

UNIVERSIDAD DE LA CUENCA DEL PLATA

Facultad de Ingeniería y Tecnología

Arquitectura de Computadoras

ESTUDIANTES:

Tarnowski, Tobías

Gonzalez, Joaquín

Piedrafita, Augusto

CARRERA: Ingeniería en Sistemas de Información

CÁTEDRA: Arquitectura de Computadoras

AÑO: Primero

PROFESOR COMISIÓN A:

Ing. Juan de Dios Benitez

Trabajo Práctico Integrador

Cátedras de Arquitectura de Computadoras, Análisis Matemático y Programación estructurada.

Justificación:

Los alumnos a lo largo de las asignaturas estuvieron viendo las herramientas tecnológicas y matemáticas que le permitirán llevar a cabo un proyecto el cual abarque las tres asignaturas; para esto se elige el desarrollo e implementación de un controlador PID (Progresivo – Integrador – Diferencial) el cual hace uso de los conceptos aprendidos en las asignaturas.

Contexto:

Se requiere controlar la temperatura de un horno eléctrico a un valor estable y confiable, para poder hacer uso de este como horno de resoldado de placas con sistemas de montajes superficial BGA. Para ello, se considera que la mejor forma de lograr eso es con la implementación de un control de temperatura a través de un PID. Para esto se solicita a los alumnos de 1° año de ingeniería en sistemas de información que desarrollen un prototipo de control PID utilizando la tecnología ARDUINO que cumpla con las siguientes características:

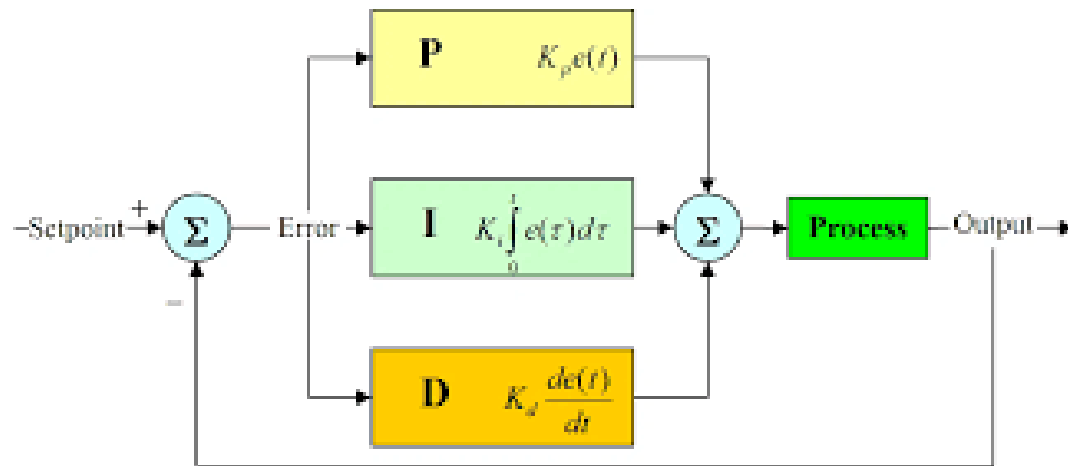
- Temperatura final estable (Set_Point) : Configurable en el rango de 150° a 350°
- Representación por medio de un display de la temperatura actual del horno vs la temperatura se Set_Point
- Banda proporcional: configurable entre 3% y 15% del Set_Point

El horno en cuestión tiene las siguientes características:

- Con las resistencias que cuenta el horno cuando se encienden las mismas la temperatura promedio del horno asciende a una tasa de 30° por minuto
- Una vez apagadas las resistencias, el horno tiene una inercia térmica que continúa con la tendencia ascendente o descendente de 10 segundos.
- Luego del tiempo de inercia térmica, el horno posee una aislación que le permite mantener la temperatura adentro por un lapso de 2 min y luego comienza a descender por perdidas de aislamiento a una tasa de 3° por minuto promedio.
- La temperatura máxima que puede alcanzar el horno es de $T_{max} = 500^{\circ}$
- La temperatura inicial a la cual comienza a levantar temperatura es la temperatura ambiente
- El horno tiene un tiempo característico (tiempo que transcurre hasta que alcanza el 63,3% de la temperatura máxima) igual a 10 minutos

El módulo PID trabaja con PWM por lo que el tiempo de ciclo del PWM tiene que ser mucho menor que el tiempo característico del horno, por lo menos 10 veces menor por lo que se debe setear el Tc entre los rangos de hasta 60 segundos.

