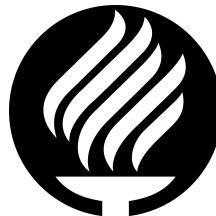


**INSTITUTO TECNOLÓGICO Y DE
ESTUDIOS SUPERIORES DE MONTERREY**
CAMPUS MONTERREY



**Caracterización mediante instrumentación
electrónica**

Reporte Final Reto

**Sistema de control de temperatura en aires
acondicionados**

Nombre	Matrícula
---------------	------------------

Isaac Alexis López Paredes	A01747148
Arturo Hiram Benavides Figueroa	A01177863
Alejandro Hernández Gómez	A01742267
Adriel Jael Santamaría Hernández	A01753918

Profesores:

Valery Moreno
Ricardo Sotelo
Alonso Sosa
Salvador Rodríguez
José Manuel Rodríguez

Fecha

5/05/2024

1 Introducción

Uno de los requerimientos más importantes en una gran variedad de procesos industriales es el control de temperatura. Desde sistemas alimenticios hasta la construcción de microprocesadores, el continuo monitoreo y ajuste de temperatura se ha vuelto de gran importancia. Además es importante incluso en aplicaciones directas hacia el consumidor, como el confort térmico deseado en una sala de espera. Por este motivo es de gran importancia la medición de temperatura, así como tener métodos automatizados capaces de activar un sistema ya, sea de enfriamiento o de calefacción, que asegure la constante temperatura deseada en un espacio.

En el presente trabajo se presentó la problemática de un establecimiento de la cadena *Citi Cinema* el cual cuenta con una serie de salas de cine. Con el objetivo de reducir el consumo energético y seguir asegurando el confort de los clientes se requiere implementar un sistema de medición de temperatura en las salas individuales, acompañado de una estrategia de control tipo ON-OFF con la cual se pueda regular el encendido y apagado del enfriamiento de las salas.

Para el desarrollo del presente trabajo se realizará como propuesta el diseño y prototipado de un sistema electrónico que incluya el monitoreo de temperatura y sistema de control ya mencionado. Adicionalmente se enviarán las medidas tomadas a la plataforma *Thingspeak*, con el objetivo de poder visualizar la información en internet en tiempo real y poder hacer un futuro análisis de la eficiencia energética, así como posibles mejoras al sistema.

Para llevar a cabo el desarrollo de este trabajo se requerirán varias consideraciones, siendo algunas la elección de instrumentación correcta, la medición de variables, el acondicionamiento de señales, la programación e implementación del sistema de control y el registro de datos en la nube. Para lograr los objetivos propuestos se hizo una división de tareas entre los miembros del equipo. En la tabla 1 se muestran estas tareas.

Integrante	Tareas
Adriel Santamaría	Diseño y construcción del prototipo; sistema de setpoint
Isaac López	Diseño del circuito; programación
Arturo Benavides	Sistema de visualización; armado del circuito
Alejandro Hernández	Envío de datos a la nube; caracterización de sensores

Table 1: Tabla de responsabilidades

2 Esquema

Para dar solución a la problemática planteada se propuso un diseño experimental que consiste en cuatro sensores de temperatura ambiente y dos para el sistema de refrigeración, siendo uno para medir la temperatura de entrada y otro de salida. Además, con el uso de un relevador se controla la activación del sistema de refrigeración, a partir de la temperatura promedio medida por los sensores.

También se diseñó un sistema para modificar el *set-point* o temperatura ideal con un botón y un potenciómetro, así como una forma de visualizar los resultados con una LCD. Para el procesamiento de datos se propone utilizar un Arduino UNO junto con una tarjeta ESP32, la cual se utilizará para enviar datos a la nube. En la figura 1 se muestra el diseño del circuito para solucionar la problemática. Se muestran de izquierda a derecha y de arriba hacia abajo:

- los 4 sensores DHT-22;
- el Arduino UNO;
- el botón y potenciómetro para el establecimiento del set-point de temperatura;
- el circuito de división de voltaje para medir la resistencia en los dos termistores;
- LED's para visualizar el funcionamiento;
- el relevador con un diodo, un transistor y una resistencia;
- y el ESP-32.

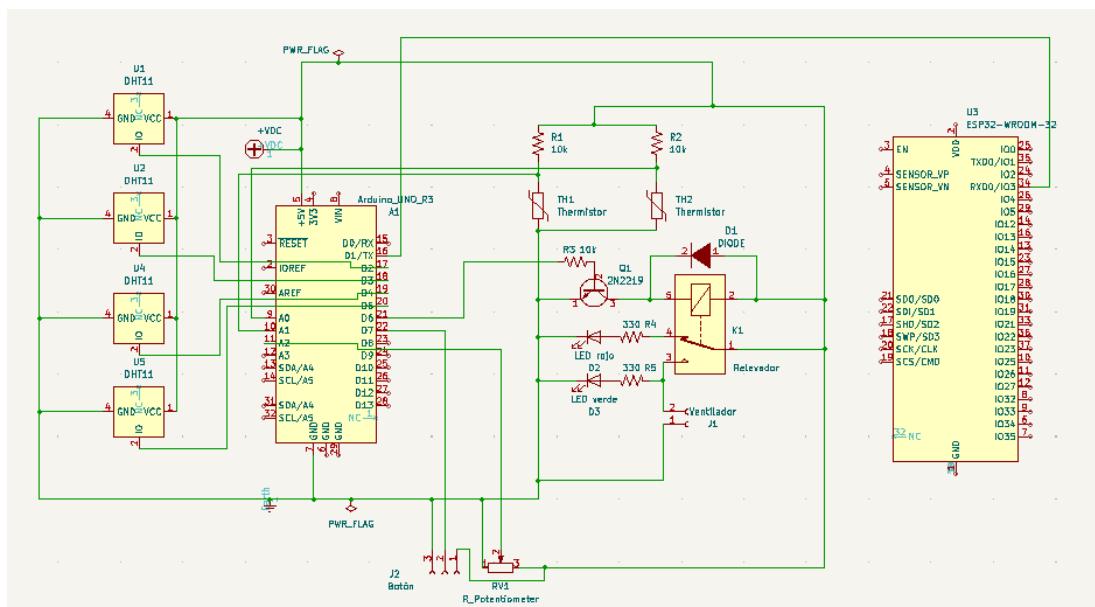


Figure 1: Esquema del circuito

Además, se diseñó la PCB del circuito. Esta se muestra en la figura 2.

