

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

CAMPUS MONTERREY



**Tecnológico
de Monterrey**

**Caracterización experimental mediante
instrumentación electrónica**

Reto

Control de temperatura en aires acondicionados y
monitoreo de presión y energía

Equipo 4

Alberto Anaya Velasco	A01252512
Daniela Cruz Alvarez	A00572205
Carlos Gabriel Espinosa Contreras	A01198290
Arif Morán Velázquez	A01234442

Profesores:

José Manuel Rodríguez Delgado
Valery Moreno Vega
Alonso Sosa Avendaño
Ricardo Sotelo Mora
Salvador Rodríguez López

18 de marzo de 2024

1. Introducción

Un cine perteneciente a *CitiCinemas* ha experimentado un exceso de consumo de energía en los últimos años. Tras analizar la problemática, la administración ha concluido que esto se podría solucionar atacando ciertos problemas relacionados con el sistema de aire acondicionado en cada una de sus salas de cine. Entre las áreas de mejora identificadas están el analizar la referencia de temperatura con que se trabajan las salas, la estrategia de control ON-OFF, la eficiencia del sistema de ductos de aire, la eficiencia del A/C, así como evaluar el consumo del medidor y definir un control electrónico.

En este informe se presenta el diseño y la elaboración del control electrónico, el cual permitirá adquirir datos, transmitirlos y visualizarlos en tiempo real para realizar algunos de los análisis descritos previamente. La elaboración de este sistema requirió de la utilización de sensores y actuadores, microcontroladores y otros componentes, así como la creación de algoritmos de control, transmisión y visualización de datos, entre otros aspectos. Al tomar en cuenta estas consideraciones, se ha creado un sistema robusto y confiable para el control y monitoreo de la temperatura y presión de las salas de cine para ayudar a solucionar la problemática de *CitiCinemas*.

Tarea	Responsable
Comunicación serial Arduino-NodeMCU	Carlos Espinosa
Lógica del circuito	Alberto Anaya
Envió de datos a Firebase	Daniela Cruz
Despliegue de datos (pagina Web)	Arif Morán
Diseño del circuito	Carlos Espinosa
Impresión PCB	Alberto Anaya
Montar el circuito	Arif Moran y Alberto Anaya
Caracterización experimental	Daniela Cruz

Tabla 1: Tareas del Reto y sus responsables

2. Desarrollo de la solución

Para el desarrollo de la solución, se consideraron las diferentes funciones que debe ofrecer el sistema con base en la problemática a resolver. Por una parte, este sistema debe ser capaz de realizar la medición de temperatura en las salas y en las entradas y salidas del aire acondicionado y del refrigerante, así como la medición de presión del refrigerante en el evaporador y condensador. Por otro lado, es necesario contar con un algoritmo de control ON-OFF eficiente para encender dos compresores y un ventilador de recirculación. También es importante contar con una pantalla que muestre los datos adquiridos por los sensores. Por último, es indispensable contar con un sistema de comunicación WiFi para transmitir los datos a la nube en tiempo real y generar un display gráfico para la visualización de las diferentes variables a medir.

Además, el objetivo de optimizar se atacó mediante el uso adecuado de los compresores para minimizar el consumo eléctrico del cine. Para ello se optó por un control ON-OFF mediante histéresis con base en la diferencia de la temperatura promedio (entre los 4 termistores de la sala) y la temperatura de referencia (SET-POINT), la cual se puede modificar a través de dos botones. El primer compresor

se enciende cuando la temperatura en la sala es igual o mayor a 2°C por encima de la temperatura de referencia. De manera similar, el segundo compresor sólo se enciende hasta que la diferencia es mayor a 4°C. Finalmente, el primer compresor, no se apagará hasta que la temperatura esté 2°C por debajo de la temperatura de referencia, esto para que la sala permanezca a una temperatura cercana a la temperatura de referencia por mayor tiempo.

Tras definir lo anterior, se comienza a considerar el diseño del prototipo, con los componentes a utilizar y las conexiones entre ellos. Por una parte, la medición de temperatura se realiza con la utilización de termistores. Los termistores son resistores cuya resistencia varía logarítmicamente dependiendo de la temperatura, de manera más pronunciada que una resistencia estándar. Se utilizaron termistores en un arreglo de divisor de voltaje, y se incluyeron dos capacitores de 10 y 0.1 μF conectados en paralelo para evitar ruido generado al convertir la señal de voltaje de analógica a digital con el microcontrolador.

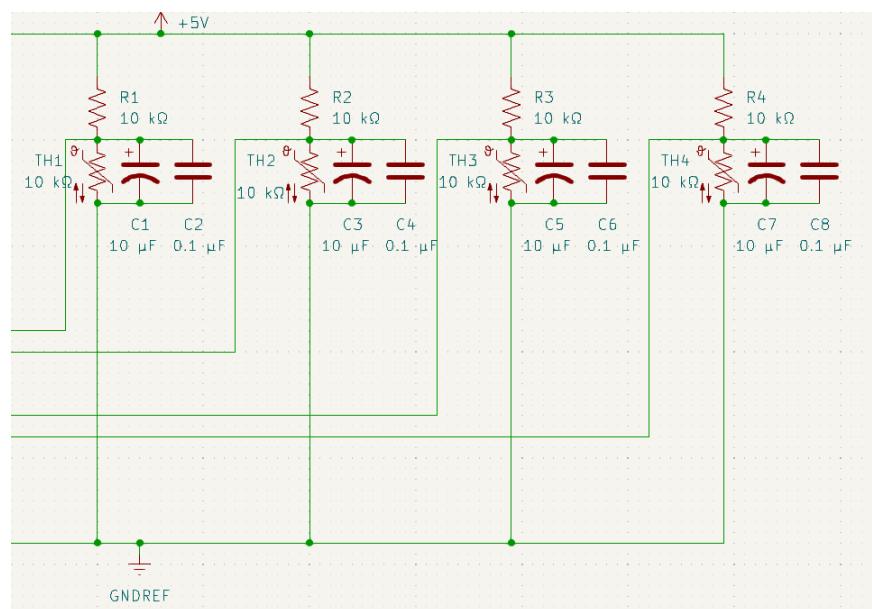


Figura 1: Diagrama de conexión de termistores

Además, se utilizó un arreglo de transistores y relevadores para energizar a los compresores y al ventilador. Este arreglo consiste en la conexión del colector del transistor en el lado negativo de la bobina y el emisor a tierra, mientras que la base va conectada a través de una resistencia a la salida digital del microcontrolador para activar la bobina y conectar la terminal del relevador que encenderá los compresores o el ventilador.

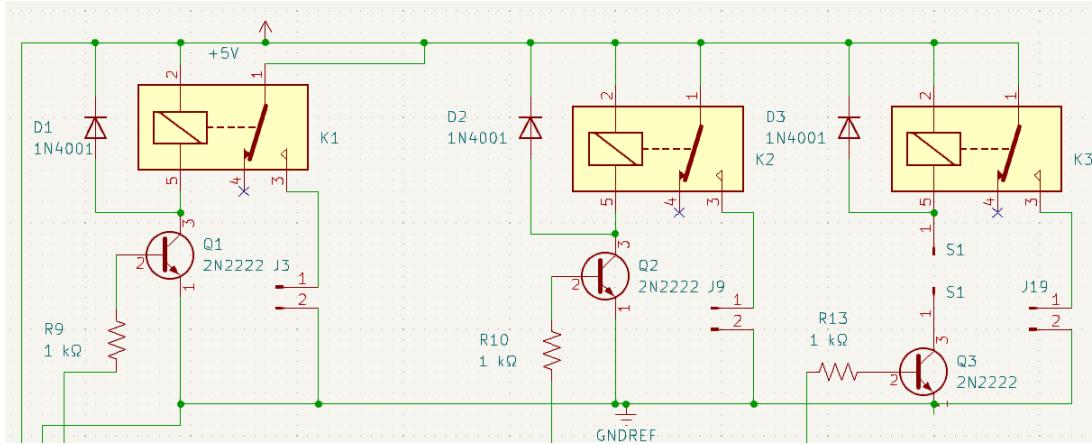


Figura 2: Conexión de sistema de relevadores

Por último, se incluyó una entrada para los sensores de presión, los cuales trabajan con 24V y devuelven un voltaje de 0 a 10 V. A falta de este tipo de sensores de uso industrial, se simuló su funcionamiento utilizando un arreglo de resistencias y un divisor de voltaje, utilizando un potenciómetro, para lograr una salida máxima de 5V para la lectura analógica de la señal utilizando el microcontrolador.

