Heater Control System Report

Embedded Systems Intern – upliance.ai

Author: Vamshi Krishna

Date: July 22, 2025

---

0. Introduction

This document describes a project designed during the Embedded Systems Internship at upliance.ai. It showcases how basic components can be integrated to build a functional, safety-aware heater controller using real-time temperature sensing and automation.

1. Abstract / Objective

The goal of this project is to build a heater control system that monitors temperature using a DHT22 sensor, and reacts appropriately to protect users from overheating. The system is simple yet demonstrates essential embedded concepts like control flow, state machines, and real-world I/O interaction.

2. Components Used

Below are the key components used to construct and simulate the heater controller. All components are easily available and chosen for simplicity and clarity.

Component	**Purpose**
Arduino Uno	Central microcontroller to execute program logic
DHT22	Digital temperature sensor for accurate readings
I2C LCD (16x2)	Shows current temperature and system state
Buzzer	Alerts user when overheating is detected
LED	Represents a heating element in simulation
Power Supply	Provides stable voltage to Arduino and components
Jumper Wires	Connects all parts together on breadboard

3. Communication Protocols

This section highlights how the components talk to each other during operation. Simpler protocols were chosen to make testing and debugging easier.

- **I2C** – Used to control the LCD module with just two wires.
- **Digital I/O** – Used for reading the DHT22 sensor and toggling LED/Buzzer.
- **Serial** – Used for monitoring data through USB on Serial Monitor.

4. Block Diagrams

To better understand the system, two block diagrams are provided — one for the simulation environment and one for the real-world setup.

4.1 Simulation-Level Architecture

Insert simulation_architecture.png here
(Space reserved for simulation block diagram)

4.2 Real-World Architecture

Insert hardware_architecture.png here
(Space reserved for real-world block diagram)

5. System Working & State Machine

The heater control logic is divided into five different states. Based on temperature changes, the system switches between these states to manage heating and safety.

State	**Temp Range (℃)**	**Action**
Idle	Startup	System initializes; heater OFF.
Heating	Temp ¡ 28	Heater turns ON to increase the temperature.

Stabilizing	28 Temp ¡ 32	Heater remains ON, nearing target.
Target Reached	32 Temp ¡ 60	Heater turns OFF, holding temperature.
Overheat	Temp 60	Buzzer activates; heater shuts OFF.

6. LCD Display Logic

The LCD helps users visually understand the system's current temperature and state. It updates every 2 seconds in real-time.

- **Line 1:** Shows the temperature (e.g., `Temp: 32.5 C`)
- **Line 2:** Shows current state (e.g., `State: Heating`)

7. Serial Monitor Output

This helps with debugging and remote observation during simulation. A typical output is:

```
Temp: 29.2 C | State: Stabilizing
```

8. Simulation Link

To view or test this project in a virtual environment, visit the Wokwi simulation below:

- https://wokwi.com/projects/437113575167255553

9. Future Improvements

The system was designed with expansion in mind. Below are possible future features:

- **Wi-Fi Enabled ESP32:** Allow wireless control and data streaming.
- **Mobile App:** Create a mobile app to monitor heater status.
- **FreeRTOS Tasking:** Use task-based architecture for better efficiency.
- **Over-the-Air Updates:** Push software changes remotely to the system.
- **Cloud Logging:** Use services like Firebase or AWS IoT to store temperature logs.

- **Touch Interface:** Replace LCD with touchscreen for better UX.
- **AI Forecasting:** Predict future temperature based on usage history.

10. Conclusion

This project demonstrated how a simple microcontroller setup can mimic real-world heating systems with safety and automation. It proves that even beginner-friendly components can be powerful when used with structured logic and good design thinking.