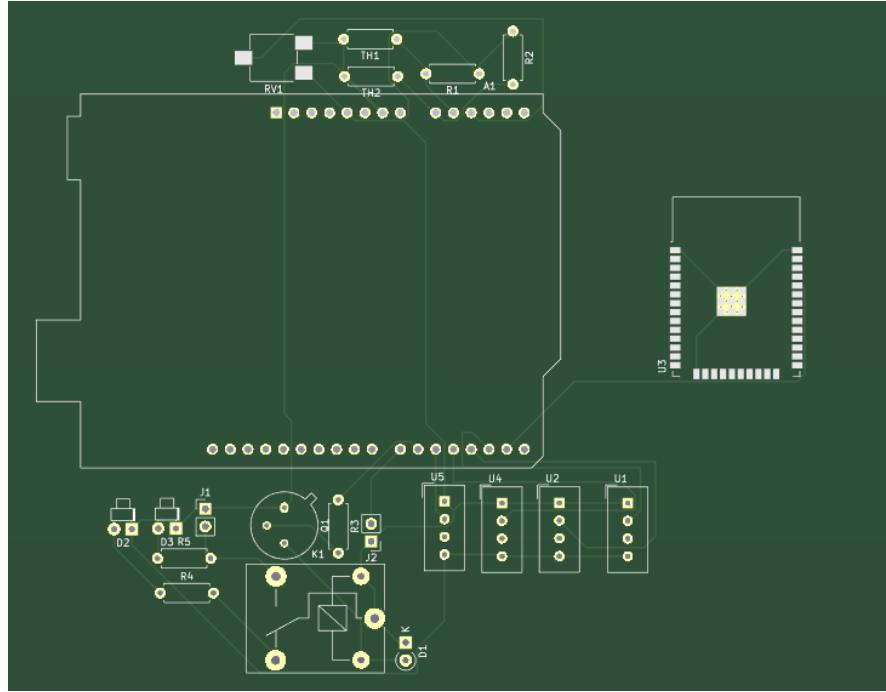


Figure 2: PCB diseñada

3 Hardware

3.1 Sensores y actuadores

Para medir la temperatura de la sala se utilizaron en total 6 sensores, 4 sensores DHT22 para medir la temperatura de ambiente y 2 termistores NTC de 10k para medir la temperatura de entrada y salida del aire acondicionado. El DHT22 es un sensor digital de temperatura y humedad relativa de buen rendimiento y bajo costo. Integra un sensor capacitivo de humedad y un termistor para medir el aire circundante, y envía los datos mediante una señal digital directamente al pin de datos del microcontrolador.

Por otro lado, un termistor es un resistor que cambia de valor con la temperatura. En particular, el termistor NTC tiene un coeficiente de temperatura negativo, es decir, la resistencia disminuye al aumentar la temperatura. El valor de 10k indica que a 25°C este tiene un valor de 10 kΩ. Este sensor es analógico. Para medir la temperatura se debe medir primero la resistencia y para ello se debe implementar un circuito de división de voltaje. Este se muestra en la figura 3. Consiste en dos resistencias en serie, y entre ellas se obtiene una señal analógica de voltaje que recibe el microcontrolador. Después de convertirlo a digital este valor se puede utilizar para calcular la resistencia como:

$$R_t = \frac{R_c \cdot V_t}{V_{in} - V_t}, \quad (1)$$

donde R_t es la resistencia del termistor, R_c es una resistencia conocida, que en este caso fue de 10 kΩ, V_{in} es el voltaje de entrada, en este caso 5V, y V_t es el voltaje que recibe el microcontrolador entre ambos resistores.

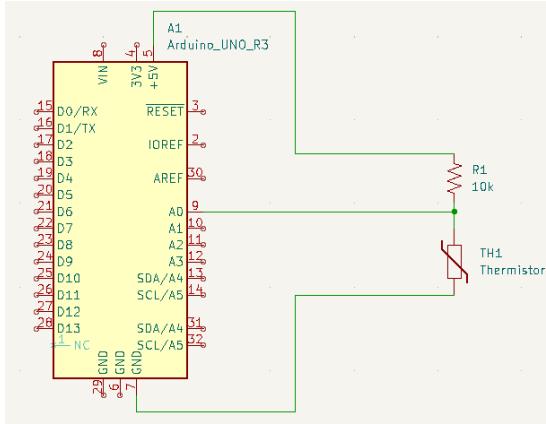


Figure 3: Circuito de división de voltaje

En cuanto al sistema de control de temperatura, este se implementó con un módulo de relevador de 5V para Arduino. Este es un dispositivo que funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de un contacto que permite abrir o cerrar otros circuitos eléctricos independientes. La ventaja de utilizar el módulo es que este se conecta directamente al microcontrolador.

3.2 Elementos restantes

3.2.1 Microcontroladores

Para realizar el procesamiento de datos se utilizó principalmente una tarjeta Arduino UNO, cuyas conexiones se muestran en la figura 1. En este caso se utilizaron los pines digitales 3, 4, 5 y 8 como entradas para medir los valores de temperatura de los cuatro sensores DTH22. Por otro lado, se utilizaron los pines analógicos A0 y A1 para medir el voltaje de los divisores de corriente utilizados para conocer la temperatura medida por ambos termistores, como descrito en la sección anterior. Para el sistema de establecimiento de Setpoint se utilizó el pin A2 como entrada del potenciómetro y los pines 2 y 13 para el botón de control. Como salidas se utilizaron el pin 6 para controlar el relevador y el pin 7 para controlar un LED adicional de encendido. La comunicación con la LCD se manejó a través del Serial, utilizando los pines A4 como SDA y A5 como SCL. Finalmente se utilizaron los pines tx y rx en el Arduino, además de rx2 y tx2 en la ESP32 para establecer de manera similar una comunicación serial con la tarjeta ESP32. El código utilizado para la lectura y procesamiento de datos se describirá en la sección de Software.

Adicional al Arduino UNO se utilizó una tarjeta ESP32. El motivo de esto fue que para cumplir con los objetivos se requería de un módulo de internet, por lo que se decidió utilizar esta tarjeta. Una vez procesados los datos por el primer microcontrolador, estos se mandan a los pines rx2 y tx2 que funcionan como entradas de la ESP32. Esta tarjeta establece una conexión a internet y sube los datos de temperatura ambiente, temperatura de entrada y temperatura de salida a un canal de *Thinkspeak* donde se pueden visualizar a tiempo real. El código utilizado para esta función se explicará más a detalle en la sección de Software.

3.2.2 Display LCD

Se decidió que además de mandar los datos a la nube era conveniente poder visualizarlos en el mismo sistema. Por este motivo se decidió utilizar un display tipo LCD 16x2. Para simplificar el uso del display, así como reducir la cantidad de pines y cables necesarios, se utilizó un módulo de comunicación I2C adicional al display. Este permitió que la comunicación fuera serial, utilizando el protocolo ya mencionado, el cual requiere de un canal de comunicación para datos SDA y un canal para la señal de reloj SCL. Se utilizaron los pines A4 y A5 respectivamente para estas funciones. Para desplegar los datos se hizo un ciclo e el cual se despliegan la temperatura de entrada, seguida por la de salida y finalmente la temperatura ambiente, la cual se muestra por más tiempo.

3.2.3 Componentes eléctricos varios

Adicional a los microcontroladores, el display y los sensores utilizados, también se hizo uso de los componentes mostrados en el esquema 1. Esto incluye LEDs y resistencias, los cuales fueron incluidos de acuerdo a las necesidades del prototipo. Los diodos y transistor que acompañan al relevador ya venían incluidos en el módulo de Arduino. Para ensamblar el prototipo, a pesar de que se diseñó una tarjeta PCB, se utilizó una protoboard, haciendo las conexiones necesarias utilizando *jumpers*.

