**Submitted by : Priyanka.K [2448347]**

**Lab 1 16.06.2025**

**1. Aim and Objective**

**Aim:**  
To utilize environmental sensors for real-time monitoring of temperature within a room, and to implement responsive actions based on sensor readings.

**Objectives:**

* **Interface temperature sensors with a microcontroller.**
* **Process sensor data to trigger visual (LED color) and auditory (speaker alarm) outputs.**
* **Display real-time sensor readings and system status via the serial monitor for user feedback.**

**2. Problem Statement**

Design and implement a system that continuously measures the temperature in a room. When certain thresholds are reached, the system should automatically change the color of an LED to indicate environmental changes and activate a speaker as an alarm. Additionally, sensor values and system responses should be displayed in the serial monitor for real-time monitoring and debugging.

**3. Code**

#include <Adafruit\_LiquidCrystal.h>

const int tempPin = A0;

// RGB LED pins

const int redPin = 9;

const int greenPin = 10;

const int bluePin = 11;

// Buzzer pin

const int buzzerPin = 8;

// Temperature threshold

const float tempThreshold = 37.0;

Adafruit\_LiquidCrystal lcd\_1(0);

void setup() {

pinMode(tempPin, INPUT);

pinMode(redPin, OUTPUT);

pinMode(greenPin, OUTPUT);

pinMode(bluePin, OUTPUT);

pinMode(buzzerPin, OUTPUT);

Serial.begin(9600);

lcd\_1.begin(16, 2);

// Startup LED test

digitalWrite(redPin, HIGH);

digitalWrite(greenPin, HIGH);

digitalWrite(bluePin, HIGH);

delay(1000);

digitalWrite(redPin, LOW);

digitalWrite(greenPin, LOW);

digitalWrite(bluePin, LOW);

lcd\_1.setCursor(0, 0);

lcd\_1.print("Temp Monitor Ready");

delay(2000);

lcd\_1.clear();

}

void loop() {

// Read and convert TMP36 analog value to Celsius

int analogVal = analogRead(tempPin);

float voltage = analogVal \* (5.0 / 1023.0);

float temp = (voltage - 0.5) \* 100.0;

// Serial output

Serial.print("Temperature: ");

Serial.print(temp);

Serial.print((char)176);

Serial.println("C");

// LCD Line 1: show temperature

lcd\_1.setCursor(0, 0);

lcd\_1.print("Temp: ");

lcd\_1.print(temp, 1);

lcd\_1.print((char)223);

lcd\_1.print("C "); // clear extra chars

// Conditions and LED + LCD display

if (temp >= tempThreshold) {

// High temp: Red + buzzer alert

digitalWrite(redPin, HIGH);

digitalWrite(greenPin, LOW);

digitalWrite(bluePin, LOW);

lcd\_1.setCursor(0, 1);

lcd\_1.print("ALARM! Overheat ");

// Buzzer with pulsing tone

for (int i = 0; i < 3; i++) {

tone(buzzerPin, 1000);

delay(500);

noTone(buzzerPin);

delay(500);

}

} else if (temp >= 30.0 && temp < tempThreshold) {

// Safe: Green ON

digitalWrite(redPin, LOW);

digitalWrite(greenPin, HIGH);

digitalWrite(bluePin, LOW);

noTone(buzzerPin);

lcd\_1.setCursor(0, 1);

lcd\_1.print("Status: Safe ");

} else {

// Cool: Blue ON

digitalWrite(redPin, LOW);

digitalWrite(greenPin, LOW);

digitalWrite(bluePin, HIGH);

noTone(buzzerPin);

lcd\_1.setCursor(0, 1);

lcd\_1.print("Status: Cool ");

}

delay(1000);

}

Code Summary :

| **Temperature** | **LED** | **LCD Row 2** | **Buzzer** |
| --- | --- | --- | --- |
| < 30°C | Blue | Status: Cool | OFF |
| 30–36.9°C | Green | Status: Safe | OFF |
| >= 37°C | Red | ALARM! Overheat | Pulsing ON |

**4. Circuits – System Design**

**Set up :**

| **Component** | **Quantity** |
| --- | --- |
| Arduino Uno/Nano | 1 |
| TMP36 Temperature Sensor | 1 |
| RGB LED (Common Cathode) | 1 |
| 16x2 LCD with I2C backpack | 1 |
| Buzzer (5V) | 1 |
| Resistors (220–330Ω) | 3–4 |
| Breadboard + Jumper Wires | 1 set |

### **Pin Connections :**

#### **Temperature Sensor (TMP36)**:

| **TMP36 Pin** | **Connected to** |
| --- | --- |
| Left (Vs) | 5V (Vcc) |
| Middle (Vout) | A0 |
| Right (GND) | GND |

