Desarrollo e Implementación de un Planificador de Tiempo Real Ejecutivo Cíclico en un Microcontrolador de Gama Media para Algoritmos de Control con Lazos Independientes

Alfonso R. Alfonsi B., Alfonso S. Alfonsi S.

***Resumen*— Este trabajo presenta el desarrollo e implementación de un planificador de tiempo real ejecutivo cíclico en un microcontrolador, capaz de guiar la ejecución de algoritmos de control con lazos independientes. El desarrollo del software se fundamenta en metodologías orientadas al flujo de datos de tiempo real y en el ciclo de vida iterativo. Como plataforma de hardware se seleccionó *Arduino Diecimila*. Como planta se utilizó un sistema de calefacción de cinco lazos independientes, regulados por algoritmos de control PID (Proporcional, Integral y Derivativo) y planificación ejecutivo cíclico. Las pruebas fueron realizadas en el simulador *ISIS Proteus*, donde se efectuaron experiencias con y sin el planificador, poniendo a prueba el comportamiento del mismo a nivel temporal y lógico. El plan cíclico fue cumplido en su totalidad, respetando las restricciones temporales de cada tarea y garantizando el buen desempeño de los lazos de control.**

***Palabras claves*— Planificación de Tiempo Real, Sistemas de Control Empotrado, Simulación, Microcontroladores.**

I. Introducción

En la electrónica actual, es evidente el impacto que representan los microcontroladores en áreas como la instrumentación, control, automatización industrial, robótica, domótica, e incluso en dispositivos de la vida diaria. Además, [1] afirma que en estos tiempos, existe una orientación estratégica hacia el desarrollo de software de tiempo real en las áreas antes mencionadas.

Un sistema de control basado en microcontroladores, puede atender uno o varios dispositivos externos y sus respectivas funcionalidades. Ante esta situación, una de las vías para el desarrollo del control es utilizar lazos independientes. Adicionalmente estos sistemas, en su mayoría, son catalogados como sistemas de tiempo real, caracterizándose por la obligación de completar sus actividades en determinados plazos de tiempo [2], provocando que el sistema controlado funcione incorrectamente, si lo anterior no se cumple. Para garantizar que todos los lazos independientes de control puedan desempeñar sus actividades dentro de estas restricciones temporales, es necesaria una política de planificación ya sea por ejecutivo cíclico o por prioridades, en donde cada lazo sea considerado como una unidad de ejecución [3]. Como ejemplos destacan los llamados sistemas de control empotrado [4]-[6].

El objetivo que se persigue es presentar el desarrollo e implementación de un planificador de tiempo real ejecutivo cíclico en un microcontrolador de gama media, donde no se necesite una cantidad considerable de memoria, ni de manejo de recursos, como para incluir un núcleo de tiempo real empotrado que guíe la ejecución del sistema.

Para lograr lo anterior se utiliza la estructura global de software *mikro_STR* [7] para desarrollar la aplicación desde sus requerimientos lógicos/temporales, hasta la codificación del programa en el microcontrolador.

Como procesador base se emplea el *ATmega328* [8], de gama media, bajo el sistema de desarrollo *Arduino Diecimila*, de amplia utilización en proyectos empotrados de hardware libre [9]. En cuanto a la planta a controlar se toma un sistema de calefacción *OVEN*, cuyo modelo y características son suministrados en [10].

La organización del artículo está estructurada como sigue. Siguiendo los subsistemas que componen una aplicación basada en microcontroladores, hardware y software [11], las secciones dos y tres están dedicadas a estos puntos. En la sección cuatro se presenta lo referente al desarrollo del sistema de control

II. Subsistema Hardware

Como plataforma de hardware para el desarrollo de la investigación se seleccionó *Arduino Diecimila*, basado en el microcontrolador *Atmel AVR ATmega328*, tomando como

Artículo recibido el 02 de Febrero de 2012. Este artículo forma parte del proyecto de investigación CI-020402-1739-11, financiado por el Consejo de Investigación UDO.

A.R.A.B. y A.S.A.S. están con la Universidad de Oriente, Grupo de Investigación Arquitecturas de Sistemas de Control (GASC), Ave. Argimiro Gabaldón, Barcelona, Estado Anzoátegui, Venezuela. E-mail: alf.rafa@gmail.com, alfonso_alfonsi@udo.edu.ve

referencia los intereses actuales en cuanto al desarrollo e implementación de hardware libre en Venezuela. Arduino es una plataforma de hardware libre basada en una sencilla placa con entradas y salidas (E/S), analógicas y digitales, y en un entorno de desarrollo que implementa el lenguaje *Processing/Wiring*. El *software* de *Arduino*, *Arduino IDE* (*Integrated Development Environment*), se ejecuta en sistemas operativos *Windows*, *Macintosh OSX* y *GNU/Linux*.

Arduino Diecimila es la placa *Arduino* más popular. Su corazón puede estar conformado por el microcontrolador *Atmel AVR ATmega168* o *ATmega328* [12], en la Tabla 1 se presentan algunas de las características técnicas de estos microcontroladores. En la Fig. 1 se muestra la disposición de los pines del microcontrolador para la placa *Arduino Diecimila*. Entre otras características, posee conexión *USB*, un conector para alimentación, una cabecera *ICSP* (*In-Circuit Serial Programming*), y un botón de *reset*. Es un sistema de desarrollo sencillo y de bajo costo que permite la realización de múltiples diseños.

TABLA 1
CARACTERÍSTICAS TÉCNICAS DE LOS MICROCONTROLADORES DE LA PLACA *ARDUINO DIECIMILA*.

