Universidad de Oriente.

Nucleó Anzoátegui.

Escuela de Ingeniería y Ciencias Aplicadas.

Departamento de Ingeniería.

Arquitectura Avanzada del Computador.



Desarrollo de una Aplicación para el Control de Temperatura en Hornos de una Empresa ubicada en la zona norte del Estado Anzoátegui.

Tutor Profesor Ing. Alfonso Alfonsí

Grupo N° 1. Integrantes:

María Cardoza C.I: 6.142.718

Luis Correa C.I: 19.840.230

Copyright © 2015 por Luis Correa & María Cardoza. Todos los derechos reservados.

Barcelona, 22 de Junio de 2015

**Tabla de Contenidos**

**I.** **Introducción** 3

**II.** **Subsistema de Hardware** 3

**III.** **Subsistema de software** 4

A. Metodologías y Herramientas 4

B. Planificador Main 5

**IV.** **Desarrollo del sistema de control** 7

A. Requisitos Funcionales 7

B. Requisitos No Funcionales 7

C. Arquitectura del Sistema: Arquitectura Lógica 8

D. Arquitectura del Sistema: Arquitectura Física 9

E. Diseño Detallado 10

F. Pruebas 12

**V.** **Resultados y discusión** 12

**VI.** **Conclusiones** 13

**VII.** **Lista de referencias** 14

**VIII.** **Glosario de términos** 15

# **Introducción**

En el desarrollo de los sistemas electrónicos, 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 (sensores que detectan movimientos y prenden bombillos). Además se afirma que en estos tiempos existe una orientación estratégica hacia el desarrollo de software para la codificación y control de los microcontroladores. Un ejemplo de ello es el Arduino IDE, que es un software con aplicaciones 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.

El objetivo que se persigue es presentar el Desarrollo de una Aplicación para el Control de Temperatura en Hornos de una empresa ubicada en la zona norte del Estado Anzoátegui, donde no se necesite una cantidad considerable de memoria, ni de manejo de recursos.

Para lograr lo anterior como procesador base se emplea el Atmega328P de gama media, bajo el sistema de desarrollo Arduino Uno, de amplia utilización en proyectos empotrados de hardware libre. En cuanto a los Hornos, cuya temperatura se va a controlar se hará mediante el integrado LM35 utilizado como sensor de temperatura.

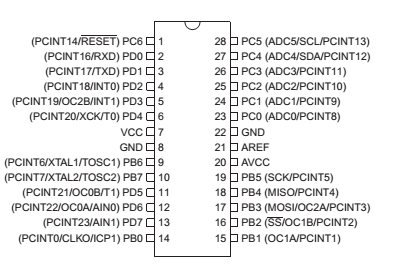
Este trabajo está estructurado de la siguiente manera:

* Subsistema Hardware.
* Subsistema Software.
* Desarrollo del Sistema de Control.

# **Subsistema de Hardware**

Como plataforma de hardware para el desarrollo de la investigación se seleccionó Arduino Uno, tomando como referencia los intereses actuales en cuanto al Desarrollo e Implementación de Hardware Libre en Venezuela. El cómo software para el control del Arduino se usó el Arduino IDE.

Arduino Uno es la placa Arduino más popular. Su cerebro puede estar conformado por el microcontrolador Atmel AVR ATmega328P. En la Figura N° 1 se muestra la disposición de los pines del microcontrolador. Entre otras características el Arduino Uno posee una conexión USB, un conector para la 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, en la tabla 1 se muestran las características técnicas del ATmega328P.



**Figura N° 1. Arduino Uno ATmega328P**

**Tabla N° 1. Características técnicas**

**de los microcontroladores ATmega328P**

|  |  |
| --- | --- |
| **PARÁMETROS** | **VALORES** |
| Flash | 32 Kbytes |
| RAM | 2 Kbytes |
| Cantidad Pines | 28 |
| Frecuencia máxima de operación | 20 MHz |
| CPU | 8-bit AVR |
| Número de Canales variables | 16 |
| Pines máximos de E/S | 26 |
| Interrupciones externas | 24 |

# **Subsistema de software**

### Metodologías y Herramientas

Desarrollo de una Aplicación para el Control de Temperatura en Hornos de una empresa ubicada en la zona norte del Estado Anzoátegui.

