UNIVERSIDAD DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL BIOMÉDICA

CBM 426 - Taller De integración

Proyecto Incubadora

Alumnos:

Josefa Belen Miranda Aravena
Maximiliano Antonio Gaete Pizarro
Maria-Ignacia Rojas

Profesores:

Gonzalo Tapia Cabrera
Antonio José Rienzo Renato
Luis Togo Arredondo Gamboa

Índice

1. Introducción **4**

2. Objetivos **4**

3. Diseño del Prototipo **5**

3.1. Materiales 5

3.2. Dibujo Técnico 5

3.2.1. Tapa 5

3.2.2. Caja 6

3.2.3. Caja Cerrada 6

3.3. Diseño del Circuito 7

3.3.1. Calculo de componentes a usar 7

3.3.2. Esquemático 8

3.3.3. Circuito Impreso 8

4. Código para el control de la incubadora **9**

4.1. Código Arduino 9

4.2. Código Python 11

5. Ensayos y Resultados **13**

5.1. Ensayos de Control de Temperatura 13

5.1.1. Prueba 1: 100 % de potencia 14

5.1.2. Prueba 2: 80 % de potencia 14

5.1.3. Prueba 3: 60 % de potencia 14

5.1.4. Prueba 4: 40 % de potencia 15

5.1.5. Prueba 5: 20 % de potencia 15

5.2. Resultados de Respuesta Escalón de Temperatura 15

5.2.1. Cálculo de la Potencia de Radiación 16

5.2.2. Analisis Prueba 1: 100 % de potencia 17

5.2.3. Analisis Prueba 2: 80 % de potencia 17

5.2.4. Analisis Prueba 3: 60 % de potencia 17

5.2.5. Analisis Prueba 4: 40 % de potencia 18

5.2.6. Analisis Prueba 5: 20 % de potencia 18

5.2.7. Resultados Finales 18

5.3. Ensayos de Control de Velocidad del Ventilador 18

5.3.1. Prueba 1: 100 % de Potencia, Ventilador a 12v, 9,5v y 7v 19

5.3.2. Prueba 2: 80 % de Potencia, Ventilador a 12v, 9,5v y 7v 19

5.3.3. Prueba 3: 60 % de Potencia, Ventilador a 12v, 9,5v y 7v 19

5.3.4. Prueba 4: 40 % de Potencia, Ventilador a 12v, 9,5v y 7v 19

5.3.5. Prueba 5: 20 % de Potencia, Ventilador a 12v, 9,5v y 7v 19
5.4. Resultados de Respuesta Escalón de Velocidad del Ventilador 19
5.4.1. Analisis Prueba 1: 100 % de Potencia, Ventilador a 12v, 9,5v y 7v . 19
5.4.2. Analisis Prueba 2: 80 % de Potencia, Ventilador a 12v, 9,5v y 7v . 19
5.4.3. Analisis Prueba 3: 60 % de Potencia, Ventilador a 12v, 9,5v y 7v . 19
5.4.4. Analisis Prueba 4: 40 % de Potencia, Ventilador a 12v, 9,5v y 7v . 19
5.4.5. Analisis Prueba 5: 20 % de Potencia, Ventilador a 12v, 9,5v y 7v . 19
5.4.6. Resultados Finales 19

6. Modelo Matemático de la Incubadora de Control de Temperatura 20
6.1. Función de Transferencia 20
6.2. Ecuación Diferencial en el Dominio del Tiempo 20
6.3. Solución en Estado Transitorio y Estacionario 20
6.4. Diagrama de Bloques 21

7. Temperaturas Alcanzables y Tiempo de Estabilización 21
7.1. Rango de Temperaturas Alcanzable 21
7.2. Tiempo de Respuesta del Sistema 22
7.3. Resumen del Control de Temperatura 22
7.4. Simulación en Python 22
7.5. Explicación del Código 24

8. Algoritmo de Control PID en Arduino 25
8.1. Cálculo de los Parámetros del PID 25
8.2. Implementación en Arduino 25

9. Resultados Control de Temperatura con PID 27

10. Actividades y Cronograma 27

11. Conclusion 27

1. Introducción

Se desarrollará un prototipo de incubadora neonatal que utiliza una ampolleta de 12V como fuente de calor, un ventilador para la circulación de aire, y un sensor NTC para medir la temperatura.

La finalidad del prototipo y controlar de manera precisa la temperatura interna, creando un ambiente adecuado para un neonato. Para lograr esto, se implementará un sistema de control basado en Arduino que permite regular la temperatura de forma automática y eficiente. Además, se registrará la evolución de los parámetros obtenidos en diferentes escenarios, lo cual contribuye al análisis y mejora continua del prototipo.

2. Objetivos

1. Desarrollar un prototipo de incubadora utilizando una caja plástica, con una ampolleta como calefactor, un ventilador y un sensor NTC para medir la temperatura.
2. Diseñar un circuito controlado por Arduino que regule la potencia de la ampolleta, mida la velocidad del ventilador y registre la temperatura.
3. Programar un software en Python que permita la comunicación del sistema con el computador a través de un puerto serial y controle los elementos mencionados.
4. Realizar ensayos de control de temperatura y registrar los resultados en archivos CSV para su análisis posterior.
5. Repetir los ensayos incluyendo el ventilador a distintas velocidades y generar gráficos de los resultados obtenidos.
6. Desarrollar un modelo matemático que describa el comportamiento de la temperatura en función de la potencia del calefactor.
7. Definir el rango de control de temperatura y los tiempos de respuesta del sistema.
8. Implementar un algoritmo de control en Arduino para mantener la temperatura dentro del rango especificado.
9. Verificar el funcionamiento del sistema y determinar el error máximo en la temperatura controlada.
10. Crear una base de datos para registrar y visualizar el estado del prototipo en tiempo real.
11. Implementar un sistema de alarmas luminoso y auditivo para alertar de fallos en el sistema.
12. Registrar las alarmas del sistema dentro de la plataforma informática.

3. Diseño del Prototipo

3.1. Materiales

Cuadro 1: Materiales utilizados en el prototipo.