Característica	ATmega168	ATmega328
Número de pines	28	28
E/S Digital (máximo)	20	20
Pines de salida *PWM*	6	6
Entradas analógicas	6	6
Comparadores analógicos	1	1
Timers	8 bits(2), 16 bits(1)	8bits (2), 16 bits(1)
Memoria *Flash*	16 KB	32 KB
Memoria *SRAM*	1 KB	2 KB
Memoria *EEPROM*	512 bytes	1 KB
Frecuencia de operación	16 MHz	16 MHz

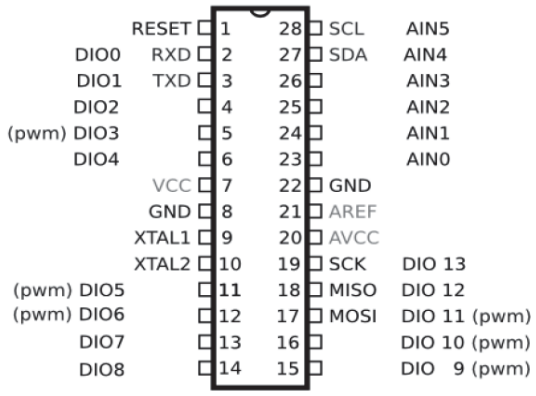

FIG. 1. Disposición de pines para *Arduino Diecimila* con *AVR ATmega168/328*.

III. SUBSISTEMA SOFTWARE

A. Metodologías y Herramientas

Según [13], la aplicación de metodologías para el desarrollo de sistemas se fundamenta en la transformación de una necesidad o idea en un producto, y la selección de alguna de ellas depende del tipo de sistema y de los objetivos que se persiguen. En un sistema de tiempo real, estos procedimientos deben cumplir con las condicionantes en la implementación de este tipo de sistemas, donde el tiempo es fundamental [14].

Para esta investigación, las reglas y patrones que constituyen el diseño y la realización general del software fueron tomadas de [7], en donde el autor desarrolla una estructura global de software, llamada *mikro_STR*, fundamentada en las metodologías de análisis y diseño estructurado orientadas al flujo de datos con extensiones de tiempo real, como lo son *RTSAD (Real Time Structured Analysis and Design)* y *DARTS (Design Approach for Real Time Systems)* [15], apoyado en el ciclo de vida iterativo para este tipo de sistemas (Fig. 2).

Tal y como hace referencia el proceso de análisis y diseño estructurado [16], y sus extensiones de tiempo real [15], se tiene que la estructuración del sistema se divide en dos partes fundamentales: los requerimientos lógicos, los cuales están caracterizados por el modelado del comportamiento de los datos, funciones y control entre el sistema y el ambiente (descomposición funcional del sistema en la forma de Diagramas de Flujos de Datos [16]); y los requerimientos temporales, que permiten predecir el comportamiento y garantizar la ejecución del sistema en el peor caso cuando se aplica un algoritmo de planificación, debido a que dentro del contexto de los sistemas de tiempo real, es necesario eliminar el indeterminismo que produce la ejecución concurrente [13], [17].

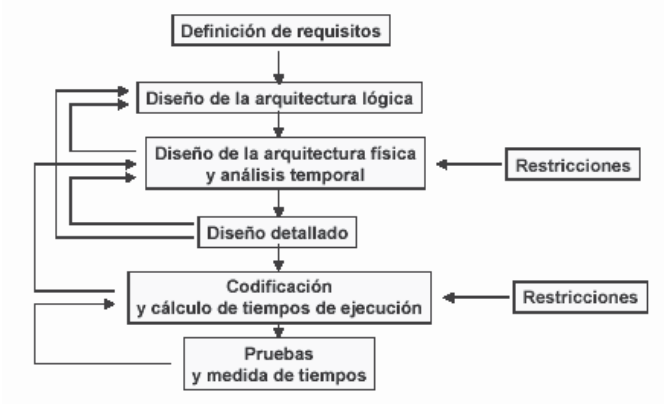

FIG. 2. Proceso de desarrollo iterativo.

Entre las herramientas presentadas en [7] para cumplir los con los requerimientos temporales del sistema se encuentran las pruebas de planificabilidad (las cuales dependerán del planificador que se desee implementar), y los temporizadores e interrupciones propias del microcontrolador, siendo estas dos últimas muy importantes, ya que como es bien sabido, la forma

ISBN: 978-980-7185-02-8

de trabajo de los microcontroladores es secuencial, y la única manera que el planificador pueda en todo momento estar monitorizando y controlando la ejecución de las tareas, es a través de estas herramientas.

Las herramientas computacionales utilizadas fueron, por un lado el entorno de desarrollo *Arduino IDE* [12], donde se realizó el código que fue almacenado en el microcontrolador. Por otro lado, la construcción del plan cíclico fue realizada con la herramienta *CICLIC* [18], [19]. Y por último, para el diseño y simulación de cada una de las etapas y experiencias del sistema se utilizó *ISIS Proteus*, en su versión *Lite* [20].

B. Planificador Ejecutivo Cíclico

Uno de los planteamientos válidos para realizar la implementación de la gestión de tareas en sistemas de tiempo real periódicos lo constituye la utilización de planificadores cíclicos [21]. Un planificador cíclico es un procedimiento iterativo que permite planificar la ejecución de un conjunto de procesos periódicos en un procesador de forma determinista, tal que los tiempos de ejecución sean predecibles.

Los ejecutivos cíclicos o planificadores cíclicos ejecutan las tareas en secuencia según unas tablas de planificación o planes de ejecución, su construcción depende de los requisitos temporales y el conjunto de tareas implicadas en el mismo. A partir de esos requisitos [18] y [21] explican que se debe construir un plan principal, que define la secuencia de tareas que deben ejecutarse durante un periodo fijo de tiempo llamado ciclo principal o hiperperiodo, el cual no es más que el mínimo común múltiplo de los periodos de las tareas. Este plan principal se divide en uno o más planes secundarios los cuales describen la secuencia de tareas que deben ejecutarse durante un periodo fijo de tiempo llamado ciclo secundario. Por lo general los planes secundarios de un mismo plan principal contendrán secuencias de distintas tareas, dependiendo de las características temporales de las tareas [13].

