Heater Control System - Design Document

1. System Overview

The heater control system is designed to automatically control heating based on real-time temperature feedback. It uses a temperature sensor (DS18B20) and a microcontroller (ESP32) to turn the heater ON or OFF depending on predefined temperature thresholds. A state machine is implemented to manage system behavior and ensure safety during operation.

2. Minimum Sensors Required

Primary Sensor
- DS18B20 Temperature Sensor
  - Range: -55ÁC to +125ÁC
  - Accuracy: -0.5ÁC
  - Interface: 1-Wire protocol
  - Purpose: To measure ambient temperature and help control the heater

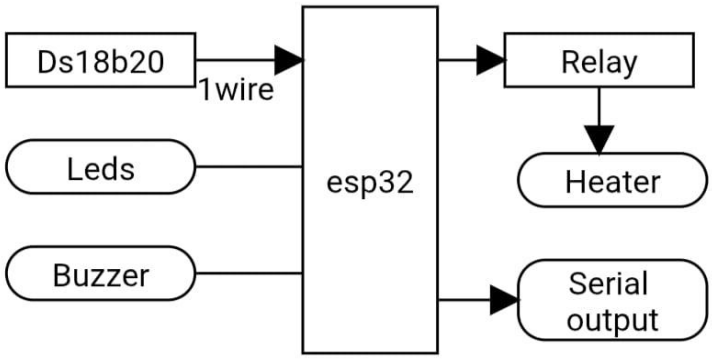
Optional Sensors for Future Enhancements
- Secondary Temperature Sensor - Redundancy in case the main sensor fails
- Humidity Sensor (e.g., DHT22) - Useful for environmental monitoring
- Current Sensor - To monitor heater power and detect electrical issues

3. Communication Protocol

Recommended Protocol: 1-Wire

Justification:
- The DS18B20 sensor uses the 1-Wire protocol by default
- Requires only one data pin, which reduces wiring complexity
- Supports multiple devices on a single wire, allowing future expansion
- Well-supported in Arduino and ESP32 environments

4. System Block Diagram



5. System States (State Machine)

IDLE - Heater OFF, monitoring temperature
HEATING - Heater ON, trying to reach target temperature
STABILIZING - Heater ON, waiting to confirm stable temperature
TARGET_REACHED - Heater OFF, desired temp achieved
OVERHEAT - Heater OFF, buzzer and warning activated

State Transitions:
- IDLE - HEATING: Temp < Target - Hysteresis
- HEATING - STABILIZING: Temp >= Target
- STABILIZING - TARGET_REACHED: Temp remains within Target ± Hysteresis for 30 sec
- TARGET_REACHED - HEATING: Temp drops below Target - Hysteresis
- Any State - OVERHEAT: Temp > Safety Threshold (42ÁC)

6. Key System Parameters

Target Temp	:35ÁC
Hysteresis	:2ÁC
Overheat Threshold	:42ÁC
Sampling Rate	:1 Hz
Stabilization Time	:30s

7. Future Roadmap

As the system matures, the following features can be considered:

Enhanced Safety
- Dual temperature sensors
- Manual reset after overheat

Real_Time Control
- FreeRTOS for multitasking

Advanced Control
- Multiple heating profiles
- Schedules and eco modes

Smart Connectivity
- BLE/MQTT for remote control
- Google Home/Alexa integration

Machine Learning
- Predictive heating and energy optimization

8. Hardware Prototype

This system was tested both in simulation and on physical hardware.
Wokwi Simulation: https://wokwi.com/projects/436891097299493889
Prototype Demo Video: https://drive.google.com/file/d/1sPjoh1X56OnBt9ygG4l6JknpRuhZzQ5J/view?usp=drivesdk
GitHub Repository: https://github.com/KOWSHIK8T/heater-control-system-esp32

Watch the hardware demo. The image below shows the working prototype, built using an ESP32 microcontroller, a DS18B20 temperature sensor, and supporting components such as LEDs and a buzzer for feedback.

Prototype Photo: