1. PURPOSE AND REQUIREMENTS SPECIFICATIONS

1.1. Purpose :-

Develop a smart temperature control solution using IoT technology to simulate and display appropriate actions such as turning on air conditioning or fans when simulated room temperatures fall below or exceed desired thresholds, ensuring simulated comfort conditions. Additionally, integrate functionality to prompt users to turn on heating when simulated temperatures are too low, enhancing simulated room comfort and usability.

1.2. Behavior :-

This Smart Temperature Control System For Homes will continuously monitor both internal room temperature and external temperature. It will analyze the temperature data in real-time and display prompts for user action when the room temperature falls below or exceeds preset comfort thresholds. The system will generate notifications in three different ways: through a buzzer, an LCD display, and LED indicators.

- **When the room temperature exceeds the upper threshold** (e.g., too hot):
    - The LCD display will show a message prompting the user to turn on the air conditioner or fan.
    - The system will automatically turn on the air conditioner if the temperature exceeds the preset threshold.
    - The buzzer will emit a sound to alert the user that the temperature is too high.
    - The LED indicator will turn red to indicate the room is too hot.
- **When the room temperature falls below the lower threshold** (e.g., too cold):
    - The LCD display will show a message prompting the user to turn on the heater.
    - The buzzer will sound to notify the user that the temperature is too low.
    - The LED indicator will turn blue to signal that the room is too cold.
- **In normal conditions**, when the room temperature is within the desired range:
    - The LCD display will show that the room is at a comfortable temperature.
    - The LED indicator will remain green to indicate optimal conditions, and the buzzer will remain silent.

1.3. Data Collection Requirements:-

The system will gather data from the following sensors:

- Temperature sensor (TMP36) to monitor the room temperature.
- Data will be collected at regular intervals (e.g., every minute) to ensure real-time temperature monitoring.

1.4. Data Analysis Requirements:-

Once data is collected, the system will:

- Compare internal room temperature with user-defined or pre-set thresholds.
- Analyze external temperature to adjust the room temperature accordingly.
- If the outside temperature is high and the room temperature exceeds the threshold, prompt for AC/fan activation.
- If the temperature falls below a certain threshold, prompt for heating.
- Simulate automated decision-making, such as adjusting AC/fan/heater status based on the data.
- This analysis will be displayed to the user, showing instructions on controlling the devices.

1.5. System Management Requirements:-

The system will provide the following management capabilities:

- Onboard monitoring functionality that displays real-time data on an LCD or user interface.
- Simulated manual override that allows users to take control of the AC, fan, or heater if needed.
- Real-time feedback loop to adjust based on both user interaction and external temperature changes.

1.6. Data Privacy And Security Requirements:-

For This System, Data Privacy And Security Requirements Include Implementing User Authentication And Access Controls.

1.7. User Interface Requirements:-

This Application Will Be Deployed Locally On The Device But It Should Include Intuitive, Easy-To-Navigate Dashboards, Real-Time Data Visualization, Customizable Reports, And User-Friendly Tools For Data Analysis And System Monitoring, Ensuring Accessibility And Efficiency For Various User Roles.