4 Software

4.1 Arduino

En esta sección se describe el código implementado para medir los datos de temperatura de los sensores, implementar el sistema de control y la visualización de los datos en la LCD. EL código completo se puede consultar en el anexo A.1.

Se comenzó incluyendo las bibliotecas necesarias para la manipulación de componentes electrónicos, como el LCD, utilizando el protocolo de comunicación serial I2C, junto con la biblioteca para los sensores de temperatura y humedad. A continuación, se define el tipo de sensores que se están utilizando y los pines a los que se enviará la información. Además, se definen variables globales fuera de las funciones para procesos como el control de un botón para ajustar el "setpoint" de la temperatura deseada.

En la función `setup()`, se inicializan la pantalla LCD, los sensores, el puerto serial y se configuran los pines de entrada/salida del Arduino necesarios para el circuito. También se define uno de los pines como una interrupción, que se utilizará para cambiar la variable global que ajusta el "setpoint" cada vez que se presiona un botón.

Dentro de la función `loop()`, primero se verifica si el botón ha sido presionado (mediante la variable global que cambia con la interrupción). Si se ha presionado, se lee constantemente el valor de un potenciómetro conectado a un pin analógico para ajustar el "setpoint" de temperatura, mientras que esta información se muestra en el LCD. Una vez que el botón se presiona nuevamente, el programa sale

de esta condición y procede a leer las temperaturas de los sensores instalados.

Para el sensor DHT22, se utiliza su biblioteca para obtener la temperatura de cuatro sensores y se muestra el promedio en el LCD. En cuanto a los termistores, se calcula la temperatura mapeando el valor analógico del pin al cual están conectados, determinando el voltaje y a partir de la ecuación de un divisor de voltaje se calcula la resistencia y posteriormente la Beta (cuyo cálculo se explicará más adelante en el documento). Finalmente, se calcula la temperatura del aire acondicionado dependiendo del termistor utilizado (entrada o salida).

Después de calcular las temperaturas, se activa o desactiva el aire acondicionado mediante un relé. La función del relé determina si la temperatura ambiental está dentro del rango aceptable (± 2 grados del "setpoint"). Si está fuera de este rango, se enciende o apaga el aire acondicionado según sea necesario.

Finalmente, se imprimen los resultados de las temperaturas obtenidas tanto en el puerto serial para procesarlo a thingspeak, como en el LCD para visualización y registro.

4.2 Implementación del Envío de Datos a ThingSpeak con ESP32

El código implementa una funcionalidad para que un módulo ESP32 se conecte a una red WiFi y envíe datos a la plataforma ThingSpeak. Estos datos se recogen desde un puerto serial, procesan y se envían a través de Internet.

4.2.1 Inclusión de Bibliotecas

Se utilizan dos bibliotecas principales para facilitar la conexión WiFi y el envío de datos por HTTP:

```
1 #include <WiFi.h>
2 #include <HTTPClient.h>
```

4.2.2 Variables de Configuración

Se configuran las credenciales para la conexión a la red WiFi y la URL para la API de ThingSpeak:

```
1 const char* ssid = "iPhone de Alex";
2 const char* password = "C188PD729";
3 const char* serverName = "http://api.thingspeak.com/update?api_key=
ZW4DBKXBBO53LX9S";
```

4.2.3 Función setup()

Esta función inicializa los puertos seriales y establece la conexión WiFi:

```
1 void setup() {
2     Serial.begin(9600);
3     Serial2.begin(9600, SERIAL_8N1, 16, 17);
4     WiFi.begin(ssid, password);
5     while (WiFi.status() != WL_CONNECTED) {
6         delay(500);
```

```

7     Serial.print(".");
8 }
9 Serial.println("Conectado a WiFi!");
10}

```

4.2.4 Función loop()

En el bucle principal, se verifica la conexión WiFi, se leen los datos del puerto serial, se limpian, validan y envían a ThingSpeak:

```

1 void loop() {
2   if (WiFi.status() == WL_CONNECTED) {
3     if (Serial2.available() > 0) {
4       String value1 = cleanInput(Serial2.readStringUntil('\n'));
5       ...
6       HttpClient http;
7       http.begin(serverName);
8       String httpRequestData = "field1=" + value1 + "&field2=" +
9       value2 + "&field3=" + value3;
10      int httpResponseCode = http.POST(httpRequestData);
11      ...
12      http.end();
13    } else {
14      Serial.println("No hay datos disponibles en el puerto serial.");
15    }
16  } else {
17    Serial.println("Error en la conexión WiFi");
18    WiFi.reconnect();
19  }
20  delay(20000);
}

```

4.2.5 Funciones Auxiliares

Las funciones `cleanInput()` y `validateNumber()` ayudan a preparar los datos para el envío, asegurando que sean números decimales válidos:

```

1 String cleanInput(String data) {
2   String result = "";
3   for (int i = 0; i < data.length(); i++) {
4     char c = data[i];
5     if (isdigit(c) || c == '.') {
6       result += c;
7     }
8   }
9   return result;
10}
11
12 String validateNumber(String num) {
13   if (num.indexOf('.') == -1) {
14     num += ".0";
15   }
16   return num;
17}

```

Este desglose explica cómo el código configura la conexión del ESP32 a una red WiFi, procesa datos seriales y los envía a una plataforma de IoT, proporcionando una base para aplicaciones de monitoreo remoto como en este caso el monitoreo de temperaturas.

5 Caracterización

En esta sección se describe el proceso para caracterizar los termistores con los que se contaban y así encontrar una relación que permitiera obtener la temperatura a partir de la resistencia medida con el divisor de voltaje.

5.1 Modelos para la temperatura de un termistor NTC

Se probaron varios modelos de caracterización para el sensor, de manera que se buscaba la mayor simplicidad y al mismo tiempo la mayor precisión. Los tres modelos que se probaron fueron un modelo lineal, un modelo de Steinhart-Hart completo (con parámetros A, B y C) y un modelo de Steinhart-Hart reducido (solo parámetro Beta).

El primero que se probó fue el de Steinhart-Hart. Está dado por:

$$\frac{1}{T} = A + B \ln(R) + C(\ln(R))^3 \quad (2)$$

donde T es la temperatura en Kelvin, R es la resistencia del termistor, y A , B , y C son coeficientes determinados experimentalmente. Para calcular estos parámetros, se necesitan al menos tres mediciones de temperatura y resistencia que deben cubrir el rango de temperaturas donde se utilizará el termistor. Se forma un sistema de ecuaciones utilizando las mediciones seleccionadas. Por cada par de temperatura y resistencia (T_i, R_i) , se substituye en la ecuación para formar:

$$\frac{1}{T_i} = A + B \ln(R_i) + C(\ln(R_i))^3$$

Este sistema de ecuaciones no lineales se resolvió usando métodos numéricos para encontrar los valores de A , B , y C que mejor se ajusten a los datos experimentales. Específicamente se utilizó la función *curvefit* de la librería *scipy* en python. Con ello se obtuvo el resultado que se muestra en la figura 4 y el código utilizado en el anexo A.2.

