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 Constanza Rojas Garcia

Profesores:

Gonzalo Tapia Cabrera
Antonio José Rienzo Renato
Luis Togo Arredondo Gamboa

Índice

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 es 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 permitirá 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

2.1. General

Diseñar y desarrollar un sistema de control basado en Arduino para regular la temperatura de un prototipo de incubadora, integrando componentes electrónicos y software para su monitoreo, control y evaluación.

2.2. Específicos

1. Construir el prototipo de la incubadora empleando los componentes disponibles (ampolleta, ventilador y sensor NTC) e identificar sus características mediante mediciones, planos CAD y fotografías.

2. Diseñar un circuito que controle la potencia del calefactor, mida la velocidad del ventilador y registre la temperatura mediante Arduino.

3. Desarrollar un software en Python para la comunicación entre Arduino y un computador, capaz de controlar los elementos del circuito y registrar los datos obtenidos.

4. Implementar un modelo matemático que describa la relación entre la potencia del calefactor y la temperatura interna del prototipo, y usarlo para definir un rango de control y tiempos de respuesta.

5. Integrar un sistema de alarmas visuales y auditivas para indicar fallos, y crear una plataforma informática con base de datos para registrar y visualizar datos en tiempo real.

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

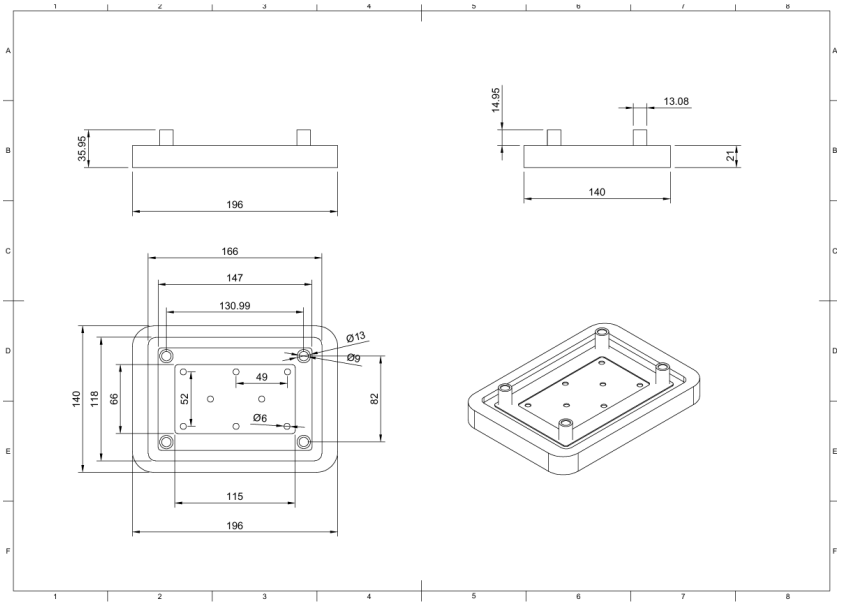

Figura 1: Dibujo técnico de la tapa.

3.2.2. Caja

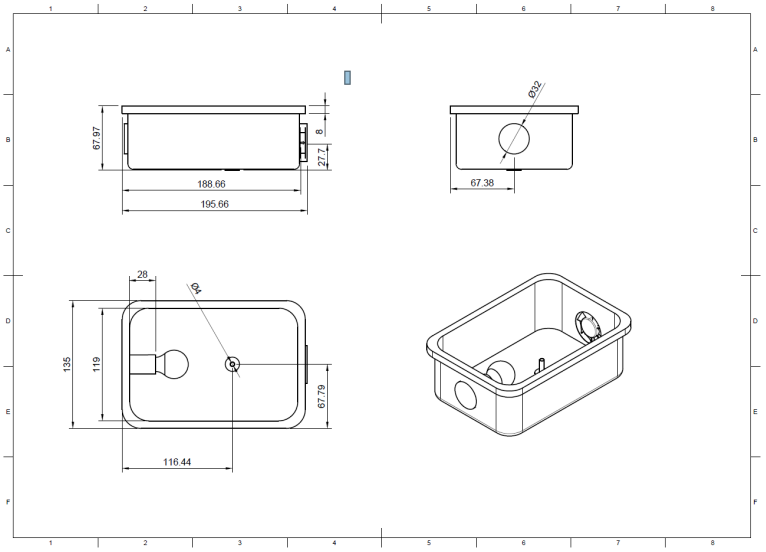

Figura 2: Dibujo técnico de la caja plástica.

3.2.3. Caja Cerrada

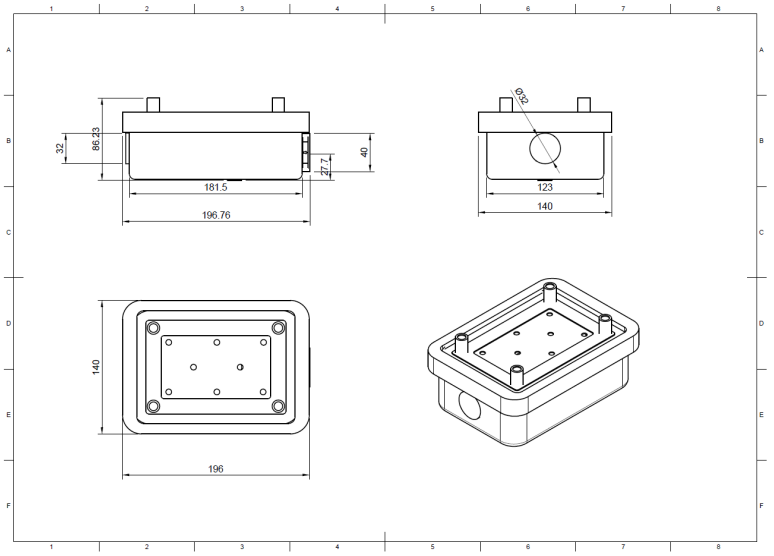

Figura 3: Dibujo técnico de la caja plástica con la tapa puesta.

3.3. Imágenes del prototipo

3.3.1. Modelo de incubadora

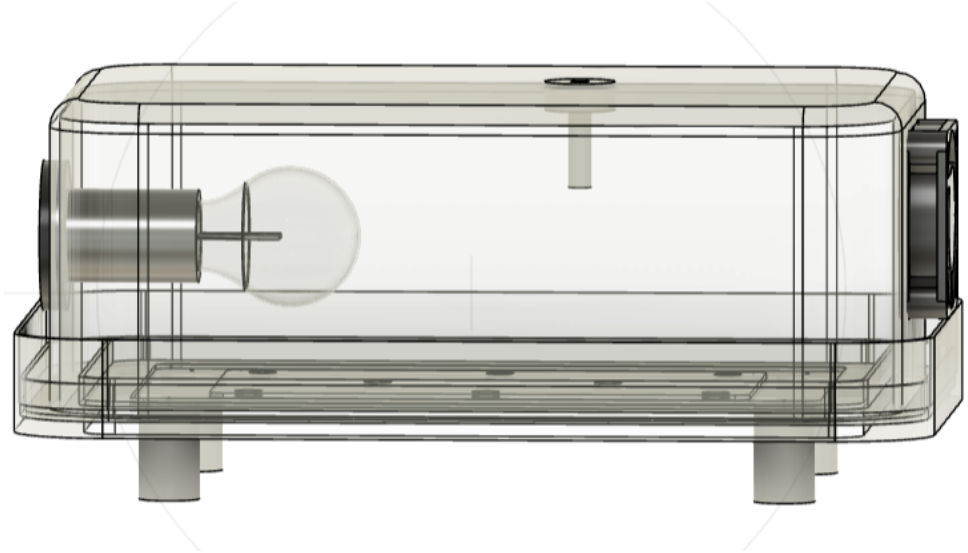

Figura 4: Diseño del modelo conceptual de la incubadora realizado en Fusion 360.

3.3.2. Prototipo incubadora real

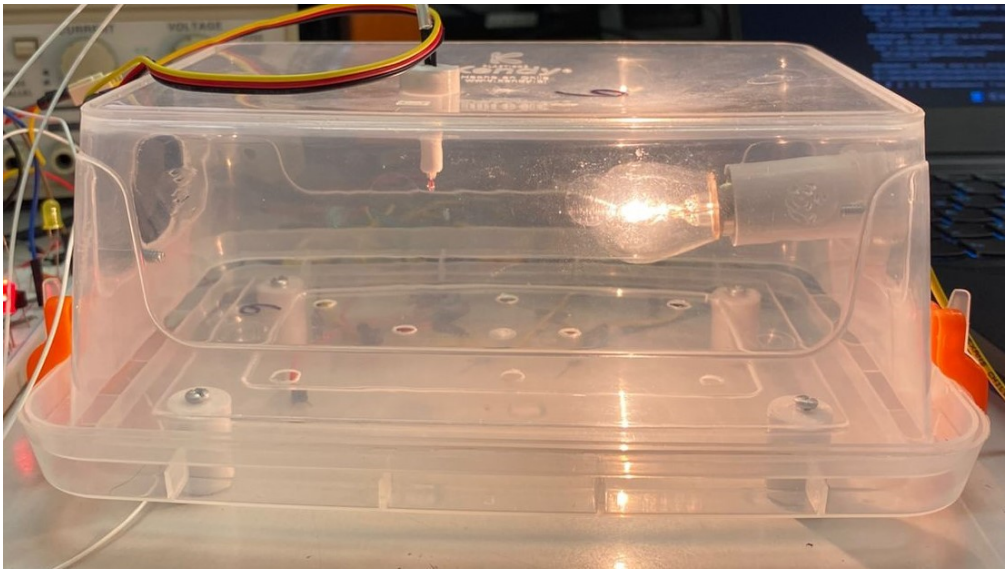

Figura 5: Prototipo físico funcional de la incubadora construido con una caja plástica.