Siguiendo los pasos de la metodología propuesta en [7], se analizaron los requerimientos lógicos y temporales del sistema, necesarios para el desarrollo del planificador ejecutivo cíclico, obteniendo como resultado final, la codificación del algoritmo en el lenguaje propio del sistema de desarrollo. En la Fig. 3 se puede observar la descomposición funcional del planificador de tareas en la forma de diagrama de flujo de datos, correspondiente al desarrollo de los requerimientos lógicos del sistema. La estructura del planificador cíclico queda entonces conformada únicamente por la función Política de planificación, la cual contendrá el algoritmo de la tabla o plan de ejecución de las tareas.

En cuanto a los requerimientos temporales, se tiene como primer paso realizar la prueba por construcción para comprobar si el sistema cumple o no con las restricciones temporales antes de implementarlo; se debe obtener la tabla de planificación o plan de ejecución del sistema que será almacenado en la memoria del microcontrolador. Se incorpora la señal de control Tiempo cumplido (Fig. 3), proveniente de un temporizador interno del microcontrolador, la cual indica al algoritmo de planificación el fin de un ciclo secundario y el comienzo del siguiente, sincronizando el sistema y de alguna manera garantizando la correcta ejecución de la secuencia de tareas correspondiente por cada ciclo secundario. Esta señal de control se realizará en forma de interrupción periódica, donde el período será igual al tiempo de duración del ciclo secundario, obtenido en la prueba por construcción.

FIG.3. Diagrama de flujo de datos para un planificador de tareas ejecutivo cíclico.

Finalmente, la codificación del algoritmo del planificador en el lenguaje del entorno de desarrollo del microcontrolador, cumpliendo los patrones establecidos en los requerimientos lógicos y temporales, queda de la siguiente manera:

```
void ejecutivo_ciclico()
{
  static byte sub_periodo = 1;
  static byte sub_total = n; //n = subperíodos
                             //totales del sistema.
                             //Obtenido de la prueba
                             //por construcción.
  switch(sub_periodo){
    case 1:
      /*
      llamadas a las tareas asociadas
      a este subperíodo
      */
      break;
    case 2:
      /*
      llamadas a las tareas asociadas
      a este subperíodo
      */
      break;
    case n:
      /*
      llamadas a las tareas asociadas
      a este subperíodo
      */
      break;
  }
  if(sub_periodo < sub_total) sub_periodo++;
  else sub_periodo = 1;
}
```

Del código anterior, se resaltan los siguientes aspectos:

• El hiperperiodo está representado por la estructura de control *Switch-Case*, y se repetirá de forma infinita durante la ejecución del sistema.

• Se garantiza la ejecución correcta del hiperperiodo con la estructura de control *if-else* colocada al final de la función.

• La variable *sub_periodo* y la constante *sub_total* están definidas como tipo *byte*, esto con la finalidad de ahorrar espacio en memoria (sólo ocupan un registro en la memoria del microcontrolador).

ISBN: 978-980-7185-02-8

IV. Desarrollo del Sistema de Control

Siguiendo el proceso de desarrollo iterativo de la Fig. 2, a continuación se definen los requisitos funcionales y no funcionales.

A. Requisitos Funcionales

Para probar el funcionamiento del planificador cíclico se diseñó el control para un sistema de calefacción con cinco lazos independientes (cuatro de control y uno de monitoreo), representando cada uno de ellos un conjunto finito de tareas críticas, periódicas, independientes y apropiables; donde el mismo fue validado mediante una serie de simulaciones. La planta utilizada como base es un sistema de calefacción *OVEN* [10], regulado por algoritmos de control PID [22], de amplia utilización en esta área del conocimiento.

El sistema de control de tiempo real propuesto tiene la finalidad o propósito de realizar el control de temperatura de cuatro hornos calefactores idénticos, pero donde cada uno tiene que servir a una temperatura específica diferente. Además, el sistema debe informar al usuario en todo momento el valor de la temperatura a la cual se encuentra cada horno. El usuario tendrá la posibilidad de graduar la temperatura de uno de los cuatro hornos, mientras que los otros tres se mantendrán a una temperatura fija. Cada tarea del sistema estará formada por un lazo independiente, siendo cada una de ellas responsable de manejar a un proceso en particular del sistema. El diagrama esquemático del sistema puede verse en la Fig. 4.

Se establecieron las siguientes magnitudes de parámetros funcionales del modelo, proporcionadas en [10]:

Temperatura ambiente (°C) = 25
Resistencia térmica al ambiente (°C/W) = 0,7
Constante de tiempo del horno (s) = 10
Constante de tiempo del calentador (s) = 1
Coeficiente de temperatura (V/°C) = 1
Potencia de calentamiento (W) = 120

Para obtener la respuesta del sistema en lazo abierto ante una entrada escalón (curva de reacción), se utiliza el sistema de análisis interactivo de *ISIS Proteus*.