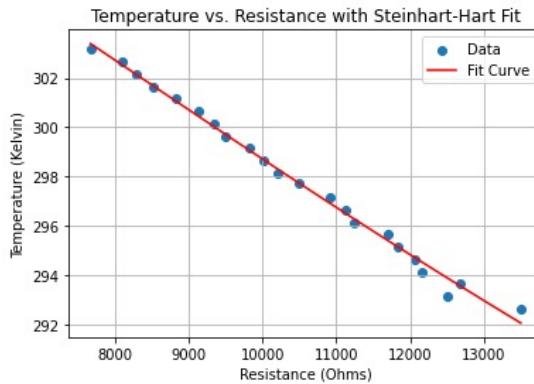


Figure 4: Caracterización usando modelo de Steinhart-Hart completo

Se puede observar que la curva se ajusta bastante bien a los datos experimentales, sin embargo, al momento de validarla en el circuito para medir la temperatura se obtenían valores que no eran congruentes con un medidor conocido, por lo que se optó por probar el modelo reducido. Este está dado por:

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{\beta} \ln \left(\frac{R_T}{R_0} \right) \quad (3)$$

Esta ecuación proviene del despejar la temperatura de la siguiente relación para la resistencia de un termistor:

$$R_T = R_0 e^{\beta \left(\frac{1}{T} - \frac{1}{T_0} \right)} \quad (4)$$

donde:

- R_T : Resistencia del termistor a la temperatura T . Es la variable que se desea calcular o medir.
- R_0 : Resistencia del termistor a una temperatura de referencia T_0 . Esta es una constante para un termistor dado y se determina experimentalmente.
- β : Es una constante del material del termistor, también determinada experimentalmente. Este valor describe la sensibilidad del termistor al cambio de temperatura.
- T : Temperatura actual del termistor en Kelvin.
- T_0 : Temperatura de referencia en Kelvin, a la cual se conoce la resistencia R_0 .

La constante β se calcula como:

$$\beta = \frac{\ln \left(\frac{R_{T2}}{R_{T1}} \right)}{\left(\frac{1}{T_2} - \frac{1}{T_1} \right)}, \quad (5)$$

donde R_{T1} y R_{T2} son las resistencias del termistor a dos temperaturas conocidas T_1 y T_2 , respectivamente. Para validar este modelo se escribió el código en python encontrado en el anexo A.3, probando varias combinaciones de T_1 y T_2 y encontrando la β que minimizara el error. Los resultados se muestran en la figura 5.

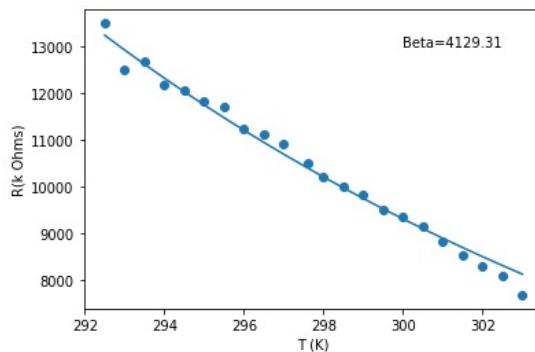


Figure 5: Caracterización usando modelo Steinhart-Hart reducido (modelo Beta)

Este modelo también se ajusta bien y la β obtenida es parecida a datos oficiales de datasheets de otros termistores NTC. Sin embargo tampoco se logró obtener valores de temperatura apropiados al momento de aplicarla en el código.

Por último, se probó una regresión lineal, la cual resultó tener un buen coeficiente R^2 (figura 6). Sin embargo tampoco se obtuvieron datos congruentes en las pruebas de software y al no estar físicamente fundamentado no fue elegida.

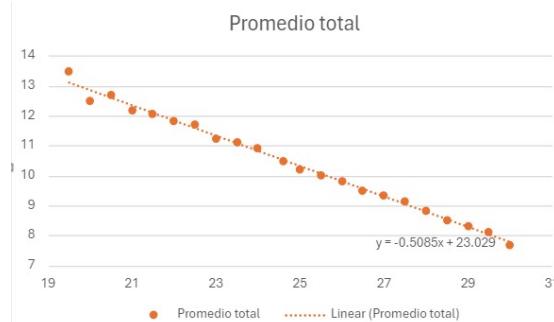


Figure 6: Caracterización usando modelo Lineal

Después de hacer pruebas con estos tres modelos se encontró una relación variable entre el parámetro Beta y la resistencia a partir de los datos experimentales, por lo tanto se optó por el uso del modelo de Steinhart reducido usando una regresión cuadrática de Beta.

5.2 Experimentación y cálculos

Como se mencionó anteriormente, se observó que, en teoría, Beta podría considerarse como un valor constante, pero en la práctica experimental del termistor, su comportamiento se asemejaba más a que la variable Beta tenía un valor dvariable. Por lo tanto, se decidió investigar la relación entre el valor de Beta y la resistencia, que es el valor que se obtendrá al conectar el termistor al circuito y procesarlo a través del Arduino.

Inicialmente, se utilizó la ecuación del modelo de regresión lineal obtenida a partir de la caracterización (la cual relaciona la resistencia y la temperatura) para calcular las resistencias en un rango de temperatura discretizado con incrementos de 0.5 °K. La razón por la que no se utilizaron directamente los datos experimentales de resistencia fue por la falta de discretización y la presencia de valores atípicos que resultaban en una variación excesiva en el valor de Beta. Luego, utilizando las resistencias calculadas mediante regresión lineal en un rango de temperaturas específico (293 K - 303 K), se determinó Beta para cada par de temperaturas consecutivas con la ecuación 5 previamente establecida.

Como siguiente paso, se graficó la relación entre Beta y la resistencia, como se puede observar en la figura 7 ,confirmando la hipótesis inicial de que Beta varía en función de la resistencia y, por ende, de la temperatura. Finalmente, se obtuvo la siguiente ecuación que describe el comportamiento de Beta en relación con la resistencia, la cual se implementó en el software:

$$\beta = 5 \cdot 10^{-5} \cdot \text{Resistencia}^2 - 1.5354 \cdot \text{Resistencia} + 15007 \quad (6)$$

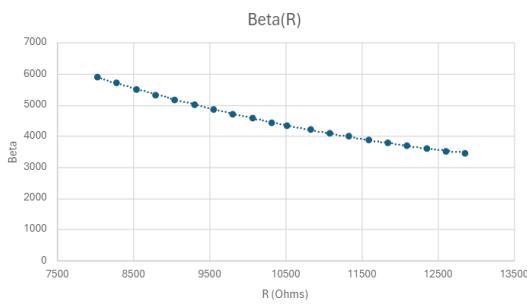


Figure 7: Gráfica de relación entre resistencia y Beta

De esta manera, al ingresar la resistencia medida por el termistor, se calculará el valor correspondiente de Beta y se mejorará la precisión del cálculo de temperatura.