2. PROCESS SPECIFICATION

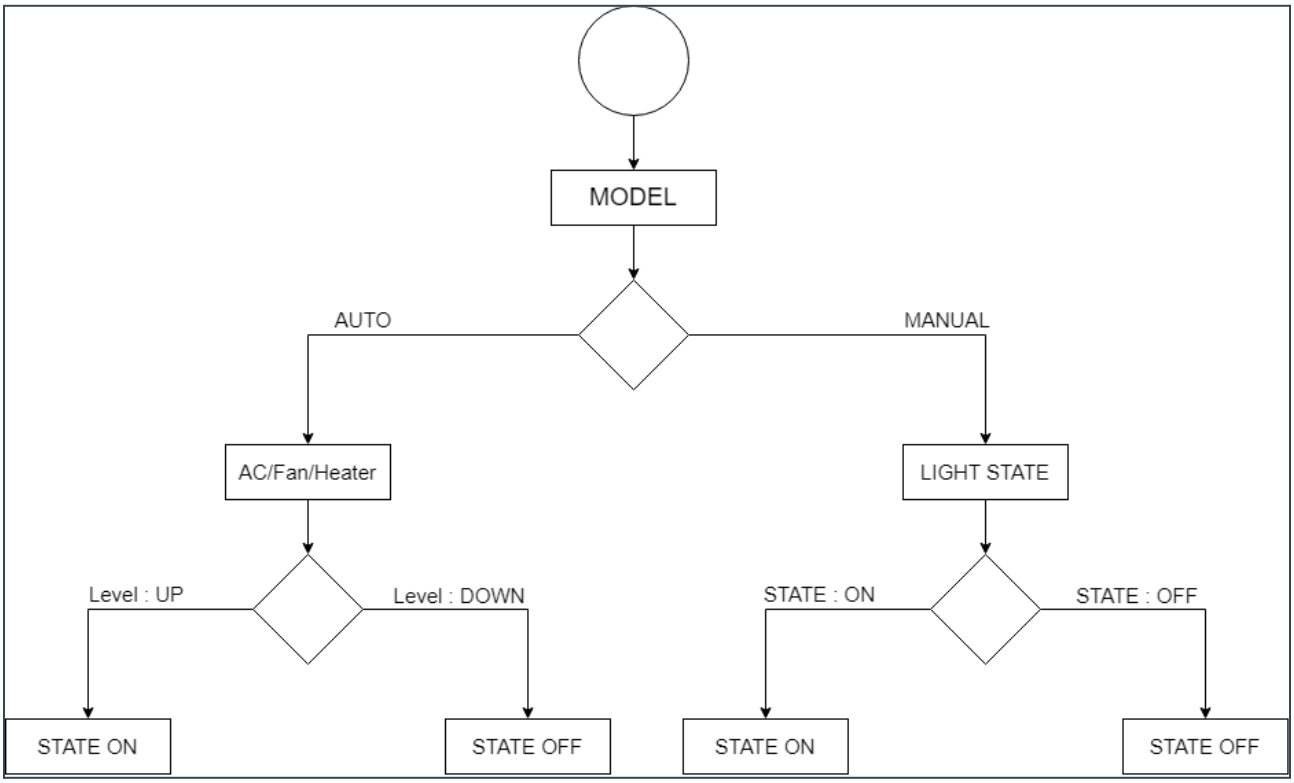

Fig. 1(a) Process Specification

- Displays room temperature readings.
- Shows user prompts when temperature thresholds are exceeded (too hot/cold).
- Offers real-time data updates, presenting comfort levels or device status (fan, heater).