Material	**Cantidad**
Caja plástica	1
Ampolleta de 12V	1
Ventilador de 12V	1
NTC 100K	1
Arduino Uno	1
Protoboard	1
Resistencia de 47Ω	1
Resistencia de 100Ω 1W	1
Resistencia de 5.6KΩ	1
Resistencia de 100KΩ	1
Transistor NPN 2N2222	1
Transistor NPN TIP41C	1
Optoacoplador 4N26	1
Fuentes de alimentación 12V	1

3.2. Dibujo Técnico

3.2.1. Tapa

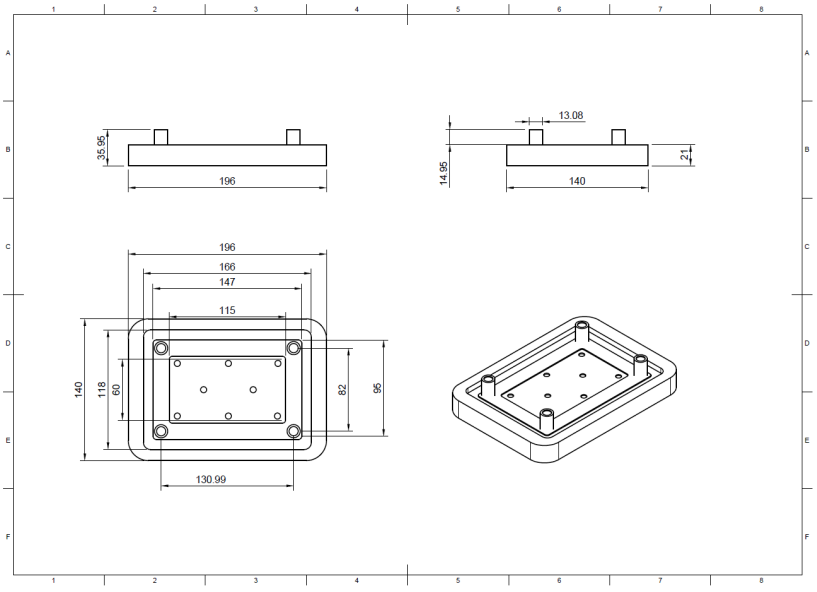

Figura 1: Dibujo técnico de la tapa.

3.2.2. Caja

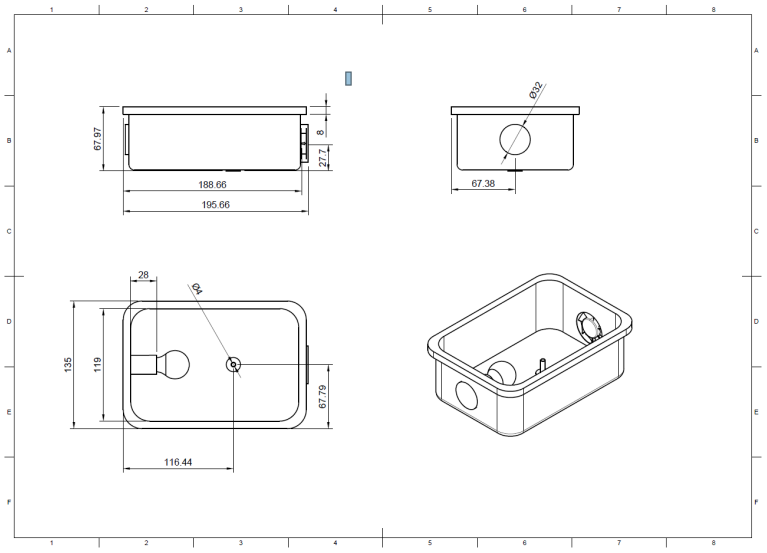

Figura 2: Dibujo técnico de la caja plástica.

3.2.3. Caja Cerrada

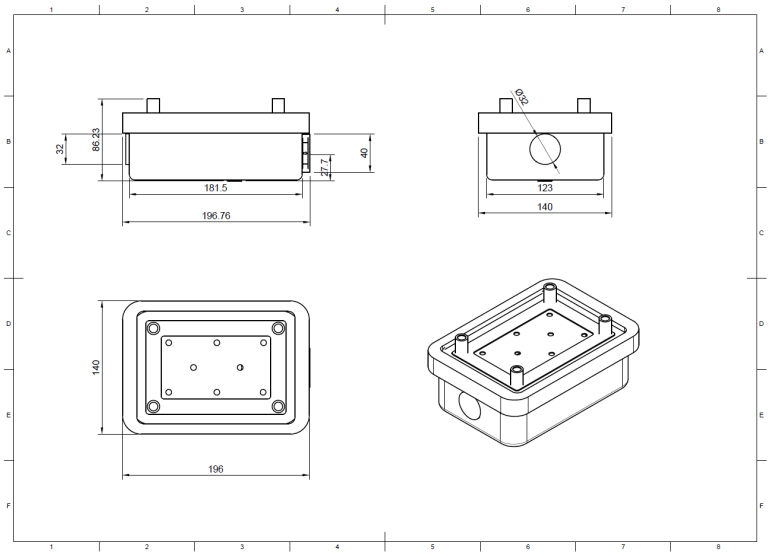

Figura 3: Dibujo técnico de la caja plástica con la tapa puesta.

3.3. Diseño del Circuito

3.3.1. Calculo de componentes a usar

Ampolleta de 12V y 1.7A

Se utilizará un transistor TIP41C para controlar la ampolleta, el cual tiene un rango de funcionamiento adecuado para la corriente que se manejará.

$$i_A = 1{,}7\,\text{A}$$

El rango de h_{FE} para el transistor TIP41C es entre 15 y 75. En este caso, utilizamos un valor de h_{FE} mínimo:

$$h_{FE} = 15$$

La corriente de base i_b se calcula como:

$$i_A = h_{FE} \cdot i_b$$
$$i_b = \frac{1{,}7}{15}$$
$$i_b = 0{,}113\,\text{A}$$

La resistencia de base R_b se calcula con la siguiente fórmula:

$$R_b = \frac{12 - 0{,}7}{0{,}113}$$
$$R_b = 100\,\Omega$$

Ventilador de 12V y 0.06A

Se utilizará un transistor 2N2222 para controlar el ventilador, el cual tiene un rango de funcionamiento adecuado para la corriente que se manejará.

$$i_V = 0{,}06\,\text{A}$$

El rango de h_{FE} para el transistor 2N2222 es entre 30 y 75. Utilizaremos el valor mínimo de h_{FE}:

$$h_{FE} = 30$$

La corriente de base i_b se calcula como:

$$i_b = \frac{i_C}{h_{FE}}$$

$$i_b = \frac{0{,}06}{30} = 0{,}002\,\text{A}$$

La resistencia de base R_b se calcula de la siguiente manera:

$$R_b = \frac{12 - 0{,}7}{0{,}002}$$

$$R_b = 5650\,\Omega$$

3.3.2. Esquemático

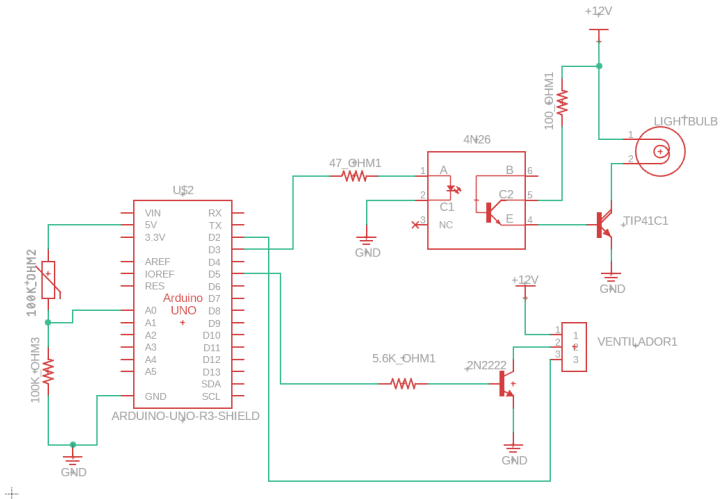

Figura 4: Diseño del circuito de control de la incubadora.

3.3.3. Circuito Impreso

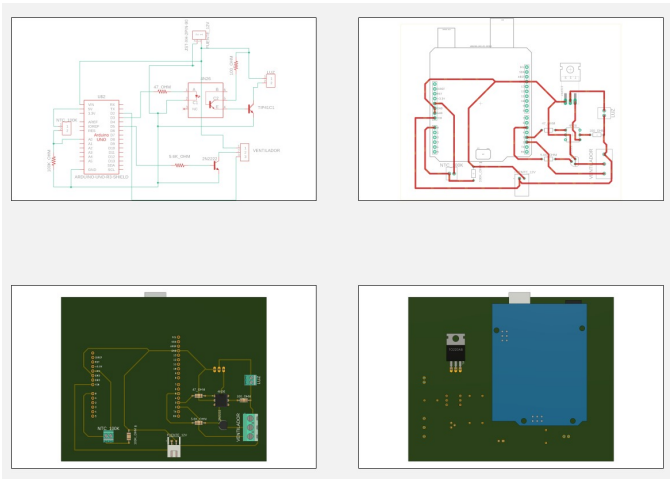

Figura 5: Diseño del circuito impreso de control de la incubadora.

4. Código para el control de la incubadora

Este proyecto integra dos componentes principales: un microcontrolador basado en Arduino y una interfaz gráfica desarrollada en Python utilizando la biblioteca PyQt6. El sistema permite controlar la potencia de radiación de una ampolleta, la velocidad de giro del ventilador, y medir la temperatura registrada por un sensor NTC. A continuación, se detallan los códigos implementados en ambas plataformas, explicando su funcionamiento.

4.1. Código Arduino

El código de Arduino es responsable de controlar los sensores y actuadores de la incubadora. Se utiliza un controlador PID (Proporcional, Integral y Derivativo) para mantener la temperatura estable, y se implementan las siguientes funciones:

- **Controlar la potencia de radiación de la ampolleta:** La ampolleta se controla utilizando modulación por ancho de pulso (PWM) en el pin 11 de Arduino. El valor de salida del controlador PID ajusta dinámicamente la potencia de la luz en función de la temperatura medida, o se puede ajustar manualmente desde la interfaz Python. Esto se logra con la función `analogWrite`.
- **Controlar y medir la velocidad de giro del ventilador:** El ventilador se controla mediante PWM en el pin 9. La velocidad de giro se mide con un tacómetro conectado al pin 2, utilizando interrupciones para contar los impulsos por revolución. Estos valores se imprimen periódicamente por el puerto serial y se utilizan tanto para la visualización en la interfaz como para ajustar el ventilador.
- **Medir la temperatura registrada por la NTC:** La resistencia del sensor NTC se mide en el pin A0, y se utiliza la ecuación de Steinhart-Hart para convertir la resistencia a una temperatura en grados Celsius. El valor de temperatura es utilizado por el PID para controlar la ampolleta.

A continuación se presenta el código de Arduino que implementa estas funciones:

```
#include <Arduino.h>
#include <PID_v1.h>

// Definicion de pines y constantes
#define NTC_PIN A0
#define LUZ_PIN 11
#define VENT_PIN 9
#define VEL_PIN 2

// Coeficientes de la ecuacion de Steinhart-Hart para el sensor NTC
#define A_COEFF 0.5458630405e-3
#define B_COEFF 2.439180157e-4
#define C_COEFF -0.0003705076153e-7
```

```
// Variables para mediciones y control
float resistenciaNTC, logResistencia, tempKelvin, tempCelsius;
int lecturaNTC, potenciaVentilador, potenciaLuz;
volatile int contadorImpulsos = 0;
volatile float velocidadRPM = 0;
unsigned long tiempoAnteriorTemp, tiempoAnteriorVel, tiempoImprimir,
    ultimoTiempoRPM;

// Variables PID
double Setpoint, Input, Output;
double Kp = 30.0, Ki = 1.0, Kd = 0.0;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

// Inicializacion de funciones y configuracion de pines
void setup() {
    Serial.begin(115200);
    pinMode(NTC_PIN, INPUT);
    pinMode(LUZ_PIN, OUTPUT);
    pinMode(VENT_PIN, OUTPUT);
    attachInterrupt(digitalPinToInterrupt(VEL_PIN), medirVelocidad,
        FALLING);
    myPID.SetMode(AUTOMATIC);
    myPID.SetOutputLimits(0, 100);
}

void loop() {
    // Logica de medicion de temperatura y control PID
    // Medicion de la NTC, calculo de temperatura y control de la
        ampolleta
    if (millis() - tiempoAnteriorTemp >= 250) {
        lecturaNTC = analogRead(NTC_PIN);
        resistenciaNTC = 100000.0 * ((1023.0 / lecturaNTC) - 1.0);
        logResistencia = log(resistenciaNTC);
        tempKelvin = 1.0 / (A_COEFF + B_COEFF * logResistencia + C_COEFF
            * pow(logResistencia, 3));
        tempCelsius = tempKelvin - 273.15;

        // Control de la luz
        Input = tempCelsius;
        myPID.Compute();
        analogWrite(LUZ_PIN, map(Output, 0, 100, 0, 255));

        tiempoAnteriorTemp = millis();
    }

    // Control y medicion de la velocidad del ventilador
    if (millis() - tiempoAnteriorVel >= 1000) {
        unsigned long tiempoActual = millis();
```

```
        unsigned long tiempoTranscurrido = tiempoActual - ultimoTiempoRPM
            ;
        int pulsos = contadorImpulsos;
        contadorImpulsos = 0;
        if (tiempoTranscurrido > 0) {
            velocidadRPM = (pulsos / 2.0) * (60000.0 / tiempoTranscurrido
                );
        }
        ultimoTiempoRPM = tiempoActual;
        tiempoAnteriorVel = tiempoActual;
    }

    // Impresion de datos por serial
    if (millis() - tiempoImprimir >= 300) {
        Serial.print(tempCelsius);
        Serial.print(";");
        Serial.print(velocidadRPM);
        Serial.print(";");
        Serial.println(Output);
        tiempoImprimir = millis();
    }
}
```

Este código maneja la lectura de la temperatura y la velocidad del ventilador, ajustando la potencia de la luz en función del controlador PID.