El esquema del sistema tiene esta forma



Consignas Arquitectura de Computadoras:

Deberá realizar el prototipo en Arduino simulando todas las variables a través de temporizadores, al mismo tiempo que deberá ir mostrando en un display lo solicitado, también deberá poder setear el valor de la temperatura de Set Point por algún medio que considere adecuado.

Deberá presentar el trabajo funcionando y exponerlo por medio de una presentación de no mas de 10 min.

A su vez deberá confeccionar un informe escrito donde explique el funcionamiento del sistema desde el punto de vista de la materia de arquitectura de computadoras, Análisis matemático y programación estructurada.

Desarrollo

Informe sobre Nuestro Desarrollo y Implementación de un Controlador PID en Arduino desde una Perspectiva de Arquitectura de Computadoras

Introducción: Queremos compartir los detalles de nuestro viaje al desarrollar e implementar un controlador PID utilizando la plataforma Arduino para regular la temperatura de un horno eléctrico. Este emocionante proyecto surgió en el marco de nuestras materias de Arquitectura de Computadoras, Análisis Matemático y Programación Estructurada, siendo un desafío que nos permitió aplicar de manera práctica los conocimientos adquiridos en estas asignaturas.

Justificación: La elección de utilizar un controlador PID se fundamenta en la necesidad de mantener la temperatura de un horno eléctrico para realizar el resoldado de placas con sistemas de montajes superficial BGA. Este proyecto integrador nos brindó la oportunidad de fusionar conceptos de programación, análisis matemático y comprender la arquitectura de computadoras, todo ello a través de la versátil plataforma Arduino.

Desarrollo del Prototipo: Como equipo de tres, optamos por la plataforma Arduino debido a su facilidad de programación y versatilidad. El controlador PID que diseñamos busca asegurar un funcionamiento estable del horno, utilizando la técnica de modulación por ancho de pulsos (PWM) para controlar las resistencias del mismo y ajustar la temperatura de manera precisa.

Características del Arduino: La elección de Arduino se basó en su flexibilidad y la disponibilidad de una amplia gama de módulos y bibliotecas. La arquitectura de Arduino, basada en el microcontrolador ATmega, se ajustó perfectamente a nuestras necesidades para este proyecto embebido.

Uso de Temporizadores: Nos sumergimos en el uso de temporizadores para simular las características del horno, tales como el tiempo de ascenso y descenso de temperatura. Esta estrategia nos permitió modelar la inercia térmica y las pérdidas de aislamiento, ajustándonos a las especificaciones del proyecto.

Display Utilizado: Elegimos un display LCD para mostrar la temperatura actual y el Set Point. Este tipo de pantalla se seleccionó por su compatibilidad con Arduino y su capacidad para presentar información de manera clara y legible.

Seteo del Set Point: La interacción con el usuario se diseñó mediante un potenciómetro conectado a una de las entradas analógicas de Arduino, proporcionando una forma intuitiva de ajustar la temperatura deseada.

Resumen de Datos:

- Temperatura Final Estable (Set Point): Puede configurarse entre 150° y 350°.
- Display LCD: Muestra la temperatura actual del horno y el Set Point.
- Banda Proporcional (Histeresis): Puede configurarse entre 3% y 15% del Set Point.
- Ascenso al Encender: 30° por minuto.

- Inercia Térmica: 10 segundos continuando la tendencia ascendente o descendente.
- Temperatura Después de Apagar: 2 minutos con aislación.
- Descenso al Apagar: 3° por minuto promedio.
- Temperatura Máxima del Horno (T° Max): 500°.
- Temperatura Inicial: Temperatura ambiente (18°C a 25°C).
- Tiempo Característico del Horno: 10 minutos.
- Módulo PID con PWM: El tiempo de ciclo del PWM debe ser al menos 10 veces menor que el tiempo característico del horno.

