Heater Control System

Candidate Name: Suraj Naiku Satwar

Platform: Arduino Uno (Simulated on Wokwi)

Language: Embedded C

Sensor: DHT22 (Real ambient temperature sensing)

UI: 16x2 LCD with I²C interface

1. Problem Statement

Design and implement a basic heater control system that reads ambient temperature using a digital sensor, controls a simulated heater, and provides feedback via an LCD and buzzer. The system must track and display one of five operational states: Idle, Heating, Stabilizing, Target Reached, and Overheat.

2. Minimum Sensors Required

- **DHT22** – Digital temperature (and humidity) sensor
    - Uses a proprietary single-wire communication protocol
    - Provides real-time ambient temperature readings
    - Suitable for low-frequency periodic measurements

3. Communication Protocol

DHT22 → Microcontroller

The DHT22 communicates with the microcontroller using a **proprietary single-wire digital protocol**. This is not a standard protocol like I²C, SPI, or UART, but is widely supported via libraries such as DHT.h.

Justification:

1. Simple Wiring: Only one data line is required, reducing the number of GPIO pins used.
2. Library Support: Easily integrated with Arduino using the DHT.h library, which manages the strict timing requirements of this protocol.

3. No ADC Needed: Unlike analog sensors (e.g., LM35), DHT22 gives digital temperature values directly, which simplifies programming.
4. Reliable for Low-Speed Applications: It doesn't require high-speed data, making it ideal for ambient temperature monitoring.
5. Widely Used in Embedded Systems: It is a standard beginner-friendly sensor for learning sensor integration and digital communication.

LCD 16x2 → Microcontroller

The LCD is interfaced using an **I²C (Inter-Integrated Circuit)** protocol via an I²C backpack module.

Justification:

1. Pin Efficiency: I²C requires only two pins (SDA and SCL), compared to 6–8 digital pins in traditional parallel LCD interfacing. This leaves more GPIOs free for other components.
2. Multiple Device Support: I²C supports connecting multiple devices (e.g., other sensors or displays) on the same bus without additional pins.
3. Stable Communication: Data transmission is synchronized with a clock signal, which improves accuracy over longer distances and avoids timing issues common in one-wire LCDs.
4. Standardized Protocol: I²C is widely used in industry and academia. It is supported by most microcontrollers and development boards.
5. Readability & Reliability: Libraries like LiquidCrystal_I2C.h provide stable performance and easy-to-use APIs.

In this project, using I²C simplifies LCD integration and makes the system modular and scalable for future enhancements.

4. Block Diagram

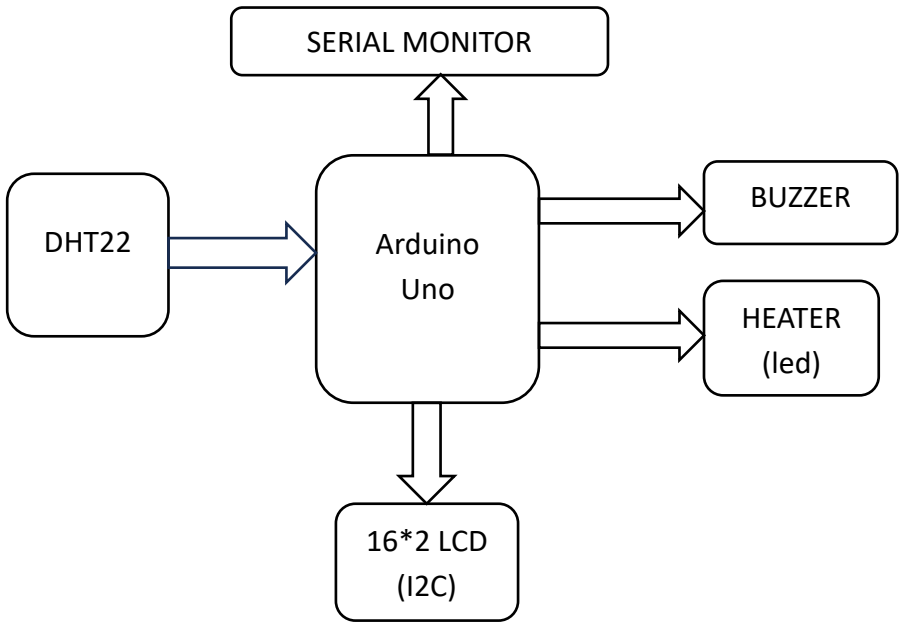

Fig: Block Diagram

System States & Logic

State	Condition	Heater (LED)	Buzzer	LCD Display
Idle	Startup/init	OFF	OFF	“IDLE”
Heating	Temp < Target − 2°C	ON	OFF	“HEATING”
Stabilizing	Target − 2°C ≤ Temp < Target	ON	OFF	“STABILIZING”
Target Reached	Target ≤ Temp < Overheat	OFF	OFF	“TARGET OK”
Overheat	Temp ≥ Overheat threshold	OFF	ON	“OVERHEAT!!”

5. Future Roadmap

❖ **Overheating Protection**

- Add a second temperature sensor for redundancy
- Use a relay for emergency heater cutoff
- Auto cool-down cycle before reactivation

❖ **Multiple Heating Profiles**

- Add buttons or LCD menu to select profiles (Low/Med/High)
- Store user preferences using EEPROM
- Control profiles remotely via BLE/Wi-Fi and mobile app

❖ **Advanced Features**

- Use PID control for stable temperature
- Add a graphical LCD or touchscreen UI
- Integrate Free RTOS for multitasking
- Enable remote alerts and cloud data logging

- Simulation Link

https://wokwi.com/projects/437174775546135553

- Github Link

https://github.com/SurajSatwar/Simple-Heater-control