4.2. Código Python

El código en Python implementa una interfaz gráfica que permite al usuario monitorear y controlar en tiempo real los valores de temperatura, velocidad del ventilador y la potencia de la luz. Las principales funciones son:

- **Controlar la potencia de la luz:** Desde la interfaz, el usuario puede ajustar manualmente la potencia de la luz usando un slider, o puede activar el modo automático que delega el control a Arduino. El control se realiza enviando comandos seriales que son interpretados por el Arduino.
- **Controlar la velocidad del ventilador:** De forma similar, el usuario puede ajustar la velocidad del ventilador manualmente o permitir que el sistema controle automáticamente en función de la temperatura.
- **Monitorear la temperatura:** La temperatura leída por el sensor NTC en Arduino se envía al programa Python, donde se muestra en la interfaz gráfica junto con gráficos en tiempo real que muestran las tendencias de temperatura, velocidad y potencia.

A continuación se presenta un extracto del código Python que implementa estas funciones:

```
import serial
from PyQt6 import uic, QtWidgets
from PyQt6.QtCore import QTimer, QDateTime

class Plataforma(QMainWindow):
    def __init__(self):
        super().__init__()
        uic.loadUi("Plataforma.ui", self)
        self.serial_port = serial.Serial('COM12', 115200, timeout=1)
        self.timer = QTimer()
        self.timer.timeout.connect(self.leer_datos)
        self.timer.start(300)

    def leer_datos(self):
        while self.serial_port.in_waiting > 0:
            linea = self.serial_port.readline().decode('utf-8').strip()
            if linea:
                datos = linea.split(';')
                if len(datos) == 3:
                    temp = float(datos[0])
                    velocidad = float(datos[1])
                    potencia_luz_aplicada = float(datos[2])
                    # Mostrar datos en la GUI
                    self.label_temperatura.setText(f'Temperatura: {temp
                        :.1f} C')
                    self.label_velocidad.setText(f'Velocidad: {int(
                        velocidad)} RPM')
                    self.label_potencia_luz.setText(f'Potencia Luz: {
                        potencia_luz_aplicada:.0f}%')

    def enviar_luz(self, valor):
        comando = f'LUZ {valor}\n'
        self.serial_port.write(comando.encode('utf-8'))

    def enviar_ventilador(self, valor):
        comando = f'VENT {valor}\n'
        self.serial_port.write(comando.encode('utf-8'))

    def enviar_setpoint(self, valor):
        comando = f'SETPOINT {valor}\n'
        self.serial_port.write(comando.encode('utf-8'))
```

Este código se encarga de la interacción entre el usuario y el sistema a través de una interfaz gráfica, mostrando la temperatura y la velocidad del ventilador, y permitiendo ajustar la potencia de la luz y la velocidad del ventilador. Los comandos enviados por el puerto serial son procesados por el Arduino para ajustar los actuadores correspondientes.

Gráficos en tiempo real

El programa utiliza PyQtGraph para generar gráficos en tiempo real que muestran la evolución de la temperatura, la velocidad del ventilador y las potencias de los actuadores a lo largo del tiempo.

```
def actualizar_graficos(self):
    self.curve_temp.setData(self.tiempos, self.temperaturas)
    self.curve_vel.setData(self.tiempos, self.velocidades)
    self.curve_potencia_vent.setData(self.tiempos, self.
        potencias_ventilador)
```

Estos gráficos permiten al usuario monitorear el desempeño de la incubadora de manera visual y en tiempo real, lo que facilita el control del sistema y la toma de decisiones.

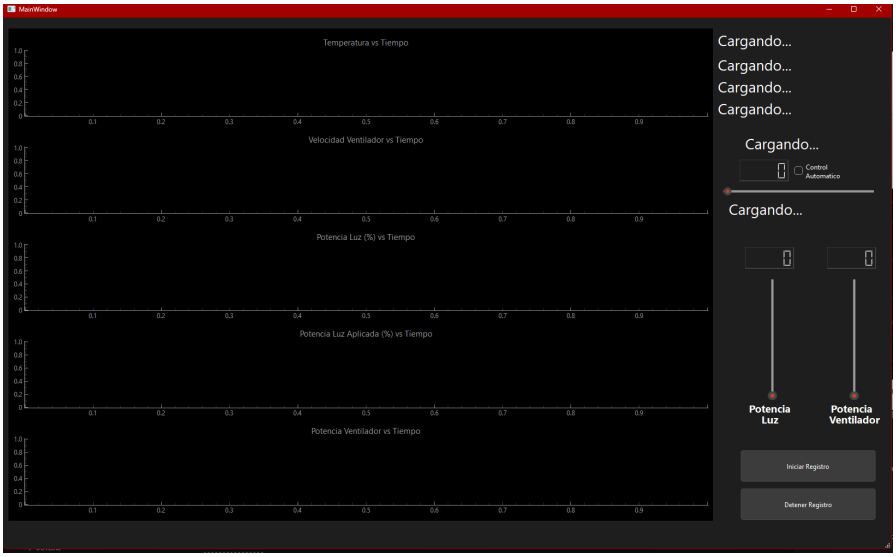

Figura 6: Interfaz gráfica de la plataforma de control de la incubadora.

5. Ensayos y Resultados

5.1. Ensayos de Control de Temperatura

Para realizar los ensayos se realizaron 5 pruebas de respuesta tipo escalon, variando la potencia de la ampolleta de 0 a 100 % en incrementos de 20 %. Los resultados obtenidos se muestran a continuación.

5.1.1. Prueba 1: 100 % de potencia

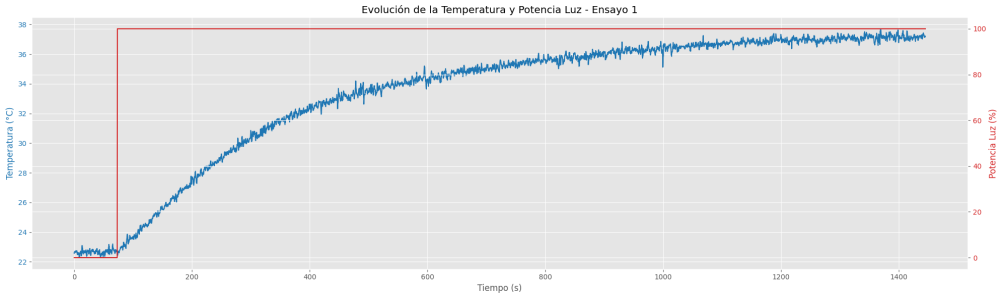

Figura 7: Respuesta de la temperatura a un escalón de 100 % de potencia.

5.1.2. Prueba 2: 80 % de potencia

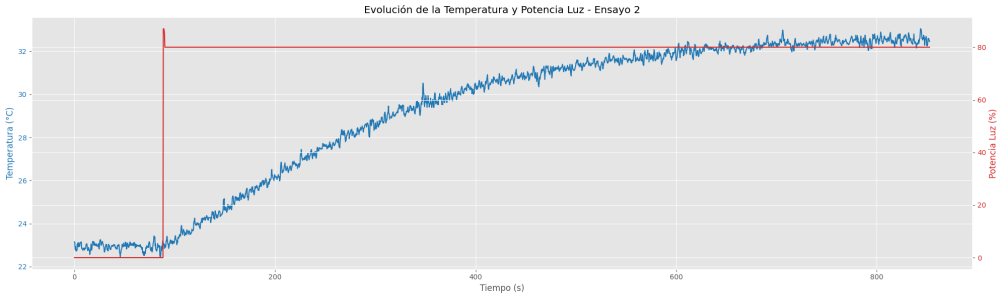

Figura 8: Respuesta de la temperatura a un escalón de 80 % de potencia.

