Heater Control System Design Document

1. Project Overview

This project is a simulation of a temperature-based heater control system. It uses an NTC temperature sensor to monitor real-time temperature and responds by controlling a simulated heater, RGB LED, and buzzer to indicate different states of the system such as Idle, Heating, Stabilizing, Target Reached, and Overheat. The system is implemented using Arduino Uno on the Wokwi simulation platform.

2. Minimum Sensors & Actuators

- NTC Temperature Sensor: Measures temperature and provides analog voltage to microcontroller.
- RGB LED: Visual feedback for states using different colors.
- Buzzer: Audio feedback, alerts in case of Overheat.
- Simulated Heater: Controlled via digital output.

3. Communication Protocol

No external communication protocol is used in this simulation. Serial communication is used for logging and debugging.

4. Hardware / Wiring

| Part | Pin |

| NTC sensor OUT | **A0** |

| NTC sensor VCC / GND| 5 V / GND |

| RGB LED **Red** | **D8** |

| RGB LED **Green** | **D7** |

| RGB LED **Blue** | **D6** |

| Buzzer (+) | **D9** |

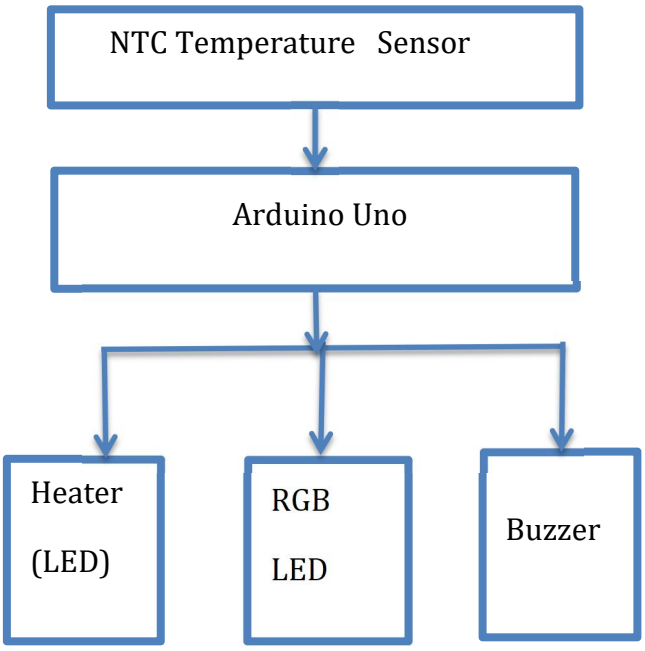
| Buzzer (–) | GND |

| Heater LED / relay | **D10** |

| Common LED cathode | GND |

| Current-limit resistors | 220 Ω (one per LED channel) |

5. Block Diagram



6. Future Roadmap

- Implement BLE communication to broadcast current heating state to external devices.
- Add support for FreeRTOS to manage multiple tasks concurrently.
- Support multiple heating profiles.
- Implement overheating protection shutdown with logging.

7. System States & Behaviour

State	Temperature Range (°C)	LED Color	Buzzer
Idle	< 25°C	OFF	OFF
Heating	25°C - 45°C	GREEN	OFF
Stabilizing	45°C - 48°C	GREEN	OFF
Target Reached	48°C - 50°C	BLUE	OFF
Overheat	> 50°C	RED	ON (continuous)

8. Embedded C Code (Arduino)

```
#define RED_PIN    8
#define GREEN_PIN  7
#define BLUE_PIN   6
#define BUZZER_PIN 9
#define HEATER_PIN 10
#define NTC_PIN    A0

int temperature = 0;
String state = "Idle";

void setup( ) {
  Serial.begin(9600);
  pinMode(HEATER_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(RED_PIN, OUTPUT);
  pinMode(GREEN_PIN, OUTPUT);
  pinMode(BLUE_PIN, OUTPUT);
  digitalWrite(HEATER_PIN, LOW);
  noTone(BUZZER_PIN);
  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, LOW);
  digitalWrite(BLUE_PIN, LOW);
}

void setLEDColor(bool red, bool green, bool blue) {
  digitalWrite(RED_PIN, red);
  digitalWrite(GREEN_PIN, green);
  digitalWrite(BLUE_PIN, blue);
}

void loop() {
  int analogVal = analogRead(NTC_PIN);
  temperature = map(analogVal, 1000, 100, -24, 80);

  if (temperature < 25) {
    state = "Idle";
    digitalWrite(HEATER_PIN, LOW);
    noTone(BUZZER_PIN);
    setLEDColor(LOW, LOW, LOW);
  } else if (temperature >= 25 && temperature < 45) {
    state = "Heating";
    digitalWrite(HEATER_PIN, HIGH);
     noTone(BUZZER_PIN);
    setLEDColor(LOW, HIGH, LOW);
  } else if (temperature >= 45 && temperature < 48) {
    state = "Stabilizing";
    digitalWrite(HEATER_PIN, HIGH);
    noTone(BUZZER_PIN);
```

