Project Report: Smart Temperature-Based Fan and LED Control System

Project Title :

Smart Temperature-Based Fan and LED Control System

Objective :

The primary objective of this project is to design a system that automatically controls a fan and RGB LED based on the surrounding temperature. It features a display to show the current temperature and fan speed, with a manual override option for the fan. Additionally, a buzzer is activated when the temperature exceeds a certain threshold, indicating dangerously high temperatures

Components Used :

Hardware:

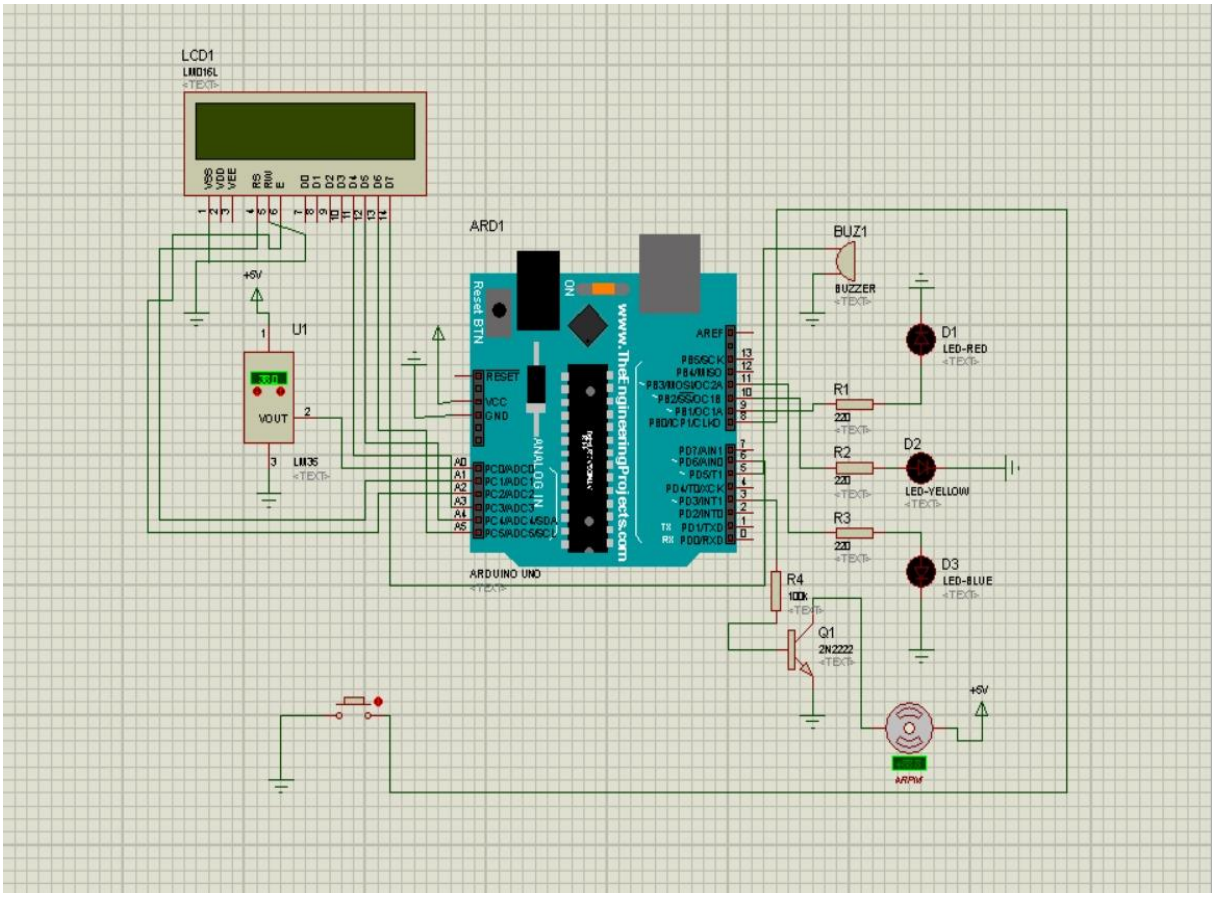
Arduino Uno (or compatible microcontroller)