3.4. Diseño del Circuito

3.4.1. Calculo de componentes a usar

Ampolleta de 12V y 1.7A

Se utilizará un transistor TIP41C para controlar la ampolleta, ya que este cuenta con un rango de funcionamiento adecuado para la corriente que se manejará.

En primer lugar, determinamos el valor de i_A, el cual obtuvimos de manera experimental al medir la corriente consumida por la ampolleta cuando operaba a su máxima potencia.

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

El rango de ganancia de corriente (h_{FE}) para el transistor TIP41C está comprendido entre 15 y 75. En este caso, utilizamos un valor mínimo de $h_{FE} = 15$, ya que al seleccionar el límite inferior del rango nos aseguramos de que el diseño funcione incluso si el transistor presenta una ganancia de corriente reducida dentro de las especificaciones.

Ya conociendo estos valores, podemos calcular la corriente de base i_b con la siguiente fórmula:

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

Sustituyendo los valores, obtenemos:

$$i_b = \frac{1{,}7}{15} = 0{,}113\,\text{A}$$

Finalmente, para calcular la resistencia de base (R_b), usamos la fórmula:

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

Donde 0,7 V corresponde a la caída de potencial típica en la unión base-emisor de un transistor de silicio. Sustituyendo los valores, obtenemos:

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

$$R_b = 100\,\Omega$$

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.

En primer lugar, determinamos el valor de la corriente colector (i_C) correspondiente al ventilador. Este valor se obtuvo de manera experimental y es:

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

El rango de ganancia de corriente (h_{FE}) para el transistor 2N2222 está comprendido entre 30 y 75. En este caso, utilizamos el valor mínimo de $h_{FE} = 30$, ya que al seleccionar el límite inferior del rango nos aseguramos de que el diseño funcione incluso si el transistor presenta una ganancia de corriente reducida dentro de las especificaciones.

La corriente de base i_b se calcula como:

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

Sustituyendo los valores obtenemos:

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

Finalmente, para calcular la resistencia de base (R_b), usamos la fórmula:

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

Donde 0,7 V corresponde a la caída de potencial típica en la unión base-emisor de un transistor de silicio. Sustituyendo los valores, obtenemos:

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

$$R_b = 5650\,\Omega$$

3.4.2. Esquemático

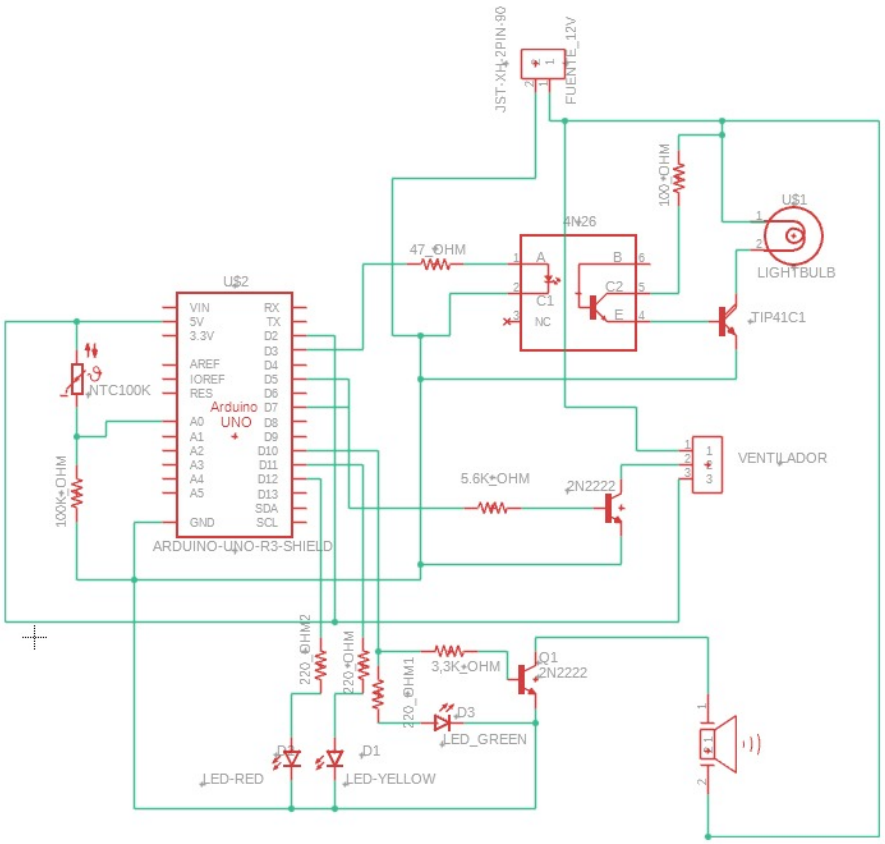

Figura 6: Diseño del circuito de control de la incubadora.

Razon de uso del optoacoplador

El **optoacoplador 4N26** se utiliza para proporcionar **aislamiento eléctrico** entre el Arduino y el circuito de potencia, protegiendo al microcontrolador de sobrecargas, interferencias y picos de voltaje. Este aislamiento permite que el Arduino controle la lámpara de manera segura y estable.

Resistencia del optoacoplador

La resistencia colocada en serie con el LED interno del optoacoplador tiene la función principal de limitar la corriente que atraviesa el LED para evitar daños y garantizar su operación adecuada. El cálculo de esta resistencia se realiza utilizando la Ley de Ohm:

$$R = \frac{V_{\text{fuente}} - V_{\text{LED}}}{I_{\text{LED}}}$$

Donde:

- V_{fuente} es el voltaje suministrado por el Arduino (5 V).
- V_{LED} es la caída de voltaje típica del LED interno del optoacoplador (1,2 V).
- I_{LED} es la corriente recomendada para el LED (10 mA a 20 mA).

Sustituyendo valores:

$$R = \frac{5 - 1{,}2}{0{,}02} = \frac{3{,}8}{0{,}02} = 190\,\Omega$$

Aunque el cálculo teórico sugiere un valor cercano a **190** Ω, en este circuito se utiliza una resistencia de **47** Ω, para permitir una corriente ligeramente mayor que asegure la activación confiable del optoacoplador. Si la resistencia fuera demasiado alta, el LED no se activaría; si fuera demasiado baja, podría dañarse. Este valor es un equilibrio que garantiza la seguridad y eficacia del circuito.

3.4.3. Conexiones del circuito

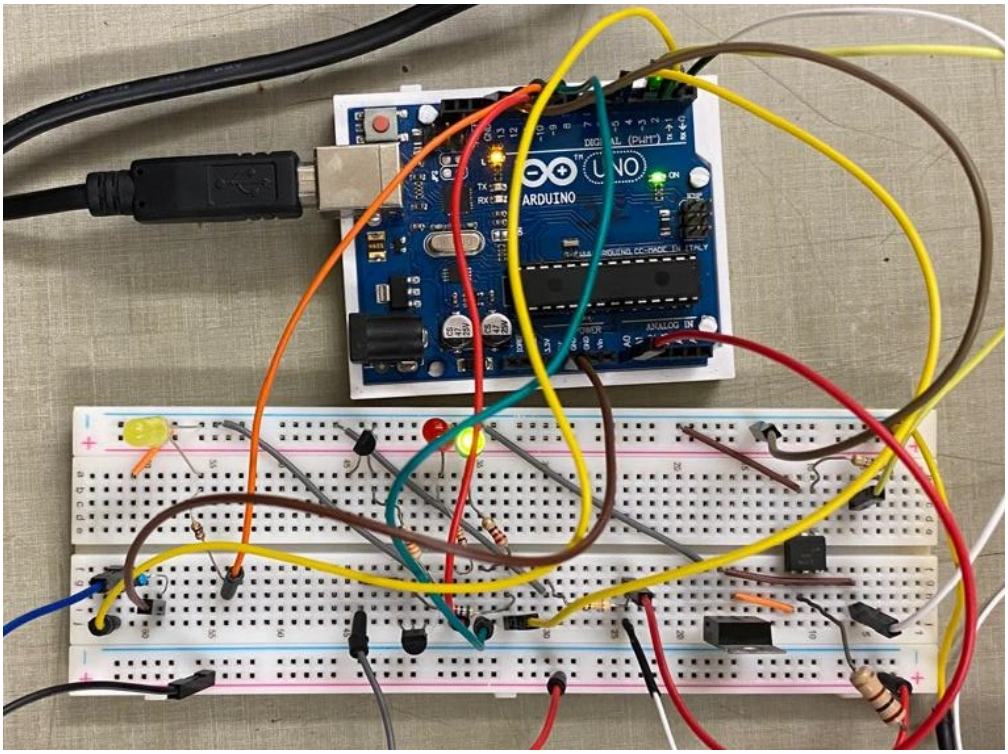

Figura 7: Protoboard con componentes conectados.

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.

  La ecuación de Steinhart-Hart se utiliza para convertir la resistencia medida de un sensor NTC a una temperatura en grados Kelvin. Esta ecuación está dada por:

$$\frac{1}{T} = A + B\ln(R) + C\ln^3(R) \tag{1}$$

  Donde:

  - T: Temperatura en Kelvin (K).
  - R: Resistencia medida del sensor NTC (Ω).
  - A, B, C: Coeficientes específicos del sensor, determinados por calibración.

La temperatura en grados Celsius (T_C) se calcula como:

$$T_C = T - 273{,}15 \tag{2}$$

Estos cálculos son implementados en el código para obtener la temperatura en tiempo real a partir de los valores de resistencia del sensor.

El código completo que implementa estas funciones puede consultarse en el anexo A.

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.

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. Se presenta un extracto del código Python que implementa estas funciones en el anexo B.

Interfaz gráfica de la plataforma de control de la incubadora y Gráficos en Tiempo Real

El programa emplea la biblioteca PyQtGraph para generar gráficos en tiempo real que ilustran la evolución de variables clave, como la temperatura, la velocidad del ventilador y las potencias de los actuadores, a lo largo del tiempo. A continuación, se muestra el fragmento de código encargado de actualizar los gráficos:

```
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)
```

Se presentan dos ejemplos que ilustran el funcionamiento de la interfaz durante las pruebas de desempeño del código. Estas pruebas permiten analizar su comportamiento en condiciones controladas, identificar posibles áreas de mejora y realizar los ajustes necesarios para optimizar su rendimiento.

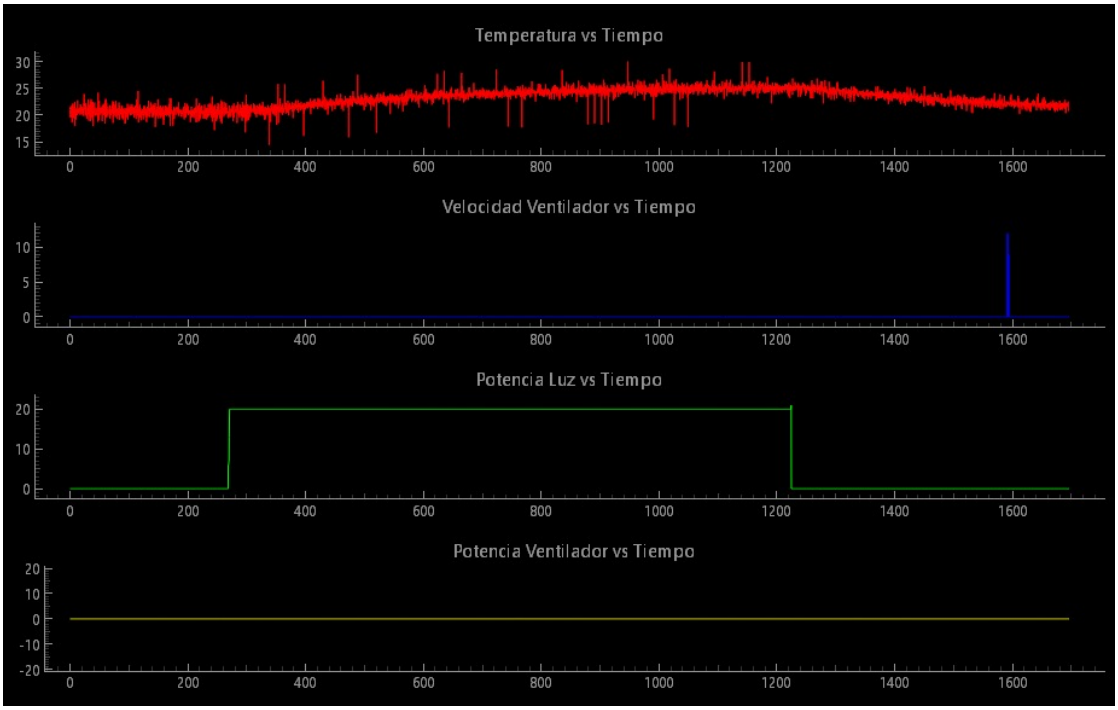

Figura 8: Interfaz en funcionamiento 1.

En la Fig.8, visualizamos el comportamiento del sistema de la incubadora durante un periodo de funcionamiento sin el uso del ventilador. La gráfica de temperatura (rojo) refleja un incremento gradual, estabilizándose en un rango cercano a los 25-30 °C, y la potencia de luz (verde) presenta incrementos en etapas a lo largo del tiempo.

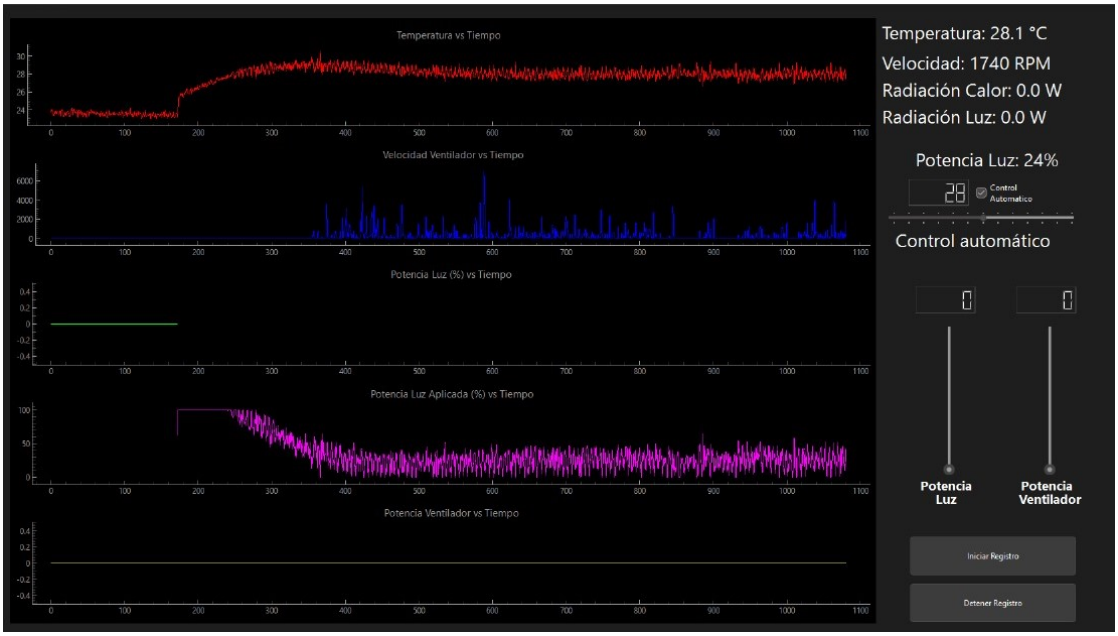

Figura 9: Interfaz en funcionamiento 2 (Control automático.)

En la Fig.9, visualizamos el monitoreo en tiempo real del sistema de la incubadora, donde la temperatura (rojo) aumenta hasta estabilizarse en un rango controlado. La velocidad del ventilador (azul) presenta fluctuaciones constantes, mientras que la potencia de luz aplicada (morado) disminuye progresivamente tras un periodo inicial elevado, ajustándose dinámicamente para mantener la estabilidad térmica.

5. Ensayos y Resultados

5.1. Ensayos de Control de Temperatura con Calefactor

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 en el gráfico de la Figura 10, donde podemos visualizar la evolución de la temperatura y la potencia de luz aplicada de estos cinco ensayos diferentes, con niveles de potencia constantes (20 %, 40 %, 60 %, 80 %, y 100 %).

Las curvas de temperatura presentan un comportamiento exponencial con un aumento inicial rápido seguido de una estabilización progresiva, alcanzando valores más altos en los ensayos con mayor potencia. Las líneas horizontales punteadas representan la potencia constante aplicada durante cada ensayo, evidenciando una relación directa entre la intensidad de la luz y la temperatura alcanzada. Este análisis permite evaluar cómo la potencia de luz influye en la respuesta térmica del sistema.

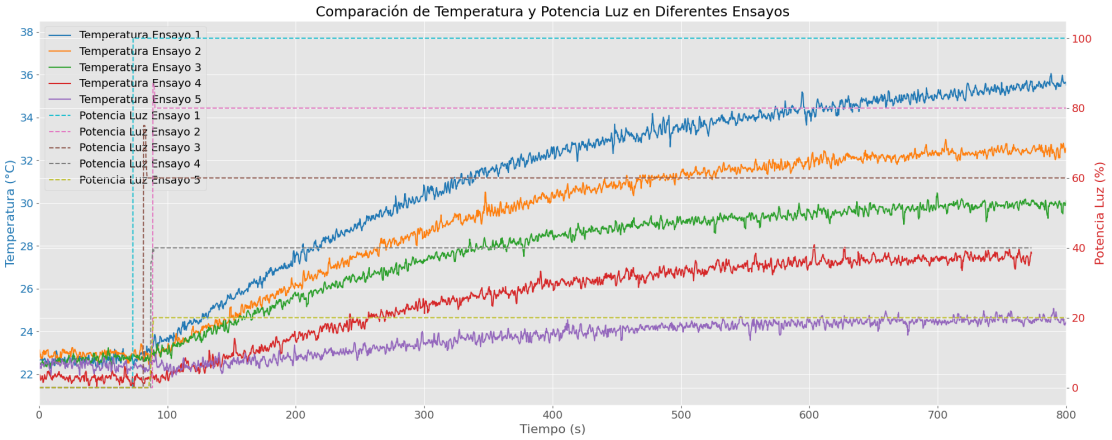

Figura 10: Gráfica de potencias y temperaturas

5.1.1. Resultados

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 asentamiento t_s, la diferencia de temperatura, la temperatura maxima para cada ensayo y la potencia de radiación.

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

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

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 considerar una eficiencia lumínica estimada de la ampolleta del 0.05 %, basada en valores típicos de lámparas incandescentes, se puede calcular la potencia de radiación.

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

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

Cálculo de la Ganancia Estática (K)

La **ganancia estática (K)** de un sistema térmico se define como la relación entre el cambio en la salida del sistema (ΔT, cambio en la temperatura) y el cambio en la entrada ($P_{\text{Radiación}}$, potencia de radiación aplicada). Matemáticamente, se expresa como:

$$K = \frac{\Delta T}{P_{\text{Radiación}}} \tag{3}$$

La ganancia estática mide la sensibilidad del sistema térmico, es decir, cuánto cambia la temperatura del sistema por cada watt de potencia aplicada.

- Una **mayor ganancia estática** indica que el sistema es más sensible a los cambios en la potencia de radiación.
- Una **menor ganancia estática** sugiere que el sistema es más inerte o menos reactivo.

La ganancia estática permite caracterizar el comportamiento del sistema en términos de su respuesta térmica. Y es de gran importancia para el desarrollo de actividades posteriores, ya que define la relación directa entre la potencia de radiación ($P_{\text{Radiación}}$) y el cambio de temperatura (ΔT). Este indicador resulta fundamental para el diseño y ajuste de controladores térmicos

Se presentan los resultados obtenidos en la siguiente tabla:

Ensayo	Potencia (%)	Temp. Inicial (°C)	Temp. Final (°C)	Cambio en Temp. (°C)	Potencia de Radiación (W)	Ganancia estática (K)
1	100	23.25	37.13	13.88	18.6	0.1389
2	80	22.95	32.52	9.57	14.9	0.1195
3	60	22.88	30.01	7.13	11.2	0.1188
4	40	21.79	27.46	5.67	7.4	0.1417
5	20	22.43	24.60	2.17	3.7	0.1085

Cuadro 2: Resultados de los ensayos de temperatura sin ventilador.

5.2. Ensayos de Temperatura con Distintas Velocidades 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.

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

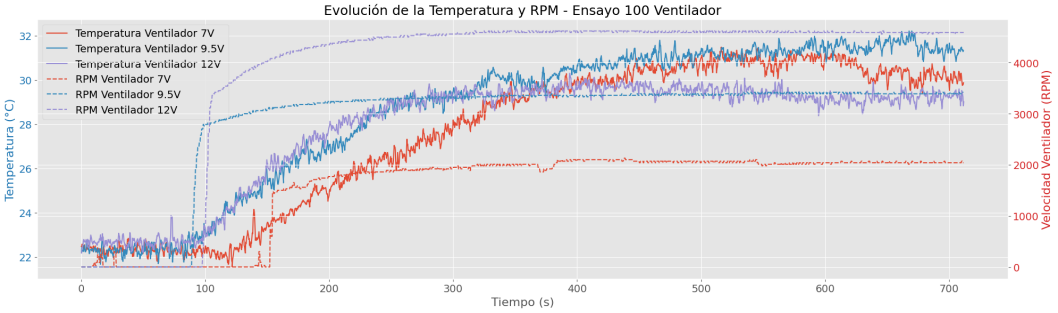

Figura 11: Respuesta de la temperatura a un escalón de 100 % de potencia y 3 velocidades de ventilador.

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

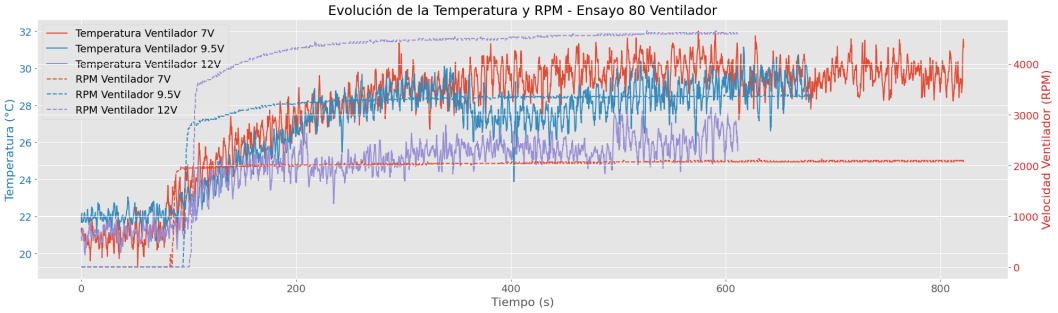

Figura 12: Respuesta de la temperatura a un escalón de 80 % de potencia y 3 velocidades de ventilador.

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

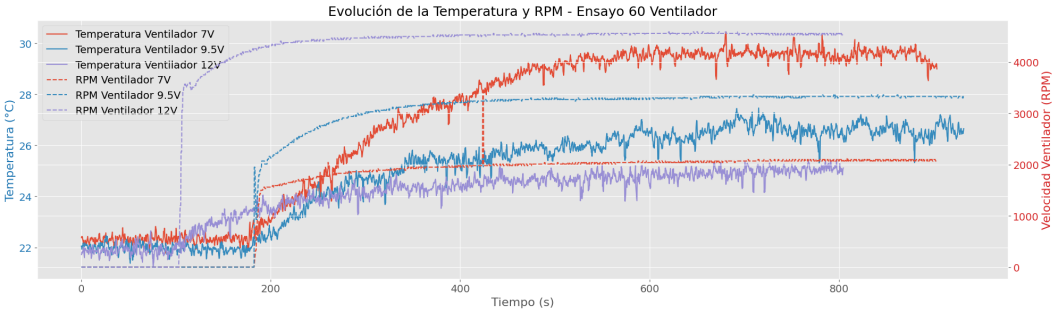

Figura 13: Respuesta de la temperatura a un escalón de 60 % de potencia y 3 velocidades de ventilador.

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

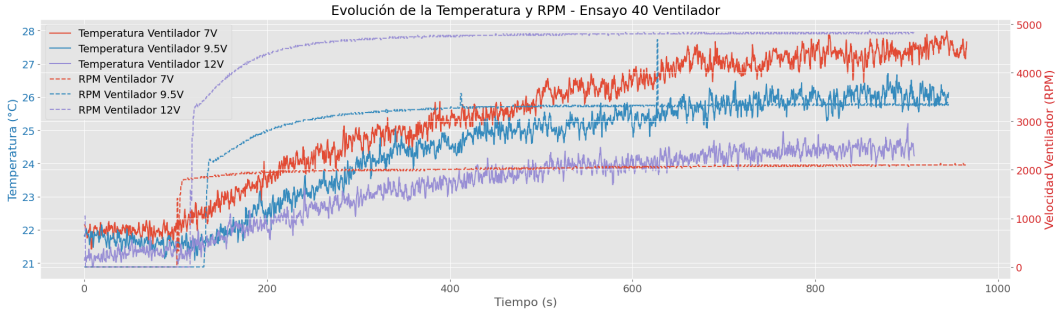

Figura 14: Respuesta de la temperatura a un escalón de 40 % de potencia y 3 velocidades de ventilador.

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

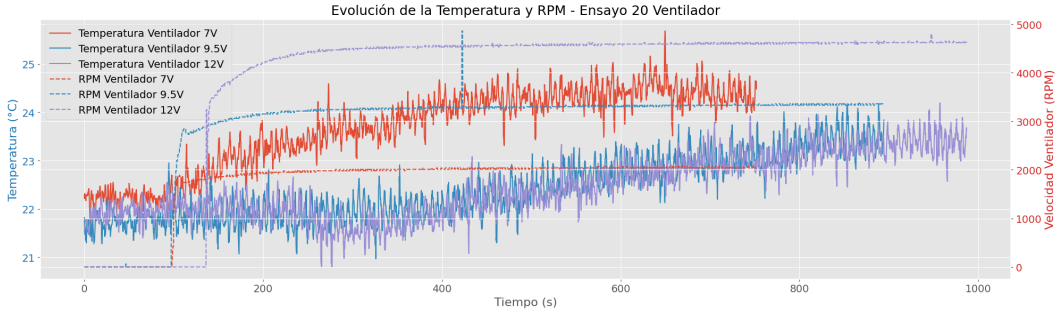

Figura 15: Respuesta de la temperatura a un escalón de 20 % de potencia y 3 velocidades de ventilador.

5.2.6. Resultados

En los ensayos realizados, se concluye que el ventilador a 12V es el más eficaz en todas las condiciones de potencia, proporcionando un enfriamiento rápido y estable, especialmente en potencias altas. Esto, según los valores obtenidos para el tiempo de estabilización (t_s) y la ganancia estática (K). Por ejemplo, para un escalón del 100 % de potencia, el ventilador a 12V logró un tiempo de estabilización de 60 segundos, mientras que a 9.5V y 7V los tiempos fueron de 70 y 80 segundos respectivamente. Además, el valor de la ganancia estática K disminuye a medida que aumenta la velocidad del ventilador, reflejando una mayor estabilidad térmica.

El tiempo de estabilización (t_s) se define como el tiempo requerido para que la temperatura alcance el 95 % de su valor final tras un cambio en la potencia aplicada. Este criterio se utiliza para evaluar la rapidez con la que el sistema responde a cambios en las condiciones operativas y estabiliza su comportamiento.

Y la ganancia estática (K) se calcula como la relación entre el cambio total de temperatura (ΔT_{final}) y el porcentaje de potencia aplicada (P %), dado por la ecuación:

$$K = \frac{\Delta T_{\text{final}}}{\text{P}\,\%} \tag{4}$$

En la siguiente tabla se presentan los resultados de los ensayos de temperatura con ventilador:

Potencia (%)	**Ventilador (V)**	T_{inicial} (°C)	T_{final} (°C)	ΔT_{final} (°C)	t_s (s)	K (°C/P %)
100	7	21.83	30.28	8.45	80	0.0845
100	9.5	22.77	31.28	8.51	70	0.0851
100	12	22.77	29.01	6.24	60	0.0624
80	7	22.74	31.14	8.40	75	0.1050
80	9.5	22.59	28.33	5.74	65	0.0718
80	12	22.02	25.54	3.52	60	0.0440
60	7	22.38	29.17	6.79	70	0.1132
60	9.5	22.05	26.65	4.60	60	0.0767
60	12	21.95	25.11	3.16	55	0.0527
40	7	22.15	27.66	5.51	60	0.1377
40	9.5	22.08	26.13	4.05	55	0.1012
40	12	21.82	24.48	2.66	50	0.0665
20	7	21.37	24.23	2.86	55	0.1430
20	9.5	21.82	23.84	2.02	50	0.1010
20	12	21.77	23.47	1.70	45	0.0850

Cuadro 3: Resultados de los ensayos con ventilador a diferentes velocidades y potencias.

En el cuadro 3, hemos incorporado el cálculo de la ganancia estática y el tiempo de estabilización a los resultados. Esta información es necesaria para comprender cómo la velocidad del ventilador impacta la temperatura alcanzada en el sistema, lo que a su vez ayudará a ajustar los parámetros operativos en tiempo real para futuras tareas.

5.3. Comparación Pruebas Con y Sin Ventilador

Al comparar las pruebas con y sin ventilador, se observa en el Cuadro 4 una clara reducción de la temperatura final cuando el ventilador está encendido, independientemente de su velocidad. Esto resalta el efecto significativo del ventilador en la reducción de temperatura.

Además podemos notar que el impacto del ventilador es más evidente a potencias altas, donde la reducción de temperatura es más significativa.

Potencia (%)	**Ventilador (V)**	T_{final} (°C) sin Ventilador	T_{final} (°C) con Ventilador	ΔT_{final} (°C) sin Ventilador	ΔT_{final} (°C) con Ventilador
100	7	37.13	30.28	13.88	8.45
100	9.5	37.13	31.28	13.88	8.51
100	12	37.13	29.01	13.88	6.24
80	7	32.52	31.14	9.57	8.40
80	9.5	32.52	28.33	9.57	5.74
80	12	32.52	25.54	9.57	3.52
60	7	30.01	29.17	7.13	6.79
60	9.5	30.01	26.65	7.13	4.60
60	12	30.01	25.11	7.13	3.16
40	7	27.46	27.66	5.67	5.51
40	9.5	27.46	26.13	5.67	4.05
40	12	27.46	24.48	5.67	2.66
20	7	24.60	24.23	2.17	2.86
20	9.5	24.60	23.84	2.17	2.02
20	12	24.60	23.47	2.17	1.70

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 $T_0 = 22°C$ (T° ambiente).

6.1. Función de Transferencia

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

$$G(s) = \frac{K}{\tau s + 1} \tag{5}$$

Donde:

- $G(s)$ es la función de transferencia del sistema.
- K es la ganancia estática promedio del sistema ($0{,}1255\,°C/P\,\%$).

Se calcula como el cambio en la temperatura ΔT dividido por el cambio en la potencia aplicada ΔP. Experimentalmente, seleccionamos un nivel de potencia del 0 % al 100 %, donde la temperatura cambia de $22°C$ a $34{,}55°C$, calculando tenemos:

$$K = \frac{34{,}55 - 22}{100 - 0} = 0{,}1255\,°C/\,\%P$$

- τ es la constante de tiempo promedio del sistema (243.91 s).

  Para calcular la constante de tiempo, primero determinamos el 63.2 % del cambio total, el cual proviene de la ecuación característica de un sistema de primer orden y está relacionado con el tiempo que el sistema tarda en alcanzar el 63.2 % de su cambio total hacia el estado estacionario. Esto se ve representado en la gráfica de la Figura 17.

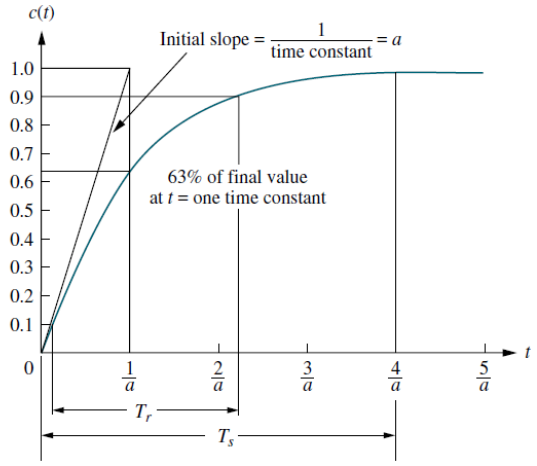

Figura 16: Salida de un sistema de primer orden a una entrada escalon unitario

$$T_{63,2} = T_{\text{inicial}} + 0{,}632 \cdot (T_{\text{final}} - T_{\text{inicial}})$$

Reemplazamos en la ecuación los valores de $T_{\text{inicial}} = 22°C$ y $T_{\text{final}} = 34{,}55°C$.

$$T(t_{63,2}) = 22 + 0{,}632 \cdot (34{,}55 - 22) = 29{,}93°C.$$

En los datos experimentales, se observa que $T = 29{,}93°C$ ocurre en $t = 243{,}91\,\text{s}$.

Por lo tanto:

$$\tau = 243{,}91\,\text{s}.$$

- s es la variable compleja del dominio de Laplace.

Sustituyendo los valores dados, obtenemos:

$$G(s) = \frac{0{,}1255}{243{,}91s + 1} \tag{6}$$

6.2. Ecuación Diferencial en el Dominio del Tiempo

La ecuación diferencial equivalente en el dominio del tiempo que describe la dinámica del sistema térmico es:

$$243{,}91\frac{dT(t)}{dt} + T(t) = 0{,}1255 \cdot P(t) \tag{7}$$

Donde:

- $T(t)$: Temperatura del sistema en función del tiempo ($^{\circ}C$).
- $P(t)$: Potencia aplicada al sistema en función del tiempo (%).
- $\tau = 243{,}91\,\text{s}$: Constante de tiempo que define la rapidez con la que el sistema responde a cambios en la potencia.
- $K = 0{,}1255\,^{\circ}C/\,\%$: Ganancia estática que relaciona el cambio de temperatura con el cambio de potencia aplicada.

6.3. Solución en Estado Transitorio y Estacionario

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

$$T(t) = T_0 + K \cdot P_0 \cdot \left(1 - e^{-t/\tau}\right) \tag{8}$$

En estado estacionario ($t \to \infty$), el sistema alcanza un estado de equilibrio donde la temperatura es constante y está dada por:

$$T_\infty = T_0 + K \cdot P_0 \tag{9}$$

Este modelo permite predecir cómo evoluciona la temperatura en función de la potencia aplicada y evaluar el comportamiento del sistema térmico en diversas condiciones operativas.

6.4. Diagrama de Bloques

A continuación se presenta el diagrama de bloques del sistema:

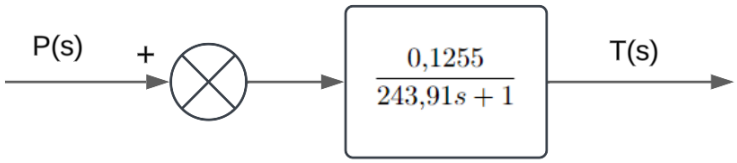

Figura 17: Diagrama de bloques del sistema de lazo abierto.

Este diagrama de bloques representa el funcionamiento de un sistema de lazo abierto, donde la entrada ($P(s)$) se aplica directamente al sistema dinámico, representado por la función de transferencia $G(s)$, para generar la salida ($T(s)$).

En un sistema de lazo abierto, la salida no se retroalimenta al sistema, por lo que no se realizan ajustes automáticos para corregir errores, lo que hace de este modelo una herramienta básica para analizar el comportamiento inicial del sistema y sentar las bases para los desarrollos posteriores.

6.5. Ejemplo del comportamiento del módelo en relación con los datos experimentales

Analizamos el comportamiento del modelo matemático propuesto en comparación con los datos experimentales obtenidos para una potencia aplicada del 50 %. Para validar la precisión, se calculan las temperaturas predichas por el modelo y se comparan con las temperaturas experimentales medidas en tiempos específicos.

- Temperatura Modelo para una potencia aplicada $P_0 = 50\,\%$:

$$T(t) = 22 + 0{,}1255 \cdot 50 \cdot (1 - e^{-t/243{,}91})$$

- El error porcentual se calcula como:

$$\text{Error } (\,\%) = \frac{|T_{\text{exp}} - T_{\text{modelo}}|}{T_{\text{modelo}}} \times 100$$

El modelo matemático describe de manera adecuada el comportamiento térmico del sistema, aunque presenta ligeras diferencias entre las temperaturas predichas y las experimentales. A pesar de estas discrepancias, el modelo captura correctamente la tendencia general de la evolución de la temperatura.