5.3 Análisis estadístico

Una vez que se contaban con los datos de temperatura en función de la resistencia (tabla 3 en anexo) se realizó el análisis de los datos en Excel (los cálculos completos y los datos para la caracterización se pueden encontrar en el siguiente [documento](#)). En primer lugar se calculó el error absoluto y se obtuvo un promedio de 0.97. En cuanto a la desviación estándar esta fue en promedio de 0.74, lo cual indica una muy buena precisión en el método utilizado para encontrar la temperatura, así como en el desempeño del sensor.

Por otro lado, se encontró que el sensor cuenta con cierta histéresis pues se obtuvo una diferencia significativa entre las mediciones en ascenso y descenso. Esto se puede observar en la figura 8. Además, el error de no linealidad calculado fue de 0.8636. Esto valida que el modelo que se está utilizando para interpretar las mediciones de los sensores es adecuado para el rango de operación.

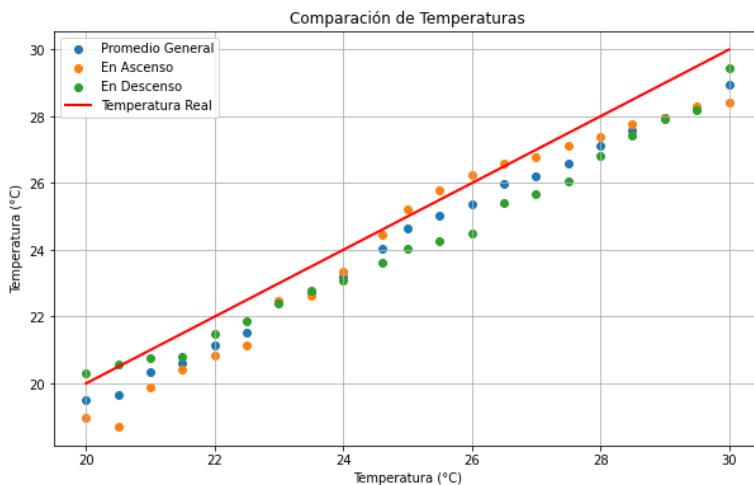


Figure 8: Comparación de temperaturas

6 Resultados

Una vez implementado el circuito mostrado en la figura 1 y los códigos ya descritos, se construyó un prototipo de la propuesta de solución. En la figura 9 se muestra la cara frontal del prototipo.

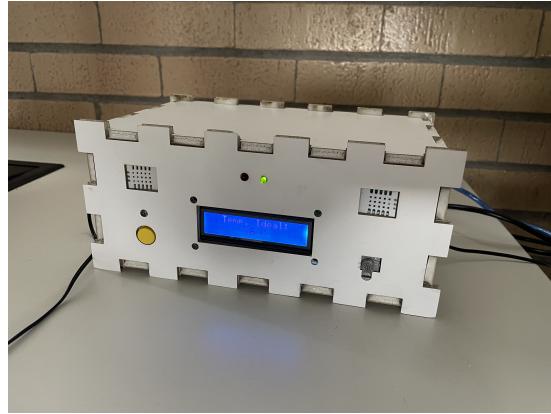


Figure 9: Vista frontal del prototipo final

Como se muestra en el diseño se incluye el display LCD y LEDs de señalamiento. Los dos LEDs superiores indican si el sistema de enfriamiento está encendido, lo cual dependerá del estado del relevador, que a su vez cambia con dependiendo de la comparación entre temperatura medida y temperatura deseada. Para indicar la temperatura deseada, o el Setpoint, se utiliza el potenciómetro y el botón de la izquierda. El modo de operación es el siguiente. Primero se ingresa el Setpoint y se presionó el botón amarillo. Esto encenderá un LED azul el cual indica que el sistema de sensores está activo y, en caso de que sea necesario, se mandará un voltaje a través de los cables mostrados en la figura 10, los cuales están intencionados para activar un compresor.

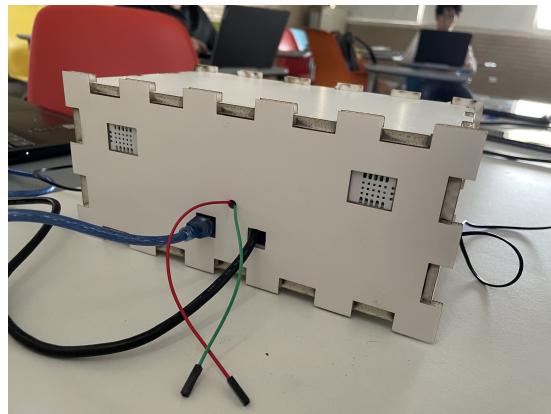


Figure 10: Vista posterior del prototipo final

A continuación se observa el prototipo en los dos diferentes estados de encendido y apagado del relevador dependiendo de la temperatura actual y temperatura ideal elegida, si la temperatura actual sobrepasa la ideal se enciende el relevador para refrigerar, lo contrario sucede si la temperatura actual es menor que la ideal:



(a) Vista del prototipo en estado encendido



(b) Vista del prototipo en estado apagado

Figure 11: Casos de control de temperatura

Finalmente, en la figura 12 se presentan los resultados de medición automática con el dispositivo completamente ensamblado y operacional, se muestran las gráficas de los datos subidos a ThingSpeak.



(a) Temperatura ambiente promedio



(b) Temperatura de entrada



(c) Temperatura de salida

Figure 12: Muestra del envío de datos de temperatura a ThingSpeak

Como se puede observar se logró subir los datos de temperatura registrados a ThingSpeak, en sus respectivos campos de manera que la monitorización de cada una independientemente es facilitada.

7 Conclusión Colectiva

A lo largo del desarrollo de este trabajo se lograron con éxito los objetivos establecidos. Se diseñó e implementó un circuito el cual permitiera leer las medidas de sensores térmicos, tanto digitales como analógicos, para poder hacer un análisis de datos e implementar un sistema de control. A su vez se utilizó el microcontrolador Arduino y el lenguaje C++ para realizar la programación pertinente, tanto del sistema de lectura de sensores, como las salidas y visualización de datos. Finalmente se logró con éxito utilizar la tarjeta ESP32 para enviar información medida a la nube a través de la plataforma *Thingspeak*. Como se demostró el

funcionamiento, tanto del sistema de medición de temperatura ambiente, como del sistema de control, fue el adecuado. Esto se ve en los resultados obtenidos, pues el error y análisis estadístico realizado al comparar con mediciones hechas con otros métodos de medición muestran coherencia y precisión en los datos. Además, el sistema de control funciona como esperado, enviando voltaje a través de los cables de salida cuando la temperatura sube más del límite establecido por el Setpoint y cortando la corriente cuando se encuentra dentro de este. El prototipo presentado permite que el usuario controle el Setpoint desde el mismo dispositivo y además manda los datos de medición en tiempo real a la nube, lo cual permite su visualización y futuro análisis.

