Embedded Systems Intern Assignment - upliance.ai

Assignment: *Build a Basic Heater Control System*

Problem Statement:

Design and implement a simple heater control system using a temperature sensor and an actuating mechanism (simulated heating). The goal is to simulate or build a basic embedded system that turns a "heater" on or off based on temperature thresholds.

Keywords: embedded system, control system, sensors

PART 2- EMBEDDED IMPLEMENTATION

I. System Overview

This project presents a basic heater control system using the Arduino Uno and a DHT22 temperature and humidity sensor. It controls a simulated heater based on real-time temperature readings and incorporates safety features such as overheat detection and visual/audible indicators.

The system uses a temperature-based **state machine** with five states: *Idle, Heating, Stabilizing, Target Reached,* and *Overheat*. Each state triggers specific outputs such as turning the heater (LED) ON or OFF, or activating a buzzer in case of overheating.

II. Platforms and Tools Used

1. Hardware Platform: Arduino Uno
2. Programming Language**:** Arduino C/C++
3. Development Environment: Wokwi Online Simulator
4. Sensor Used: DHT22 (temperature & humidity sensor)
5. Actuators Used: LED (heater indicator), Buzzer (overheat alert)
6. Libraries: Adafruit DHT sensor library

III. Features Implemented

- ✓ 5-State Temperature-Based Heater Logic
- ✓ Continuous sensor reading
- ✓ Serial logging of temperature, humidity, and system state
- ✓ Visual feedback using LED

- ✓ Overheat alert using Buzzer

IV. Simulation and Observations

The simulation was created using Wokwi’s online Arduino UNO environment.

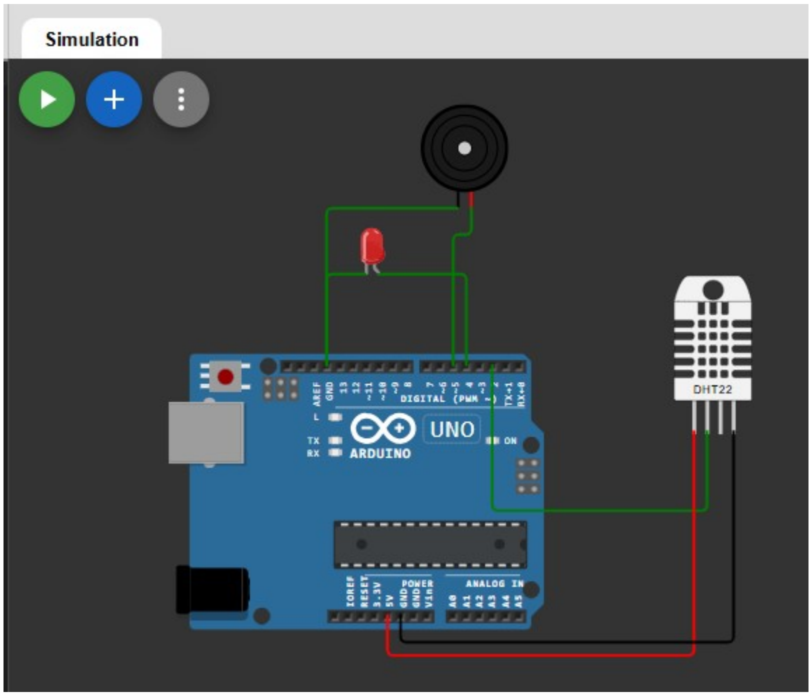
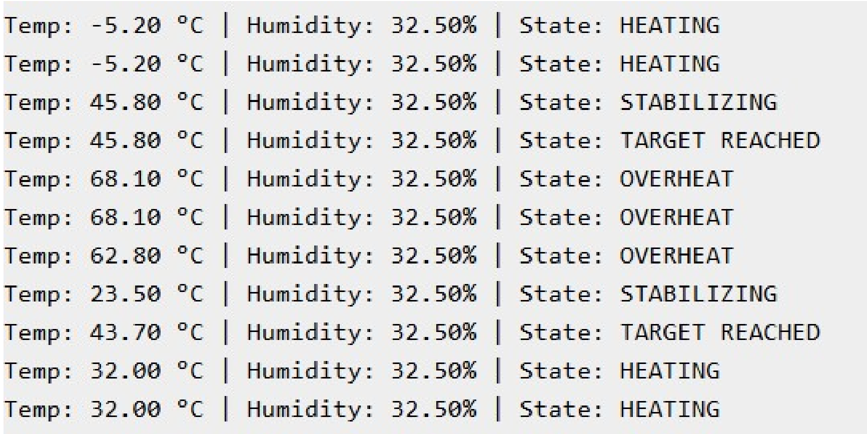