Tiempo (s)	Temperatura Experimental ($°C$)	Temperatura Modelo ($°C$)	Error (%)
0	22.00	22.00	0.00
100	24.50	24.11	1.62
200	26.80	25.51	5.05
300	28.20	26.44	6.65
400	28.60	27.06	5.70
500	28.70	27.45	4.55

Cuadro 5: Comparación entre las temperaturas experimentales y las predichas por el modelo, más el cálculo del error porcentual.

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

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

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

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

En base a estos cálculos, el rango de temperaturas alcanzable, con un $T_0 = 22°C$ es:

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

Sin embargo, si la temperatura ambiente es mayor o menor, este rango no es representativo y debe ajustarse proporcionalmente. Por lo que finalmente definimos un rango

operativo seguro de:

$$24°C \leq T \leq 33°C$$

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: 24 °C (sin potencia aplicada).
  - Máxima: 33 °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. 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.

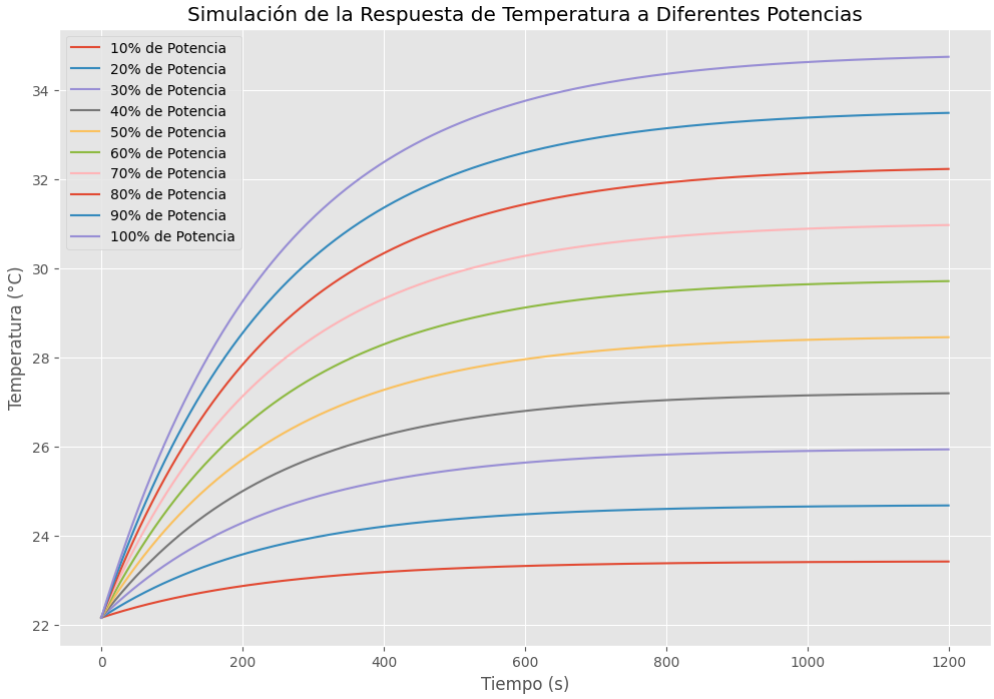

Figura 18: Simulación de la Respuesta de Temperatura a Diferentes Potencias

8. Algoritmo de Control PI 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 PI

El controlador PI 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.

Para determinar los parámetros del controlador PI, se empleó el método de Ziegler-Nichols como punto de partida, el cual permite calcular valores iniciales para K_p y K_i al provocar oscilaciones sostenidas en el sistema mediante ajustes incrementales de K_p. A partir de estos valores iniciales, se realizaron simulaciones iterativas en Python para optimizar los parámetros, ajustando el modelo del sistema con el objetivo de lograr una respuesta rápida, estable y con un error mínimo en estado estacionario. A continuación, se detalla el procedimiento seguido para los cálculos.

Determinación del Valor Crítico ($K_{p,\textbf{crítico}}$)

Primero, se configuró el sistema utilizando únicamente control proporcional, estableciendo los parámetros $K_i = 0$ y $K_d = 0$. De esta manera, solo K_p afectaba el comportamiento del sistema. A partir de un valor inicial bajo, se incrementó K_p gradualmente mientras se observaba la respuesta del sistema.

Durante este proceso, se monitoreó la salida del sistema para identificar el punto en el que las oscilaciones comenzaban a ser sostenidas, es decir, cuando la amplitud de las oscilaciones se mantenía constante con el tiempo. Este punto marcó el límite de estabilidad del sistema.

En ese momento, se registraron dos valores clave:

- $K_{p,\textbf{crítico}}$: El menor valor de K_p que provocó oscilaciones sostenidas en la salida.
- $T_{\textbf{crítico}}$: El período de las oscilaciones, medido como el tiempo entre dos picos consecutivos en la señal de salida.

Cálculo de los Parámetros Iniciales con Ziegler-Nichols

Para calcular los parámetros iniciales del controlador PI, se calcularon los valores iniciales de K_p y K_i utilizando las siguientes ecuaciones de Ziegler-Nichols:

$$K_p = 0{,}45 \cdot K_{p,\text{crítico}}$$

$$K_i = \frac{K_p}{T_{\text{crítico}}}$$

Luego sustituimos los valores críticos obtenidos experimentalmente en la ecuación.

- $K_{p,\text{crítico}} = 60$: Valor crítico del controlador proporcional que provocó oscilaciones sostenidas en el sistema.
- $T_{\text{crítico}} = 20\,\text{s}$: Período de las oscilaciones sostenidas.

$$K_p = 0{,}45 \cdot 60{,}0 = 27{,}0$$

$$K_i = \frac{27{,}0}{20{,}0} = 1{,}35$$

Ajuste Iterativo de los Parámetros

Los valores obtenidos mediante Ziegler-Nichols ($K_p = 27{,}0$, $K_i = 1{,}35$) fueron utilizados como punto de partida. Para garantizar un desempeño óptimo del sistema, los parámetros se ajustan iterativamente mediante simulaciones.

Tras estas simulaciones, se determinaron los valores finales óptimos para el sistema de control de temperatura:

- $K_p = 30{,}0$: 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.

El código utilizado para la simulación de los valores PI se detalla en el Anexo B.

8.2. Análisis de la Simulación de la Respuesta PI

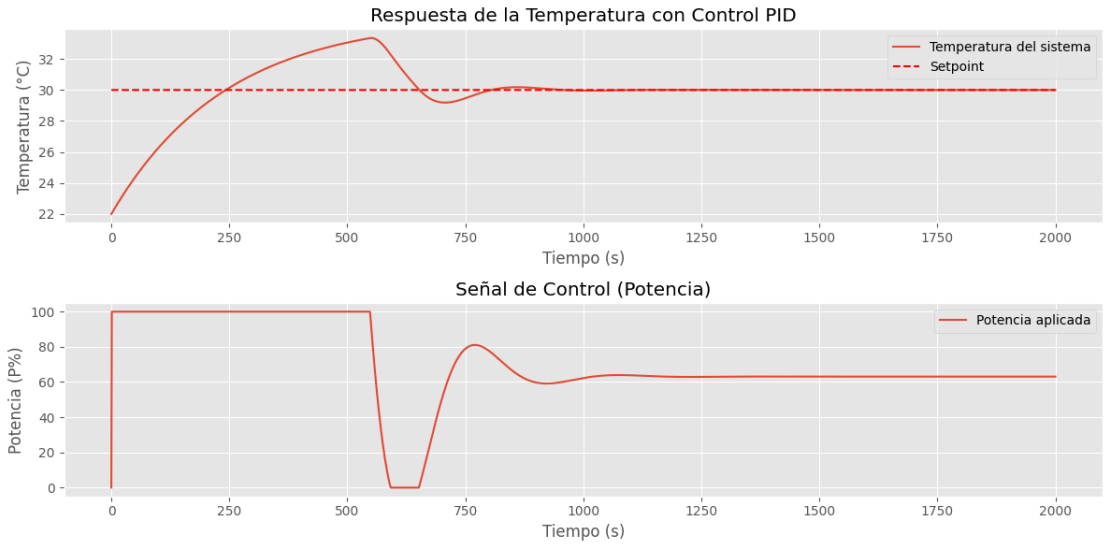

Figura 19: Simulación de la Respuesta de Temperatura con Control PI

La Figura 19 muestra los resultados de la simulación del sistema térmico controlado mediante un controlador PI. En la primera gráfica, se observa la respuesta de la temperatura del sistema, mientras que la segunda gráfica ilustra la evolución de la señal de control (potencia) aplicada.

En la gráfica superior, se evidencia cómo la temperatura del sistema evoluciona en función del tiempo al aplicar un control PI. Inicialmente, se produce un **sobreimpulso** donde la temperatura supera brevemente el setpoint deseado de 30 °C antes de estabilizarse. Este comportamiento es característico de sistemas controlados por PI y ocurre debido al ajuste proporcional del error inicial. Posteriormente, el sistema logra estabilizarse de manera eficiente alrededor del setpoint, con oscilaciones mínimas, lo que refleja un buen desempeño del controlador.

La gráfica inferior muestra la evolución de la potencia aplicada para controlar la temperatura. Al inicio, se aplica la potencia máxima (100 %) para compensar rápidamente el error inicial y alcanzar el setpoint. Conforme la temperatura se aproxima al valor objetivo, la potencia disminuye gradualmente y presenta ligeras fluctuaciones que reflejan el ajuste dinámico del controlador para mantener el sistema estable en el setpoint.