- *LM35 Temperature Sensor*
- *RGB LED (or separate red, yellow, and blue LEDs)*
- *DC Fan (low-power 5V)*
- *Buzzer*
- *16x2 LCD Display*
- *Resistors (220Ω for LEDs, 1kΩ for fan transistor)*
- *NPN Transistor (e.g., 2N2222) for fan control*
- *Push Button (for manual fan override)*
- *Power Supply (for Arduino and fan)*

Software:

1. *Arduino IDE for programming*
2. *Proteus for circuit simulation*

Circuit Diagram :



The circuit consists of:

- *The LM35 sensor connected to the analog pin A0 of the Arduino.*
- *RGB LED connected to pins 9 (red), 10 (yellow), and 11 (blue).*
- *The DC fan is connected via a transistor to pin 3 (PWM control).*
- *Buzzer connected to pin 5.*
- *16x2 LCD connected to pins A1 (RS), A2 (E), and A3-A5 (D4-D6).*
- *Push button connected to pin 8 with a pull-up resistor.*

This circuit can be simulated in Proteus using the appropriate components.

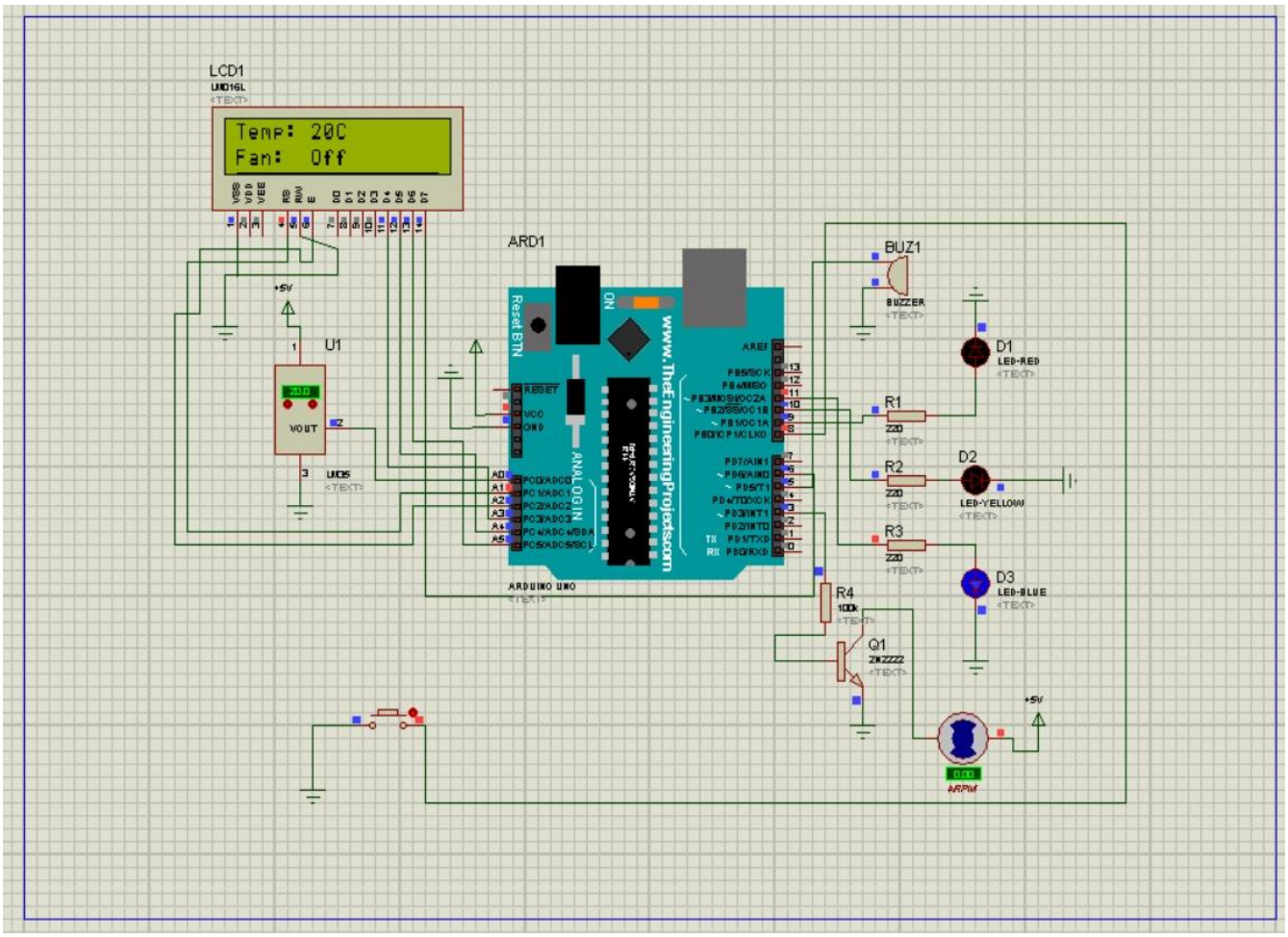
Working Principle :