#### **RGB LED** (Common Cathode):

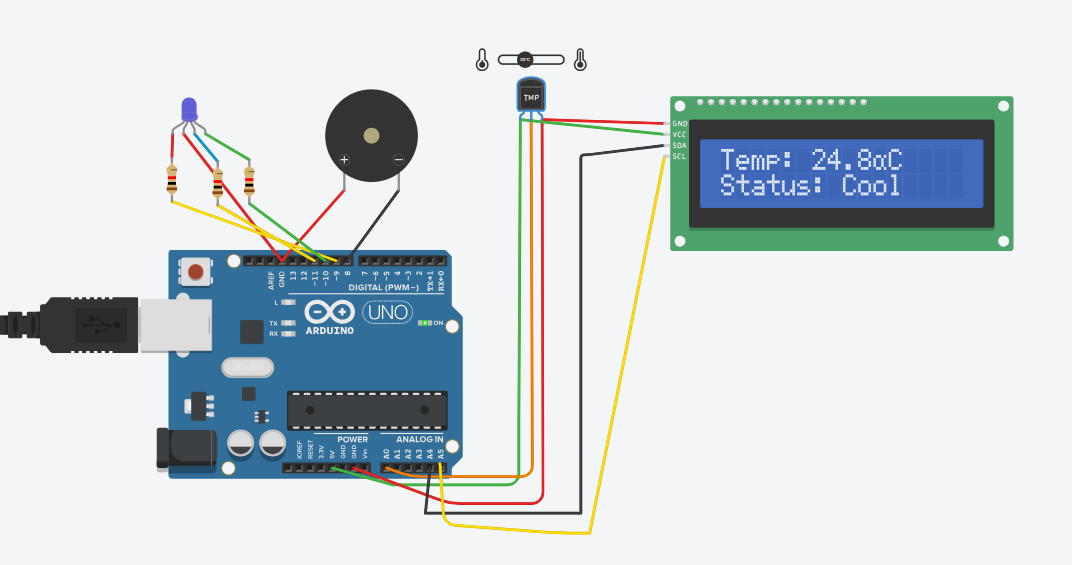
| **RGB LED Pin** | **Connected to** | **Notes** |
| --- | --- | --- |
| Red (R) | Pin 9 | With 220Ω resistor |
| Green (G) | Pin 10 | With 220Ω resistor |
| Blue (B) | Pin 11 | With 220Ω resistor |
| Cathode (Common) | GND | Connect directly to ground |

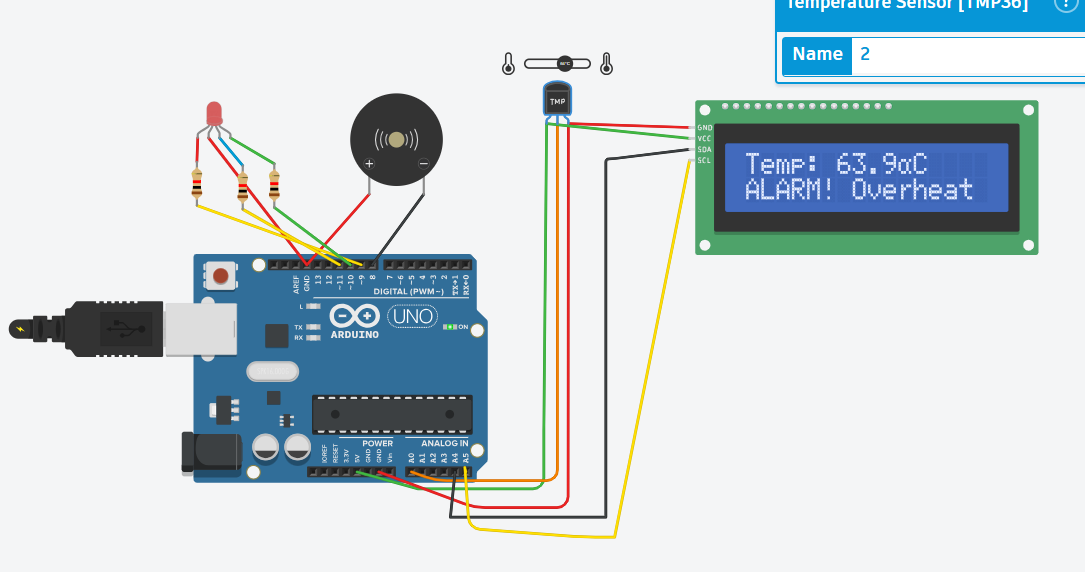
#### **Buzzer**:

| **Pin** | **Connected to** |
| --- | --- |
| +ve | Pin 8 |
| -ve | GND |

#### **16x2 LCD with I2C** (Adafruit\_LiquidCrystal):

| **LCD Pin** | **Arduino Pin** |
| --- | --- |
| SDA | A4 |
| SCL | A5 |
| VCC | 5V |
| GND | GND |





**5. Sample Output**

Temperature: 57.04°C

Temperature: -33.87°C

Temperature: -40.22°C

Temperature: -40.22°C

Temperature: -40.22°C

Temperature: -40.22°C

Temperature: -40.22°C

Temperature: -23.12°C

Temperature: 28.20°C

Temperature: 57.04°C

Temperature: 23.80°C

Temperature: 20.87°C

Temperature: -23.12°C

Temperature: -23.12°C

Temperature: -23.12°C

Temperature: -23.12°C

Temperature: 50.20°C

Temperature: 82.94°C

**6. Challenges**

* **Sensor Calibration:** Ensuring accurate readings from the temperature sensor.
* **Interfacing Multiple Outputs:** Managing LED color changes and speaker activation simultaneously.
* **Real-time Feedback:** Displaying updated sensor values and system status in the serial monitor.
* **Threshold Setting:** Determining appropriate thresholds for triggering alarms.

**7. Application Design**

The system consists of a microcontroller (e.g., Arduino) connected to a temperature sensor an RGB LED, and a buzzer. The microcontroller reads sensor data, processes it, and controls the LED and buzzer based on predefined thresholds. The serial monitor provides real-time feedback for monitoring and troubleshooting.

**8. Reflection**

This lab provided valuable hands-on experience in interfacing sensors with microcontrollers and designing responsive systems. Challenges such as sensor calibration and managing multiple outputs were addressed through careful design and testing. The project demonstrates the practical application of sensor-based monitoring and automation in real-world scenarios.