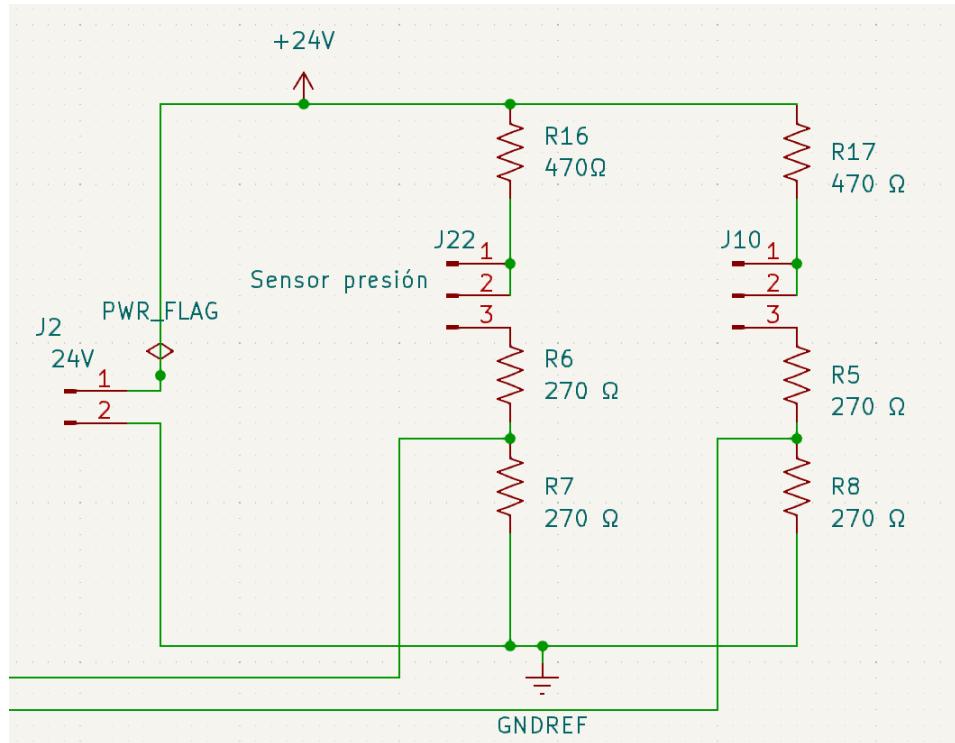


Figura 3: Diagrama de conexión de sensores de presión

El diagrama del circuito se puede apreciar en la Figura 4 y en la Tabla 2 un listado de los componentes utilizados.

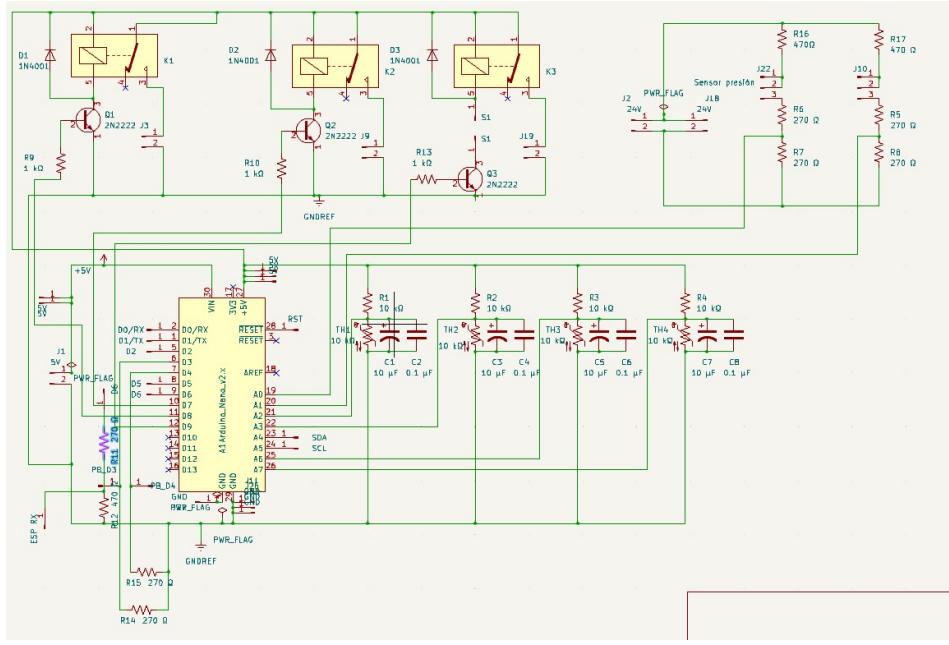


Figura 4: Diagrama del circuito desarrollado en KiCad

2.1. Descripción de las entradas y salidas

Descripción de la instrumentación (sensores) utilizados y de los actuadores (salidas, por ejemplo, relés, utilizados).

- Termistores: Los termistores son resistencias sensibles a la temperatura, es decir, su resistencia cambia a medida que la temperatura varía. Son muy sensibles y reaccionan a cambios de temperatura muy pequeños. En este caso, se trabaja con termistores de $10\text{ k}\Omega$, en un rango de 0 -50 grados Celsius.
 - Potenciómetros: El potenciómetro es una resistencia cuyo valor de resistencia en vez es variable, y se manipula mediante una perilla. Este, se usa para simular un sensor de presión con una entrada de 24 Volts y salida de 5 V.
 - Relays: Los relays son switches que son activados mediante un mecanismo inductivo , que se activa al pasarle corriente. En este caso se usa para pasar de 5v a 5v y encender un indicador de los compresores.
 - Transistor tipo npn: El transistor npn es un switch que se cierra cuando le llega un valor de 1 a la base. En el circuito , se usa para activar los relays.
 - Convertidor DC- DC 24 a 5V: Este módulo de potencia recibe los 24 V de la fuente y entrega un nivel de tensión a la salida inferior de 5 V y sirve para alimentar el arduino.
 - Switch y botón: El switch funciona para encender el circuito completo y energizar con 24 volts el convertidor. Por otro lado, los botones sirven para incrementar o disminuir la temperatura a la que se desea llegar.
 - Capacitores: Los capacitores funcionan para limpiar la señal proveniente de los termistores y que no varíe su lectura ante variaciones de voltaje de la fuente.

Arduino NANO	Conexión
A0,A1	Sensores de Presión
A4,A5	SDA,SCL(LCD I2C)
A2,A3,A6,A7	Termistores
D5,D6	Tx,Rx(Esp8266)
D9,D8,D7	Relevador 1,2,3
D3,D4	disminuir, incrementar T_{on}

Tabla 2: Conexiones de Arduino nano

2.2. Hardware

Descripción del Hardware restante (incluye el microcontrolador, la LCD, etc., y conexión de sensores y actuadores (por ejemplo, arduino y conexión a los pines de la plataforma).

Arduino Nano: Se utiliza como la plataforma principal para el desarrollo del proyecto. Es una pequeña placa que permite un rápido prototipado sobre una protoboard. Este microcontrolador sera el encargado de leer los sensores y enviar de manera serial las mediciones de los sensores al NodeMCU.

NODEMCU (ESP8266): Actúa como un módulo de comunicación inalámbrica que se conecta al Arduino a través de una comunicación serial. El NODEMCU puede recibir datos del Arduino y enviarlos a través de WiFi a servicios en la nube, como Firebase.