Luego, con la respuesta a lazo abierto del sistema, y por el análisis en lazo abierto de Ziegler-Nichols [22], el modelo del sistema de calefacción queda definido como:

$$G(s) = \frac{K_0 e^{-s\tau_0}}{1+\gamma_0 s} = 125{,}5\frac{e^{-s}}{1+12{,}5s} \qquad (1)$$

Donde los coeficientes k_0, τ_0, γ_0 representan la ganancia del proceso, el tiempo muerto y la constante de tiempo, respectivamente. Los parámetros del controlador discreto PID obtenido en [10] a periodo de muestreo $T = 0{,}1$ s según criterio $T < \tau_0/4$, son:

$$a = K_p = 0{,}1195\,;\, b = \frac{K_p T}{T_i} = 0{,}0062\,;\, c = \frac{K_p T_d}{T} = 0{,}6215 \qquad (2)$$

Donde *Kp* representa la ganancia proporcional, *Ti* es la constante de tiempo integral y *Td* es la constante de tiempo derivativa, del controlador PID, quedando éste definido finalmente como:

$$\frac{U(z)}{E(z)} = a + \frac{b}{(1-z^{-1})} + c(1-z^{-1}) \qquad (3)$$

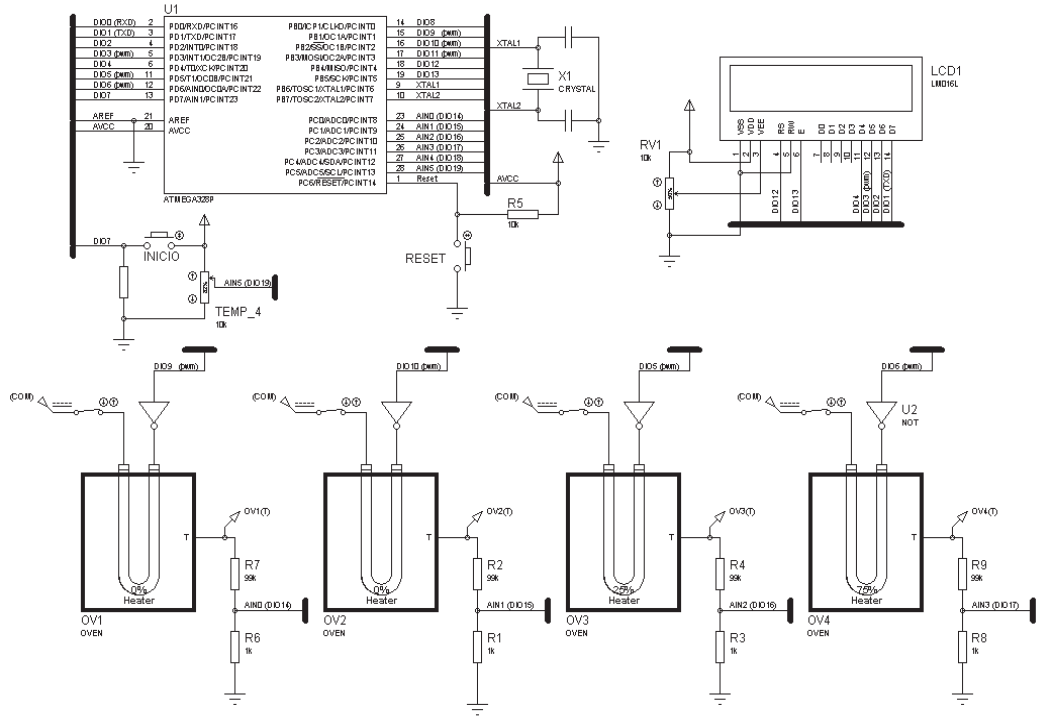

FIG. 4. Diagrama esquemático del sistema de control de calefacción.

ISBN: 978-980-7185-02-8

B. Requisitos No Funcionales

Para estos requisitos se toman los temporales, periodos y tiempo de cómputos. La operación sobre dispositivos es realizada por manejadores implementados como tareas, definidas como el patrón presentado en la Tabla 2, donde *Ci* representa el tiempo de cómputo, *Pi* el período, y *Di* el plazo de finalización.

TABLA 2
TIPOS Y DEFINICIÓN DE TAREAS.

Tareas	Definición
Control	*Ti=(Ci,Pi,Di)*
Visualización	*Tim=(Cim,Pim,Dim)*

C. Arquitectura del Sistema: Arquitectura Lógica

Para la elaboración de la estructura del software del sistema de control de calefacción, fue necesario desarrollar la arquitectura lógica del sistema, en este caso, siguiendo los lineamientos de las metodologías citadas en la sección tres. Esto es, la identificación de entidades externas, descomposición funcional del sistema en la forma de diagramas de flujos de datos, entre otros requisitos, Fig. 5.

D. Arquitectura del Sistema: Arquitectura Física

Para analizar el cumplimiento de los requerimientos temporales del sistema, es necesario conocer de antemano las características temporales de cada tarea, Tabla 2.

Como primer paso, se determinó el tiempo de cómputo estimado. Estos se obtuvieron analizando el código ejecutable, al descomponer en bloques secuenciales, donde cada uno representaba una tarea del sistema. Se asoció a cada tarea un pin digital del microcontrolador, activándolo al inicio de la tarea en cuestión y desactivándolo al finalizar la misma. Se realizó un análisis digital de las salidas configuradas para cada tarea (Fig. 6).

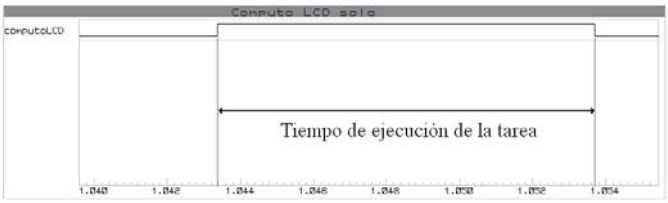

FIG. 6. Análisis digital para los tiempos de cómputo.

El resto de las características temporales fueron determinadas en función del tipo de tarea. En la Tabla 3 se muestra el resumen de las características temporales y la función de cada tarea.