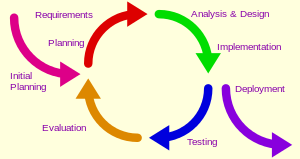
Según la aplicación de metodologías para el desarrollo de un Sistema para el Control de Temperatura en Hornos, dependerá del tipo de sistema y de los objetivos que se persiguen. En un sistema embebido estos procedimientos deben cumplir con las condicionantes en la implementación de este tipo de sistemas.

La estructura del sistema se divide en dos partes fundamentales:

* Los requerimientos lógicos, los cuales están caracterizados por el Modelado del Comportamiento de Datos, Funciones y Control entre el Sistema y el ambiente (descomposición funcional del sistema en la forma de Diagramas de Flujo de Datos).
* 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.

Los procedimientos que se deben cumplir con las condicionantes en la implementación (ver figura 1) son:

* Planificación inicial.
* Planificación y Requerimiento.
* Análisis y Diseño, Implementación 🡪 Despliegue.
* Prueba.
* Evaluación 🡪 Iteración.



**Figura N° 2. Proceso de Desarrollo Iterativo.**

Entre las herramientas presentadas para cumplir con los requerimientos temporales del sistema se encuentran las pruebas y los temporizadores e interrupciones propias del microcontrolador, siendo estas dos últimas muy importantes, ya que como es bien sabido, la forma 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 la tareas, es a través de estas herramientas.

Las herramientas computacionales utilizadas fueron el Arduino IDE, donde se realizó el código que fue almacenado en el microcontrolador, para la simulación de cada una de las etapas del sistema se utilizó ISIS Proteus, en su versión 7.2.

### Planificador Main

Uno de los planeamientos válidos para realizar la Implementación de la Gestión de Tareas en Sistemas Embebidos lo constituye la utilización de Planificadores Cíclicos. 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.

El Planificador Main ejecuta 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.

****

**Figura N° 3. Diagrama de Flujo de Datos para un**

**Planificador Main**

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:

byte sub\_periodo=1;

const byte sub\_total=4;

void ejecutivo\_ciclico(){

switch(sub\_periodo){

case 1:

//código

break;

case 2:

//código

break;

case 3:

//código

break;

case 4:

//código

break;

}

sub\_periodo=(sub\_periodo==sub\_total)?1: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 el uso de ternarios 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 (solo ocupan un registro en la memoria del microcontrolador).

