Project Report

Smart Automatic Temperature Controller

IoT-Based Thermal Protection System Using ESP32-C3 (Simulated in Wokwi)

📌 1. Project Title:

Smart Automatic Temperature Controller for Small Heat-Sensitive Devices

---

🧠 2. Objective:

To design a compact, automated temperature control system using a microcontroller that:

- Monitors temperature in real-time,
- Automatically turns ON a cooling fan when temperature rises,
- Turns OFF the fan when the temperature drops back to a safe level.

This project simulates how **smart thermal management** can be applied in **compact embedded systems** or **IoT devices**.

🧰 3. Hardware Used:

Component	Function
ESP32-C3 DevKit	Main controller (compact, low-power, IoT-ready)
Potentiometer	Simulates temperature sensor (in Wokwi)
Blue LED	Simulates mini cooling fan
Serial Monitor	Displays temperature and fan status

> 🔧 Note: The **buzzer was intentionally removed** to maintain silent operation and simplicity.

🌐 4. Real-World Use Case:

This system is applicable in:

- **Small PCB circuits**, ICs, lithium batteries

- **Wearables and embedded IoT devices**
- **Mini control units or drones**

Any place where **thermal damage** must be prevented automatically, without manual intervention.

🔄 5. Control Logic:

Condition	Action
Temperature > 38°C	Fan (LED) turns ON
Temperature < 35°C	Fan turns OFF

💡 6. Working Principle:

- The **potentiometer** simulates variable temperature input.
- The **ESP32-C3** maps the analog input to a simulated temperature range (20°C – 60°C).
- Based on temperature thresholds:
  - The **cooling fan (LED)** turns ON or OFF accordingly.
  - All data is printed via **Serial Monitor** for observation.

🔌 7. Circuit (Wokwi Simulation)

Connections:

Component	ESP32-C3 Pin	Purpose
Potentiometer	GPIO 2 (A0)	Temperature input
Fan LED (Blue)	GPIO 3	Fan control
Pot VCC	3.3V	Power
Pot GND, LED GND	GND	Ground

Simulated using **Wokwi ESP32-C3 DevKit**.

💻 8. Arduino Code:

CopyEdit

```
#define TEMP_PIN 2          // Analog input pin
```

```
#define FAN_PIN 3         // Digital output for fan (LED)

void setup() {
  pinMode(FAN_PIN, OUTPUT);
  Serial.begin(115200);
}

void loop() {
  int analogVal = analogRead(TEMP_PIN);
  float temperature = map(analogVal, 0, 4095, 20, 60); // Simulated temp in °C

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");

  if (temperature > 38) {
    digitalWrite(FAN_PIN, HIGH); // Turn ON fan
  } else if (temperature < 35) {
    digitalWrite(FAN_PIN, LOW);  // Turn OFF fan
  }

  delay(500);
}
```

🧠 9. My Contribution:

- Designed circuit logic and flow based on real-world use cases.
- Used **Wokwi simulator** to prototype the system without physical hardware.
- Ensured **clean and noise-free operation** by excluding unnecessary components (e.g., buzzer).
- Focused on **minimalist design** suitable for space-constrained electronics.

🤖 10. Use of AI Tools:

I used **AI (ChatGPT by OpenAI)** to assist in:

- Writing and optimizing Arduino code
- Structuring control logic clearly
- Generating documentation in professional format

This helped speed up the process, validate ideas, and present the solution more effectively.

✅ 11. Conclusion:

The **Smart Automatic Temperature Controller** is a highly efficient and low-footprint solution for small devices prone to heat. It demonstrates how **embedded systems and automation** can enhance device safety without the need for complex hardware.

🧱 12. Challenges Faced:

1. Size Constraints

- Most traditional temperature controllers are **bulky or made for industrial use**.
- My project targets **very small components** (like microcontrollers, power ICs, etc.), where traditional systems **won't fit**.

2. Sensor Simulation in Wokwi

- Since no real sensors were used, I had to simulate temperature using a **potentiometer**.
- Mapping analog values to accurate temperature range took testing and calibration.

3. Removing Buzzer Logic

- Many tutorials focus on buzzers for alerts. But I wanted a **silent, practical solution**.
- Ensured alerting was still effective via **Serial Monitor logs** or future Wi-Fi add-ons.

4. Keeping It Minimal

- Designing a solution that works with **just 1 input (temp) and 1 output (fan)** while staying smart and automated.
- Making it simple, clean, and fit for IoT embedding.

🧭 13. Why This Project — If Temperature Controllers Already Exist?

This question is very common in **viva, interviews, and reports**.

Here's a strong, clear answer you can include:

> ✅ **Yes, standard temperature controllers already exist**, but they are often:
>
> - Designed for **larger industrial systems**
> - Use **expensive sensors and hardware**
> - Not suited for **compact embedded electronics**

🔧 **My project solves a specific problem:**

- What if you have a **tiny circuit or battery** that heats up?
- You can't use a full thermostat or industrial controller.
- You need a **tiny, low-power, smart system** — like this.

🛠️ So I built this:

- **For learning**: to understand embedded sensing and actuation.
- **For scaling**: can be embedded into real IoT devices.
- **For customizing**: this basic logic can be extended to Wi-Fi, mobile alerts, or dashboards.

✅ 14. Final Summary (For Interview):

If the interviewer asks:

> "Why did you make this when it already exists?"

You say:

> "Because existing controllers aren't made for **very small electronics**. I wanted to build a **minimal, embedded-friendly version** using ESP32-C3. It's simple, fast, and scalable — ideal for small devices like IoT boards, battery packs, or sensor nodes. This is also great for learning and demonstrating embedded automation."