Fig1. Simulation setup of basic heater system

Fig1. illustrates the hardware simulation: Arduino Uno, DHT22 sensor, LED (heater), and buzzer.

- The DHT22 sensor provides real-time temperature and humidity readings, which are used to control outputs.
- LED (D4) indicates heating. Buzzer (D5) alerts when overheat (>50°C) is detected.
- The Serial Monitor logs temperature, humidity, and the current system state every 2 seconds.
- Temperature and humidity can be adjusted using sliders in the DHT22 component to test each state.

All components are powered via Arduino's 5V and GND pins. The temperature slider in the simulation allows testing all state transitions



```
Temp: -5.20 °C | Humidity: 32.50% | State: HEATING
Temp: -5.20 °C | Humidity: 32.50% | State: HEATING
Temp: 45.80 °C | Humidity: 32.50% | State: STABILIZING
Temp: 45.80 °C | Humidity: 32.50% | State: TARGET REACHED
Temp: 68.10 °C | Humidity: 32.50% | State: OVERHEAT
Temp: 68.10 °C | Humidity: 32.50% | State: OVERHEAT
Temp: 62.80 °C | Humidity: 32.50% | State: OVERHEAT
Temp: 23.50 °C | Humidity: 32.50% | State: STABILIZING
Temp: 43.70 °C | Humidity: 32.50% | State: TARGET REACHED
Temp: 32.00 °C | Humidity: 32.50% | State: HEATING
Temp: 32.00 °C | Humidity: 32.50% | State: HEATING
```

Fig3. Output in serial monitor

Fig3. Shows the output observed in serial monitor when the toggle for temperature and humidity in the DHT22 sensor is moved to simulate different temperature and humidity level. Based on these levels the LED glows or the Buzzer indicated buzzing.

Below is the state trigger action table:

State	Trigger Condition	Heater (LED)	Buzzer	Serial Output
Idle	System start or temp ≥ target & < overheat	OFF	OFF	Logs current temp, humidity, state
Heating	Temp < (target - hysteresis)	ON	OFF	"state: heating"
Stabilizing	Temp ≥ (target - hysteresis) and < target or re-entry	ON	OFF	"state: stabilizing"
Target reached	Temp ≥ target and < overheat	OFF	OFF	"state: target reached"
Overheat	Temp ≥ overheat threshold (≥ 50 °c)	OFF	ON	"state: overheat" + alert
Return from overheat	Temp drops below Target	ON	OFF	Transitions to STABILIZING

Target Temperature**:** 40.0°C

Hysteresis: 1.0°C

Overheat Limit: 50.0°C

V. Conclusion

The implemented heater control system successfully demonstrates real-time temperature monitoring, multi-state decision-making, and actuation using an Arduino Uno and DHT22 sensor. By leveraging a clear 5-state logic, the system manages heater behavior effectively, ensuring both comfort and safety.

Core functionalities like continuous sensor reading, serial logging, and actuator control (LED and buzzer) were implemented and verified through simulation. While bonus features like BLE and multitasking (via RTOS) were not implemented on Arduino Uno, they are acknowledged and proposed for future versions using more advanced microcontrollers like the ESP32.

This project lays the groundwork for scalable, smart heating solutions in low-cost embedded environments.