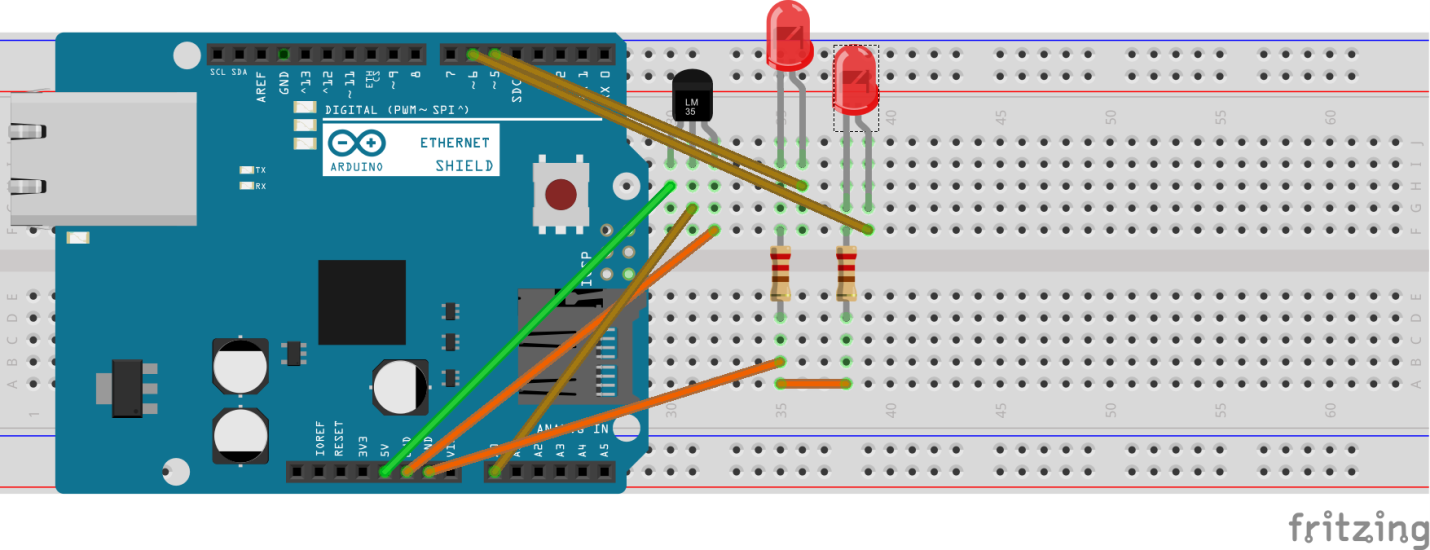
# **Desarrollo del sistema de control**

### Requisitos Funcionales

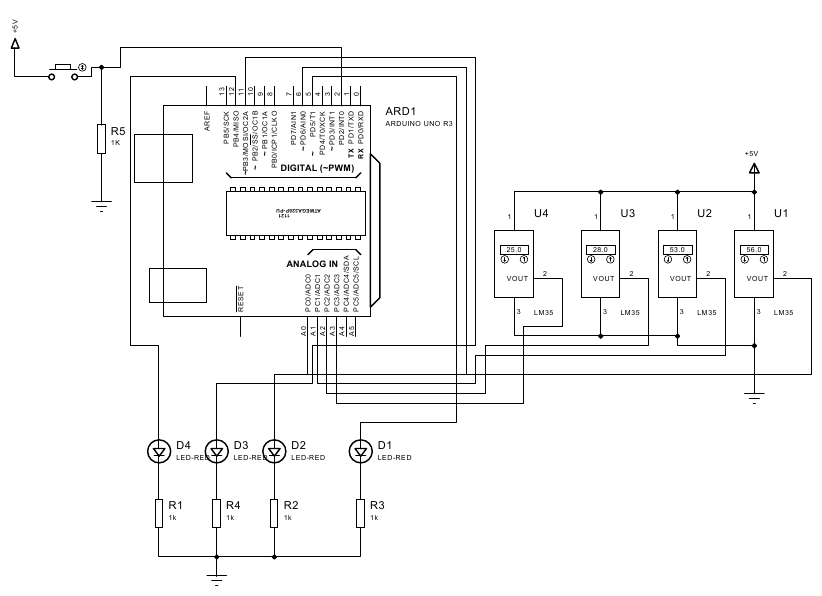
* El sistema debe informar al usuario en todo momento el valor de la temperatura a la cual se encuentra cada horno.
* Para el Sistema de Control se utilizó cinco lazos independientes (cuatro de control y uno de monitoreo), representando cada uno de ellos un conjunto finito de tareas críticas, periódicas, independientes y apropiables; donde el mismo fue validado mediante una serie de simulaciones.
* Para la planta se utiliza como base un sistema de control con el sensor LM35, de amplia utilización en esta área del conocimiento.
* Cada tarea del sistema estará formada por un lazo independiente, siendo cada una de ellas responsable de manejar un proceso en particular del sistema.

### Requisitos No Funcionales

* La operación sobre dispositivos es realizada por manejadores implementados como tareas.