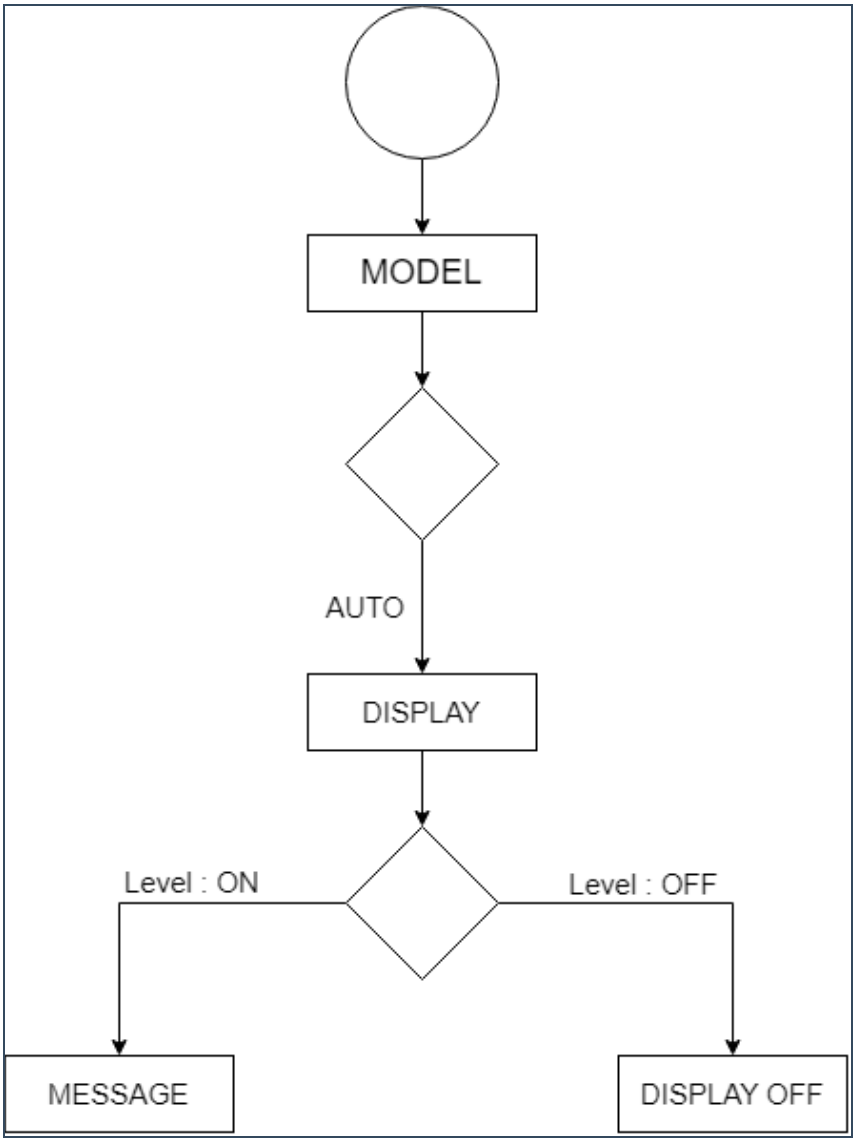

Fig. 1(b) LED Display

3. DOMAIN MODEL SPECIFICATION

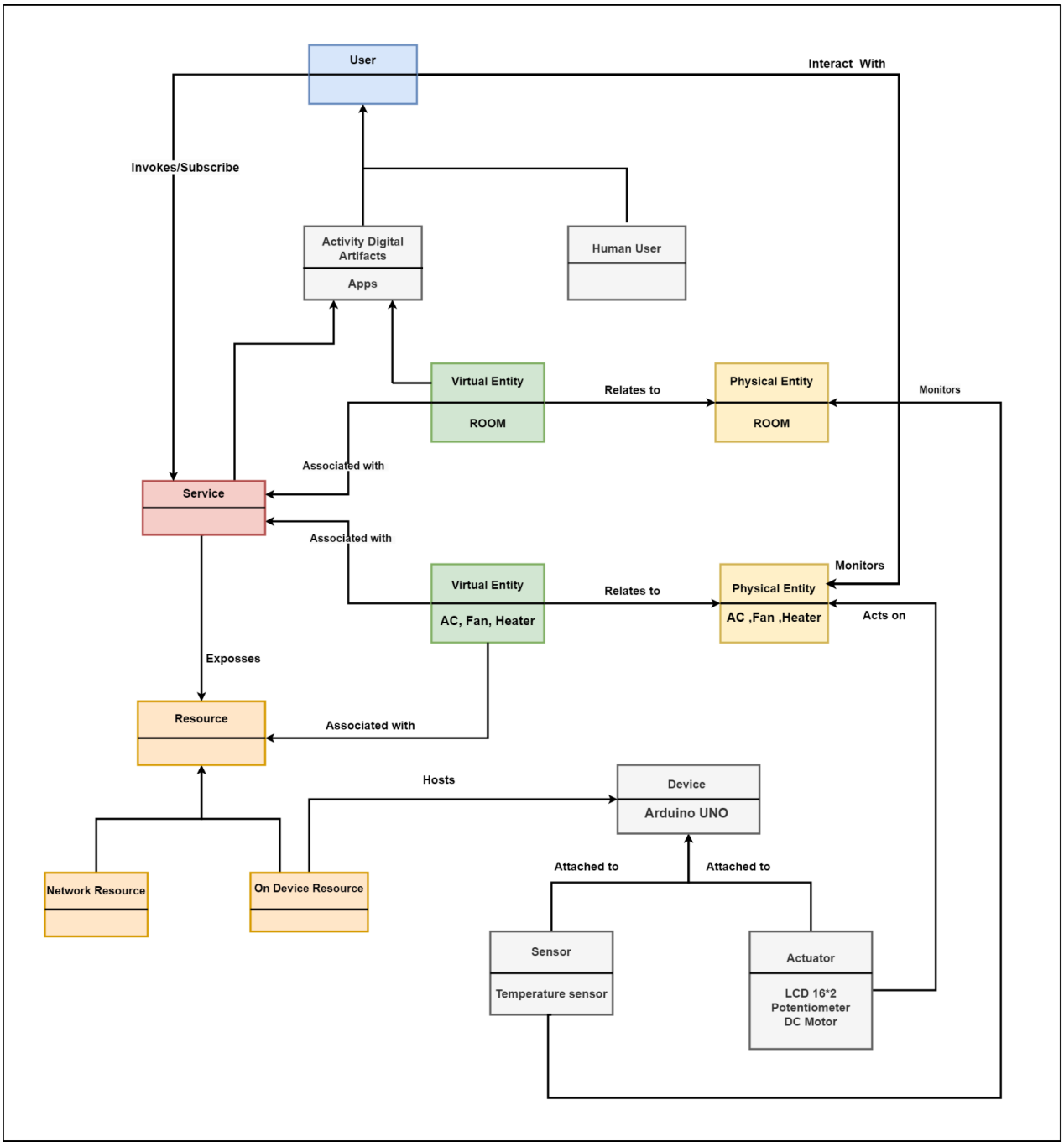

Fig. 2 Domain Model Specification

- Represents entities involved in the temperature control system.
- Entities include sensors (temperature), actuators (fans, heaters), and user interfaces.
- Shows relationships between these entities and how they interact for temperature management.

3.1. Entities and Objects

Entities: AC, Fan, Heater

Objects: Building

3.2. Attributes of the objects and relationship between objects

Temperature Sensor(TMP36):

- **Unique Identifier:** ID assigned to Temperature Sensor.
- **Temperature Detection:** Detects the Temperature in The Room.

AC:

- **Unique Identifier:** ID assigned to the AC.
- **Activation Status:** indicates Whenever Room Temperature is Too Hot then it Will ON.

Fan:

- **Unique Identifier:** ID assigned to the Fan.
- **Activation Status:** indicates Whenever Room Temperature is Too Hot or Below Hot then it Will ON Along With AC.

Heater:

- **Unique Identifier:** ID assigned to the Heater.
- **Activation Status:** indicates Whenever Room Temperature is Too Cold then it Will ON.

Buzzer:

- **Unique Identifier:** ID assigned to the Buzzer.
- **Activation Status:** Indicates whether the Temperature in The Room Changed

LED:

- **Unique Identifier:** ID assigned to the LED.
- **Activation Status:** Indicates whether the LED is turned on Based on Temperature.

3.3. Specific Technology or Platform

- Sensor Technologies used for detecting Temperature in The Room and Turn ON AC, Fan, Heater along With Buzzer, Message And LED Lights.