El presente trabajo muestra la utilidad de los sistemas eléctricos para realizar mediciones y automatizar procesos, en este caso un sistema de control de temperatura. Es evidente la utilidad de este tipo de sistemas, no solo para controlar la temperatura, sino para en general poder automatizar cualquier tipo de medición que pueda ser utilizada para realizar una acción de acuerdo a los datos recolectados, incluso, utilizando la nube para poder hacer un análisis más profundo. Además de esto, se muestra la importancia de realizar una correcta caracterización, en este caso de sensores de temperatura, pues de esta depende que las mediciones sean adecuadas y no existan faltas mayores causadas por un mal procedimiento experimental. Finalmente, el uso de métodos estadísticos fue de gran ayuda para al final comprobar la precisión y exactitud del sistema de medición, lo cual es importante al momento de buscar implementar un sistema similar en una aplicación industrial o comercial.

8 Conclusiones Individuales

Adriel Santamaria: En este proyecto y curso he aprendido muchos métodos, elementos y componentes que me resultaron bastante aplicables y de valor para mi futuro. Los temas vistos en clase y en especial este reto me dieron una perspectiva más amplia de lo que puedo hacer con mis conocimientos y como trasladar una idea a una solución tangible como lo fue esta caja de medición para la temperatura. Quedé satisfecho con el resultado del trabajo en equipo que realizamos en este proyecto, todos propusimos ideas e integraron un aspecto fuera de las especificaciones que me pareció bastante innovador, el cual fue el botón para ajustar el set point. A modo de cierre, estoy orgulloso de nuestros resultados y de los conocimientos adquiridos y aplicados en este proyecto.

Isaac López: A través del transcurso de esta materia he tenido que enfrentar varios retos, sin embargo, considero que la solución de este reto fue la oportunidad perfecta para superarlos. Pude resolver, junto con el apoyo de mi equipo, problemas que se presentaron en la programación, en el diseño del circuito y en las conexiones. Considero que fue un proyecto muy interesante y completo, pues tuvimos que aplicar los conocimientos teóricos y prácticos que adquirimos en los distintos módulos de la materia. Desde caracterizar el sensor experimental, diseñar y armar el circuito, entender el funcionamiento de componentes como el relevador, fortalecer mis habilidades de programación en un lenguaje como C++ en el que no estaba acostumbrado, todas son habilidades que me serán útiles en el futuro.

Alejandro Hernández: Este proyecto me ha permitido aplicar y expandir mis conocimientos en programación y diseño de sistemas electrónicos de una manera muy concreta y significativa. A través de la implementación del sistema de control de temperatura, he adquirido una comprensión más profunda de cómo los componentes de hardware, como los sensores y los microcontroladores, pueden interactuar eficientemente para lograr objetivos específicos. Además, he aprendido sobre la importancia de la precisión en la programación para el manejo de datos en tiempo real y cómo pequeños errores en el código pueden tener impactos significativos en el rendimiento del sistema. Trabajar en este proyecto también ha reforzado mi habilidad para trabajar en equipo, destacando la importancia de la comunicación clara y efectiva, especialmente en situaciones donde el proyecto tiene múltiples componentes interdependientes. La experiencia de colaborar estrechamente con mis compañeros y de compartir nuestros diferentes enfoques y soluciones ha sido increíblemente valiosa, no solo para el éxito del proyecto, sino también para mi desarrollo personal y profesional.

Arturo Benavides: Despues de haber completado este proyecto desarolle muchísimas habilidades distintas a las que estaba acostumbrado a lo largo de mi carrera. Empezando por todos los requerimientos prácticos, desde las actividades en clase hasta la implementación física del prototipo propuesto. Al tener que implementar el circuito tuve que aprender en el camino a como eficientizar el diseño, mejorar el acomodo de las conexiones y los elementos, y procurar otros factores como el correcto contacto entre elementos y la estabilidad mecánica del prototipo. De igual forma aprendí bastante sobre los métodos utilizados para medir magnitudes físicas, lo cual disfruté demasiado porque me permite visualizar aplicaciones industriales a todo lo que he aprendido a lo largo de la carrera, permitiéndome entender como se le puede dar un uso a los principios físicos que ya conozco. También al tomar medidas me di cuenta de la importancia de hacer trabajos con una metodología bien establecida, pues es de suma importancia poder hacer repeticiones para considerar el error en las medidas y poder hacer un correcto análisis estadístico que permita darle confiabilidad al proceso o poryecto que intente implementar.

References

- [1] 5V 5-Pin Relay, 9 2017. URL: <https://components101.com/switches/5v-relay-pinout-working-datasheet>.
- [2] Serial input, 2 2018. URL: <https://wiki-content.arduino.cc/en/Tutorial/LibraryExamples/LiquidCrystalSerialDisplay>.
- [3] Khan Academy. Divisor de voltaje. URL: <https://es.khanacademy.org/science/electrical-engineering/ee-circuit-analysis-topic/ee-resistor-circuits/a/ee-voltage-divider>.
- [4] Arrow Electronics. ¿Qué es un termistor?, 1 2023. URL: <https://www.arrow.com/es-mx/research-and-events/articles/how-does-a-thermistor-work>.
- [5] NayLamp Mechatronics. Sensor de temperatura y humedad relativa DHT22 (AM2302). URL: <https://>

[naylampmechatronics.com/sensores-temperatura-y-humedad/
58-sensor-de-temperatura-y-humedad-relativa-dht22-am2302.html](http://naylampmechatronics.com/sensores-temperatura-y-humedad/58-sensor-de-temperatura-y-humedad-relativa-dht22-am2302.html).

- [6] Handson Technology. I2C Serial Interface 1602 LCD Module. URL: http://www.handsontec.com/dataspecs/module/I2C_1602_LCD.pdf.