This system uses an LM35 temperature sensor to measure the ambient temperature and controls the speed of a fan and the colour of an RGB LED based on the temperature readings.

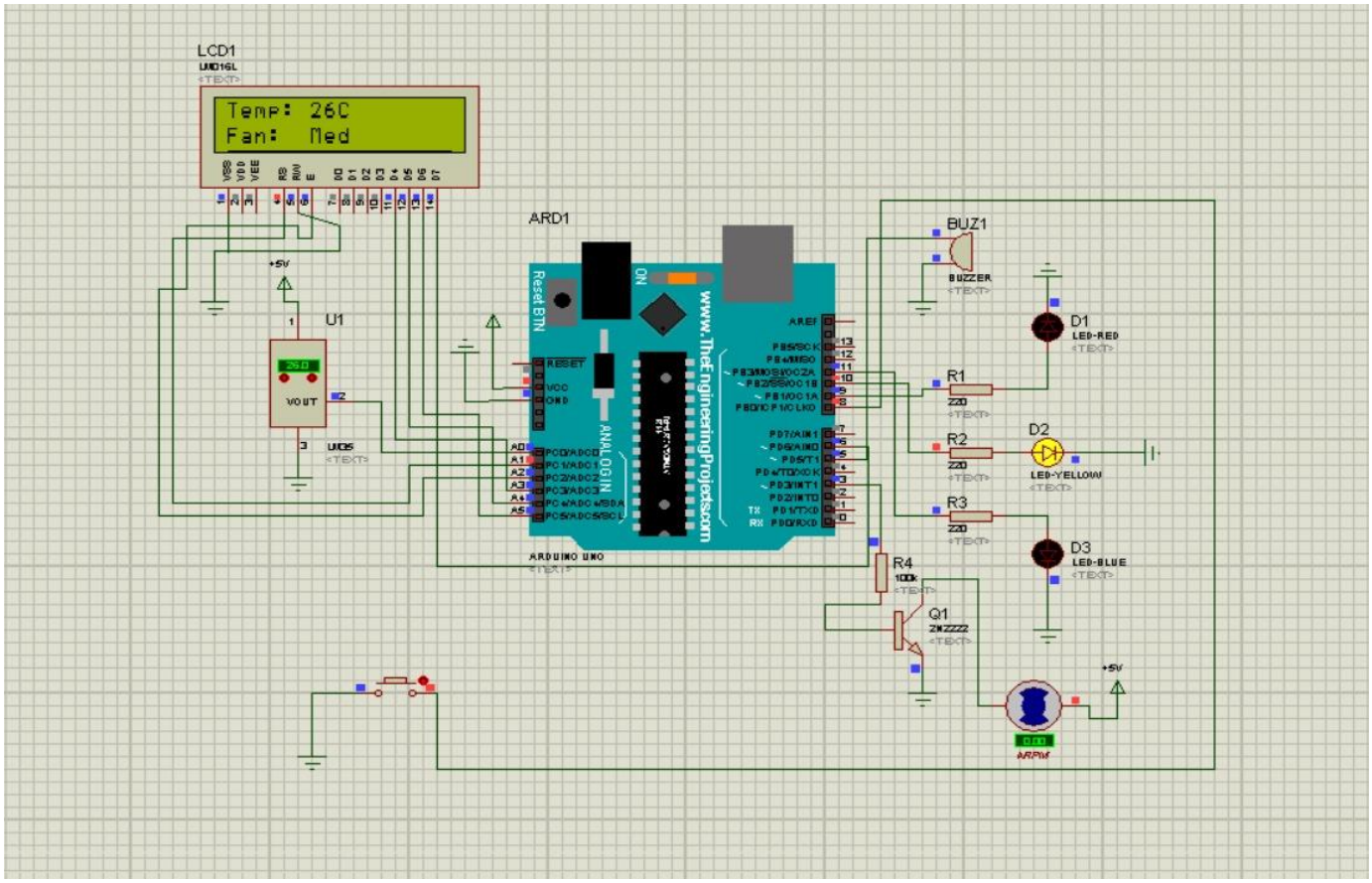
Temperature Measurement: *The LM35 sensor outputs a voltage that is linearly proportional to the temperature in °C (10mV per degree Celsius). The Arduino reads this analog voltage and converts it into a temperature reading*