LCD: El LCD con protocolo I2C funciona como un contenedor para un componente maestro I2C y utiliza un componente maestro, en este caso el Arduino, para recibir instrucciones. En el circuito lo utilizamos para desplegar la temperatura promedio y la temperatura de encendido.

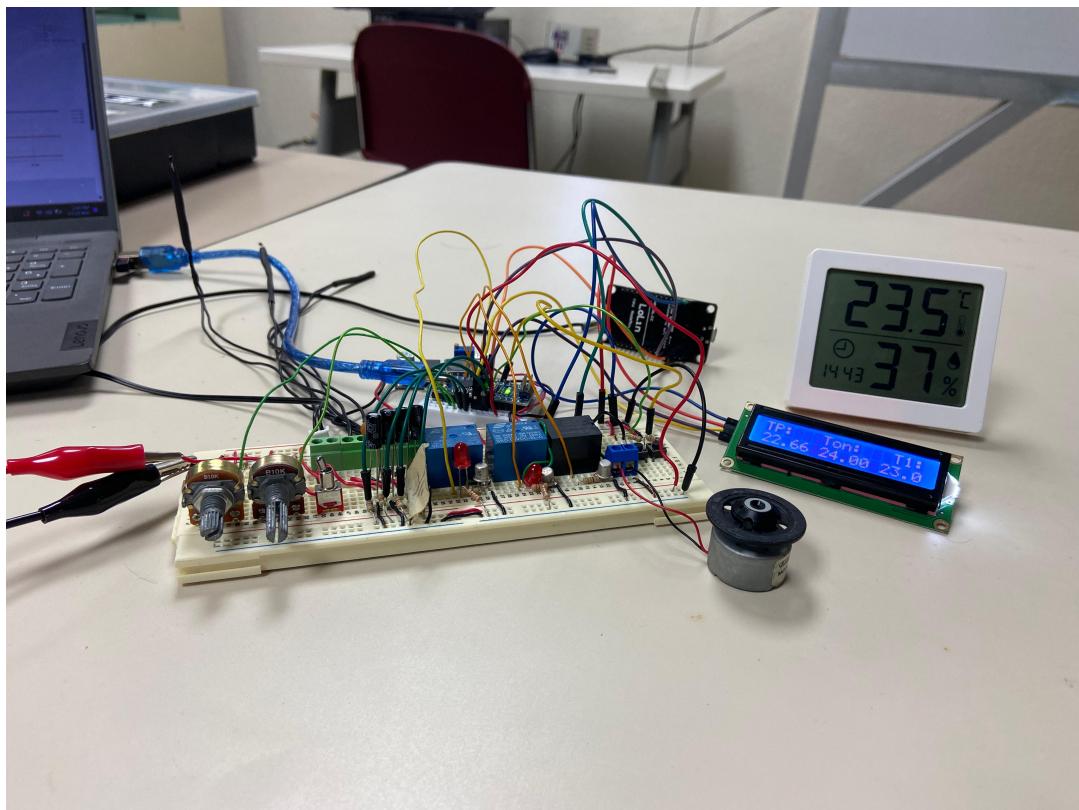


Figura 5: Circuito electrónico de la solución

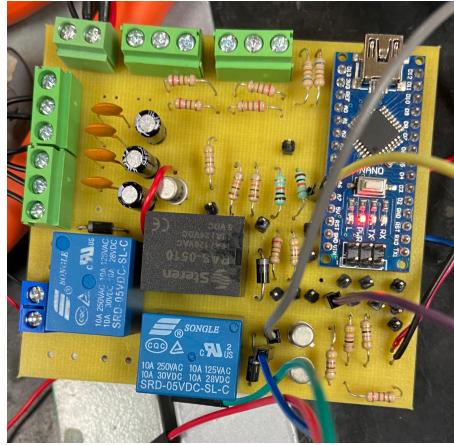


Figura 6: *Printed circuit board* (PCB) de la solución

2.3. Software

El software está compuesto por 3 códigos, dentro de ella hay cuatro partes esenciales, la comunicación serial Arduino-NodeMCU, el envío de datos a Firebase, la lógica del circuito, y por último el despliegue de datos.

1. Lógica del circuito: esta parte del código se corre en el Arduino, en donde se toma la lectura del voltaje de los pines analógico; se hace la transducción a unidades físicas; se calcula la temperatura promedio; el hace el control de los compresores y ventilador, por medio de los relevadores según la temperatura de encendido y la promedio; se imprimen algunos datos en el LCD; finalmente, se hace la comunicación con el nodeMCU.
 - a) Configuración inicial: Se incluyen las librerías necesarias para el proyecto y se declaran variables para almacenar datos y configuraciones iniciales.
 - b) Función r2t: Esta función convierte resistencias en temperaturas utilizando una fórmula logarítmica del datasheet.
 - c) Configuración Inicial en la Función setup: Se inicializa la pantalla LCD, se configuran los pines y se inicia la comunicación serial para establecer el entorno de trabajo.
 - d) Función loop: Controla el flujo principal del programa, incluyendo la lectura de botones, cálculos con sensores y el control del sistema de calefacción.
 - e) Operaciones Matemáticas y Lecturas de Sensores: Se realizan cálculos matemáticos, lecturas de sensores analógicos y conversiones de datos para obtener información relevante.
 - f) Comunicación Serial y Visualización de Datos: Se envían datos a través de comunicación serial para su visualización o procesamiento externo, y se muestra información en la pantalla LCD.
2. Comunicación serial Arduino-NodeMCU: En esta parte se tiene la comunicación entre los microcontroladores, en donde se tiene la escritura en el puerto serial del arduino, la lectura del puerto serial del NodeMCU, y el procesamiento de la lectura. La comunicación serial entre el Arduino y el

NODEMCU se establece mediante la configuración de un puerto serial virtual utilizando la librería SoftwareSerial en el Arduino. Asimismo, se inicia la comunicación serial en ambos dispositivos a una velocidad de 115200 baudios para garantizar una transmisión de datos eficiente y sincronizada. Una vez configurada la comunicación serial, se envían datos de temperatura y presión desde el Arduino al NODEMCU a través de este puerto serial virtual. Estos datos pueden ser utilizados por el NODEMCU para realizar diversas acciones, mostrar la información en una pantalla del LCD, enviarla a un servidor o realizar un control basado en los valores recibidos.

3. Envió de datos a Firebase: En esta parte del código el nodeMCU envía los datos a Firebase de la siguiente manera:
 - a) Se definen constantes para la conexión a internet y a Firebase.
 - b) Se establece la conexión a internet.
 - c) Se establece la conexión a Firebase con lo definido previamente.
 - d) Al haber recibido los datos del Arduino, se verifica la conexión y se envían a Firebase.
4. Despliegue de datos: En esta parte el código se establece una conexión Wi-Fi, se lee datos de los termistores y sensores presión, y los envía a Firebase para su almacenamiento y visualización en tiempo real y actualización de las gráficas.
 - a) Configuración inicial: Se incluyen las bibliotecas necesarias y se definen las credenciales de red y la clave API del proyecto Firebase.
 - b) Función de configuración: Se inicia la comunicación serial y se conecta a la red Wi-Fi. Se realiza un registro anónimo en Firebase y se inicia la generación de tokens para la autenticación.
 - c) Bucle principal: Se lee la entrada serial para recibir datos del sensor en formato de cadena. Se tokeniza la cadena para extraer los valores de temperatura y presión. Luego, se envían estos datos a la base de datos en tiempo real de Firebase.
 - d) Funciones de envío de datos: Se utilizan funciones de Firebase para enviar los datos de temperatura y presión a ubicaciones específicas en la base de datos.

