pp. 1-5, 2011.

[12] Bobe, N., Ploger, M., Puschman, F. and Lillwitz, S., Thing are Speeding up: FPGA-based simulation models for electric drives, Hanser Automotive, pp. 1-5, 2012.

[13] National Instruments. Tutorial: LabVIEW FPGA in Hardware-in-the-Loop Simulation Applications. Available: http://www.ni.com [Citado 10 de Mayo de 2013].

[14] Hui-Ling, P. y Ya-Lin, N., Design of serial communication interface based on FPGA. IEEE International Computer Science and Automation Engineering (CSAE) Conf. Shangai, China, pp. 410-414, Junio 2011.

[15] Machado, F. and Borromeo, S., Diseño de sistemas digitales con VHDL, Universidad Rey Juan Carlos, Madrid, España, 2010.

[16] Altera. Cyclone IV FPGA Starter Development Board Reference Manual. San Jose, CA, USA, Octubre 2006.

[17] Brown, S. and Vranesic, Z., Fundamentals of Digital Logic with VHDL Design, McGraw-Hill, University of Toronto, Toronto, Canada, 2005.

[18] Ogatha, K., Ingeniería de Control Moderna. Prentice Hill 4th ed., Madrid, España, 2003

[19] Quintero-Gómez, M., Uso de Labview para sistemas de control en ingeniería química. Dyna, 169, pp. 150-157, 2011.

[20] Skoog, D., Principios de Analisis Instrumental. Cengage Learning, México D.F., Mexico, 2008.

[21] Cassandras, C. and Lafortune, S., Introduction to discrete event systems. Springer Science+Business Media, New York, NY, USA, 2008.

[22] Perry, D., VHDL: Programming by Example. McGraw-Hill Education, 2002.

[23] Mathworks. Discrete PID Controller Help. Avaliable: http://www.mathworks.com [Citado 10 de Mayo de 2013].