5.1.3. Prueba 3: 60 % de potencia

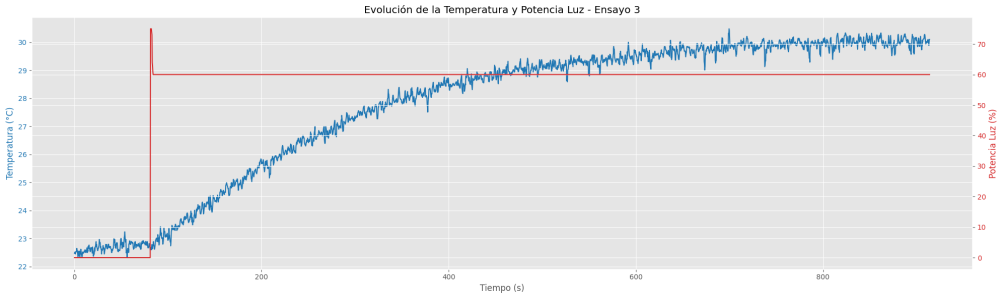

Figura 9: Respuesta de la temperatura a un escalón de 60 % de potencia.

5.1.4. Prueba 4: 40 % de potencia

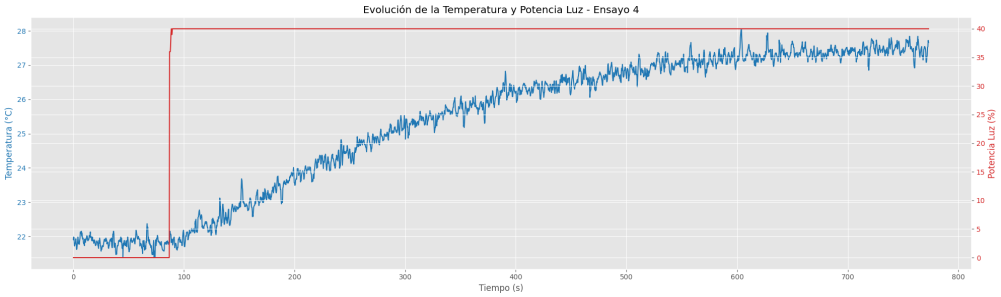

Figura 10: Respuesta de la temperatura a un escalón de 40 % de potencia.

5.1.5. Prueba 5: 20 % de potencia

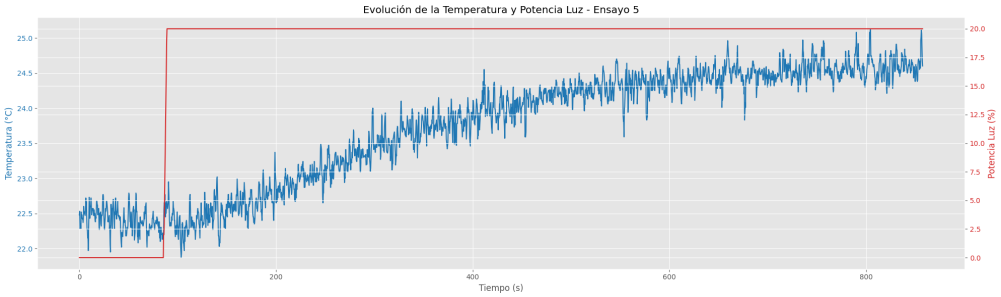

Figura 11: Respuesta de la temperatura a un escalón de 20 % de potencia.

5.2. Resultados de Respuesta Escalón de Temperatura

Las pruebas de respuesta a escalón permiten evaluar la capacidad de control del sistema y determinar su desempeño en función de la potencia aplicada. La Potencia al 100 % representa la máxima capacidad de calentamiento de la ampolleta, mientras que la potencia al 0 % representa la condición de apagado total.

Para realizar el análisis de los resultados, se calcula el tiempo de establecimiento t_s, el sobrepaso máximo M_p, la constante de tiempo K, la temperatura maxima para cada ensayo.

Fórmulas para el Analisis de la Respuesta a Escalón

Cambio en la Temperatura Final (ΔT_{final}):

$$\Delta T_{\text{final}} = T_{\text{final}} - T_{\text{inicial}}$$

Ganancia Estática (K):

$$K = \frac{\Delta T_{\text{final}}}{\Delta P}$$

Constante de Tiempo (τ):

$$T_\tau = T_{\text{inicial}} + 0{,}632 \times \Delta T_{\text{final}}$$

Tiempo de Asentamiento (t_s):

$$t_s \approx 4 \times \tau$$

Sobrepaso (M_p) (si ocurre):

$$M_p = \frac{T_{\text{max}} - T_{\text{final}}}{T_{\text{final}}} \times 100$$

5.2.1. Cálculo de la Potencia de Radiación

La potencia de radiación de la ampolleta se calcula como:

$$P = V \cdot I$$

Donde $V = 12\,\text{V}$ es el voltaje de la ampolleta y I es la corriente medida en cada ensayo.

En este caso la potencia total es de 20.4 W ya que los valores maximos alcanzados por la ampolleta son $V = 12\,\text{V}$ y $I = 1{,}7\,\text{A}$.

Al tener la eficiencia luminica de la ampolleta que es $0.05\,\%$, se puede calcular la potencia de radiación.

$$P_{\text{Radiación Luminica}} = P \cdot \text{Eficiencia}$$

$$P_{\text{Radiación calorica}} = P_{\text{Total}} - P_{\text{Radiación Luminica}}$$

Esto se puede observar en el cuadro 2 para cada ensayo.

Ensayo	Potencia (%)	Potencia de Radiación. (W)
1	100	18.6
2	80	14.9
3	60	11.2
4	40	7.4
5	20	3.7

Cuadro 2: Potencia de radiación según la potencia aplicada.