The screenshot shows the Firebase Realtime Database interface. On the left, there's a sidebar with navigation links like 'Realtime Database', 'Authenticación', 'Hosting', 'Extensions', 'Releases Mobile...', 'Categorías de producto', 'Compilación', 'Lanzamiento y supervisión', 'Analytics', 'Participación', and 'Todos los productos'. The main area is titled 'Realtime Database' and has tabs for 'Datos', 'Reglas', 'Copias de seguridad', 'Uso', and 'Extensiones'. A modal window titled 'Protege tus recursos de Realtime Database contra los abusos, como fraudes de facturación o phishing.' is open, with a link to 'Configurar la Verificación de aplicaciones'. The database structure is shown as a tree: 'https://c-citicinemas-default.firebaseio.com/' -> 'Sensor' -> 'Pressure1: 0', 'Pressure2: 0', 'Temperature: 25.18', 'Temperature2: 24.72999542', 'Temperature3: 24.29', 'Temperature4: 0', 'Temperature5: 0', 'Temperature6: 0', 'Temperature7: 0', 'Temperature8: 0', and 'Time: 906.843994141'. At the bottom, it says 'Ubicación de la base de datos: Estados Unidos (us-central)'.

Figura 7: Base de datos en firebase

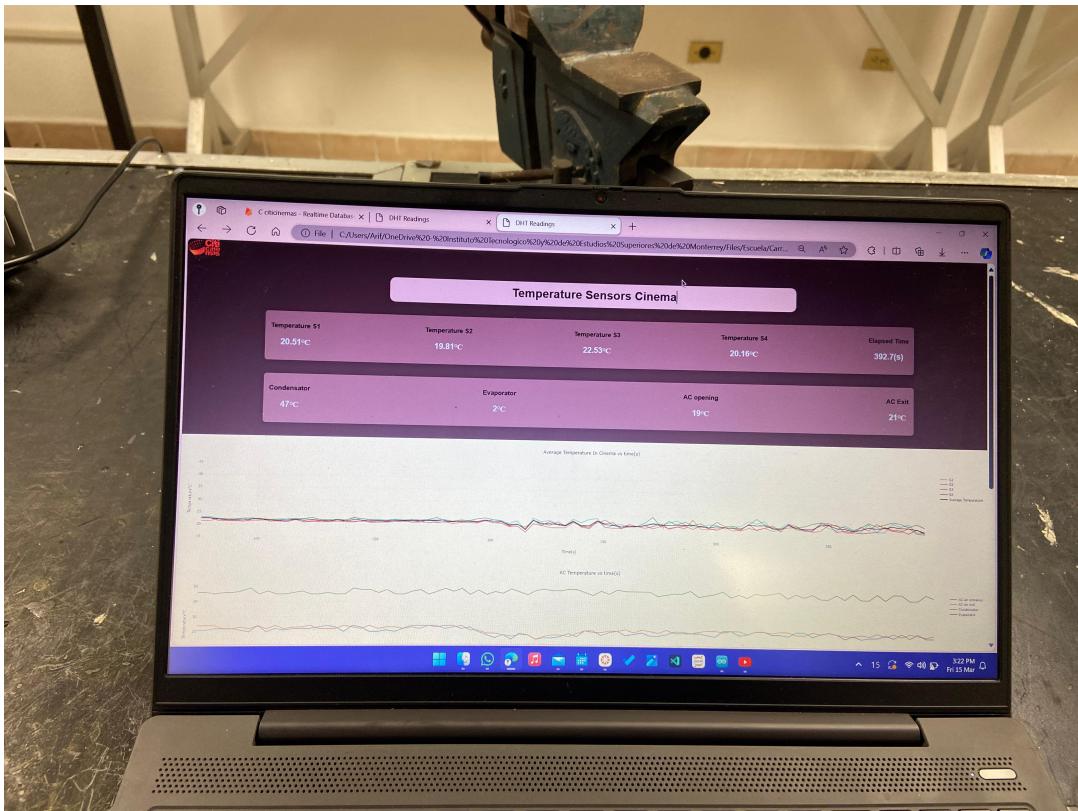


Figura 8: HTML

3. Caracterización

Caracterización experimental de un sensor de los utilizados.

Utilizando registros de medición en diferentes rangos del sensor utilizado y un instrumento patrón

Se realizó la caracterización experimental para un sensor utilizado para las siguientes temperaturas (basado en el instrumento patrón) 19.1, 23.7, 25, 27, 29,

y 30 grados Celsius. Se obtuvo que el error promedio del sensor es de 0.975 grados Celsius. Por lo que, cada que se calcule una temperatura con el sensor, el valor de la temperatura real se encontrara en un rango de

$$[medido - errorpromedio, medido + errorpromedio] \quad (1)$$

como se muestra en la figura 9.

Temperatura Real	S1
19,1	18,94
23,7	23,14
25	24,02
27	24,91
29	27,51
30	29,43

Tabla 3: Comparación entre las temperaturas reales y el sensor 1

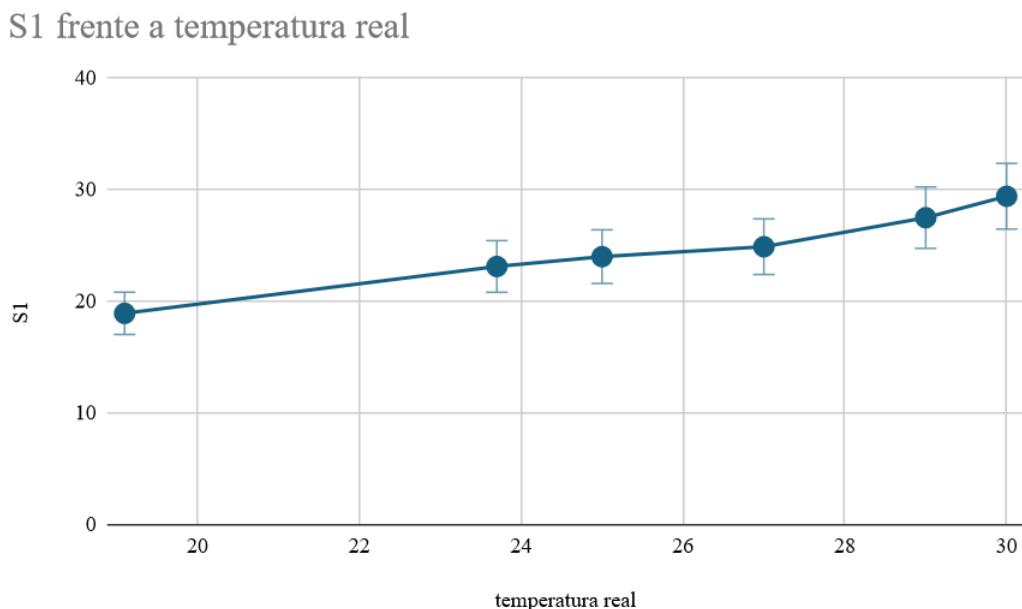


Figura 9: Gráfica de temperatura medida por un sensor contra el instrumento patrón

Asimismo, se calculo la precisión del sensor calculando su desviación estándar, haciendo 10 mediciones del sensor con la misma temperatura real de 23.7 grados centígrados, y se obtuvo una precisión de 0.452.

4. Conclusiones

A lo largo de este proyecto se trabajó con un sistema eléctrico con el fin de monitorear y optimizar el consumo de energía por aires acondicionados en un cine. Se modeló una situación similar mediante un circuito electrónico de Arduino y un ESP8266 con la capacidad de monitorizar la temperatura de la sala, el aire acondicionado y de la presión del refrigerante en los aires. Asimismo,

el sistema cuenta con la capacidad de mandar estos datos de manera inalámbrica a una base de datos en Firebase, donde se puede acceder por una página web que diseñamos. Durante el curso de 5 semanas se desarrolló dicho sistema, con el objetivo de realizar el circuito final en una PCB así como un algoritmo que optimizara el consumo. En un principio se montó el circuito completamente en una PCB, sin embargo, no se logró que funcionara de manera adecuada. Gracias a esto pudimos entender la dificultad y la precisión que requiere este proceso. Ya que, el proceso mismo requería de varios días para diseño y para checar errores. Por lo que hizo falta planear la PCB con más tiempo para checar funcionamiento y hacer correcciones. Sin embargo, debido al tiempo tan reducido, no se logró entrar a estas etapas de corrección.