****

**Figura N° 4.Diagrama Esquemático del Sistema de Control de Temperatura.**



**Figura N° 5.Diagrama Esquemático del Sistema de**

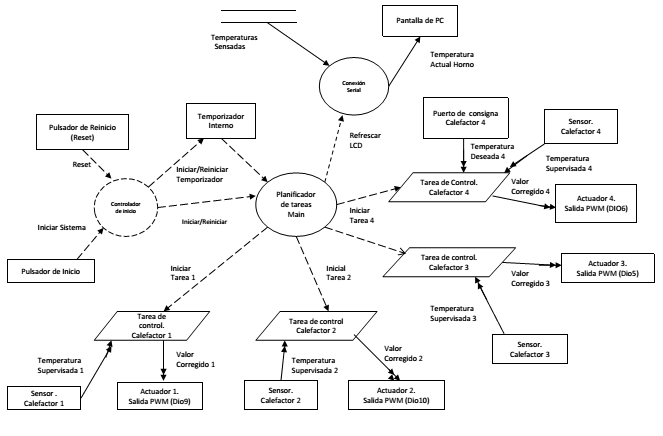
**Control con cuatro sensores de temperatura.**

### Arquitectura del Sistema: Arquitectura Lógica

Para la elaboración de la estructura del software del Sistema de Control de Temperatura, 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 Figura número 6.

### Arquitectura del Sistema: Arquitectura Física

Como primer paso 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 puede visualizar el Diagrama de Flujo de Datos de la Figura número 5.



**Figura N°6. Diagrama de Flujo de Datos nivel 1.**

### 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.

const byte sensor1 = 0;

const byte sensor2 = 1;

const byte sensor3 = 2;

const byte sensor4 = 3;

const byte ledRojo1 = 5;

const byte ledRojo2 = 6;

const byte ledRojo3 = 11;

const byte ledRojo4 = 12;

byte sub\_periodo=1;

const byte sub\_total=4;

unsigned long miliVolts;

unsigned long temperatura;

unsigned int brillo;

boolean inicio;

const byte botoninicio=13;

volatile boolean ON=LOW;

void setup(){

Serial.begin(9600);

pinMode(botoninicio,INPUT);

pinMode(ledRojo1,OUTPUT);

pinMode(ledRojo2,OUTPUT);

pinMode(ledRojo3,OUTPUT);

pinMode(ledRojo4,OUTPUT);

attachInterrupt(0, parpadeo, LOW);// el 0 pin digital 2

}void loop(){

switch(ON){

case HIGH:

main();

break;

case LOW:

analogWrite(ledRojo1,0);

analogWrite(ledRojo2,0);

analogWrite(ledRojo3,0);

analogWrite(ledRojo4,0);

break;

}

}void parpadeo(){

ON=!ON;

}void main(){

switch(sub\_periodo){

case 1:

miliVolts = analogRead(sensor1)\*5000L/1023;

temperatura = miliVolts/10;

brillo=map(temperatura,10,40,0,255);

brillo = constrain(brillo,0,255);

//frio valor brillo muy bajo

analogWrite(ledRojo1,brillo);

Serial.print("Temperatura calentador 1 ");

Serial.print(temperatura);

Serial.println(" grados");

delay(1000);

break;

case 2:

miliVolts = analogRead(sensor2)\*5000L/1023;

temperatura = miliVolts/10;

brillo=map(temperatura,10,40,0,255);

brillo = constrain(brillo,0,255);

//frio valor brillo muy bajo

analogWrite(ledRojo2,brillo);

Serial.print("Temperatura calentador 2 ");

Serial.print(temperatura);

Serial.println(" grados");

delay(1000);

break;

case 3:

miliVolts = analogRead(sensor3)\*5000L/1023;

temperatura = miliVolts/10;

brillo=map(temperatura,10,40,0,255);

brillo = constrain(brillo,0,255);

//frio valor brillo muy bajo

analogWrite(ledRojo3,brillo);

Serial.print("Temperatura calentador 3 ");

Serial.print(temperatura);

Serial.println(" grados");

delay(1000);

break;

case 4:

miliVolts = analogRead(sensor4)\*5000L/1023;

temperatura = miliVolts/10;

brillo=map(temperatura,10,40,0,255);

brillo = constrain(brillo,0,255);

//frio valor brillo muy bajo

analogWrite(ledRojo4,brillo);

Serial.print("Temperatura calentador 4 ");

Serial.print(temperatura);

Serial.println(" grados");

delay(1000);

break;

}

sub\_periodo=(sub\_periodo==sub\_total)?1:sub\_periodo+1;

}

### Pruebas

Se confeccionaron dos experiencias.