Resumen de PID y Ciclo PWM:

- **PID (Proportional-Integral-Derivative):** Es un algoritmo de control que ajusta una variable de salida para mantener una variable medida tan cercana como sea posible a un valor de referencia. En el código, implementamos las componentes proporcional, integral y derivativa para calcular la salida del calentador.
- **Ciclo PWM (Modulación por Ancho de Pulsos):** Es una técnica de control que ajusta el ancho de los pulsos en una señal de salida. En este caso, se utiliza para controlar la potencia entregada al calentador del horno. El tiempo de ciclo del PWM debe ser menor que el tiempo característico del horno para garantizar un control eficiente. En el código, controlamos el ciclo PWM para mantener la temperatura del horno dentro de los parámetros deseados.

Control PID y Modulación por Ancho de Pulsos (PWM) en el Contexto del Horno Eléctrico:

Control PID: El control PID (Proporcional-Integral-Derivativo) es una técnica ampliamente utilizada en sistemas de control para mantener una variable medida tan cercana como sea posible a un valor de referencia. En el contexto del horno eléctrico, el control PID se implementa para regular la temperatura de manera eficiente.

1. Componente Proporcional (P):

- Esta componente mide la diferencia entre la temperatura deseada (Set Point) y la temperatura actual del horno.
- La función **salida_proporcional_porcentaje()** calcula el porcentaje de la diferencia, considerando la histeresis (zona muerta) configurada.

2. Componente Derivativa (D):

- La componente derivativa tiene en cuenta la velocidad de cambio de la temperatura para prever futuros cambios.

- La función **salida_derivativa_porcentaje()** utiliza la pendiente de la temperatura para calcular esta componente, considerando la constante derivativa y la histeresis.

3. Componente Integral (I):

- La componente integral aborda el acumulado de errores a lo largo del tiempo.
- La función **salida_integral_porcentaje()** suma los errores acumulados, considerando la constante integral y la histeresis.

4. Selección de la Salida del Calentador:

- La función **select_calentador()** combina las tres componentes del PID y ajusta la potencia del calentador en consecuencia.
- Se asegura de que la potencia resultante esté dentro de los límites (0% a 100%).

Modulación por Ancho de Pulsos (PWM): La técnica de Modulación por Ancho de Pulsos se utiliza para controlar la potencia entregada al calentador del horno. En este contexto, el ciclo PWM regula la cantidad de tiempo durante el cual el calentador está encendido en cada ciclo.

1. Configuración del Ciclo PWM:

- Se establece un ciclo PWM con una duración definida por la constante **millis_ciclo**. Este ciclo es esencialmente el tiempo en el que se ajusta la potencia del calentador.
- La función **PWM_set()** gestiona este ciclo, determinando cuándo se debe activar o desactivar el calentador según el porcentaje de PWM configurado.

2. Control del Calentador mediante PWM:

- La función **select_calentador()** utiliza el resultado del control PID para determinar la potencia del calentador.
- La variable **pwm_porcentaje** representa el porcentaje de tiempo durante el ciclo PWM durante el cual el calentador está activo.

3. Actualización de la Temperatura:

- Durante el tiempo en que el calentador está activo, la temperatura del horno aumenta según la tasa de calentamiento configurada (**dt_calentar**).
- Cuando el calentador está apagado, la temperatura disminuye debido a las pérdidas de aislamiento, según la tasa de pérdidas configurada (**dt_perdidas**).

Integración del PID y PWM: El control PID ajusta la potencia del calentador, y la técnica de PWM implementa este ajuste temporalmente. Esto asegura que el sistema responda de manera eficiente a los cambios en la temperatura ambiente y alcance el Set Point de manera controlada, evitando oscilaciones excesivas.

Perspectiva de la Programación Estructurada en el Ejercicio del Horno Eléctrico, resumiendo las líneas más importantes.

Código del Prototipo:

Explicación Detallada del Código del horno con Control PID y PWM:

Definición de Pines y Pantalla LCD:

```
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
```

```
const int pin_subir{ 9 };
```

```
const int pin_bajar{ 8 };
```

```
const int pin_led{ 6 };
```

```
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

Se establecen los pines para la conexión de la pantalla LCD y los botones de control, junto con la inicialización de la pantalla.

Variables de Temperatura y Definición de las variables del horno:

```
float temperatura_actual{ 28 };
```

```
float set_point{ 200 };
```

```
float histeresis{ 5 };
```

Constructor y Configuración Inicial del Horno

```
void inicializarHorno() {
```

```
    t_ambiente = 18;
```

```
    t_horno = t_deseada;
```

```
    t_max = 500;
```

```
}
```

```
void lecturaTermometro(float temperatura) {
```

```
    t_horno = temperatura;
```

Se crea una instancia del **horno** con valores iniciales y se realiza la configuración inicial, como la inicialización de la pantalla LCD y la activación del horno.

Funciones del horno para el Control PID:

```
float salida_proporcional_porcentaje() {
```

```
    // (Cálculo de la componente proporcional)
```

```
}
```

```
float salida_derivativa_porcentaje() {
```

```
    // (Cálculo de la componente derivativa)
```

```
}
```

```
float salida_integral_porcentaje() {
```

```
    // (Cálculo de la componente integral)
```

```
} float select_calentador() {
```

```
    // (Selección de la salida del calentador mediante el control PID)
```

```
}
```

Estas funciones dentro de **horno** implementan los cálculos necesarios para las componentes proporcional, derivativa e integral del controlador PID. La función **select_calentador()** combina estas componentes para determinar la salida del calentador.

Funciones de Interfaz con el Horno:

```
void lectura_termometro(float temperatura) {
```

```
    t_horno = temperatura;
```

```
}
```

La función **lectura_termometro()** actualiza la temperatura del horno con la lectura actual.

Funciones de Interfaz con el Usuario:

```
void cambiar_display() {
```

```
    // (Cambiar la visualización en la pantalla LCD)
```

```
}
```

```
void actualizar_display() {
```



```
// (Actualizar la visualización en la pantalla LCD)
```

```
}
```

Estas funciones gestionan la interfaz con el usuario, cambiando y actualizando la información mostrada en la pantalla LCD.

Funciones de Control del Ciclo PWM:

```
bool PWM_set() {
```

```
// (Control del ciclo PWM)
```

```
}
```

La función **PWM_set()** gestiona el ciclo PWM, determinando si se debe aplicar la potencia al calentador en un determinado momento.

Función Principal de Ejecución Continua:

```
void loop() {
```

```
// (Función principal que se ejecuta continuamente)
```

```
}
```

Esta función se ejecuta de manera continua y maneja la interacción con el usuario, actualización de parámetros de control del horno, control del ciclo PWM y actualización de la pantalla LCD.

Perspectiva del Análisis Matemático en el Ejercicio del Horno Eléctrico:

1. Ascenso de Temperatura (Encendido de Resistencias):

$$\frac{dT}{dt} = dt_calentar$$

Donde:

- $\frac{dT}{dt}$ es la tasa de cambio de la temperatura.
- $dt_calentar$ es la tasa de calentamiento, que es 30° por minuto.

2. Descenso de Temperatura (Apagado de Resistencias):

$$\frac{dT}{dt} = -dt_perdidas$$

Donde:

- $\frac{dT}{dt}$ es la tasa de cambio de la temperatura.
- $dt_perdidas$ es la tasa de pérdida de calor, que es 3° por minuto.

3. Inercia Térmica después de Apagar las Resistencias:

$$\frac{dT}{dt} = -\frac{T - T_{\text{malmate}}}{10}$$

Donde:

- $\frac{dT}{dt}$ es la tasa de cambio de la temperatura.
- T es la temperatura actual del horno.
- T_{ambiente} es la temperatura ambiente.
- El 10 en el denominador representa el tiempo característico de 10 segundos.

4. Control PID (Componente Proporcional):

Donde:

- P es la componente proporcional del control PID.
- T_{desenda} es la temperatura deseada (Set Point).
- porcentaje_histeresis es el porcentaje de histeresis configurado.

5. Control PID (Componente Derivativa):

$$D = \frac{K_4 \cdot (E - E_{\text{nntarhin}})}{\text{delay_en_mis}}$$

Donde:

- D es la componente derivativa del control PID.
- K_d es la constante derivativa.
- E es el error actual ($T_{denenda} - T$).
- $E_{anterior}$ es el error anterior.
- $delay_en_ms$ es el tiempo de retardo en milisegundos.

6. Control PID (Componente Integral):

Donde:

- I es la componente integral del control PID.
- K_i es la constante integral.
- $suma_errores$ es la suma acumulativa de errores.

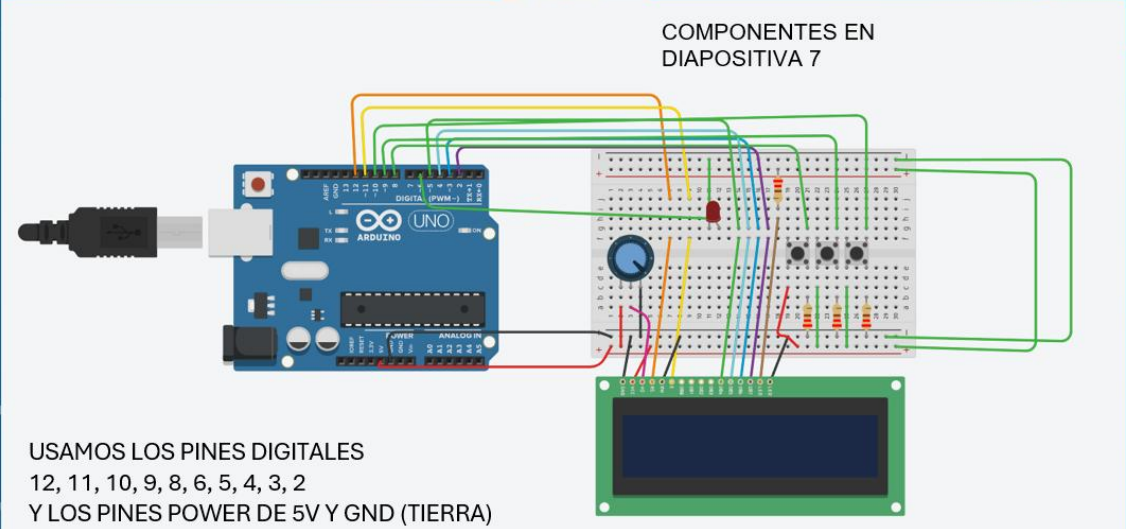
Link al tinkercad:

<https://www.tinkercad.com/things/7zaPoiphSbs-horno-definitivo>

ARQUITECTURA Y CARACTERISTICAS

(extraído de nuestro powerpoint)

CIRCUITO HORNO



COMPONENTES EN DIAPOSITIVA 7

USAMOS LOS PINES DIGITALES
12, 11, 10, 9, 8, 6, 5, 4, 3, 2
Y LOS PINES POWER DE 5V Y GND (TIERRA)

Características del Arduino UNO R3



Microcontrolador:	ATmega328P
Voltaje de operación:	5V
Entrada de alimentación:	7-12V
Pines digitales I/O:	14
Pines PWM:	6
Pines analógicos:	6
Corriente por pin:	20 mA
Corriente del pin 3.3V:	50 mA max
Memoria Flash:	32 KB (0.5 KB usados por el bootloader)
SRAM:	2 KB
EEPROM:	1 KB
Velocidad de reloj:	16 MHz
LED programable integrado:	13
Dimensiones:	68.6 x 53.4 mm
Peso:	25 gramos

DISPLAYS A USAR

(DE NORMAL)
LCD 16 X 2



(POR SI HAY SOLO ESTE)
LCD 16 X 2 (I2C)