En conjunto, los resultados demuestran un **tiempo de estabilización corto**, con una respuesta rápida, donde las oscilaciones y el error en estado estacionario son mínimos, validando los valores seleccionados para los parámetros del controlador ($K_p = 30{,}0$, $K_i = 1{,}0$, $K_d = 0{,}0$).

8.3. Implementación y Resultados del Control PI en Arduino

El diseño e implementación del controlador PI en Arduino (ver Anexo A) permite gestionar de manera eficiente la temperatura del ambiente controlado, ajustando dinámicamente la potencia de la luz sin generar cambios bruscos. La implementación modular permite operar tanto en modo manual como automático, asegurando flexibilidad para futuras modificaciones.

El desempeño del controlador fue evaluado experimentalmente mediante pruebas con cuatro valores de setpoint ($33°C$, $30°C$, $28°C$ y $25°C$), donde se analizaron parámetros clave como el sobrepaso, el error máximo porcentual y el tiempo de estabilización. A continuación, se presentan los resultados:

Para un setpoint de $33°C$, los resultados fueron:

- **Sobrepaso:** $1,57°C$.
- **Error máximo porcentual:** $4,76\,\%$.
- **Tiempo de estabilización:** $252,256$ segundos.
- **Temperatura media en estado estacionario:** $33,129°C$.
- **Variacion maxima en estado estacionario:** $+/-1{,}57°C$.
- **Error porcentual en estado estacionario:** $1,08\,\%$.

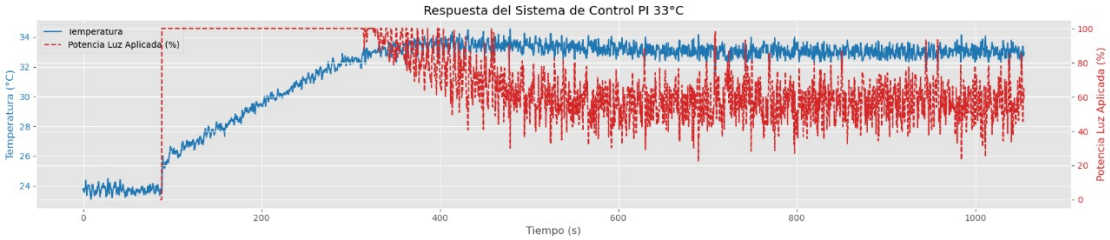

Figura 20: Control de Temperatura con PI en Arduino para un setpoint de $33°C$.

Para un setpoint de $30°C$, los resultados fueron:

- **Sobrepaso:** $0{,}91°C$.
- **Error máximo porcentual:** $3{,}03\,\%$.
- **Tiempo de estabilización:** $106,194$ segundos.
- **Temperatura media en estado estacionario:** $29{,}996°C$.
- **Variacion maxima en estado estacionario:** $+/-0{,}9°C$.
- **Error porcentual en estado estacionario:** $0,39\,\%$.

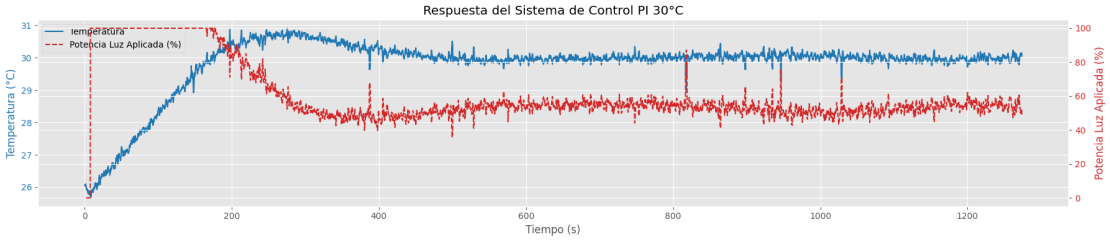

Figura 21: Control de Temperatura con PI en Arduino para un setpoint de $30°C$.

Para un setpoint de $28°C$, los resultados fueron:

- **Sobrepaso:** $2,52°C$.
- **Error máximo porcentual:** $9{,}00\,\%$.
- **Tiempo de estabilización:** $199,458$ segundos.
- **Temperatura media en estado estacionario:** $28,075°C$.
- **Variacion maxima en estado estacionario:** $+/-1{,}5°C$.
- **Error porcentual en estado estacionario:** $1,39\,\%$.

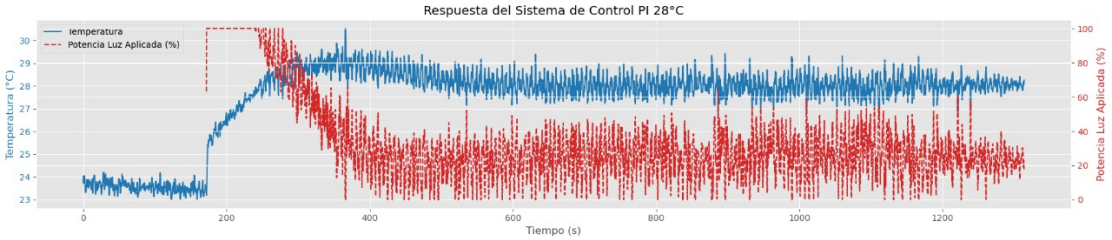

Figura 22: Control de Temperatura con PI en Arduino para un setpoint de 28°C.

Para un setpoint de 25°C, los resultados fueron:

- **Sobrepaso:** 1,75°C.
- **Error máximo porcentual:** 7,00 %.
- **Tiempo de estabilización:** 70, 543 segundos.
- **Temperatura media en estado estacionario:** 24,996°C.
- **Variacion maxima en estado estacionario:** $+/-$ 0,76°C.
- **Error porcentual en estado estacionario:** 0, 50 %.

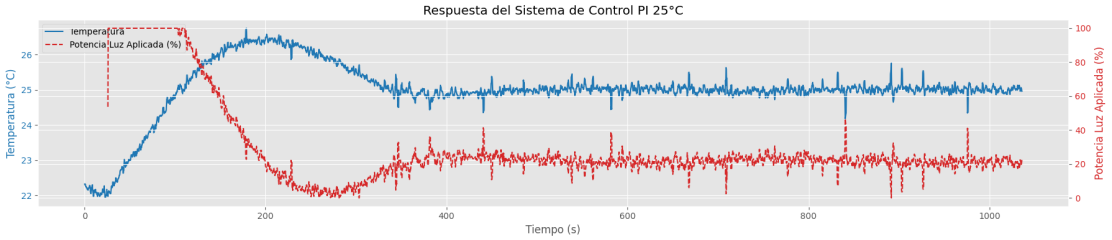

Figura 23: Control de Temperatura con PI en Arduino para un setpoint de 25°C.

Los resultados muestran que, para todos los setpoints evaluados, el sistema logra alcanzar la temperatura deseada en un tiempo aproximado de 400 segundos (6,6 minutos). En el caso del setpoint de 33°C, el sistema presenta un sobrepaso moderado (1,57°C) y una estabilización con oscilaciones menores, lo que indica una respuesta precisa pero con una dinámica que podría beneficiarse de una mayor suavidad. Para 30°C, el sistema exhibe un comportamiento más eficiente, con un tiempo de estabilización reducido (106, 194 segundos) y oscilaciones iniciales mínimas, manteniendo la temperatura deseada de manera consistente con un error en estado estacionario insignificante.

En los setpoints de 28°C y 25°C, se observa un aumento en el sobrepaso (2,52°C y 1,75°C, respectivamente) y una mayor oscilación en el caso del setpoint de 28°C. Sin embargo, el sistema logra estabilizarse en ambos casos con un error porcentual aceptable. Estos resultados reflejan que, aunque el diseño del controlador PI muestra buen

desempeño general, podría beneficiarse de ajustes adicionales para reducir las oscilaciones y mejorar la respuesta dinámica en setpoints más bajos. En general, la selección de los parámetros del controlador ($K_p = 30{,}0$, $K_i = 1{,}0$, $K_d = 0{,}0$) permite mantener la temperatura del ambiente controlado dentro del rango deseado, validando la efectividad del diseño implementado para la gestión térmica con estabilidad y confiabilidad.

9. Actualización del Modelo y Diagrama de Bloques

En esta sección se presenta el modelo actualizado del sistema de control de temperatura, que incluye un controlador proporcional-integral (PI) en lazo cerrado. Esta modificación tiene como objetivo mejorar la estabilidad y precisión del sistema, permitiendo un control eficiente de la temperatura en el rango deseado.

9.1. Modelo Compensado

El sistema térmico inicial se representó mediante la siguiente función de transferencia:

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

que modela la dinámica del sistema térmico con una ganancia estática de $0{,}1255\,^{\circ}\mathrm{C}/\,\%$ y una constante de tiempo de 243,91 s. Para compensar el comportamiento del sistema y mejorar su desempeño, se añadió un controlador PI descrito por la función de transferencia:

$$C_{PI}(s) = K_p + \frac{K_i}{s},$$

donde K_p y K_i son las constantes proporcional e integral, respectivamente. Los valores optimizados de estas constantes se determinaron para garantizar un tiempo de estabilización corto y un error estacionario mínimo.

9.2. Diagrama de Bloques del Sistema Compensado

La Figura 24 muestra el diagrama de bloques del sistema compensado. Este diagrama incluye los siguientes elementos principales:

- **Entrada de referencia ($P(s)$):** Representa la potencia deseada, que se ajusta para alcanzar la temperatura objetivo.
- **Suma de errores:** Calcula el error $E(s)$ como la diferencia entre la referencia ($P(s)$) y la salida medida ($T(s)$).
- **Controlador PI ($C_{PI}(s)$):** Procesa el error para generar la señal de control que ajusta la potencia aplicada al sistema térmico.