* La primera generando el hex en el Arduino IDE y luego realizando la simulación en el software ISIS Proteus.
* La segunda generando el hex en el Arduino IDE y luego cargando el programa en el Arduino Uno para poder comprobar físicamente el funcionamiento del código y del circuito.

# **Resultados y discusión**

Se pueden apreciar los resultados del análisis digital realizado en el simulador, incorporando el Planificador Main. La respuesta del sistema para los cuatro Hornos (análisis interactivo) se muestra en la Figura N° 9. 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.

En cuanto al análisis de la respuesta del sistema a lazo cerrado, las cuatro temperaturas lograron posicionarse en el valor deseado transcurrido cierto tiempo, se puede decir entonces que además de cumplir con las restricciones temporales, se lograron concretar todas las acciones de control del sistema.

De igual manera, la rutina de visualización en la pantalla del computador se llevó a cabo de acuerdo a lo previsto. Por tratarse de una tarea de visualización de datos, se pudo realizar un análisis profundo para corroborar lo propio al poder cargar el programa en el Arduino Uno y mostrar que funcionaba lo simulado.

# **Conclusiones**

* El Desarrollo de una Aplicación para Control de Temperatura en Hornos ha permitido comprobar las variaciones de temperatura y dar respuesta a las temperaturas no deseadas.
* Mediante la aplicación creada, se pudo visualizar por el computador la conexión serial del Arduino Uno y las variaciones de temperatura por el computador.
* Mediante la construcción del circuito se pudo cargar el programa en el Arduino Uno y realizar las conexiones necesarias para que el circuito funcionara correctamente.

# **Lista de referencias**

* Alfonsí, A. (2013). Técnica Dinámica para Ajustar las Necesidades Energéticas de los Sistemas Empotrados de Control de Tiempo Real Autónomos. Trabajo de Ascenso Profesor Titular. Departamento de Computación y Sistemas. Universidad de Oriente, Núcleo Anzoátegui, Barcelona, Venezuela.
* James Rumbaugh, Ivar Jacobson, Grady Booch 2007. El Lenguaje de Modelado Unificado Ed 2. Madrid. Addison Wesley.

# **Glosario de términos**

* Arduino Uno: Es una plataforma de [hardware libre](https://es.wikipedia.org/wiki/Hardware_libre), basada en una [placa](https://es.wikipedia.org/wiki/Circuito_impreso) con un [microcontrolador](https://es.wikipedia.org/wiki/Microcontrolador) y un [entorno de desarrollo](https://es.wikipedia.org/wiki/Entorno_de_desarrollo), diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares, basado en el microcontrolador Atmel AVR ATmega328P, implementa el lenguaje Processing/Wiring, con el cual se puede crear el circuito (hardware para detectar temperatura) y cargar el hex creado con el programa Arduino IDE.
* Arduino IDE (Integrated Develoment Enviromment): Entorno de Desarrollo Integrado, se ejecuta en sistemas operativos Windows, OS X, Linux, para programar con el lenguaje Processing/Wiring para las placas de Arduino.
* Atmega328P: Es un [chip](https://es.wikipedia.org/w/index.php?title=Circuito_Integrado&action=edit&redlink=1) [microcontrolador](https://es.wikipedia.org/wiki/Microcontrolador) creado por [Atmel](https://es.wikipedia.org/wiki/Atmel) y pertenece a la serie [megaAVR](https://es.wikipedia.org/w/index.php?title=MegaAVR&action=edit&redlink=1).
* ISIS Proteus: Es una [compilación](https://es.wikipedia.org/wiki/Suite_ofim%C3%A1tica) de [programas](https://es.wikipedia.org/wiki/Software) de diseño y simulación [electrónica](https://es.wikipedia.org/wiki/Electr%C3%B3nica), desarrollado por [Labcenter Electronics](https://es.wikipedia.org/w/index.php?title=Labcenter_Electronics&action=edit&redlink=1) que consta de los dos programas principales: Ares e Isis, y los módulos VSM y Electra.