A Anexos

A.1 Código de Arduino para lectura, control y visualización de temperatura

```
1 #include <LiquidCrystal_I2C.h>
2 #include "DHT.h"
3
4 #define DHTPIN1 8
5 #define DHTPIN2 3
6 #define DHTPIN3 4
7 #define DHTPIN4 5
8
9 #define DHTTYPE1 DHT22
10 #define DHTTYPE2 DHT22
11 #define DHTTYPE3 DHT22
12 #define DHTTYPE4 DHT22
13
14 DHT dht1(DHTPIN1, DHTTYPE1);
15 DHT dht2(DHTPIN2, DHTTYPE2);
16 DHT dht3(DHTPIN3, DHTTYPE3);
17 DHT dht4(DHTPIN4, DHTTYPE4);
18
19 LiquidCrystal_I2C lcd(0x27,16,2);
20
21 // Contadores y variables de ayuda
22 int a=1;
23 //int modificar;
24 volatile bool modificar = 0;
25 bool plata = 0;
26 bool estado = 0;
27
28 void setup()
29 {
30     lcd.init();
31     lcd.clear();
32     lcd.backlight();
33
34     // Bot n
35     pinMode(2,INPUT_PULLUP);
36     pinMode(13,INPUT);
37
38     // LED set_point
39     pinMode(7,OUTPUT);
40
41     // Potenci metro
42     pinMode(A2,INPUT);
43
44     // Termistores
45     pinMode(A0,INPUT);
46     pinMode(A1,INPUT);
47
48     // Relevador
49     pinMode(6,OUTPUT);
50
51     Serial.begin(9600);
52     dht1.begin();
53     dht2.begin();
```

```

55 dht3.begin();
56 dht4.begin();
57
58 attachInterrupt(digitalPinToInterruption(2), cambio_setpoint, RISING);
59
60 }
61
62 void loop()
63 {
64 // DEFINIR SET POINT //
65
66 // bool boton = digitalRead(13);
67 // Serial.println("-Empieza-");
68 // if (modificar==0){
69 //     modificar=1;
70 // }
71 // else {
72 //     modificar=0;
73 // }
74 // Serial.println("-Empieza-");
75 while (plata==0){
76     float pot = analogRead(A2);
77     int set_point = map(pot,0,1023,14,34);
78     // Serial.print("Set point = ");
79     // Serial.println(set_point);
80     delay(500);
81     lcd.clear();
82     lcd.setCursor(2,0);
83     lcd.print("Temp. Ideal:");
84     lcd.setCursor(6,1);
85
86     lcd.print(set_point);
87     lcd.print(" C");
88 }
89
90 // MEDIR TEMPERATURA //
91 float temperaturaPromedio = tempAmb();
92 float temperaturaEntrada = tempTermistor(A0);
93 float temperaturaSalida = tempTermistor(A1);
94
95 // EVALUAR RELEVADOR //
96 float pot = analogRead(A2);
97 int set_point = map(pot,0,1023,14,34);
98 relevador(temperaturaPromedio, set_point);
99
100 // MANDAR RESULTADOS A ESP-32
101
102 // VISUALIZAR RESULTADOS //
103
104 // Monitor serial
105 // Serial.print("Promedio: ");
106 // Serial.println("--");
107 Serial.println(temperaturaPromedio);
108 // Serial.print("Termistor 1: ");
109 Serial.println(temperaturaEntrada);
110 // Serial.print("Termistor 2: ");
111 Serial.println(temperaturaSalida);
112
113 // Display
114 mostrarDisplay(8000,2,temperaturaPromedio,temperaturaEntrada,

```

```

        temperaturaSalida);
115 }

116
117
118 // FUNCIONES
119
120 void cambio_setpoint() {
121     if (modificar==0){
122         plata=!plata;
123         estado=!estado;
124         digitalWrite(7,estado);
125     }
126 }
127
128
129 float mapFloat( float x, float in_min, float in_max, float out_min,
130                 float out_max) {
131     return (x - in_min) * (out_max - out_min) / (in_max - in_min) +
132         out_min;
133 }
134
135 void relevador( float temperatura ,int set_point){
136     if (temperatura>=set_point+1){
137         delay(1000);
138         digitalWrite(6,LOW);
139     }
140     else if(temperatura<=set_point-1){
141         digitalWrite(6,HIGH);
142     }
143 }
144
145 float tempAmb(){
146     delay(2000);
147     float t1 = dht1.readTemperature();
148     float t2 = dht2.readTemperature();
149     float t3 = dht3.readTemperature();
150     float t4 = dht4.readTemperature();
151
152     if (isnan(t1) || isnan(t2) || isnan(t3) || isnan(t4)) {
153         Serial.println(F("Error en leer temperatura de DHT sensor!"));
154         return;
155     }
156
157
158     float Tp = (t1+t2+t3+t4)/4;
159     return Tp;
160 }
161
162
163 float tempTermistor(int pin){
164     float sensorValue = analogRead(pin);
165     float Vt = mapFloat(sensorValue,0.0,1023,0.0,5.0);
166     float Rt = (10000 * Vt) / (5 - Vt);
167     float Beta = .00004*Rt*Rt-1.4048*Rt+14542;
168     float T_log = 1 / ((log(Rt/10000.0)/Beta) + 1/(25+273.15));
169     float temperatura = T_log-273.15;
170     // Serial.print("Resistencia: ");
171     // Serial.println(Rt);

```

```

172 // Serial.print("Beta: ");
173 // Serial.println(Beta);
174
175 return temperatura;
176
177 }
178
179 void mostrarDisplay(int tiempo, int vueltas, float
180   temperaturaPromedio, float temperaturaEntrada, float
181   temperaturaSalida){
182   int espera = (tiempo)/3;
183
184   lcd.clear();
185   lcd.setCursor(2,0);
186   lcd.print("Temperatura");
187   lcd.setCursor(0,1);
188   lcd.print("Entrada: ");
189   lcd.print(temperaturaEntrada);
190   lcd.print(" ");
191   lcd.print("C");
192   delay(espera);
193
194   lcd.clear();
195   lcd.setCursor(2,0);
196   lcd.print("Temperatura");
197   lcd.setCursor(0,1);
198   lcd.print("Salida: ");
199   lcd.print(temperaturaSalida);
200   lcd.print(" ");
201   lcd.print("C");
202   delay(espera);
203
204   lcd.clear();
205   lcd.setCursor(2, 0);
206   lcd.print("Temperatura");
207   lcd.setCursor(0,1);
208   lcd.print("Ambiente:");
209   lcd.print(temperaturaPromedio);
210   lcd.print(" ");
211   lcd.print("C");
212   delay(espera);
213 }

```

A.2 Código para ajustar datos experimentales al modelo de Steinhart completo

```

1 import numpy as np
2 from scipy.optimize import curve_fit
3 import matplotlib.pyplot as plt
4
5 # Data provided
6 x_values = np.array([19.5, 20, 20.5, 21, 21.5, 22, 22.5, 23, 23.5,
7   24, 24.6, 25, 25.5, 26, 26.5, 27, 27.5, 28, 28.5, 29, 29.5, 30])
y_values = np.array([13.5, 12.504, 12.68666667, 12.16833333,
12.07333333, 11.83333333, 11.69833333, 11.24333333, 11.13, 10.92,
10.49333333, 10.205, 10.01166667, 9.821666667, 9.496666667,
9.348333333, 9.136666667, 8.82, 8.526666667, 8.3, 8.095,
7.671666667])

```

```

8 # Convert temperature from Celsius to Kelvin
9 x_values_kelvin = x_values + 273.15
10
11
12 # Convert resistance from kilo ohms to ohms
13 y_values_ohms = y_values * 1000
14
15 # Define the Steinhart and Hart Equation
16 def steinhart_hart_eq(R, A, B, C):
17     return 1 / (A + B * np.log(R) + C * (np.log(R))**3)
18
19 # Fit the data to the Steinhart and Hart Equation
20 initial_guess = (1e-3, 1e-4, 1e-6) # Initial guess for (A, B, C)
21 coefficients, _ = curve_fit(steinhart_hart_eq, y_values_ohms,
22                             x_values_kelvin, p0=initial_guess)
23
24 # Extract the coefficients
25 A, B, C = coefficients
26
27 # Print the coefficients
28 print("A:", A)
29 print("B:", B)
30 print("C:", C)
31
32 # Plot the data points with scatter
33 plt.scatter(y_values_ohms, x_values_kelvin, label='Data')
34
35 # Generate the fit curve
36 R_range = np.linspace(min(y_values_ohms), max(y_values_ohms), 1000)
37 T_fit = steinhart_hart_eq(R_range, A, B, C)
38
39 # Plot the fit curve
40 plt.plot(R_range, T_fit, color='red', label='Fit Curve')
41
42 # Add labels and legend
43 plt.xlabel('Resistance (Ohms)')
44 plt.ylabel('Temperature (Kelvin)')
45 plt.legend()
46
47 # Show the plot
48 plt.title('Temperature vs. Resistance with Steinhart–Hart Fit')
49 plt.grid(True)
50 plt.show()