4. INFORMATION MODEL SPECIFICATION

4.1. Virtual Entities :

AC, Fan, Heater

4.2. Attributes and Relations of Virtual Entities. :

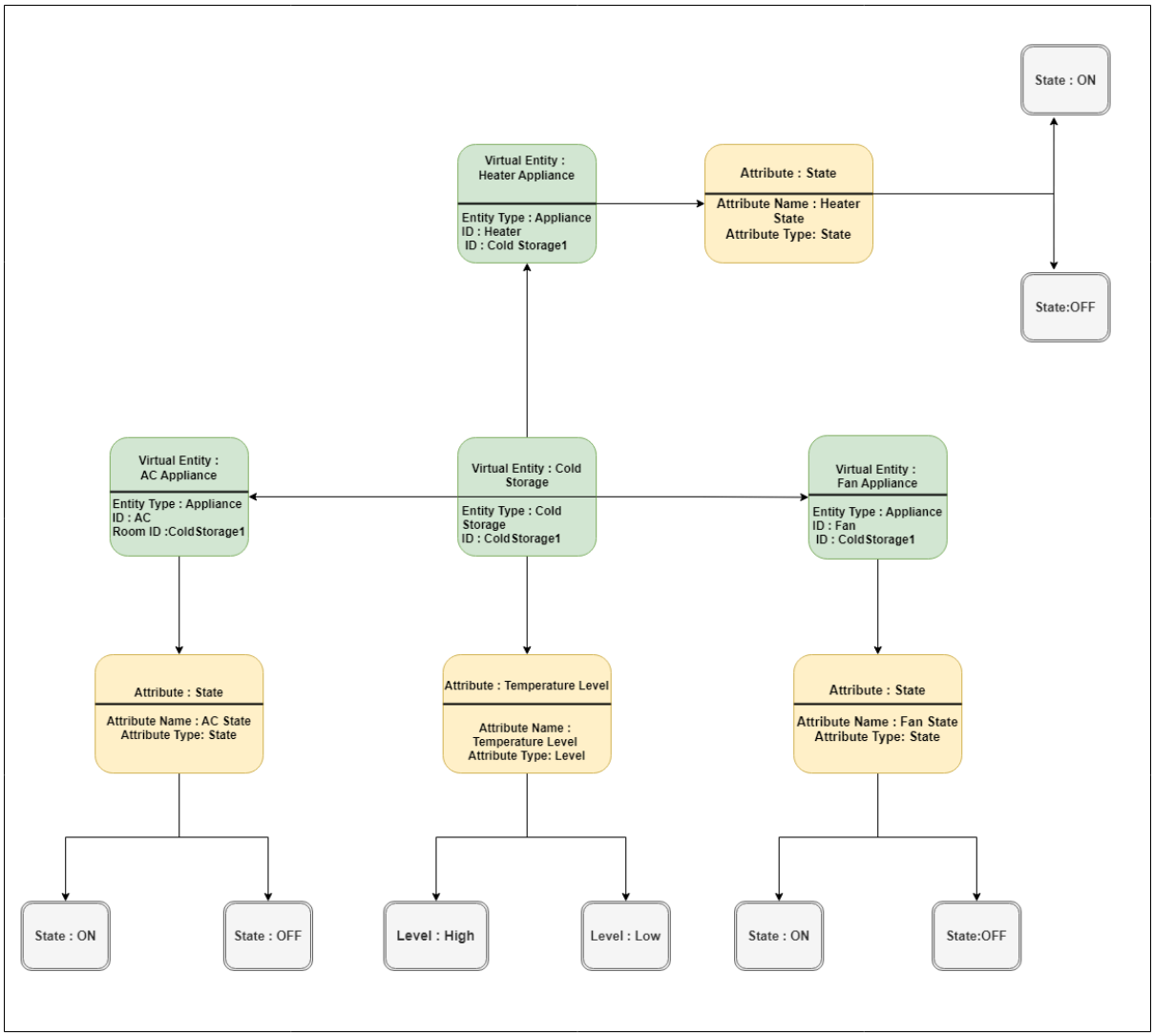

Fig. 3 Information model Specification

- Displays data structures used in the system.
- Describes attributes of virtual entities (e.g., room temperature, external temperature).
- Outlines how virtual entities relate to each other and trigger actions (e.g., turning on the fan or heater).

5. SERVICE SPECIFICATION

5.1. Service Types :

Temperature Detection Service:

- Detects Temperature Using TMP36
- Triggers alarms (buzzer, LED) based on predefined thresholds.
- Provides real-time Room Temperature(e.g., via LCD Display).