Adicionalmente, a lo anterior, pudimos darnos cuenta del papel que juega la programación y el algoritmo en la optimización de sistemas eléctricos. Lo cual definió nuestro algoritmo para nuestra implementación en el circuito de usar diferencias de temperatura entre la temperatura de encendido y temperatura ambiente promedio, en lugar de fijar un valor. Ya que esto modela de mejor manera la realidad y da mayor libertad al usuario al permitir modificar esta temperatura.

Consideramos que fue un proyecto de valor, ya que es útil y aplicable, no solo en una industria, con sistemas automatizados, sino en nuestros próximos bloques para el diseño de experimentos y mediciones mediante instrumentación electrónica.

Alberto Anaya Velasco

La realización de este proyecto implicó la integración de diferentes habilidades y conocimientos que desarrollamos no sólo durante el bloque sino que también durante los semestres anteriores. Creo que fue realmente interesante y de gran utilidad el entender cómo elaborar un sistema de monitoreo de variables físicas con un fin aplicable a la vida real. No sólo tuvimos que utilizar nuestro conocimiento adquirido de electrónica y circuitos, sino que también tuvimos que utilizar nuestras habilidades de programación para automatizar el proceso y enviar los datos, así como complementar con nuestro conocimiento estadístico para la medición de error y manipulación de datos. Además, en lo personal aprendí sobre el diseño y fabricación de circuitos impresos, lo cual me parece fue muy interesante. Considero que este ha sido un reto que se destaca por la integración de conocimientos y habilidades que requirió, y lograr construir un diseño funcional exitosamente ha sido verdaderamente gratificante.

Daniela Cruz Alvarez

Este reto fue juntar todo lo aprendido durante estas cinco semanas de trabajo, se conectó todo lo que habíamos hecho con todos los profesores, desde la implementación sencilla de encender y apagar LEDs, hasta lograr hacer un diseño optimizado. Asimismo, juntar lo aprendido en la clase de Estadística, para poder caracterizar y analizar correctamente los datos que estábamos obteniendo. También, pudimos juntar lo aprendido en los circuitos teóricos, para poder calcular las resistencias adecuadas para los componentes que requerían distintas corrientes para no quemarse. Aprender a manipular los datos de los sensores para obtener los resultados que queríamos, por ejemplo, los sensores de temperatura, teníamos que hacer una transducción para obtener los grados centígrados, y que no es algo

directo que te dan los sensores. Aprender a hacer la comunicación en serie del Arduino con el NodeMCU, para poder mostrar los datos obtenidos en el pagina web. Sin embargo, no todo fue tan sencillo, debíamos ser siempre precavidos de no conectar mal los componentes, porque sino se podían quemar las cosas y debíamos empezar de nuevo, asimismo, con los códigos que realizábamos, debíamos de ser cuidadosos, porque siempre podía haber errores de tipo o configuraciones que no se hacían y podían estar ocasionando fallos en el circuito.

Carlos Gabriel Espinosa Contreras

Pudimos encontrar una solución a la problemática presente en el cine, por medio de la electrónica y el internet de las cosas, con estos se podrá tomar una rápida y efectiva acción para acondicionar correctamente las salas, considerando el consumo energético del proceso. Para esto fue fundamental hacer uso de lo impartido a lo largo del curso, aprendí a analizar circuitos y comprender las formas en que podemos manipular el voltaje y la corriente, realizar las conexiones en el circuito considerando la polaridad de los componentes, la lógica detrás de las operaciones que se pueden realizar en los circuitos y el funcionamiento de algunos protocolos de comunicación. Estos conocimientos aunados nos permitieron crear circuitos lógicos en los que se procesaba la información de entrada, los sensores principalmente, para dar una salida, como es el envío de datos a Firebase, la impresión de datos en el LCD y el control sobre otros dispositivos que regulan el aire acondicionado. Finalmente, caracterizamos un sensor relativo a un instrumento patrón, esto nos permite tener un nivel de certeza sobre las mediciones de los instrumentos utilizados.

Arif Morán Velázquez

Uno de los puntos más importantes para un diseño optimizado es la histéresis del sistema. En este caso se optó por tres escenarios: 1 compresor, 2 compresores y ningún compresor. Para dar mayor flexibilidad al usuario, la temperatura de encendido se puede manipular a través de dos botones para incrementarla y decrementarla. Esto a fin de evitar que solo se pudiese trabajar con una temperatura de encendido. Considero que esta parte es una de las importantes a seguir trabajando, ya que, la optimización de estos valores puede ayudar a reducir más el consumo del Cine. Asimismo, se puede mejorar la página estética y funcionalmente, podríamos agregar más funciones como poder monitora más variable, así como poder controlar los compresores y la temperatura de encendido. Si bien no se logró que funcionara la pcb, considero que sirvió como un ejercicio valioso como introducción y a manera de enseñanza, que es un proceso que requiere de mucho cuidado y de verificación con más tiempo. Considero en lo personal, que fue un reto muy valioso, ya que fue una situación tangible y real en la industria y en la investigación. Fue un reto interesante que disfrute y que considero será importante en los demás bloques, como base y como una herramienta de experimentación y medición de variables físicas, así como su error.

Referencias

[DIY, 2021] DIY, R. (2021). Send Data From Arduino to NodeMCU and NodeMCU to Arduino Via Serial Communication.

5. Anexos

Link a Códigos

5.1. Código Arduino Nano

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <math.h>
#include <SoftwareSerial.h>

float te=24; //temperatura de encendido

LiquidCrystal_I2C lcd(0x27, 16, 2);

float r_25 = 9900.0;
float aa = 3.354017e-3;
float bb = 2.5617244e-4;
float cc = 2.1400943e-6;
float d = -7.2405219e-8;

float tt[8]={1,1,1,1,1,1,1,1};
float V[8];
float V0[8];
float V1[8];
float denomin[8];
float denomin1[8];
float num[8];
float R[8];
float temp[8];

void r2t(float RR[], float g[]) {
    for (int i = 0; i < 8; i++) {
        float resistencia = RR[i];
        g[i]=(1/(aa + bb*log(resistencia/r_25)) + cc*pow(log(resistencia/r_25),2) + d*pow(log(resistencia/r_25),3))-273;
    }
}

SoftwareSerial espSerial(5, 6); // RX, TX

void setup() {
    lcd.init();
    lcd.backlight();
    lcd.setCursor(0, 0);
    lcd.print("TP:~~Ton:~~T1:~");
    // put your setup code here, to run once:
    Serial.begin(115200);
    espSerial.begin(115200);
    pinMode(7,OUTPUT);
    pinMode(8,OUTPUT);
    pinMode(9, OUTPUT);
    pinMode(3, INPUT);
    pinMode(4, INPUT);
}

void scal(float a, float b[],float c[]) {
    //float c[8]={1,1,1,1,1,1,1,1};
    for (int i = 0; i < 8; i++) {
        c[i] = a*b[i];
    }
}

void sum(float a, float b[],float c[]) {
    //float c[8]={1,1,1,1,1,1,1,1};
    for (int i = 0; i < 8; i++) {
        c[i] = b[i]+a;
    }
    //return c[8];
}

void mult(float a[], float b[],float c[]) {
    //float c[8]={1,1,1,1,1,1,1,1};
    for (int i = 0; i < 8; i++) {
        c[i] = b[i]*a[i];
    }
    //return c[8];
}

void pwr(float a, float b[], float c[]){
    for (int i = 0; i < 8; i++) {
        c[i] = pow(b[i],a);
    }
}

void dvs(float a[], float b[], float c[]){
    for (int i = 0; i < 8; i++) {
        c[i] = (a[i]/b[i]);
    }
}

void loop() {
    int inc = digitalRead(3);
```

```

int dec = digitalRead(4);

// Check if the increment button is pressed and the previous state was not pressed
if (inc == HIGH) {
    // Increment the counter
    te++;
    inc=LOW;
}
if (dec == HIGH) {
    // Increment the counter
    te=te-1;
    //dec=LOW;
    //Serial.println(te);
    // Print the updated counter value to the serial monitor
    //Serial.println("Counter increased: " + String(counter));
}