Fan Control: *Based on the temperature, the fan speed is adjusted using PWM (Pulse Width Modulation). The system categorizes temperatures into three ranges:*

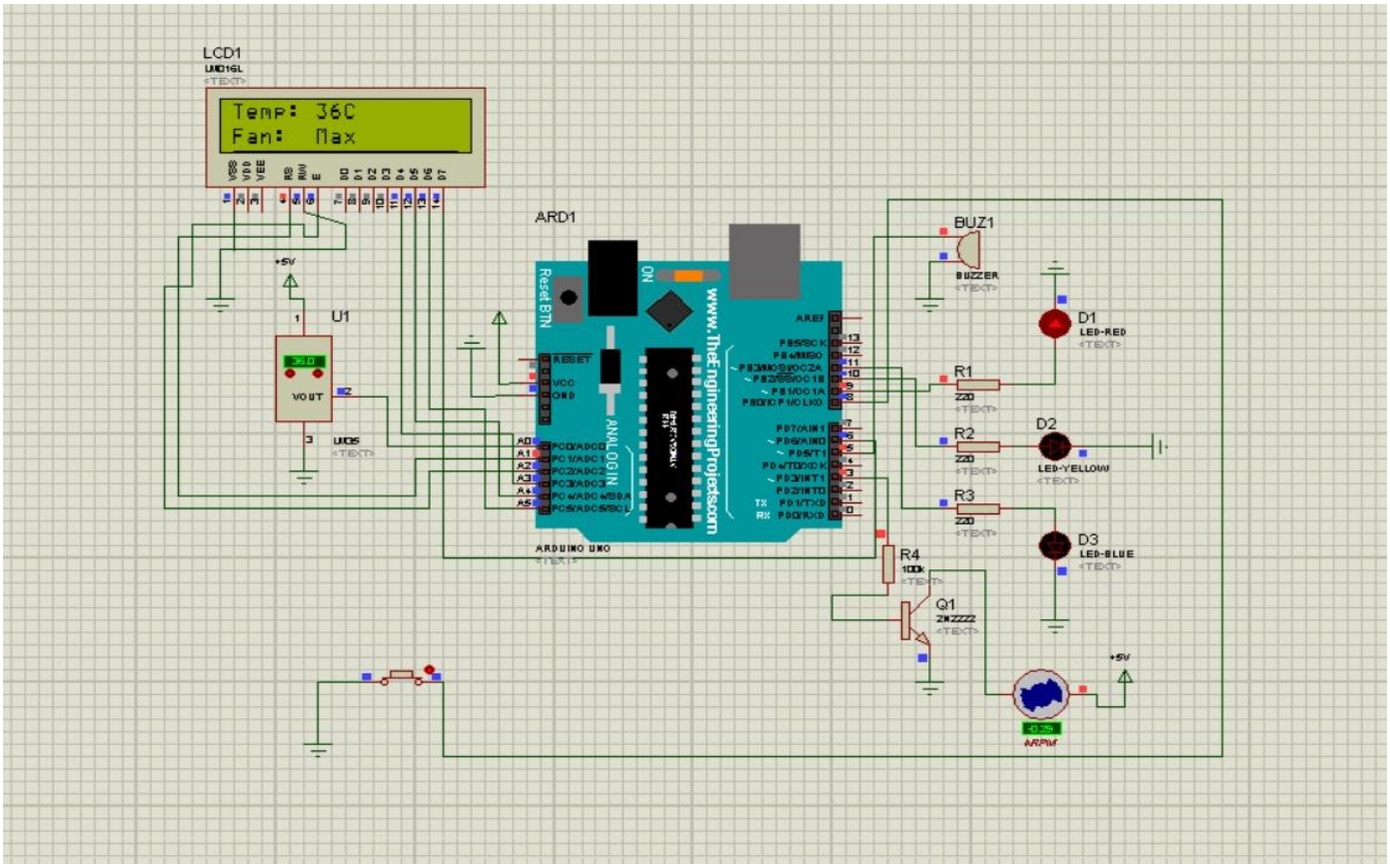
At Cool (temperature < 25°C): *Fan off, blue LED on.*