5.2. Service Inputs/Output :

Temperature Detection Service:

- **Inputs :** Sensor data (Room Temperature level measurements), threshold settings.
- **Output :** Turn ON AC, Fan if the Temperature is Too Hot along With Buzzer and RED LED and Same Goes for Heater if The Temperature is Too Cold

5.3. Service Schedules :

Temperature Detection Service:

- Continuous monitoring when the system is active.

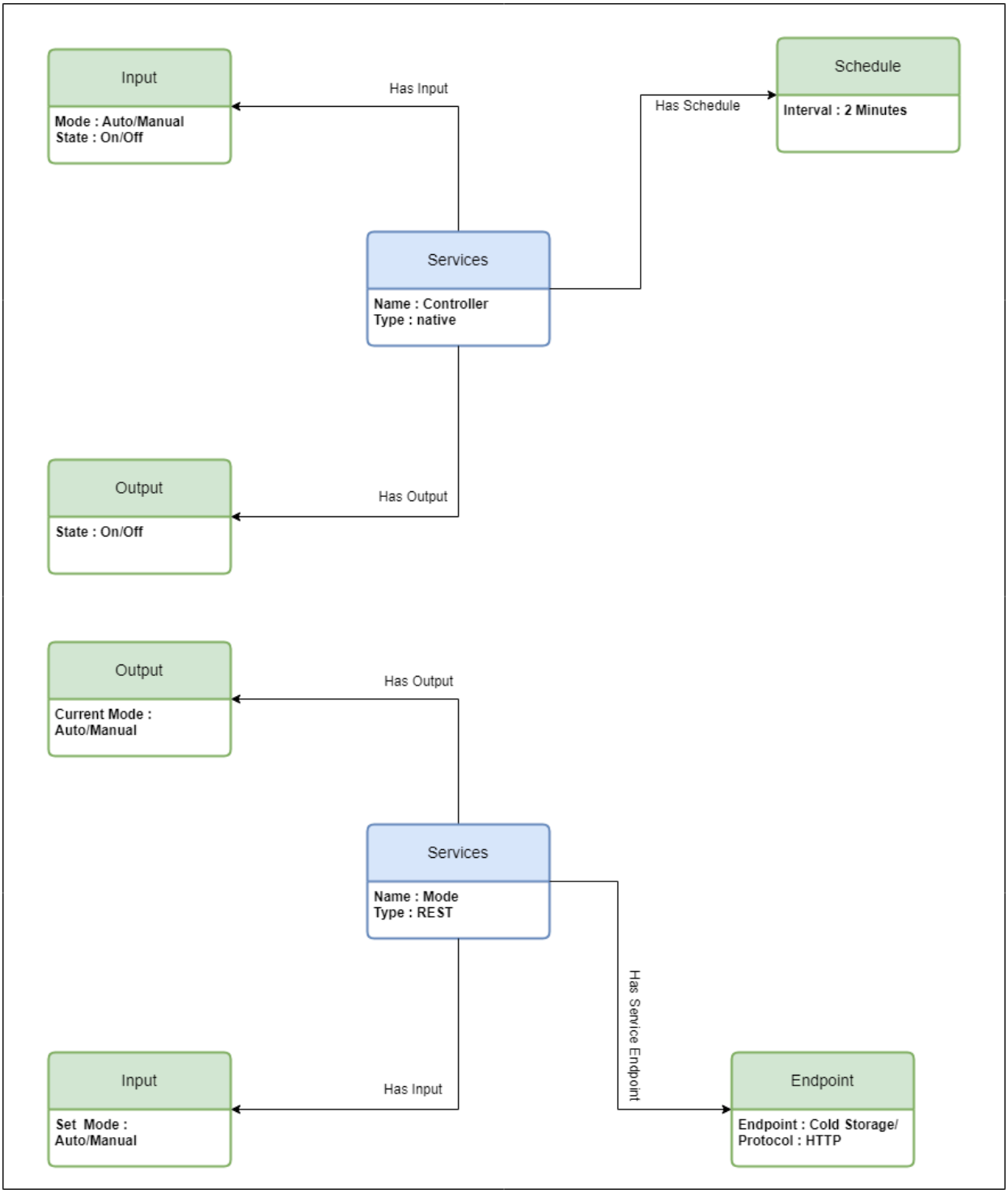

Fig. 4(a) Service Specification Diagram

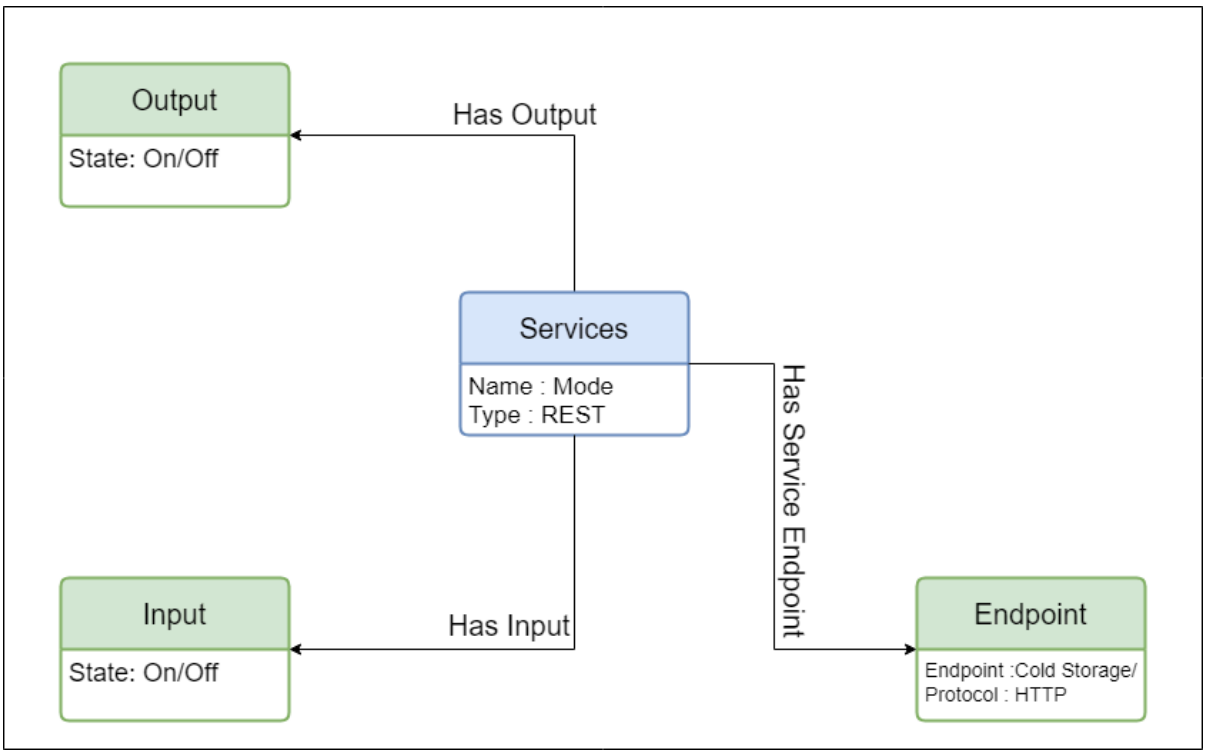

Fig. 4(b) Service Specification Diagram

- Defines different services provided by the system.
- Includes temperature monitoring, notifications, and control actions for devices.
- Details service inputs (temperature data), outputs (device actions), and endpoints (actuators).

