SYSTEM DESIGN DOCUMENT
BASIC HEATER CONTROL SYSTEM

INTERNSHIP ASSIGNMENT – UPLIANCE.AI

Assignment: Part 1 – System Design Deliverable

1. MINIMUM SENSORS AND ACTUATORS REQUIRED

A heater control system relies on a combination of input sensors and output actuators to regulate and simulate real-world thermal behavior. Selecting the right components is essential to ensure accurate sensing, reliable actuation, and responsive feedback.

Required Sensors and Actuators:

➢ Temperature Sensor (LM35 / DHT22):

- Measures ambient temperature in real-time.
- Analog (LM35) or digital (DHT22) options are available.
- Enables conditional logic for heating control based on thresholds.
- Highly accurate, easy to interface with Arduino.

➢ Heater Simulation Output (LED / Digital Pin):

- Represents the heating element.
- Simple ON/OFF control logic via digital GPIO pin.
- LED can visually simulate heating being active or inactive.

➢ Feedback Device (Optional LED or Buzzer):

- Indicates system status:
    - Blinking LED for Heating
    - Solid LED for Target Reached
    - Buzzer alert for Overheat
- Improves usability and debugging, especially in simulation environments.
- Minimum Components Table:

Components	Quantity	Purpose
Temperature Sensor	1	Reads ambient temperature
LED (Heater Sim)	1	Simulates heater ON/OFF state
LED/Buzzer (Optional)	1	Status feedback

2. RECOMMENDED COMMUNICATION PROTOCOL

A key requirement in embedded systems is efficient and clear communication between modules, especially for debugging, monitoring, or external interfacing.

Selected Protocol: UART (Serial Communication)

➤ **What is UART?**
Universal Asynchronous Receiver-Transmitter (UART) is a serial communication protocol that transmits data asynchronously between two devices using two wires: TX (transmit) and RX (receive).

Why UART is Ideal for This Project:

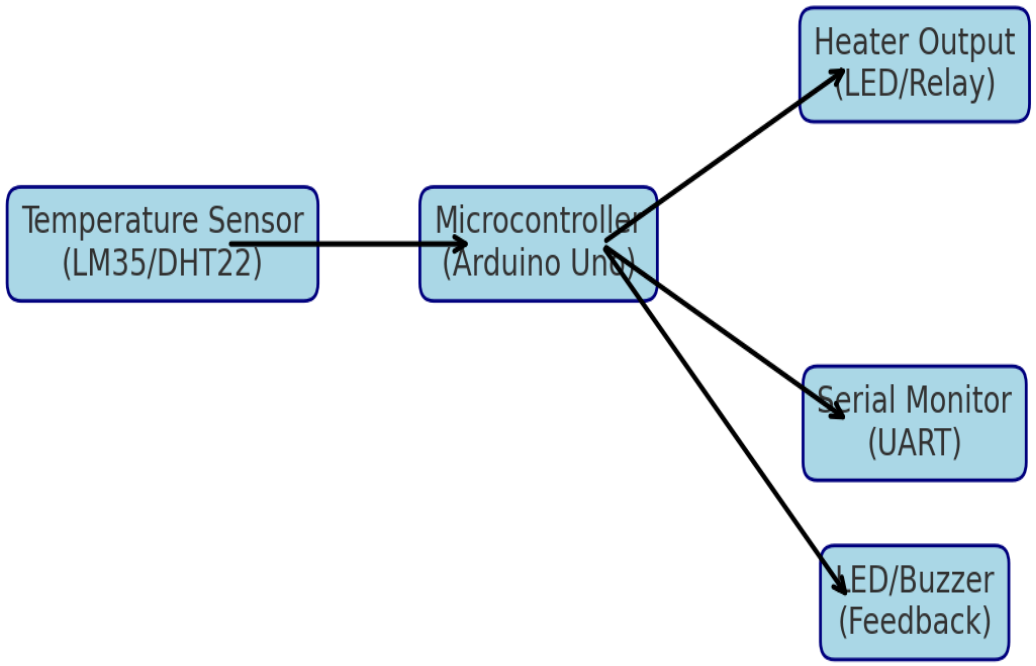
- **Native Support:** Arduino Uno comes with built-in UART over USB.
- **Easy Debugging:** Messages can be viewed using the Serial Monitor.
- **Minimal Hardware:** No external modules needed — just the onboard USB/Serial.
- **Wokwi Compatibility:** Perfectly works with virtual console output in Wokwi simulations.

Use Cases in This Project:

- Log real-time temperature values
- Show current system state (Heating, Stabilizing, Target Reached, Overheat)
- Debug sensor readings and system response

3. BLOCK DIAGRAM OF KEY MODULES

Visual representation of the system architecture enhances understanding of how individual components interact. This block diagram illustrates all functional units and their communication flow.



Modules Explained:

➢ **Temperature Sensor Module:**

- Continuously monitors the room temperature.
- Sends data to the microcontroller.

➢ **Microcontroller (Arduino Uno):**

- The brain of the system.
- Implements logic using FreeRTOS tasks.
- Controls heater ON/OFF via digital output.
- Sends logs to the serial monitor.
- Triggers visual or audio alerts.

➢ **Heater Output (LED):**

- Simulates real heating element.
- Turns ON when temperature is below threshold (e.g., 35°C).
- Turns OFF when target temperature is reached.

➢ **LED/Buzzer (Optional):**

- Provides clear feedback to the user or tester.

➢ **Serial Monitor:**

- Continuously logs system data for visibility and debugging.

4. FUTURE ROADMAP

As embedded systems evolve, it is essential to plan for scalability and safety. Below are enhancements that can transform this basic heater controller into a smart, production-grade system.

Feature	Description
Overheat Protection	System can auto-disable heater if temperature > 70°C to avoid hardware damage.
Multiple Heating Profiles	Support user-configurable modes like Eco, Normal, Turbo.
BLE Support	ESP32 can replace Arduino Uno to broadcast heating state wirelessly.
OLED/LCD Display	Real-time display of temperature and system state.
Mobile App Integration	Remote control and alerts via smartphone using BLE/Wi-Fi.
Touch/Physical Buttons	Add manual control options for mode selection or override.
PID Control	Implement Proportional-Integral-Derivative logic for smoother heating transitions.

5. CONCLUSION

The Basic Heater Control System is a robust and scalable embedded application that demonstrates fundamental control logic, sensor-actuator integration, and task scheduling with FreeRTOS. Designed using an Arduino-based architecture, the system:

- Efficiently reads and processes real-time temperature data
- Simulates heating via output control
- Logs system states via UART for easy debugging
- Supports visual or audio status indicators
- Lays the foundation for smart appliance upgrades