Computer Organization & Assembly Language
Project Proposal

Project Title:

"*ResQTemp* — Smart Temperature & Rescue Alert System"

Submitted by:

Moavia Amir (2k24_BSAI_72)
Muhammad Dawood (2k24_BSAI_31)

Semester:

3rd

Submitted to:

Prof. Ghulam Mustafa

Introduction:

ResQTemp is a microcontroller-based smart temperature control and emergency alert system designed to integrate low-level **Assembly programming** principles with **IoT functionality**. The system continuously monitors environmental temperature using an LM35 sensor, executes real-time decision logic in Assembly, and triggers appropriate actions such as turning on a cooling fan, activating an LED indicator, or initiating a **rescue alert after a 15-second safety window**.

The device also hosts a web page on an ESP8266 IoT module, displaying live temperature readings, system status, and location-sharing features for emergency conditions. This project merges **COAL concepts**—direct hardware interfacing, data manipulation, and control flow in Assembly—with **modern IoT communication**, showcasing how low-level control can coexist with network-based intelligence.

Problem Statement:

Temperature management and emergency detection are critical in environments like server rooms, labs, and industrial setups. Traditional thermostats only control temperature but lack **intelligent timing** and **alert mechanisms**.
In many low-cost systems, once overheating occurs, there is **no immediate alert** or **remote visibility**.

ResQTemp addresses this issue by combining **Assembly-level temperature control** (ensuring high precision and hardware interaction) with **IoT-based remote monitoring**, offering both automation and awareness.

Objectives:

- Implement a temperature monitoring system using **Assembly language** on an Arduino.
- Activate safety outputs (LED/Fan) when temperature crosses threshold.
- Introduce a **15-second delay** before the emergency trigger (to prevent false alarms).
- After 15 seconds, initiate a **rescue alert** through the ESP8266 module.
- Display real-time data and system status on an **IoT web dashboard**.
- Share GPS/location data in emergency mode.

System Overview

The system consists of:

- **Microcontroller (Arduino)** programmed in **Assembly** to:
    - Read temperature sensor values (LM35).
    - Control fan and LED indicators.
    - Trigger buzzer and alert flag if overheating persists.
- **IoT/GSM module (ESP8266)** to:
    - Send an alert message or location update via the internet or SMS.
    - Display the system status on a simple web page or message log.

Proposed Solution and Methodology

Hardware Component	Software Components
Arduino UNO	Assembly Language for Arduino (COAL core)
LM35 Sensor	C++ (for ESP8266 IoT communication)
LED & Cooling Fan (control output)	Web Page; HTML + CSS (for the IoT dashboard webpage)
ESP8266 Wi-Fi Module	Cooling control output
Resistors, Breadboard, Power Supply, Connecting Wires	

Working Principle:

1. The **LM35** sensor provides an analogue voltage proportional to temperature.
2. The **Arduino** reads the sensor data and executes **Assembly instructions** to:
    - Compare the temperature with the set threshold
    - Turn **ON/OFF** the **fan** or **LED**
    - Start a **15-second** timer if overheating persists
3. If the condition remains after 15 seconds, **Arduino** sends a signal to the **ESP8266**.
4. **ESP8266** uploads data to a web server and shows:
5. Current temperature
6. System status (Normal / Overheat / Rescue Mode)
7. The user can view and control the system remotely through the IoT dashboard.

This entire process takes place in **real time**, allowing responsive and accurate movement.

Project Timeline

Week 1 – Research & Planning

- Study project requirements and COAL lab objectives.
- Review interfacing of LM35 sensor, Arduino, and ESP8266.
- Finalise system design and flow diagram.

Week 2 – Assembly Coding & Simulation

- Write and test Assembly code for temperature reading and delay routine.
- Simulate control logic using Proteus or TinkerCAD.
- Verify correctness of input/output operations.

Week 3 – Hardware Implementation

- Assemble circuit components (LM35, Arduino, LEDs, buzzer, ESP8266).
- Upload the Assembly program via the Arduino interface.
- Test temperature sensing and 15-second timer control.

Week 4 – IoT Integration & Web Page Development

- Connect ESP8266 for IoT communication.
- Develop a simple HTML web page to display live temperature data and alerts.
- Test data transmission between the microcontroller and web dashboard.

Week 5 – Final Testing & Documentation

- Perform full system testing under different temperature conditions.
- Record results and verify all expected outcomes.
- Prepare final **report**, presentation slides, and project demonstration.

Expected Outcomes:

- A fully functional hardware system that reads temperature using **LM35** and controls outputs (fan/LED) through precise **Assembly-level logic**.
- Implementation of an accurate 15-second delay routine in Assembly before activating emergency mode, demonstrating **COAL** timing and control skills.

- Integration of **ESP8266** to display live temperature, device status, and alerts on a simple web page accessible via Wi-Fi.
- System triggers an **IoT**-based rescue or location-sharing alert if the high-temperature condition persists beyond the safety delay.
- A working prototype that combines **low-level** Assembly programming with modern IoT communication, showing both technical depth and innovation.

Future Enhancements:

- Integration of **ultrasonic sensor** for distance or obstacle safety.
- Addition of a **camera module** for real-time video monitoring.
- Development of a **mobile app** for remote control.
- Integration with **IoT cloud platforms** (e.g., Blynk or Thing Speak) for data logging and analysis.

Conclusion:

The ***ResQTemp* — Smart Temperature & Rescue Alert System** demonstrates how **Computer Organization and Assembly Language (COAL)** principles can be practically applied to solve real-world automation problems.
It successfully integrates **low-level hardware control** with **IoT-based communication**, achieving a system that is efficient, reliable, and responsive.
This project not only strengthens understanding of Assembly-level microcontroller programming but also prepares the students for advanced embedded and IoT-based system design in the future..