```

A.3 Código para encontrar Beta constante a partir de datos experimentales

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from itertools import combinations
4
5 def combinaciones(n):
6     # Generar todas las combinaciones posibles de 2 n meros sin
7     # repetición entre 0 y n
8     return np.array(list(combinations(range(n+1), 2)))
9
10 def rt(T, r0, beta, T0):
11     return r0*np.exp(beta*(1/T-1/T0))

```

```

12 def calculate_beta(T,R,idx1 ,idx2):
13     R1 = R[idx1]
14     R2 = R[idx2]
15     T1 = T[idx1]
16     T2 = T[idx2]
17
18     beta = np.log(R1/R2) / (1/T1-1/T2)
19     #print("Beta = ", beta)
20     return beta
21
22 def get_mse (actual , pred):
23     actual , pred = np.array (actual) , np.array (pred)
24     return np.square (np.subtract (actual , pred)). mean ()
25
26 def optimize_beta(T,R):
27     # Posibles combinaciones
28     idx = combinaciones(21)
29
30     # Iniciar con beta arbitraria
31     beta = calculate_beta(T,R,15 ,16)
32
33     # Calcular error
34     actual = R
35     pred = rt(T,10205 ,beta ,25+273)
36     mse = get_mse(actual ,pred)
37
38     # Encontrar beta que minimiza el mse
39     for i in range(0, idx.shape[0]-1):
40         idx1 = idx [i ,0]
41         idx2 = idx [i ,1]
42
43         beta_new = calculate_beta(T,R, idx1 ,idx2)
44         pred_new = rt(T,10205 ,beta_new ,25+273)
45         mse_new = get_mse(actual ,pred)
46
47         if mse_new < mse:
48             mse = mse_new
49             beta = beta_new
50             pred = pred_new
51
52     return beta , pred , mse
53
54
55 data = np.array([
56     [19.5 , 13.5] ,
57     [20 , 12.504] ,
58     [20.5 , 12.68666667] ,
59     [21 , 12.16833333] ,
60     [21.5 , 12.07333333] ,
61     [22 , 11.83333333] ,
62     [22.5 , 11.69833333] ,
63     [23 , 11.24333333] ,
64     [23.5 , 11.13] ,
65     [24 , 10.92] ,
66     [24.6 , 10.49333333] ,
67     [25 , 10.205] ,
68     [25.5 , 10.01166667] ,
69     [26 , 9.82166667] ,
70     [26.5 , 9.49666667] ,
71     [27 , 9.34833333] ,

```

```

72 [27.5 , 9.1366666667] ,
73 [28 , 8.82] ,
74 [28.5 , 8.5266666667] ,
75 [29 , 8.3] ,
76 [29.5 , 8.095] ,
77 [30 , 7.6716666667]
78 ])
79
80 T = data[:, 0] + 273
81 R = data[:, 1] * 1000
82
83 beta , Rt , mse = optimize_beta(T,R)
84 print("Beta = ",beta)
85 print("\nError cu dratico medio = ", mse)
86
87 # Plot
88 plt.scatter(T,R)
89 plt.plot(T,Rt)
90 plt.xlabel("T (K)")
91 plt.ylabel("R(k Ohms)")
92 plt.text(300 , 13000 , 'Beta=4129.3121' , fontsize=10, color='black')
93
94 plt.show()

```

A.4 Datos de caracterización del termistor

Temperatura (°C)	R en ascenso (kOhm)	R en descenso (kOhm)				
20	12.3	13.46	12.6	12.17	11.99	
20.5		13.4	12.56	12.1	11.89	
21	12.08	12.97	12.14	11.98	11.84	12
21.5	11.9	12.76	11.88	11.79	11.79	12.32
22	11.84	12.6	11.59	11.66	11.49	11.82
22.5	11.77	12.49	11.41	11.47	11.33	11.72
23	10.8	11.76	11.13	11.36	11.09	11.32
23.5		11.61	10.76	11.17	10.89	11.22
24		11.36	10.33	11.13	10.71	11.07
24.6	10.4	10.75	9.68	10.97	10.52	10.64
25	10.1	10.5	9.12	11	10.26	10.25
25.5	9.91	10.09	8.86	11.11	10.11	9.99
26	9.58	9.84	8.68	10.96	9.97	9.9
26.5	9.39	9.66	8.5	10.59	9.01	9.83
27	9.2	9.48	8.46	10.24	8.94	9.77
27.5	8.9	9.31	8.35	9.89	8.84	9.53
28	8.7	9.01	8.3	9.4	8.79	8.72
28.5	8.4	8.78	8.13	9.1	8.65	8.1
29	8.3	8.68	7.89	8.72	8.55	7.66
29.5	7.9	8.53	7.77	8.53	8.5	7.34
30	7.78	8.36	7.66	7.58	7.38	7.27

Table 2: Datos de resistencia medida en ascenso y descenso

T real (°C)	T en ascenso (°C)	T en descenso (°C)				
20	19.88	17.70	19.29	20.14	20.50	
20.5		17.94	19.47	20.36	20.78	
21	20.49	18.83	20.38	20.69	20.97	20.65
21.5	20.93	19.33	20.97	21.15	21.15	20.14
22	21.14	19.74	21.62	21.48	21.82	21.18
22.5	21.36	20.05	22.05	21.93	22.20	21.45
23	23.31	21.46	22.66	22.21	22.73	22.29
23.5		21.82	23.43	22.64	23.18	22.54
24		22.35	24.32	22.78	23.57	22.89
24.6	24.21	23.54	25.66	23.14	23.98	23.75
25	24.80	24.04	26.84	23.12	24.49	24.51
25.5	25.17	24.83	27.35	22.99	24.79	25.02
26	25.80	25.30	27.67	23.30	25.06	25.19
26.5	26.15	25.63	27.98	23.96	26.90	25.31
27	26.47	25.94	27.97	24.58	26.98	25.41
27.5	27.00	26.22	28.11	25.19	27.12	25.82
28	27.32	26.73	28.11	26.03	27.15	27.28
28.5	27.82	27.10	28.36	26.52	27.34	28.42
29	27.92	27.22	28.73	27.14	27.45	29.20
29.5	28.59	27.41	28.85	27.41	27.47	29.73
30	28.71	27.64	28.94	29.10	29.50	29.73

Table 3: Datos de ascenso y descenso de temperatura con modelo logarítmico