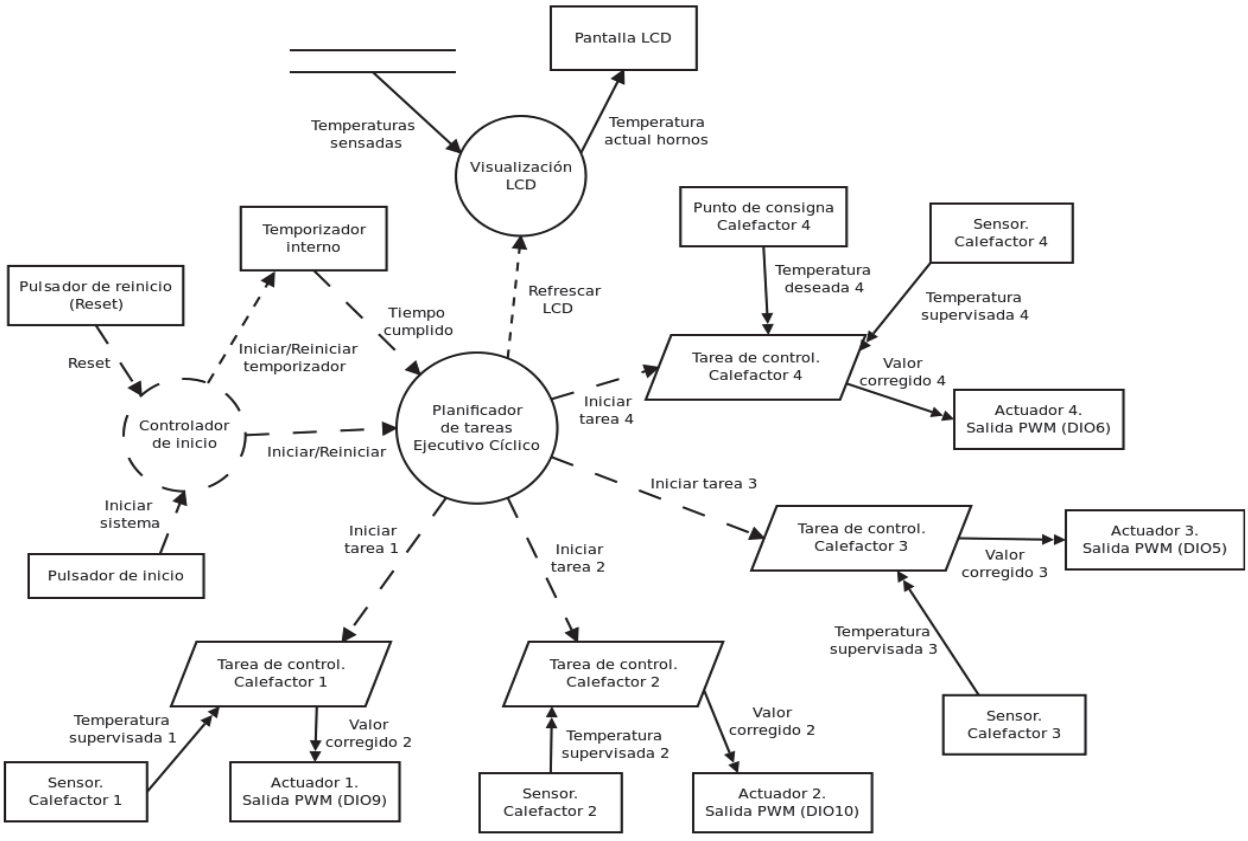

FIG. 5. Diagrama de flujo de datos de nivel 1.

ISBN: 978-980-7185-02-8

TABLA 3
CARACTERÍSTICAS TEMPORALES DE LAS TAREAS DEL SISTEMA.

Tarea	*Ci* (ms)	*Pi* (s)	*Di* (s)	Función
Tarea de control 1	0,3	0,1	0,1	Control PID, horno 1.
Tarea de control 2	0,3	0,1	0,1	Control PID, horno 2.
Tarea de control 3	0,3	0,1	0,1	Control PID, horno 3.
Tarea de control 4	0,5	0,1	0,1	Control PID, horno 4.
Visualización *LCD*	11	1	1	Mostrar en la pantalla el valor actual de las temperaturas.

Luego de establecidas las características temporales de las tareas, se procedió a realizar la prueba por construcción para obtener el diseño del planificador ejecutivo cíclico del sistema. Ésta se realizó mediante el uso de la herramienta *CICLIC*, la determinó seis posibles diseños factibles. Se tomó el de ciclos secundarios igual a 100 ms, que además, fue el recomendado por el programa. El resumen de la planificación se muestra a continuación:

Duración del hiperperiodo (s) = 1
Duración de los ciclos secundarios (ms) = 100
Tiempo ocioso (ms) = 975
Instancias totales = 41
Utilización del procesador = 2,5%

Para la sincronización de los ciclos secundarios, se utilizó el temporizador *Timer2* del microcontrolador, configurado de manera tal que generara una llamada a la función de planificación cada 100 ms. Para ello se usó la librería *MsTimer2.h* de Arduino [23].

E. Diseño Detallado

Después de haber confeccionado el plan de ejecución de las tareas, se procedió a elaborar el código detallado de los módulos que conforman el programa que fue almacenado en el microcontrolador, usando como estructura principal las directrices propuestas en [7].

El código en su totalidad se dividió en dos partes. La primera corresponde al módulo de planificación de tareas ejecutivo cíclico, adaptado el plan cíclico obtenido para sistema de control en cuestión, escrito en un archivo separado, con extensión **.c*. La segunda parte lo conformó el archivo principal del proyecto, con extensión **.pde* (extensión de los proyectos creados en *Arduino IDE*), donde se escribieron los cuerpos de las tareas que conforman al sistema de control, las configuraciones principales del sistema y el microcontrolador (variables globales, constantes, identificación de pines de entradas y salidas, configuración de la pantalla *LCD*), y la rutina principal del programa. El archivo fuente del planificador fue incluido como cabecera dentro de archivo principal del proyecto. A continuación se muestra un fragmento del código asociado al programa principal:

```
//Librerías, archivos de cabecera y variables
#include <MsTimer2.h>
#include <LiquidCrystal.h>
#include "Planificador.c"
boolean ON = 0;
boolean inicio;
int temp1,temp2,temp3,temp4; //temperaturas para LCD
LiquidCrystal lcd(12, 13, 4, 3, 2, 1); //pines LCD

void setup() {
//Función de configuración del sistema.
  pinMode(7,  INPUT);              //botón inicio
//configuración del LCD:
  lcd.begin(16, 2);                //tamaño del LCD
  lcd.print("  ::Stand By::  ");   //inicio LCD
  lcd.setCursor(0, 1);
  lcd.print("  Pulse inicio  ");   //inicio LCD
//Configuración de la interrup de ciclo secundario:
  MsTimer2::set(100,politica);    //Timer2 interrupt
}

void loop() {
  do{             //proceso "Controlador de Inicio"
    inicio = digitalRead(7); //pin pulsador inicio
    if (inicio == LOW) ON = 0;
    else if (inicio == HIGH){
      ON = 1;
      MsTimer2::start(); //iniciar interrupción
periódica
      politica();        //ir a planificador.
    }
  } while (ON == 0);

  while (ON == 1){} //bucle infinito - tiempo ocioso
}
. . .
```

Siguiendo los pasos de la metodología, el código de las tareas de control quedan fuera del bucle principal del programa, éstas son creadas en funciones apartes. A continuación se presenta el código de la función asociada a la tarea de control 1, con el algoritmo de controlador PID incluido. Las tareas restantes poseen una estructura idéntica, el único cambio es el valor de la referencia de temperatura.

```
void tarea_1(){
  static int valor, control;  //ADC y PWM
  static const float a=0.1243;//parametros del PID
  static const float b=0.0062;
  static const float c=0.6215;
  static const float temperatura_limite = 1200.0;
                        //referencia de temperatura
  static float rT,eT,iT,dT,yT,uT,uT1;//variables PID
  static float iT0, eT0 = 0.0;
  static float maximo = 1000.0; float minimo = 0.0;
                        //variables anti-windup

  valor = analogRead(A0);
  yT = 5000.0*(valor/1023.0); //escalizar salida
  temp1 = int(yT/10.0);
  rT = temperatura_limite;
  eT = rT - yT;         //cálculo señal de error
  iT = b*eT + iT0;      //cálculo termino integrativo
  dT = c*(eT - eT0);    //cálculo termino derivativo
  uT = iT + a*eT + dT; //cálculo señal de control
```

```
  uT1 = constrain(uT, minimo, maximo);
  control = map(uT1, minimo, maximo, 0, 255);
  analogWrite(9,control); //señal de control
  iT0 = iT;
  eT0 = eT;
}
```

F. Pruebas

Se confeccionaron dos experiencias. La primera, incorporando el planificador diseñado, bajo la estructura global de software propuesta en [7]. La segunda, usando una estructura convencional de programación para desarrollar los respectivos controles (sin planificación).

Para cada una de ellas se compiló el código en el *IDE* de *Arduino*; el archivo **.hex* obtenido fue almacenado en la memoria del microcontrolador, y se procedió a realizar dos tipos de análisis al sistema. Uno temporal, utilizando sistema de análisis digital (*Digital Analysis*) en modo gráfico del simulador, para observar el orden de ejecución de las tareas del sistema y verificar el cumplimiento de las restricciones temporales de las mismas. El otro análisis se basó en la observación de la respuesta del sistema de calefacción en conjunto en lazo cerrado, ante una entrada escalón de 2 V, el procedimiento y parámetros de la simulación, se muestran a continuación:

• A cada calefactor del sistema (*OVEN*) le fue asignado un punto de consigna diferente, ante la misma entrada escalón. Para los primeros tres, el valor de estado estacionario fue almacenado directamente en la memoria del microcontrolador. Estos valores fueron: Calefactor 1, 120 ºC; Calefactor 2, 90 ºC; Calefactor 3, 150 ºC.

• El valor de estado estacionario de la temperatura del Calefactor 4, el cual es leído por el sistema de forma externa, se estableció de la siguiente manera para la prueba: al inicio del sistema se colocó en su valor máximo, 200 ºC, y transcurrido la mitad del tiempo del análisis, el punto de consigna fue modificado a aproximadamente a una temperatura de 144 ºC.

• El tiempo total de simulación fue de 150 segundos del análisis, lo suficiente como para observar completamente las dinámicas de todas las plantas.

Para visualizar la respuesta de cada calefactor, se utilizó de nuevo la herramienta de modo gráfico, pero esta vez realizando un análisis interactivo (*Interactive Analysis*).

V. Resultados y Discusión

A. Incorporando el Planificador Ejecutivo Cíclico

En la Fig. 7 y Fig. 8 se pueden apreciar los resultados del análisis digital realizado en el simulador, incorporando el planificador ejecutivo cíclico. La respuesta del sistema para los cuatro calefactores (análisis interactivo) se muestra en la Fig. 9. Como se puede ver en el análisis digital de la simulación (Fig. 7 y Fig. 8), el plan cíclico creado para el conjunto de tareas fue cumplido en su totalidad, respetando las restricciones temporales y el orden de activación de cada tarea.

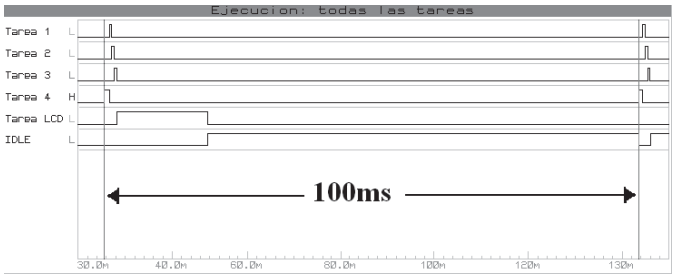

FIG. 7. Período de activación de ciclos secundarios.