5.2.2. Analisis Prueba 1: 100 % de potencia

$$T_{\max} = 37{,}43\,^{\circ}\mathrm{C}$$

$$\Delta T_{\text{final}} = 37{,}13 - 23{,}25 = 13{,}88\,^{\circ}\mathrm{C}$$

$$K = \frac{13{,}89}{100} = 0{,}1389\,^{\circ}\mathrm{C/P\,\%}$$

$$t_s = 4 \times 296{,}12 = 1184{,}48\,\mathrm{s}$$

$$M_p = \frac{37{,}43 - 37{,}13}{37{,}13} \times 100 = 0{,}81\,\%$$

5.2.3. Analisis Prueba 2: 80 % de potencia

$$T_{\max} = 32{,}69\,^{\circ}\mathrm{C}$$

$$\Delta T_{\text{final}} = 32{,}52 - 22{,}95 = 9{,}57\,^{\circ}\mathrm{C}$$

$$K = \frac{9{,}56}{80} = 0{,}1195\,^{\circ}\mathrm{C/P\,\%}$$

$$t_s = 4 \times 236{,}68 = 946{,}72\,\mathrm{s}$$

$$M_p = \frac{32{,}69 - 32{,}52}{32{,}52} \times 100 = 0{,}52\,\%$$

5.2.4. Analisis Prueba 3: 60 % de potencia

$$T_{\max} = 30{,}07\,^{\circ}\mathrm{C}$$

$$\Delta T_{\text{final}} = 30{,}01 - 22{,}88 = 7{,}13\,^{\circ}\mathrm{C}$$

$$K = \frac{7{,}13}{60} = 0{,}1188\,^{\circ}\mathrm{C/P\,\%}$$

$$t_s = 4 \times 205{,}53 = 822{,}12\,\mathrm{s}$$

$$M_p = \frac{30{,}07 - 30{,}01}{30{,}01} \times 100 = 0{,}20\,\%$$

5.2.5. Analisis Prueba 4: 40 % de potencia

$$T_{\max} = 27{,}72\,^{\circ}\mathrm{C}$$

$$\Delta T_{\text{final}} = 27{,}46 - 21{,}79 = 5{,}67\,^{\circ}\mathrm{C}$$

$$K = \frac{5{,}67}{40} = 0{,}1417\,^{\circ}\mathrm{C/P\,\%}$$

$$t_s = 4 \times 216{,}53 = 866{,}12\,\mathrm{s}$$

$$M_p = \frac{27{,}72 - 27{,}46}{27{,}46} \times 100 = 0{,}95\,\%$$

5.2.6. Analisis Prueba 5: 20 % de potencia

$$T_{\max} = 25{,}11\,^{\circ}\mathrm{C}$$

$$\Delta T_{\text{final}} = 24{,}60 - 22{,}43 = 2{,}17\,^{\circ}\mathrm{C}$$

$$K = \frac{2{,}17}{20} = 0{,}1085\,^{\circ}\mathrm{C/P\,\%}$$

$$t_s = 4 \times 264{,}67 = 1058{,}68\,\mathrm{s}$$

$$M_p = \frac{25{,}11 - 24{,}60}{24{,}60} \times 100 = 2{,}07\,\%$$

5.2.7. Resultados Finales

Para aclarar al lector, se presentan los resultados obtenidos en el cuadro 3.

Ensayo	**Potencia (%)**	**Temp. Inicial (°C)**	**Temp. Final (°C)**	**Cambio en Temp. (°C)**
1	100	23.25	37.13	13.88
2	80	22.95	32.52	9.57
3	60	22.88	30.01	7.13
4	40	21.79	27.46	5.67
5	20	22.43	24.60	2.17

Cuadro 3: Resultados de los ensayos sin ventilador.

5.3. Ensayos de Control de Velocidad del Ventilador

Para realizar los ensayos se realizaron 5 pruebas de respuesta tipo escalon, variando la velocidad del ventilador. Los resultados obtenidos se muestran a continuación.

5.3.1. Prueba 1: 100 % de Potencia, Ventilador a 12v, 9,5v y 7v

5.3.2. Prueba 2: 80 % de Potencia, Ventilador a 12v, 9,5v y 7v

5.3.3. Prueba 3: 60 % de Potencia, Ventilador a 12v, 9,5v y 7v

5.3.4. Prueba 4: 40 % de Potencia, Ventilador a 12v, 9,5v y 7v

5.3.5. Prueba 5: 20 % de Potencia, Ventilador a 12v, 9,5v y 7v

5.4. Resultados de Respuesta Escalón de Velocidad del Ventilador

5.4.1. Analisis Prueba 1: 100 % de Potencia, Ventilador a 12v, 9,5v y 7v

5.4.2. Analisis Prueba 2: 80 % de Potencia, Ventilador a 12v, 9,5v y 7v

5.4.3. Analisis Prueba 3: 60 % de Potencia, Ventilador a 12v, 9,5v y 7v

5.4.4. Analisis Prueba 4: 40 % de Potencia, Ventilador a 12v, 9,5v y 7v

5.4.5. Analisis Prueba 5: 20 % de Potencia, Ventilador a 12v, 9,5v y 7v

5.4.6. Resultados Finales

Para aclarar al lector, se presentan los resultados obtenidos en el 4.

Potencia (%)	**Ventilador (V)**	T_{inicial} (°C)	T_{final} (°C)	ΔT_{final} (°C)	τ (s)	t_s (s)	K (°C/P %)	M_p (%)
100	7	...	...	...	...	...	...	...
100	9.5	...	...	...	...	...	...	...
100	12	...	...	...	...	...	...	...
80	7	...	...	...	...	...	...	...
80	9.5	...	...	...	...	...	...	...
80	12	...	...	...	...	...	...	...
60	7	...	...	...	...	...	...	...
60	9.5	...	...	...	...	...	...	...
60	12	...	...	...	...	...	...	...
40	7	...	...	...	...	...	...	...
40	9.5	...	...	...	...	...	...	...
40	12	...	...	...	...	...	...	...
20	7	...	...	...	...	...	...	...
20	9.5	...	...	...	...	...	...	...
20	12	...	...	...	...	...	...	...

Cuadro 4: Resultados de los ensayos con ventilador a diferentes velocidades y potencias.

6. Modelo Matemático de la Incubadora de Control de Temperatura

Se presenta un modelo matemático para describir el comportamiento de la temperatura interna de un ambiente controlado como función de la potencia aplicada al calefactor. El modelo se basa en las suposiciones siguientes:

- El efecto del ventilador se ignora.
- El sistema se aproxima como un sistema de primer orden.
- La temperatura inicial es $T_0 = 22°C$.

6.1. Función de Transferencia

Un sistema de primer orden en el dominio de Laplace se expresa mediante la siguiente función de transferencia:

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

Donde:

- $G(s)$ es la función de transferencia del sistema.
- K es la ganancia estática promedio del sistema ($0{,}1255\,°C/P\,\%$).
- τ es la constante de tiempo promedio del sistema (243.91 s).
- s es la variable compleja del dominio de Laplace.