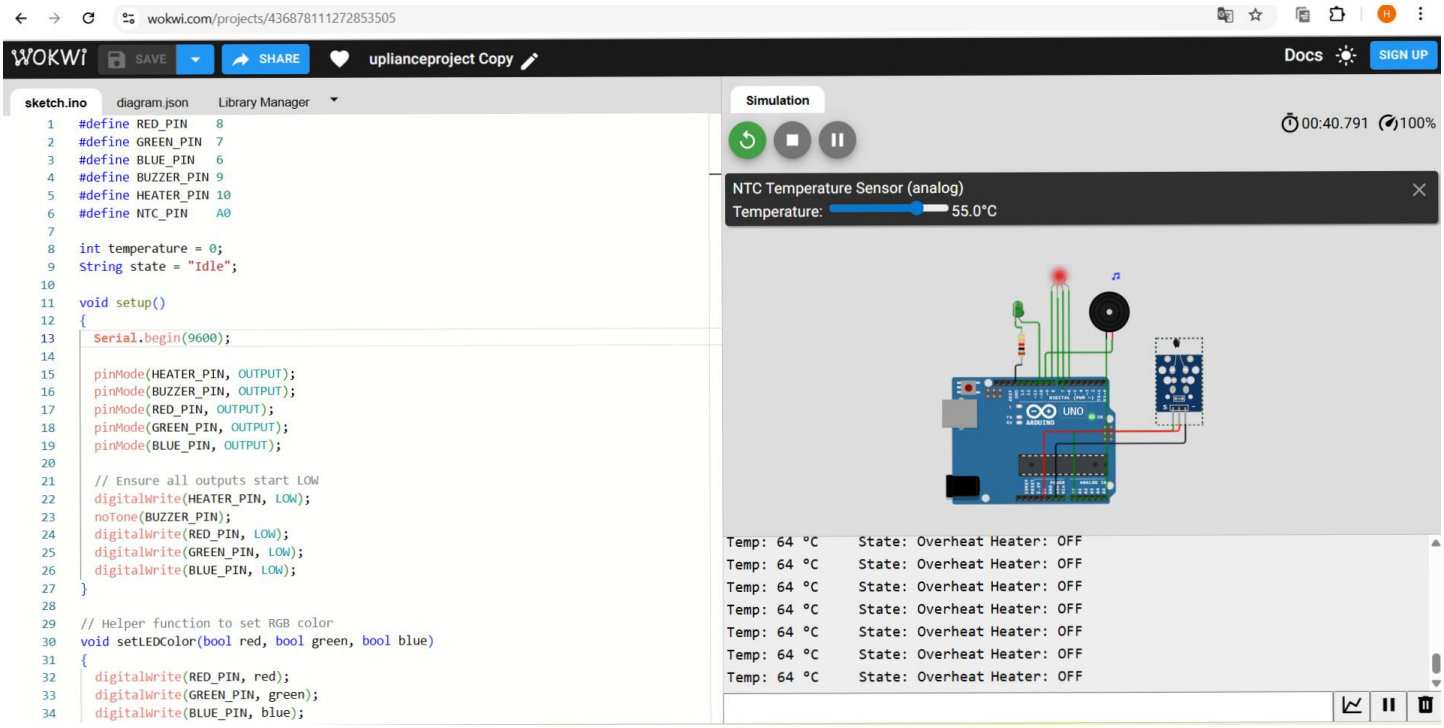
```
    setLEDColor(LOW, HIGH, LOW);
  } else if (temperature >= 48 && temperature <= 50) {
    state = "Target Reached";
    digitalWrite(HEATER_PIN, LOW);
    noTone(BUZZER_PIN);
    setLEDColor(LOW, LOW, HIGH);
  } else {
    state = "Overheat";
    digitalWrite(HEATER_PIN, LOW);
    setLEDColor(HIGH, LOW, LOW);
    tone(BUZZER_PIN, 1000);
  }

  Serial.print("Temp: ");
  Serial.print(temperature);
  Serial.print(" °C\tState: ");
  Serial.print(state);
  Serial.print("\tHeater: ");
  Serial.println(digitalRead(HEATER_PIN) ? "ON" : "OFF");

  delay(1000);
}
```

9. Running the Simulation

To test the Heater Control System on the Wokwi simulation platform, open the project and ensure all components—NTC temperature sensor, RGB LED, buzzer, and simulated heater—are connected as defined in the wokwi-diagram.json file(https://wokwi.com/projects/436878111272853505). Start the simulation by clicking the green play button.

To simulate different temperatures, One-click on the NTC temperature sensor—this opens a slider that lets you adjust the temperature from –24 °C to 80 °C. As the temperature changes, observe the RGB LED responding with different colors that reflect the current system state: green for Heating and Stabilizing, blue for Target Reached, red for Overheat, and off for Idle. The buzzer provides a continuous sound as an alert when the system enters the Overheat state.

The simulated heater (connected to pin 10) turns on during heating and stabilizing phases, and turns off when the system is idle, has reached the target temperature, or detects overheating. Meanwhile, the Serial Monitor continuously logs the current temperature, the active system state, and heater status every second. This simulation replicates the full behavior of a physical heater control system without requiring any actual hardware.



Fig(1): Simulation Sample image

10.Future Roadmap

The current simulation is implemented using Wokwi with Arduino Uno, which is limited to basic GPIO and Serial communication. However, if implemented on physical hardware like an ESP32, the system could be enhanced with additional real-world features:

BLE Advertising: Use Bluetooth Low Energy (BLE) to broadcast the current heating state to a smartphone app or nearby devices for remote monitoring.

Real-Time Visual & Audio Feedback: Maintain the RGB LED and buzzer indicators for state changes and overheat alerts.

Use of FreeRTOS or Timer Interrupts: Replace the delay()-based loop with FreeRTOS tasks or hardware timer interrupts for efficient multitasking and responsiveness.

Note: Wokwi currently does not support BLE or real-time OS simulation. These features are feasible and recommended for physical implementation with compatible microcontrollers (e.g., ESP32).

11.Submission links

Wokwi simulation link: https://wokwi.com/projects/436878111272853505

Code Repository (ZIP / GitHub): https://github.com/hemanthkumarjn05/heater-control-system.git