// put your main code here, to run repeatedly:
//pressure
int s0 = analogRead(A0);
int s1 = analogRead(A1);
//Temperature
int s2 = analogRead(A2);
int s3 = analogRead(A3);
int s6 = analogRead(A6);
int s7 = analogRead(A7);

float p1=map(s0 ,0 ,1024 ,40 ,120);
float p2=map(s1 ,0 ,1024 ,150 ,350);

float ss[8]={s2 ,s3 ,s6 ,s7 ,0 ,0 ,0 ,0};////////{t1 ,t2 ,t3 ,t4 ,t5}

float jj[8]={1024.0,1024.0,1024.0,1024.0,1024.0,1024.0,1024.0,1024.0};
scal(5.0,ss,V); //float v = s0* (5.0/1024);
dvs(V,jj ,V0);
scal(-1,V0,V1); //float resistencia = (voltage * 10000)/(5 - voltage);
sum(5.0 ,V1,denom);
scal(r_25 ,V0,num);
dvs(num,denom ,R);
r2t(R,temp);

temp[4]=random(5)+45;
temp[5]=te-random(5);
temp[6]=(te-random(3));
temp[7]=random(5);

float tp=(temp[0]+temp[1]+temp[2]+temp[3])/4;
//digitalWrite(8,HIGH);

if(round(tp-te)>2 && round(tp-te)<5{
    digitalWrite(7,HIGH);
    digitalWrite(8,LOW);
    digitalWrite(9,HIGH);

}
else if(round(tp-te)>=5{
    digitalWrite(8,HIGH);
    digitalWrite(7,HIGH);
    digitalWrite(9,HIGH);
}

else if((tp-te)<=-2){
    digitalWrite(7,LOW);
    digitalWrite(8,LOW);
}

else {
    //digitalWrite(7,LOW);
    //digitalWrite(8,LOW);
    //digitalWrite(9,LOW);
}

//scal(1000/5-)
//scal(2.0,test ,hola);
//Serial.print(tp);
//Serial.print(',');
//Serial.print(temp[0]);
Serial.print(',');
Serial.print(temp[1]);
Serial.print(',');
Serial.print(temp[2]);
Serial.print(',');
Serial.print(temp[3]);
Serial.print(',');
Serial.print(temp[4]);
Serial.print(',');
Serial.print(temp[5]);
Serial.print(',');
Serial.print(temp[6]);
Serial.print(',');
Serial.print(temp[7]);
Serial.print(',');
Serial.print(p1);
Serial.print(',');
Serial.println(p2);

lcd.setCursor(0, 1);
lcd.print(tp);

```

```

lcd.setCursor(6, 1);
lcd.print(te);

lcd.setCursor(12, 1);
lcd.print(temp[0]);

//String kk= String(temp[0])+' '+String(temp[1]);
espSerial.print(temp[0]);
espSerial.print(' ');
espSerial.print(temp[1]);
espSerial.print(' ');
espSerial.print(temp[2]);
espSerial.print(' ');
espSerial.print(temp[3]);
espSerial.print(' ');
espSerial.print(temp[4]);
espSerial.print(' ');
espSerial.print(temp[5]);
espSerial.print(' ');
espSerial.print(temp[6]);
espSerial.print(' ');
espSerial.print(temp[7]);
espSerial.print(' ');
espSerial.print(p1);
espSerial.print(' ');
espSerial.println(p2);
//espSerial.println(kk);

delay(3500);

// float temp1 = r2t(resistencia);
//temp2 = r2t(resistencia2);
//Serial.println(temperatura);
}

```

5.2. Código Esp8266

```

#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <Firebase_ESP_Client.h> //we are using the ESP82

//To provide the ESP32 / ESP8266 with the connection and the sensor type
//DHT dht_sensor(DHT_SENSOR_PIN, DHT_SENSOR_TYPE);

//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

// Insert your network credentials
#define WIFI_SSID "iPhone_de_Alberto"//"Totalplay-D6AA"//
#define WIFI_PASSWORD "bibliopiso4"//"D6AAA8DFrJyw3yyo"//

// Insert Firebase project API Key
#define API_KEY "AIzaSyAsyvcBuHnVDoPiDP55YVx-R2NEE9QBeQ"

// Insert RTDB URLdefine the RTDB URL */
#define DATABASE_URL "https://c-citicinemas-default-rtdb.firebaseio.com/"

//Define Firebase Data object
FirebaseData fbd;

FirebaseAuth auth;
FirebaseConfig config;

unsigned long sendDataPrevMillis = 0;
int count = 0;
bool signupOK = false; //since we are doing an anonymous sign in

void setup() {
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  /* Assign the api key (required) */
  config.api_key = API_KEY;

  /* Assign the RTDB URL (required) */
  config.database_url = DATABASE_URL; //"c-citicinemas-default-rtdb.firebaseio.com/"

  /* Sign up */
  if (Firebase.signUp(&config, &auth, "", "")) {
    Serial.println("ok");
    signupOK = true;
  } else {
    Serial.printf("%s\n", config.signer.signupError.message.c_str());
  }

  /* Assign the callback function for the long running token generation task */

```

```

config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h

Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);
}

void loop() {
    if (Serial.available()) {
        String gg = Serial.readStringUntil('\n');
        float time = round(micros() / 1e3) / 1000;

        // Convert String to char array
        char receivedCharArray[gg.length() + 1];
        gg.toCharArray(receivedCharArray, sizeof(receivedCharArray));

        // Initialize an array to store individual values
        float values[10];

        // Tokenize the string using strtok
        char *token = strtok(receivedCharArray, ", ");
        int i = 0;
        while (token != NULL && i < 10) {
            // Convert the token to a floating-point number and store in the array
            values[i] = atof(token);

            // Move to the next token
            token = strtok(NULL, ",");
            i++;
        }

        // Assign values to individual variables
        float t1 = values[0];
        float t2 = values[1];
        float t3 = values[2];
        float t4 = values[3];
        float t5 = values[4];
        float t6 = values[5];
        float t7 = values[6];
        float t8 = values[7];
        float p1 = values[8];
        float p2 = values[9];
    }

    if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 1000 || sendDataPrevMillis == 0)) {
        //since we want the data to be updated every second
        sendDataPrevMillis = millis();
        // Enter Temperature in to the DHT_11 Table
        if (Firebase.RTDB.setFloat(&fbdo, "Sensor/Temperature", t1)) {
            // This command will be executed even if you dont serial print but we will make sure its working
            //Serial.print("Temperature:");
            //Serial.println(t1);
        } else {
            Serial.println("Failed to Read from the Sensor");
            Serial.println("REASON: " + fbdo.errorReason());
        }
        if (Firebase.RTDB.setDouble(&fbdo, "Sensor/Temperature2", t2)) {
        }
        //else {
        //    Serial.println("Failed to Read from the Sensor");
        //    Serial.println("REASON: " + fbdo.errorReason());
        //}
        if (Firebase.RTDB.setFloat(&fbdo, "Sensor/Temperature3", t3)) {
            // This command will be executed even if you dont serial print but we will make sure its working
        }
        if (Firebase.RTDB.setDouble(&fbdo, "Sensor/Temperature4", t4)) {
            // This command will be executed even if you dont serial print but we will make sure its working
        }
        if (Firebase.RTDB.setDouble(&fbdo, "Sensor/Temperature5", t5)) {
            // This command will be executed even if you dont serial print but we will make sure its working
        }
        if (Firebase.RTDB.setDouble(&fbdo, "Sensor/Temperature6", t6)) {
            // This command will be executed even if you dont serial print but we will make sure its working
        }
        if (Firebase.RTDB.setDouble(&fbdo, "Sensor/Temperature7", t7)) {
            // This command will be executed even if you dont serial print but we will make sure its working
        }
        if (Firebase.RTDB.setDouble(&fbdo, "Sensor/Temperature8", t8)) {
            // This command will be executed even if you dont serial print but we will make sure its working
        }
        if (Firebase.RTDB.setDouble(&fbdo, "Sensor/Pressure1", p1)) {
            // This command will be executed even if you dont serial print but we will make sure its working
        }
        if (Firebase.RTDB.setDouble(&fbdo, "Sensor/Pressure2", p2)) {
            // This command will be executed even if you dont serial print but we will make sure its working
        }
        if (Firebase.RTDB.setDouble(&fbdo, "Sensor/Time", time)) {
            // This command will be executed even if you dont serial print but we will make sure its working
            //Serial.print("Elapsed Time : ");
            //Serial.println(time);
        }
    }
}

```

}

5.3. Código HTML