Sustituyendo los valores dados, obtenemos:

$$G(s) = \frac{0{,}1255}{243{,}91s + 1} \tag{2}$$

6.2. Ecuación Diferencial en el Dominio del Tiempo

La ecuación diferencial equivalente en el dominio del tiempo es:

$$243{,}91\frac{dT(t)}{dt} + T(t) = 0{,}1255 \cdot P(t) \tag{3}$$

6.3. Solución en Estado Transitorio y Estacionario

Para una entrada de potencia constante $P(t) = P_0$, la solución es:

$$T(t) = T_0 + K \cdot P_0 \cdot \left(1 - e^{-t/\tau}\right) \tag{4}$$

En estado estacionario ($t \to \infty$), la temperatura alcanzada será:

$$T_\infty = T_0 + K \cdot P_0 \tag{5}$$

Por ejemplo, si $P_0 = 50\,\%$, la temperatura final será:

$$T_\infty = 22 + 0{,}1255 \cdot 50 = 28{,}275\,°C \tag{6}$$

6.4. Diagrama de Bloques

A continuación se presenta el diagrama de bloques del sistema:

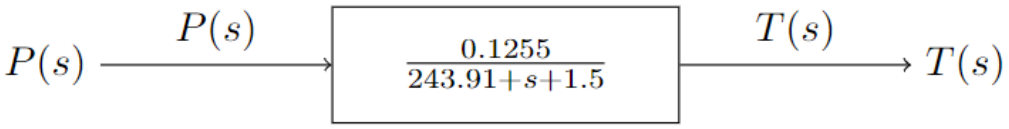

Figura 12: Diagrama de bloques del sistema de control de temperatura.

7. Temperaturas Alcanzables y Tiempo de Estabilización

Para determinar el **rango de temperaturas alcanzable** mediante el control de la potencia del calefactor y los **tiempos de respuesta asociados**, nos basamos en la función de transferencia obtenida previamente:

$$G(s) = \frac{0{,}1255}{243{,}91\,s + 1}$$

Donde:

- $K = 0{,}1255$ °C/P %: Ganancia estática del sistema.
- $\tau = 243{,}91$ s: Constante de tiempo.
- Temperatura inicial: $T_0 = 22$ °C.

7.1. Rango de Temperaturas Alcanzable

La temperatura máxima alcanzable dependerá de la potencia máxima aplicada al calefactor, que asumimos puede variar entre $0\,\%$ y $100\,\%$. Usando la ganancia estática K, se calcula la temperatura final como:

$$T_\infty = T_0 + K \cdot P_{\max}$$

Para $P_{\max} = 100\,\%$:

$$T_\infty = 22 + 0{,}1255 \cdot 100 = 34{,}55\,^\circ\text{C}$$

Para $P_{\min} = 0\,\%$:

$$T_\infty = 22 + 0{,}1255 \cdot 0 = 22\,^\circ\text{C}$$

Por lo tanto, el **rango de temperaturas alcanzable** es:

$$22\,^\circ\text{C} \leq T \leq 34{,}55\,^\circ\text{C}$$

7.2. Tiempo de Respuesta del Sistema

El tiempo de respuesta se refiere al tiempo que el sistema tarda en alcanzar un porcentaje significativo del valor final deseado. Para un sistema de primer orden, estos tiempos se calculan usando la constante de tiempo τ.

- **63.2 % del cambio total:**

$$t_{63{,}2\,\%} \approx \tau = 243{,}91\,\text{segundos} \approx 4{,}07\,\text{minutos}$$

- **95 % del cambio total:**
  El tiempo para alcanzar el 95 % del valor final es aproximadamente 5 veces la constante de tiempo:

$$t_{95\,\%} \approx 5 \cdot 243{,}91 = 1219{,}55\,\text{segundos} \approx 20{,}33\,\text{minutos}$$

7.3. Resumen del Control de Temperatura

- **Rango de temperaturas alcanzable:**
  - Mínima: 22 °C (sin potencia aplicada).
  - Máxima: 34.55 °C (con 100 % de potencia).
- **Tiempos de respuesta:**
  - 63.2 % del valor final: ≈ 4,07 minutos.
  - 95 % del valor final: ≈ 20,33 minutos.

7.4. Simulación en Python

El siguiente código en Python simula la respuesta de la temperatura para diferentes niveles de potencia. La temperatura se calcula usando una función exponencial basada en el modelo de primer orden.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Simulacion de la Respuesta de Temperatura a Diferentes Potencias

% Parametros del sistema
K = 0.1268  % Ganancia estatica promedio (C/P%)
tau = 243.54  % Constante de tiempo promedio (s)
P_max = 100  % Potencia maxima (P%)

% Asumiendo que tienes un DataFrame con la columna 'Temperatura␣(C)'
df = pd.read_csv('tu_archivo.csv')  % Reemplaza con el nombre correcto
    del archivo
temperatura = df['Temperatura␣(C)']
T_inicial = temperatura[:50].mean()  % Temperatura inicial (C)

% Vector de tiempo para la simulacion
tiempo = np.arange(0, 1200, 1)  # Desde 0 hasta 1200 s en incrementos de
    1 s

# Lista de niveles de potencia en porcentaje
niveles_potencia = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

# Crear la figura para las graficas
plt.figure(figsize=(12, 8))

for porcentaje_potencia in niveles_potencia:
    # Calculo de P y T_final para cada nivel de potencia
    delta_P = P_max * (porcentaje_potencia / 100)
    delta_T_final = K * delta_P

    # Calculo de la temperatura en funcion del tiempo
    T = T_inicial + delta_T_final * (1 - np.exp(-tiempo / tau))

    # Graficar la respuesta
    plt.plot(tiempo, T, label=f'{porcentaje_potencia}%␣de␣Potencia')

# Configuracion de la grafica
plt.title('Simulacion␣de␣la␣Respuesta␣de␣Temperatura␣a␣Diferentes␣
    Potencias')
plt.xlabel('Tiempo␣(s)')
plt.ylabel('Temperatura␣(C)')
plt.legend()
plt.grid(True)
plt.show()
```

7.5. Explicación del Código

- **Cálculo de ΔP:** La potencia aplicada se calcula como un porcentaje de la potencia máxima.
- **Cálculo de ΔT_{final}:** Se determina el aumento de temperatura esperado para cada nivel de potencia.
- **Simulación de la respuesta térmica:** La temperatura se calcula usando la ecuación:

$$T(t) = T_{\text{inicial}} + \Delta T_{\text{final}} \cdot (1 - e^{-t/\tau})$$

- **Gráfica:** Se generan curvas de temperatura en función del tiempo para diferentes niveles de potencia.

Resultados Simulaciones

El sistema permite controlar la temperatura interna entre **22 °C y 34.55 °C** según la potencia aplicada. El tiempo de respuesta para alcanzar el 95 % del valor final es de aproximadamente **20 minutos**. La simulación en Python confirma estos resultados mostrando cómo la temperatura evoluciona con diferentes niveles de potencia.

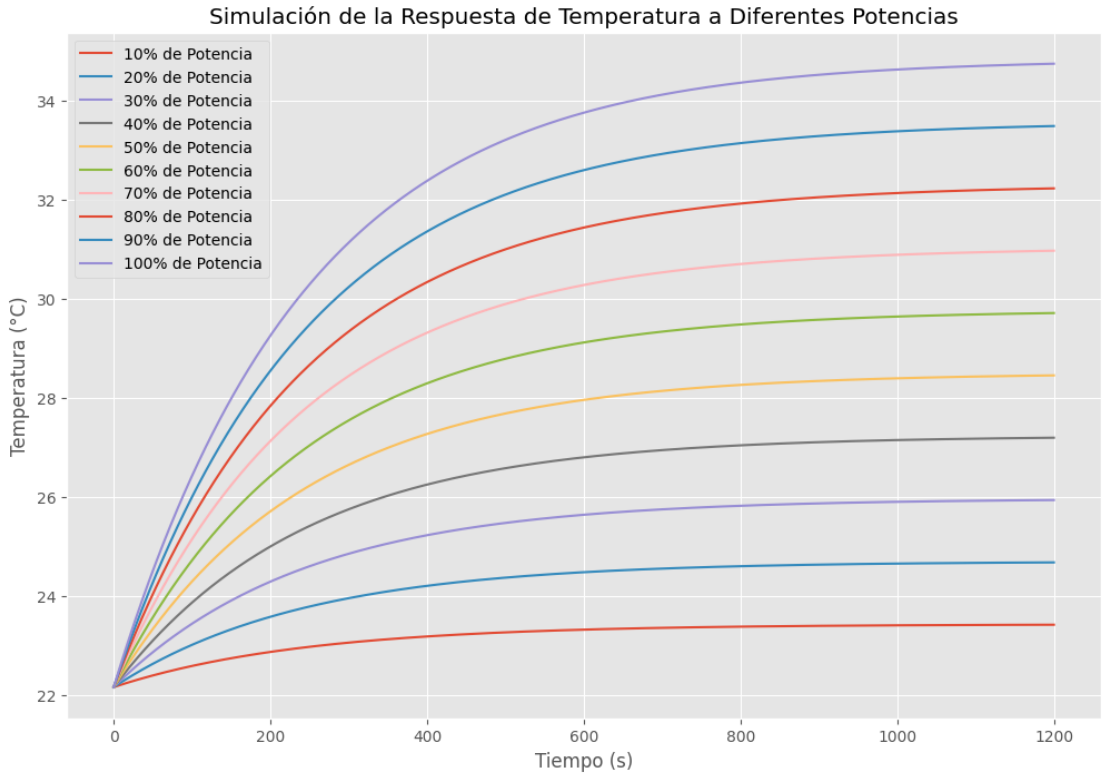

Figura 13: Simulación de la Respuesta de Temperatura a Diferentes Potencias

8. Algoritmo de Control PID en Arduino

El sistema a controlar es de naturaleza térmica, lo que implica una respuesta lenta a los cambios. Dado que no se espera un comportamiento rápido o cambios bruscos, se ha decidido utilizar un **controlador PI** (Proporcional-Integral) en lugar de un PID completo. La ausencia del término derivativo evita amplificar el ruido proveniente del sensor de temperatura.

8.1. Cálculo de los Parámetros del PID

El controlador PID se define por tres constantes:

- **Proporcional** (K_p): Controla la magnitud de la respuesta al error actual.
- **Integral** (K_i): Corrige errores acumulados en el tiempo, útil para eliminar el error en estado estacionario.
- **Derivativo** (K_d): Anticipa cambios futuros en el error, útil para sistemas rápidos. Sin embargo, no es necesario en sistemas térmicos.

Selección de los Valores:

- $K_p = 30{,}0$: Un valor alto para corregir rápidamente los errores iniciales.
- $K_i = 1{,}0$: Un valor moderado para evitar acumulación excesiva del error, proporcionando estabilidad en el estado estacionario.
- $K_d = 0{,}0$: No se utiliza el término derivativo debido a la lentitud y ruido del sistema.

Estos valores fueron determinados empíricamente mediante prueba y error usando el metodo **Ziegler-Nichols**, ajustando los parámetros hasta lograr una respuesta rápida sin oscilaciones significativas.

8.2. Implementación en Arduino

```
#include <PID_v1.h>

// Variables PID
double Setpoint, Input, Output;
double Kp = 30.0, Ki = 1.0, Kd = 0.0;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

// Control automatico
bool control_automatico = false;
float ultimasTemp[5] = {0};
int indiceTemp = 0;
```

```
void setup() {
  Serial.begin(115200);

  // Configuracion del PID
  myPID.SetMode(AUTOMATIC);
  myPID.SetOutputLimits(0, 100);
}

void loop() {
  // Medicion de temperatura cada 250 ms
  if (millis() - tiempoAnteriorTemp >= 250) {
    lecturaNTC = analogRead(NTC_PIN);
    resistenciaNTC = 100000.0 * ((1023.0 / lecturaNTC) - 1.0);
    logResistencia = log(resistenciaNTC);
    tempKelvin = 1.0 / (A_COEFF + B_COEFF * logResistencia + C_COEFF *
        pow(logResistencia, 3));
    tempCelsius = tempKelvin - 273.15;

    ultimasTemp[indiceTemp] = tempCelsius;
    indiceTemp = (indiceTemp + 1) % 5;

    if (control_automatico) {
      Input = calcularPromedio(ultimasTemp, 5);
      myPID.Compute();
      analogWrite(LUZ_PIN, map(Output, 0, 100, 0, 255));
    }

    tiempoAnteriorTemp = millis();
  }

  // Envio de datos por Serial cada 300 ms
  if (millis() - tiempoImprimir >= 300) {
    Serial.print(calcularPromedio(ultimasTemp, 5));
    Serial.print(";");
    Serial.print(velocidadRPM);
    Serial.print(";");
    Serial.println(Output);
    tiempoImprimir = millis();
  }

  // Lectura de comandos desde Serial
  if (Serial.available() > 0) {
    String comando = Serial.readStringUntil('\n');

    if (comando == "AUTOMATIC_ON") control_automatico = true;
    else if (comando == "AUTOMATIC_OFF") control_automatico = false;
    else if (comando.startsWith("SETPOINT")) Setpoint = comando.substring
```

```
        (9).toDouble();
  }
}
```

Este diseño e implementación de un controlador PI en Arduino permite gestionar la temperatura del ambiente, la potencia de la luz sin cambios bruscos. Los valores PID fueron seleccionados empíricamente para obtener una respuesta rápida y estable. La exclusión del término derivativo es adecuada dado que se trata de un sistema lento con ruido en las mediciones. La estructura modular del código permite extenderlo fácilmente para otras funcionalidades ya que al separar el control manual del ventilador, ampolleta con el control automatico permite usar uno u otro cuando se requiera.

9. Resultados Control de Temperatura con PID

10. Actividades y Cronograma

Se adjuntan las Actividades realizas para el trabajo figura 14.

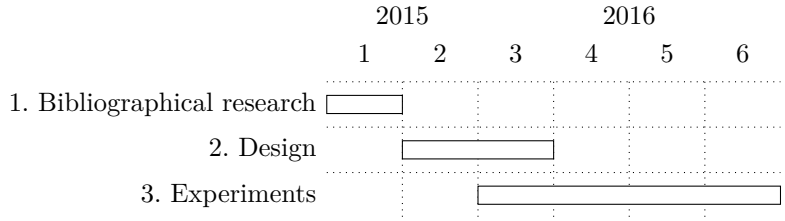

Figura 14: Schedule of activities in trimesters.

11. Conclusion