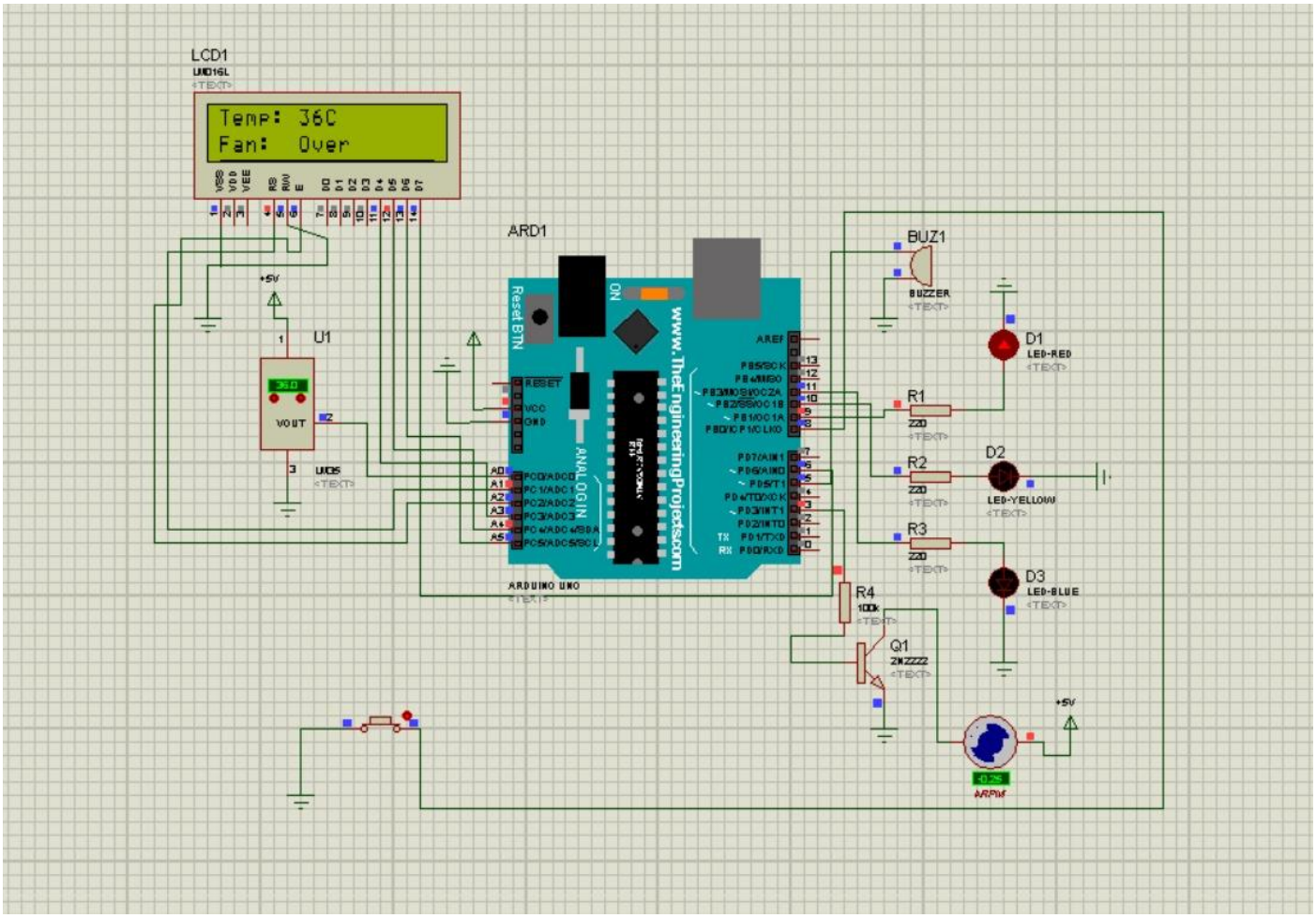
At moderate (25°C ≤ temperature ≤ 35°C): *Fan at medium speed, yellow LED on*