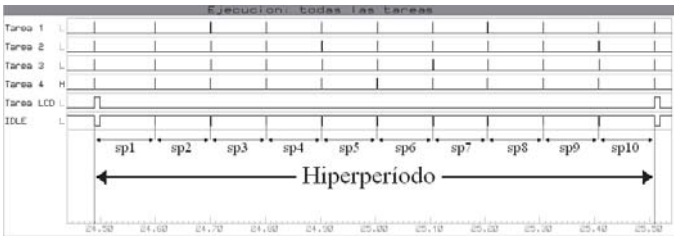

FIG. 8. Ejecución de las tareas.

En cuanto al análisis de la respuesta del sistema a lazo cerrado, se observa que los controladores PID responden de manera muy eficiente, siguiendo las referencias impuestas. Las cuatro temperaturas lograron posicionarse en el valor deseado transcurrido cierto tiempo, a pesar de tener como entrada fija un escalón de 2 V que lleva la temperatura de los hornos a 276 ºC de forma natural (Fig. 9). Se puede decir entonces que además de cumplir con las restricciones temporales, se lograron concretar todas las acciones de control del sistema.

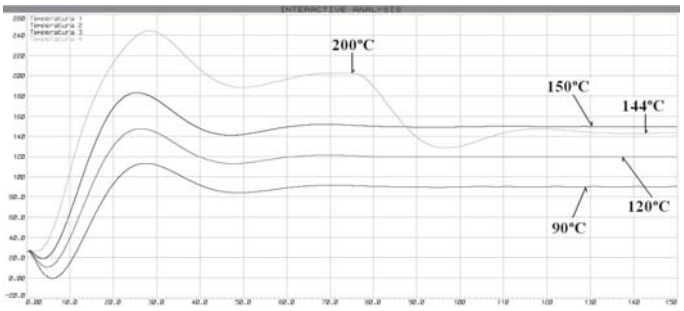

FIG. 9. Respuesta a lazo cerrado del sistema de calefacción usando el planificador ejecutivo cíclico.

De igual manera, la rutina de visualización en la pantalla *LCD* se llevó a cabo de acuerdo a lo previsto. Por tratarse de una tarea de visualización de datos, no es posible realizar un tipo de análisis más profundo para corroborar lo propio. Sin embargo, con en el análisis digital realizado se puede comprobar que la rutina de visualización en la pantalla *LCD* se ejecutaba en los lapsos de tiempo previstos.

B. Incorporando Programación Convencional

Para esta segunda experiencia, el código fue escrito de la manera como se vienen programando normalmente, es decir,

ISBN: 978-980-7185-02-8

sin ningún tipo de metodología en específico. El programa en su totalidad está conformado por un sólo bloque secuencial donde se encuentran todas las tareas del sistema, en este caso, para el entorno de programación *Arduino IDE*, todo el código se escribió en la función *loop*. Al igual que como se hizo para la primera prueba, se realizó tanto el análisis temporal (Fig. 10), como el de la respuesta del sistema a lazo cerrado (Fig 11).

Las pruebas de análisis digital fueron realizadas bajo las mismas condiciones, para un tiempo de corrida total de 1 segundo (duración del hiperperiodo). Como se pudo observar en el análisis digital, existen demasiadas activaciones en instantes muy cortos de tiempo para cada tarea, lo cual se tradujo en la inestabilidad total de los calefactores a lazo cerrado (Fig. 11), esto es debido a que los parámetros de la planta discretizada, están adecuados al período de muestreo establecido en la sección anterior. Al no cumplir con estos requisitos el modelo de la planta cambia, y por ende conduce a la inestabilidad.

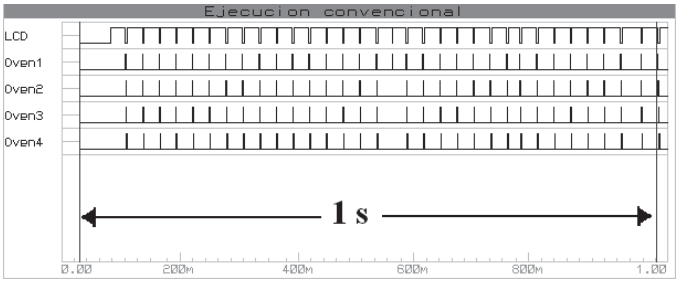

FIG. 10. Análisis digital: Codificación convencional.

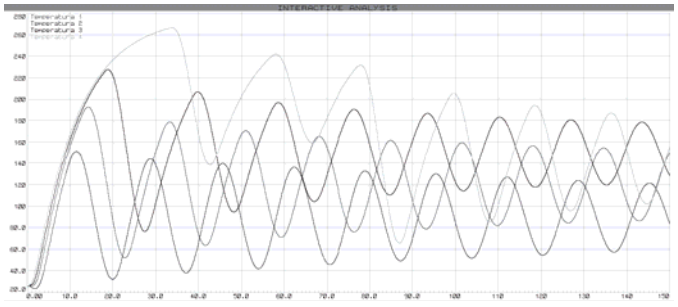

FIG. 11. Respuesta a lazo cerrado del sistema de calefacción usando programación convencional.

Otra cosa que se puede resaltar, es que el hecho de tener que actualizar muy seguido los datos en la pantalla *LCD* para la función de visualización de los datos, tiende a parpadear con mucha frecuencia, haciendo ilegible en varios instantes de tiempo, los datos presentados.

VI. Conclusiones

Se diseñó y codificó un algoritmo de planificación ejecutivo cíclico, siguiendo los lineamientos planteados en [7], para sistemas basados en microcontroladores de gama media, y apoyándose en la herramienta *CICLIC* para la generación de las alternativas de planificación, cumpliendo con la prueba de construcción que exige este algoritmo.