```
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>DHT Readings</title>
    <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
    <style type="text/css">
        body{
            background-color: #8b7373;
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }
        .data-container{
            display: flex;
            justify-content: space-between;
            width: 80%;
            margin: 50px auto;
            padding: 20px;
            background-color: #b99b9b;
            box-shadow: 0 2px 20px rgba(0, 0, 0, 0.1);
            border-radius: 10px;
        }
        .data-item{
            text-align: center;
        }
        .data-item h2{
            font-size: 24px;
            font-weight: bold;
            margin-bottom: 10px;
        }
        .data-item p{
            font-size: 30px;
            font-weight: bold;
            color: hsl(185, 16%, 85%);
        }
        .data-head{
            margin: auto;
            width: 50%;
            text-align: center;
            font-size: 45px;
            font-weight: bold;
            margin: 50px auto;
            padding: 20px;
            background-color: #e4d4d4;
            box-shadow: 0 5px 20px rgba(0, 0, 0, 0.1);
            border-radius: 20px;
        }
    </style>
</head>

<body>
    
    <script>
        var xx = [1,2,3,4];
        window.xx=xx;
    </script>
    <div class="data-head">Temperature Sensors Cinema</div>
    <div class="data-container">
        <div class="data-item">
            <h2>Temperature S1</h2>
            <p class="value" id="temperatures1"> 24 &#8451;</p>
        </div>
        <div class="data-item">
            <h2>Temperature S2</h2>
            <p class="value" id="temperatures2"> 37 &#8451;</p>
        </div>
        <div class="data-item">
            <h2>Temperature S3</h2>
            <p class="value" id="temperatures3"> 58 &#8451;</p>
        </div>
        <div class="data-item">
            <h2>Temperature S4</h2>
            <p class="value" id="temperatures4"> 99 &#8451;</p>
        </div>
        <div class="data-item">
            <h2>Elapsed Time </h2>
            <p class="value" id="Etime"> 1 </p>
        </div>
    </div>
</body>
```

```

        </div>

    </div>
<div class="data-container">
    <div class="data-item">
        <h2>Condensator </h2>
        <p class="value" id="temperatures5"> 24 &#8451;</p>
    </div>

    <div class="data-item">
        <h2>Evaporator</h2>
        <p class="value" id="temperatures8"> 37 &#8451;</p>
    </div>

    <div class="data-item">
        <h2>AC opening </h2>
        <p class="value" id="temperatures6"> 24 &#8451;</p>
    </div>

    <div class="data-item">
        <h2>AC Exit </h2>
        <p class="value" id="temperatures7"> 24 &#8451;</p>
    </div>

</div>

</div id="myDiv"></div>
<div id="myDiv2"></div>

    <div class="data-head">Pressure Sensors of Refrigerant </div>

<div class="data-container">
    <div class="data-item">
        <h2>AC Condensator Pressure</h2>
        <p class="value" id="pressure1"> 24 ;</p>
    </div>

    <div class="data-item">
        <h2>AC Evaporator pressure</h2>
        <p class="value" id="pressure2"> 37 ;</p>
    </div>
</div>

</div id="myDiv3"></div>

<script type="module">

    // Import the functions you need from the SDKs you need
    import { initializeApp } from "https://www.gstatic.com/firebasejs/10.8.0/firebase-app.js";
    import { getDatabase , ref , get , onValue} from "https://www.gstatic.com/firebasejs/10.8.0.firebaseio.js";
    import { getAnalytics } from "https://www.gstatic.com/firebasejs/10.8.0/firebase-analytics.js";
    // TODO: Add SDKs for Firebase products that you want to use
    // https://firebase.google.com/docs/web/setup#available-libraries

    // Your web app's Firebase configuration
    // For Firebase JS SDK v7.20.0 and later, measurementId is optional
    const firebaseConfig = {
        apiKey: "AlzaSyAsyvcBUhNivDoPiDP55YVx-R2NEE9QBeQ",
        authDomain: "c-citicinemas.firebaseio.com",
        databaseURL: "https://c-citicinemas-default-rtdb.firebaseio.com",
        projectId: "c-citicinemas",
        storageBucket: "c-citicinemas.appspot.com",
        messagingSenderId: "853571736012",
        appId: "1:853571736012:web:031fa3684d0aa0dcbe4d69",
        measurementId: "G-ML5W2XYHDS"
    };
    // Initialize Firebase

    // Initialize Firebase
    const app = initializeApp(firebaseConfig);
    const database = getDatabase(app);

    //getting reference to the data we want
    //Temperature
    var dataRef1 = ref(database , 'Sensor/Temperature');
    var dataRef2 = ref(database , 'Sensor/Temperature2');
    var dataRef3 = ref(database , 'Sensor/Temperature3');
    var dataRef4 = ref(database , 'Sensor/Temperature4');
    var dataRef5 = ref(database , 'Sensor/Temperature5');
    var dataRef6 = ref(database , 'Sensor/Temperature6');
    var dataRef7 = ref(database , 'Sensor/Temperature7');
    var dataRef8 = ref(database , 'Sensor/Temperature8');

    //Pressure
    var dataRef9 = ref(database , 'Sensor/Pressure1');
    var dataRef0 = ref(database , 'Sensor/Pressure2');

    //Time
    var dataReft = ref(database , 'Sensor/Time');

```

```

let temperatures1 = [];
let temperatures2 = [];
let temperatures3 = [];
let temperatures4 = [];
let temperatures5 = [];
let temperatures6 = [];
let temperatures7 = [];
let temperatures8 = [];
let pressures1 = [];
let pressures2 = [];
let ts = [];

onValue(dataRef1, (snapshot) => {
const time = snapshot.val();

// Fetch values from dataRef1
get(dataRef1).then((temp1Snapshot) => {
const temp1 = temp1Snapshot.val();

// Fetch values from dataRef2
get(dataRef2).then((temp2Snapshot) => {
const temp2 = temp2Snapshot.val();

// Fetch values from dataRef3
get(dataRef3).then((temp3Snapshot) => {
const temp3 = temp3Snapshot.val();

get(dataRef4).then((temp4Snapshot) => {
const temp4 = temp4Snapshot.val();

get(dataRef5).then((temp5Snapshot) => {
const temp5 = temp5Snapshot.val();

get(dataRef6).then((temp6Snapshot) => {
const temp6 = temp6Snapshot.val();

get(dataRef7).then((temp7Snapshot) => {
const temp7 = temp7Snapshot.val();

get(dataRef8).then((temp8Snapshot) => {
const temp8 = temp8Snapshot.val();

get(dataRef9).then((temp9Snapshot) => {
const pres1 = temp9Snapshot.val();

get(dataRef0).then((temp0Snapshot) => {
const pres2 = temp0Snapshot.val();

// Update HTML elements
document.getElementById('Etime').innerHTML = Math.round(time*10)/10 +(s);
document.getElementById('temperatures1').innerHTML = Math.round(temp1*100)/100 + "&#8451;";
document.getElementById('temperatures2').innerHTML = Math.round(temp2*100)/100 + "&#8451;";
document.getElementById('temperatures3').innerHTML = Math.round(temp3*100)/100 + "&#8451;";
document.getElementById('temperatures4').innerHTML = Math.round(temp4*100)/100 + "&#8451;";
document.getElementById('temperatures5').innerHTML = temp5 + "&#8451;";
document.getElementById('temperatures6').innerHTML = temp6 + "&#8451;";
document.getElementById('temperatures7').innerHTML = temp7 + "&#8451;";
document.getElementById('temperatures8').innerHTML = temp8 + "&#8451;";

document.getElementById('pressure1').innerHTML = pres1 + "psi";
document.getElementById('pressure2').innerHTML = pres2 + "psi";

// Update arrays and plot
ts.push(time);
temperatures1.push(temp1);
temperatures2.push(temp2);
temperatures3.push(temp3);
temperatures4.push(temp4);
temperatures5.push(temp5);
temperatures6.push(temp6);
temperatures7.push(temp7);
temperatures8.push(temp8);

pressures1.push(pres1)
pressures2.push(pres2)

updatePlotly(ts, temperatures1, temperatures2, temperatures3, temperatures4, temperatures5, temperatures6, temperatures7, temperatures8, pressures1, pressures2);
});
```

```

        //export const xx; // Export xx at the end
        //return xx;
</script>

<script>var pp= 23*3 ;
//console.log(myVariable);
</script>

<script>var xData=[1,2,3,0,5];
</script>

<script>

var layout = {
    title: 'Average Temperature In Cinema vs time(s)',
    plot_bgcolor: "rgba(0,0,0,0)",
    paper_bgcolor: "rgba(0,0,0,0)",
    yaxis: {
        title: {
            text: 'Temperature'+"&#8451;",
        },
        range: [15, 45],
    },
    xaxis: {
        title: {
            text: 'Time(s)',
        },
    },
};

var layout2 = {
    title: 'AC Temperature vs time(s)',
    yaxis: {
        title: {
            text: 'Temperature'+"&#8451;",
        },
        range: [0, 50],
    },
    xaxis: {
        title: {
            text: 'Time(s)',
        },
    },
};

var layoutp = {
    title: 'Pressure in Refrigerant vs time(s)',
    yaxis: {
        title: {
            text: 'Pressure psi',
        },
        range: [0, 370],
    },
    xaxis: {
        title: {
            text: 'Time(s)',
        },
    },
};

// Plotly.newPlot('myDiv', [data1,data2], layout);

// Update the chart dynamically when needed:
function updatePlotly(datax, T1,T2,T3,T4,T5,T6,T7,T8, TA,P1,P2) {
    var y1 = {
        x:datax,
        y: T1, // Initial data for the first plot
        type: 'scatter',
        mode: 'lines',
        line: { color: '#17BECF' },
        name: 'S1',
    };

    var y2 = {
        x:datax,
        y: T2, // Add your second set of y data here
        type: 'scatter',
        mode: 'lines',
        line: { color: '#FF5733' }, // Choose a different color for the second plot
        name: 'S2',
    };

    var y3 = {
        x:datax,
        y: T3, // Add your second set of y data here
        type: 'scatter',
        mode: 'lines',
        line: { color: '#FF5733' }, // Choose a different color for the second plot
        name: 'S3',
    };

    var y4 = {
        x:datax,
        y: T4, // Initial data for the first plot

```

```

        type: 'scatter',
        mode: 'lines',
        //line: { color: '#17BECF' },
        name: 'S4',
    };

    var Av = {
        x: datax,
        y: TA, // Add your second set of y data here
        type: 'scatter',
        mode: 'lines',
        line: { color: 'Black' }, // Choose a different color for the second plot
        name: 'Average Temperature',
    };

    var y5 = {
        x: datax,
        y: T5, // Add your second set of y data here
        type: 'scatter',
        mode: 'lines',
        line: { color: '#FF5733' }, // Choose a different color for the second plot
        name: 'Condensator',
    };

    var y6 = {
        x: datax,
        y: T6, // Add your second set of y data here
        type: 'scatter',
        mode: 'lines',
        line: { color: '#FF5733' }, // Choose a different color for the second plot
        name: 'AC air entrance',
    };

    var y7 = {
        x: datax,
        y: T7, // Add your second set of y data here
        type: 'scatter',
        mode: 'lines',
        line: { color: 'Black' }, // Choose a different color for the second plot
        name: 'AC air exit',
    };

    var y8 = {
        x: datax,
        y: T8, // Add your second set of y data here
        type: 'scatter',
        mode: 'lines',
        line: { color: 'Black' }, // Choose a different color for the second plot
        name: 'Evaporator',
    };

    var z1 = {
        x: datax,
        y: P1, // Add your second set of y data here
        type: 'scatter',
        mode: 'lines',
        line: { color: 'Black' }, // Choose a different color for the second plot
        name: 'P1 psi',
    };

    var z2 = {
        x: datax,
        y: P2, // Add your second set of y data here
        type: 'scatter',
        mode: 'lines',
        line: { color: 'red' }, // Choose a different color for the second plot
        name: 'P2 psi',
    };

    Plotly.newPlot('myDiv', [y1,y2,y3,y4,Av], layout);
    Plotly.newPlot('myDiv2', [y6,y7,y5,y8], layout2);
    Plotly.newPlot('myDiv3', [z1,z2], layoutp);

    // Plotly.update('myDiv', [data3,data4]);
};

// Call updateChart when necessary, such as after receiving new data
</script>

</body>
</html>

```

5.4. Arreglo experimental completo

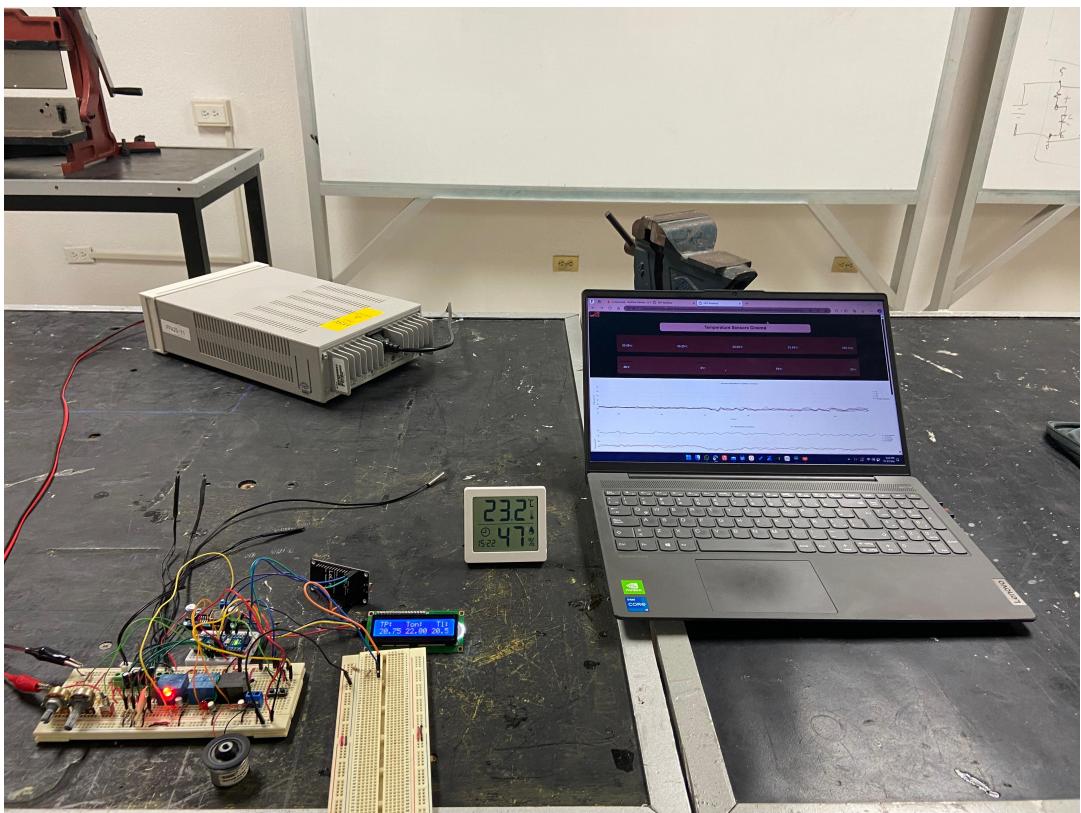


Figura 10: Arreglo experimental en función