```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
lcd.init();
lcd.setBacklight(1);
```

- 1 POTENCIOMETRO



- 1 LUZ LED



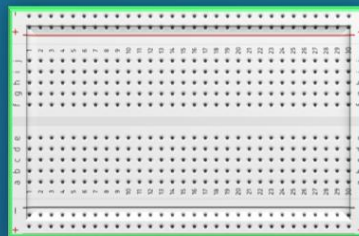
- 3 BOTONES



- 4 RESISTENCIAS (220 Ω)



- 1 PROTOBOARD (OPCIONAL)



Control PID

El Control PID es un algoritmo utilizado para ajustar una variable de salida y mantener una variable medida cerca de un valor de referencia.

Consiste en tres componentes:

Proporcional (P), Integral (I) y Derivativa (D).

1. Proporcional (P):

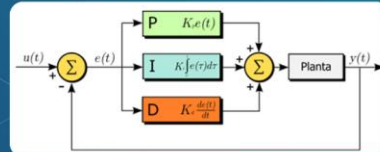
- Función:** Responde a la diferencia actual entre la temperatura deseada y la temperatura real.
- Objetivo:** Reducir el error actual ajustando proporcionalmente la potencia del calentador.

Derivativo (D):

- Función:** Considera la velocidad de cambio de la temperatura.
- Objetivo:** Prever futuros cambios y reducir oscilaciones, mejorando la estabilidad y la respuesta del sistema.

1. Integral (I):

- Función:** Aborda la acumulación de errores pasados a lo largo del tiempo.
- Objetivo:** Eliminar el error acumulado para mejorar la precisión y eliminar el sesgo a largo plazo.

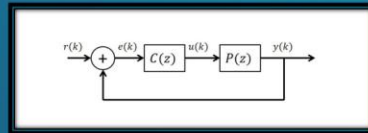


A. `salida_proporcional_porcentaje()` (Componente P):

Calcula la componente proporcional del control PID. Utiliza la diferencia entre la temperatura deseada (`t_deseada`) y la temperatura actual del horno (`t_horno`). Aplica una fórmula que considera la histeresis (`porcentaje_histeresis`).

B. `salida_derivativa_porcentaje()` (Componente D):

Calcula la componente derivativa del control PID. Utiliza la velocidad de cambio de la temperatura para prever futuros cambios. Aplica una fórmula que involucra la constante derivativa, la diferencia entre la temperatura deseada y actual, y la pendiente de la temperatura.



C. `salida_integral_porcentaje()` (Componente I):

Calcula la componente integral del control PID. Aborda el acumulado de errores a lo largo del tiempo. Aplica una fórmula que suma los errores acumulados, considerando la constante integral.

D. `select_calentador()` (Selección de Salida):

Combina las tres componentes del PID (proporcional, derivativa e integral) para determinar la salida del calentador. Ajusta la potencia del calentador en consecuencia, asegurándose de que esté dentro de los límites (0% a 100%).

Como se aplica al código?

Condición de tiempo del ciclo:

La función verifica si el tiempo transcurrido desde el inicio del ciclo PWM (`millis_ciclo`) es mayor o igual al tiempo definido para un ciclo completo.

Actualización del tiempo del ciclo:

Si se cumple la condición, se actualiza el tiempo del ciclo (`millis_pwm`) al tiempo actual (`millis()`), y se guarda el porcentaje de PWM establecido (`pwm_porcentaje`) en `pwm_porcentaje_set`. Cálculo y comparación del porcentaje de PWM:

Luego, la función calcula el porcentaje de tiempo transcurrido en el ciclo actual y lo compara con el porcentaje de PWM establecido (`pwm_porcentaje_set`).

Si el porcentaje de tiempo transcurrido es menor o igual al porcentaje de PWM configurado y este no es cero, la función devuelve `true`, indicando que el calentador debe estar encendido durante este intervalo de tiempo.

Si no se cumple la condición, devuelve `false`, indicando que el calentador debe estar apagado durante este intervalo.

```
// Función para controlar el ciclo PWM
bool PWM_set() {
    if ((millis() - millis_pwm) >= millis_ciclo) {
        millis_pwm = millis();
        pwm_porcentaje_set = pwm_porcentaje;
    }
    if ((float(millis() - millis_pwm) / float(millis_ciclo)) * 100.0 <= pwm_porcentaje_set and pwm_porcentaje_set != 0) {
        return true;
    } else {
        return false;
    }
}
```