Se demostró la validez del planificador cíclico desarrollado, mediante la simulación de un sistema de control de tiempo real, tomando como base un sistema *OVEN* con cinco lazos de control, implementado en un microcontrolador *AVR ATmega328*, bajo el sistema de desarrollo Arduino. Las pruebas del sistema se realizaron en el simulador de circuitos *ISIS Proteus*. Se determinó con las simulaciones que utilizando las políticas de planificación y las metodologías correctas para el desarrollo del software, se garantiza el buen desempeño de las actividades de los lazos de control (plazos de ejecución), no siendo así al desarrollarlo de manera convencional, donde los sistemas se vuelven inestables.

Referencias Bibliográficas

[1] E. Uzcátegui y D. Dinarle. **"Metodologías de desarrollo para sistemas de tiempo real. Un estudio comparativo"**. Revista Universidad, Ciencia y Tecnología, Universidad Nacional Experimental Politécnica "Antonio José de Sucre" (UNEXPO), Venezuela, Vol. 13, pp. 59-66, Marzo 2009.

[2] A. Crespo y A. Alonso. **"Una panorámica de los sistemas de tiempo real"**. Revista Iberoamericana de Automática e Informática Industrial, Comité Español de Automática (CEA) y Universidad Politécnica de Valencia (UPV), España, Vol. 3, pp. 7-18, 2006.

[3] L. Sha et al. **"Real time scheduling theory: a historical perspective"**. Real time system. Kluwer Academic Publishers, The Netherlands, Vol. 28, pp. 101-155, 2006.

[4] A. Noriega, C. Silva y A. Ordaz. **"Implementación de un algoritmo de control adaptable en un microcontrolador"**. Memorias del VI Coloquio de Automatización y Control, Mérida, 2008, pp. 602-609.

[5] J. Ospina et al. **"A tutorial introduction to embedded model control"**. Memorias del VI Coloquio de Automatización y Control, Mérida, 2008, pp. 563-570.

[6] M. Castillo, I. Campos y J. Barbosa. **"Controlador PID digital con el PIC16F877"**. Memorias del Encuentro de Investigación en Ingeniería Eléctrica, Zacatecas, 2006, pp. 111-115.

[7] A.R. Alfonsi. **"Diseño de una estructura global de software con gestión de tareas de tiempo real para aplicaciones de control"**. Tesis de Pregrado, Departamento de Electricidad, Universidad de Oriente, Anzoátegui, Venezuela, 2011.

[8] Atmel AVR. (Consulta en abril, 2011). **"ATmega48A/PA/88A/PA/168A/PA/328/P datasheet. Rev. 8271C–AVR–08/10"**. Disponible en www.atmel.com

[9] Fundación Cenditel (Consulta en junio, 2011). **"Proyecto de Hardware Libre"**. Disponible en www.cenditel.gob.ve

[10] Primer Congreso Virtual de Microcontroladores y sus Aplicaciones (Consulta en junio, 2011). **"Método básico para implementar un controlador digital PID en un microcontrolador PIC para desarrollo de aplicaciones a bajo costo"**. Disponible en www.electrosoft.com.ve/PrimerCongresoDeMicrocontroladores.

[11] Y. Goncalves et al. **"Titulador potenciométrico digital automático para experimentos de equilibrios en disolución"**. Revista Ciencias e Ingeniería, Universidad de los Andes (ULA), Venezuela, Vol. 32, pp. 23-30, Diciembre-Marzo 2011.

[12] Arduino (Consulta en abril, 2011). **"Arduino diecimila"**. Disponible en www.arduino.cc/en/Main/ArduinoBoardDuemilanove

[13] P. Laplante. **"Real-time systems design and analysis"**. 3º Ed. United States of America: IEEE Press and Willey-Interscience, 2004.

[14] A. Alfonsi y J. Pérez. **"Sistema de control en tiempo real para una planta piloto compacta usando software libre"**. Revista Universidad, Ciencia y Tecnología, Universidad Nacional Experimental Politécnica "Antonio José de Sucre" (UNEXPO), Venezuela, Vol. 13, pp. 189-198, Septiembre 2009.

[15] H. Gomaa. **"Software design methods for real-time systems"**. SEI Curriculum Module SEI-CM-22-1.0. United States of America: Carnegie Mellon University, 1989.

[16] R. Pressman. **"Ingeniería del software. Un enfoque práctico"**. 5ª Ed. España: McGraw-Hill Interamericana de España, S. A., 2002.

[17] F. Cottet, J. Delacroix, C. Kaiser and Z. Mammeri. **"Scheduling in real-time systems"**. United Kingdom: John Wiley & Sons Ltd, 2002.

[18] J. Yépez, J. **"Diseño e implementación de una herramienta para la planificación de ejecutivos cíclicos"**. Proyecto fin de carrera, Departamento de Ingeniería de Sistemas, Automática e Informática Industrial, Universidad Politécnica de Cataluña, Barcelona, España, 2007.

[19] J. Yépez et al. **"CICLIC: A tool to generate feasible cyclic schedules"**. Proceedings of the 6th IEEE International Workshop on Factory Communication Systems (WFCS06), Torino, 2006.

[20] Proteus Lite (Consulta en mayo, 2011). **"CAD para sistemas digitales Proteus"**. Disponible en www.ieeproteus.com/descarga.html#lite

[21] T. Baker and A. Shaw. **"The cyclic executive model and Ada real time systems"**. IEEE, Vol. 1, pp. 120-129, 1989.

[22] K. Ogata. **"Sistemas de control en tiempo discreto"**. 2º Ed. México: Prentice Hall Hispanoamericana S.A., 1996.

[23] Arduino (consulta en junio, 2011). **"Librerías"**. Disponible en http://arduino.cc/es/Reference/Libraries

ISBN: 978-980-7185-02-8