At Hot (temperature > 35°C): *Fan at maximum speed, red LED on, and a buzzer is activated as a high-temperature warning.*



Manual Override: *A push button is provided to manually override the system, forcing the fan to run at maximum speed regardless of the temperature.*



LCD Display: *The 16x2 LCD shows the current temperature and fan status (Off, Med, Max, or Over for override mode).*

Code/Logic :

```
#include <LiquidCrystal.h>

// Pin assignments
const int LM35Pin = A0;
const int fanPin = 3;
const int buzzerPin = 5;
const int redLEDPin = 9;
const int yellowLEDPin = 10;
const int blueLEDPin = 11;
const int buttonPin = 8;

// LCD pin assignments
LiquidCrystal lcd(A1, A2, A3, A4, A5, 6);
int temperatureC;
```

```
int buttonState = 0;
bool fanOverride = false;

void setup() {
  pinMode(fanPin, OUTPUT);
  pinMode(buzzerPin, OUTPUT);
  pinMode(redLEDPin, OUTPUT);
  pinMode(yellowLEDPin, OUTPUT);
  pinMode(blueLEDPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);

  lcd.begin(16, 2);
  lcd.print("Temp: ");
  lcd.setCursor(0, 1);
  lcd.print("Fan: ");
}

void loop() {

  // Correct formula to read temperature from LM35

  float voltage = analogRead(LM35Pin) * (5.0 / 1023.0
  temperatureC = voltage * 100;

  // Display temperature on LCD
  lcd.setCursor(6, 0);
  lcd.print(temperatureC);
  lcd.print("C  ");

  // Read the push button state
  buttonState = digitalRead(buttonPin);

  if (buttonState == LOW) {
// Toggle fan override when button is pressed
    fanOverride = !fanOverride;
    delay(300);  // Debounce delay
  }

  // Control fan and LEDs based on temperature
  if (!fanOverride) {
    if (temperatureC < 25) {
      analogWrite(fanPin, 0);
      digitalWrite(redLEDPin, LOW);
      digitalWrite(yellowLEDPin, LOW);
      digitalWrite(blueLEDPin, HIGH);
      lcd.setCursor(6, 1);
      lcd.print("Off ");
    } else if (temperatureC >= 25 && temperatureC <= 35) {
      analogWrite(fanPin, 128);
      digitalWrite(redLEDPin, LOW);
      digitalWrite(yellowLEDPin, HIGH);
      digitalWrite(blueLEDPin, LOW);
```

```
    lcd.setCursor(6, 1);
    lcd.print("Med ");
   } else if (temperatureC > 35) {
    analogWrite(fanPin, 255);
    digitalWrite(redLEDPin, HIGH);
    digitalWrite(yellowLEDPin, LOW);
    digitalWrite(blueLEDPin, LOW);
    lcd.setCursor(6, 1);
    lcd.print("Max ");

    // Activate buzzer when temperature is too high
    digitalWrite(buzzerPin, HIGH);
    delay(200);
    digitalWrite(buzzerPin, LOW);
   }
  } else {
   // Fan override mode (manual fan control)
   analogWrite(fanPin, 255);
   digitalWrite(redLEDPin, HIGH);
   digitalWrite(yellowLEDPin, LOW);
   digitalWrite(blueLEDPin, LOW);
   lcd.setCursor(6, 1);
   lcd.print("Over");
  }

  delay(500);
}
```

Results and Conclusion :

The Smart Temperature-Based Fan and LED Control System successfully adjusts the fan speed based on the temperature, provides clear visual feedback using RGB LEDs, and displays real-time temperature and fan status on the LCD. The manual fan override feature works reliably, allowing for user control when needed. This system is ideal for applications where temperature regulation is critical, providing both automation and manual control.