- **Planta térmica:** Representada por $G(s) = \frac{0{,}1255}{243{,}91s+1}$, modela el comportamiento dinámico del sistema.
- **Sensor NTC:** Mide la temperatura actual ($T(s)$) y proporciona retroalimentación al sistema.

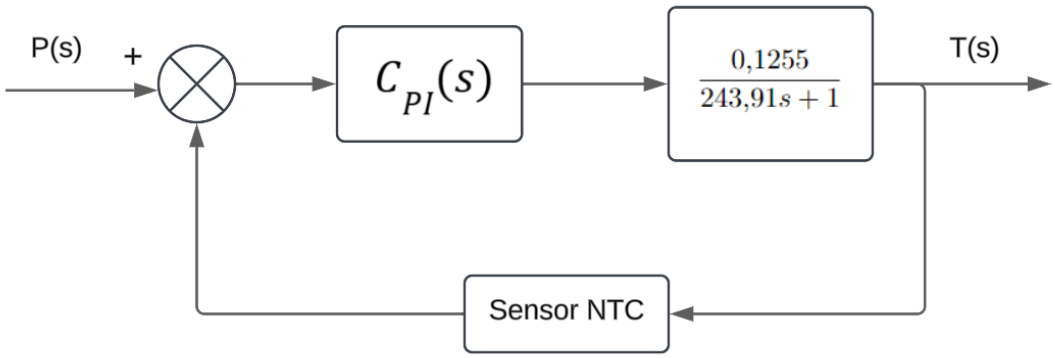

Figura 24: Diagrama de bloques del sistema compensado con controlador PI.

9.3. Funcionamiento del Sistema

El sistema opera de la siguiente manera:

1. La referencia $P(s)$ establece la potencia requerida para alcanzar el setpoint de temperatura.
2. El error $E(s) = P(s) - T(s)$ es procesado por el controlador PI, que genera una señal de control basada en los términos proporcional (K_p) e integral (K_i).
3. La señal de control modula la potencia del calefactor, que se modela mediante la planta térmica $G(s)$.
4. La salida del sistema ($T(s)$) es medida por el sensor NTC, cerrando el ciclo de control.

10. Sistema de Alarmas

El sistema de alarmas implementado utiliza indicadores luminosos y auditivos para representar el estado del prototipo, según se detalla a continuación:

- **Luz verde:** Indica que el sistema opera en condiciones normales. La temperatura está dentro del rango de control y todos los componentes funcionan correctamente.

- **Luz amarilla:** Indica una alerta moderada. La temperatura está fuera del rango (Setpoint ± 1°C).
- **Luz roja y alarma sonora:** Indica una falla crítica en el sistema. Las condiciones que activan esta alarma son:
  - Temperatura excesivamente fuera del rango (`Setpoint` ± 3°C).
  - Ventilador detenido (RPM = 0).
  - Calefactor desconectado.

10.1. Componentes a implementar en el sistema de alarmas

Se implementaron al circuito principal los siguientes componentes:

Cuadro 6: Materiales para el sistema de alarma

Material	**Cantidad**
LED rojo	1
LED verde	1
LED amarillo	1
Transistor 2N2222	1
Resistencia 3,3 kΩ	1
Buzzer 12 V	1
Resistencia de 330Ω	3

Se identificó que el buzzer generaba un sonido de baja intensidad debido a una corriente insuficiente en su terminal. Para solucionar este problema, se utilizó un transistor 2N2222 configurado como amplificador de corriente.

Cálculo de la resistencia base para el diseño del circuito con el transistor 2N2222

El transistor 2N2222 es un transistor NPN de propósito general. Tiene una ganancia de corriente (h_{FE}) típica de 100, aunque en esta situación utilizaremos un valor de 50, debido a que necesita una rápida acuación entre los estados de conducción y corte.

Determinar la corriente de colector

La corriente de colector (I_C) es la suma de la corriente del LED (I_{LED}) y la corriente necesaria para el buzzer (I_{BUZZER}).

El valor de I_{LED} se eligió como 20 mA, basado en una suposición típica para LEDs comunes, que suelen operar entre 10 mA y 20 mA para un brillo adecuado.

$$I_{\text{LED}} = 20\,\text{mA} = 0{,}02\,\text{A}$$

Para el buzzer, si es de 12V y 0.5W, entonces:

$$I_{\text{BUZZER}} = \frac{\text{Potencia}}{\text{Voltaje}} = \frac{0{,}5}{12}\,\text{A} = 0{,}042\,\text{A}.$$

Entonces:

$$I_C = I_{\text{LED}} + I_{\text{BUZZER}} = 0{,}02 + 0{,}042 = 0{,}062\,\text{A}.$$

Determinar la corriente de base

El transistor necesita una corriente de base (I_B) suficiente para saturarlo. Usamos la fórmula:

$$I_B = \frac{I_C}{\beta_{\text{sat}}}$$

Con $\beta_{\text{sat}} = 50$:

$$I_B = \frac{0{,}062}{50} = 0{,}00124\,\text{A} = 1{,}24\,\text{mA}.$$

Calcular la resistencia de base

La resistencia de base (R_B) limita la corriente suministrada por el Arduino al transistor. Usamos la Ley de Ohm considerando la tensión de salida del Arduino ($V_{\text{OUT}} \approx 5\,\text{V}$) y la caída de tensión entre la base y el emisor ($V_{\text{BE}} \approx 0{,}7\,\text{V}$):

$$R_B = \frac{V_{\text{OUT}} - V_{\text{BE}}}{I_B}.$$

Sustituyendo los valores:

$$R_B = \frac{5 - 0{,}7}{0{,}00124} = \frac{4{,}3}{0{,}00124} \approx 3467\,\Omega.$$

Usamos una resistencia comercial estándar de $3{,}3\,\text{k}\Omega$.

11. Plataforma Informática

La plataforma informática desarrollada tiene como objetivo registrar, almacenar y visualizar los datos clave del prototipo de la incubadora en tiempo real. Este sistema permite el monitoreo constante de la temperatura interna, la potencia de radiación y la velocidad de giro del ventilador, con registros realizados automáticamente cada minuto. A continuación, se describen los principales componentes y características de la implementación.

11.1. Diseño de la Base de Datos

Se diseñó una base de datos en *phpMyAdmin* para almacenar los datos generados por el sistema. La tabla principal, denominada `estado_incubadora`, contiene la siguiente estructura:

Columna	Tipo de dato	Descripción
id	INT AUTO_INCREMENT PRIMARY KEY	Identificador único de cada registro.
timestamp	DATETIME	Fecha y hora del registro.
temperatura	FLOAT	Temperatura interna del sistema (°C).
potencia_radiacion	FLOAT	Potencia de radiación calculada (W).
velocidad_ventilador	FLOAT	Velocidad del ventilador (RPM).
estado_alarma	VARCHAR(50)	Estado del sistema (verde, amarillo o rojo).

Cuadro 7: Estructura de la base de datos para registrar las mediciones del prototipo.

11.2. Funcionamiento del Sistema

Los datos son enviados desde el sistema basado en Arduino a través de una conexión serial al servidor de base de datos. Un software desarrollado en Python (ver Anexo B) procesa estos datos y realiza el almacenamiento automático cada minuto.

La Figura 25 muestra un ejemplo de los datos registrados en la base de datos, donde se incluyen mediciones de temperatura, velocidad, potencia y el estado del sistema.

		id	temperatura	velocidad	potencia	estado_alarma	timestamp
☐	Editar Copiar Borrar	14	23.73	0	12.6122	AMARILLO	2024-12-12 14:43:12
☐	Editar Copiar Borrar	15	24.91	40980	8.56332	VERDE	2024-12-12 14:44:12
☐	Editar Copiar Borrar	16	24.57	30300	0	ROJO	2024-12-12 14:46:28
☐	Editar Copiar Borrar	17	25.52	0	10.4257	AMARILLO	2024-12-12 14:50:08
☐	Editar Copiar Borrar	18	26.12	40950	11.2303	ROJO	2024-12-12 14:51:08
☐	Editar Copiar Borrar	19	25.84	0	12.4315	AMARILLO	2024-12-12 14:54:54
☐	Editar Copiar Borrar	20	26.68	30	10.4406	ROJO	2024-12-12 14:55:54

Figura 25: Vista de los datos registrados en la base de datos.

11.3. Estados del Sistema y Alarmas

El campo `estado_alarma` en la base de datos indica la condición del sistema de acuerdo con los siguientes criterios:

- **Verde:** El sistema opera dentro del rango normal, con la temperatura dentro de ±1°C del setpoint.
- **Amarillo:** La temperatura está ligeramente fuera del rango normal (±1°C), pero no en un estado crítico.
- **Rojo:** Se detecta una condición crítica, como una temperatura fuera de ±3°C del rango deseado, el ventilador detenido o el calefactor desconectado.

La plataforma informática desarrollada permite un monitoreo eficiente del prototipo, registrando los datos relevantes en tiempo real y almacenándolos de manera estructurada en una base de datos. Además, la inclusión de un sistema de alarmas asegura la detección y notificación oportuna de condiciones anómalas, contribuyendo a la estabilidad y seguridad del sistema.

12. Anexos Códigos

A. Código Arduino

B. Código Intefaz en Python

C. Código en QtDesign

D. Código del Visualizador de Archivos CSV