6. IOT LEVEL SPECIFICATION

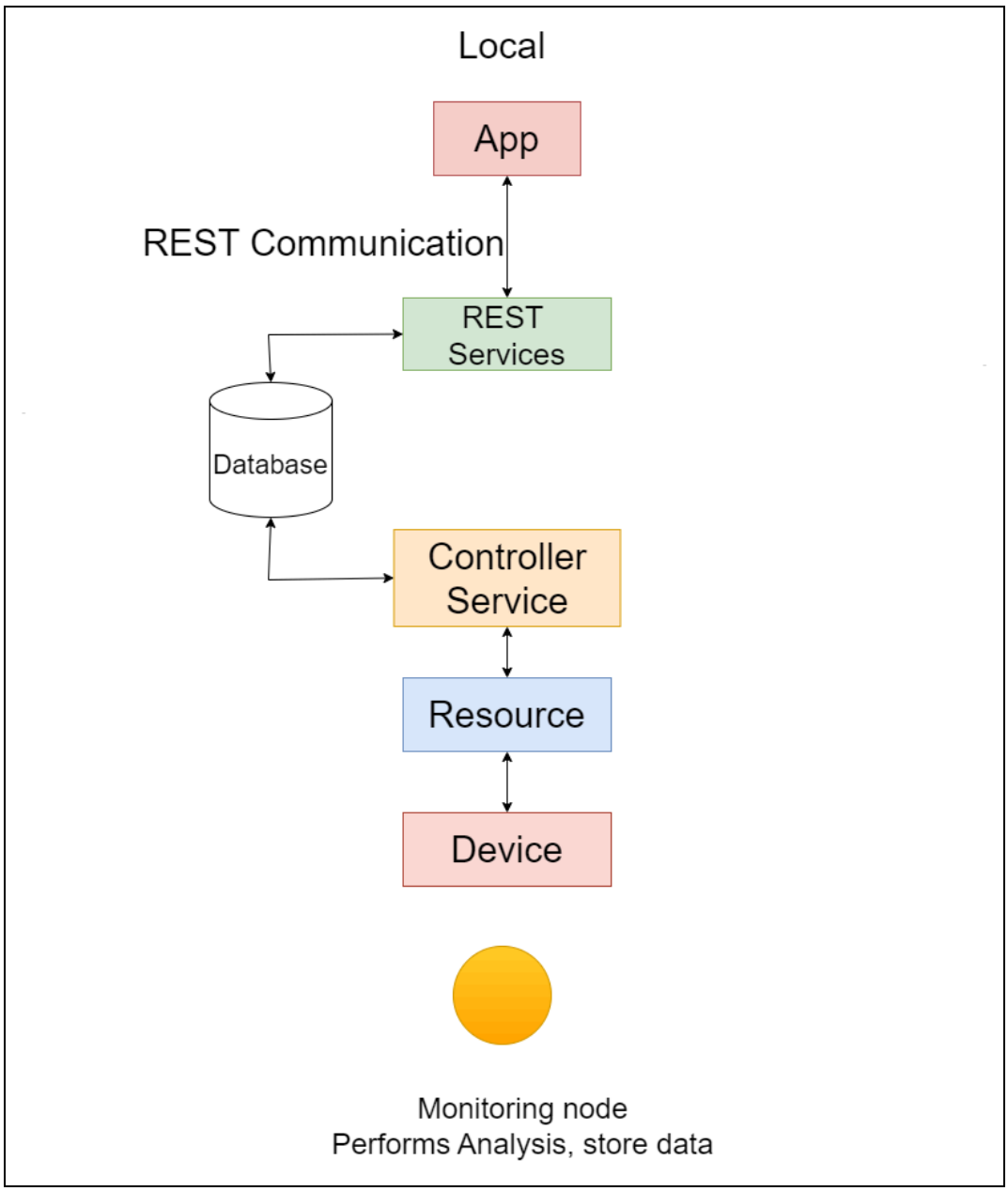

Fig. 5 IOT Level Specification

- Illustrates the architecture of the IoT system.
- Shows data flow between sensors, actuators, and the control system.
- Highlights the integration of temperature monitoring and decision-making processes.

7. FUNCTIONAL VIEW SPECIFICATION

7.1. The Functional View (FV) :

The functional view defines the defined functions of the IoT systems grouped into various Functional Groups (FGs).

7.2. Functional Groups (FGs) :

Each Functional Group either provides functionalities for interacting with instances of concepts defined in the Domain Model or provides information related to these concepts.

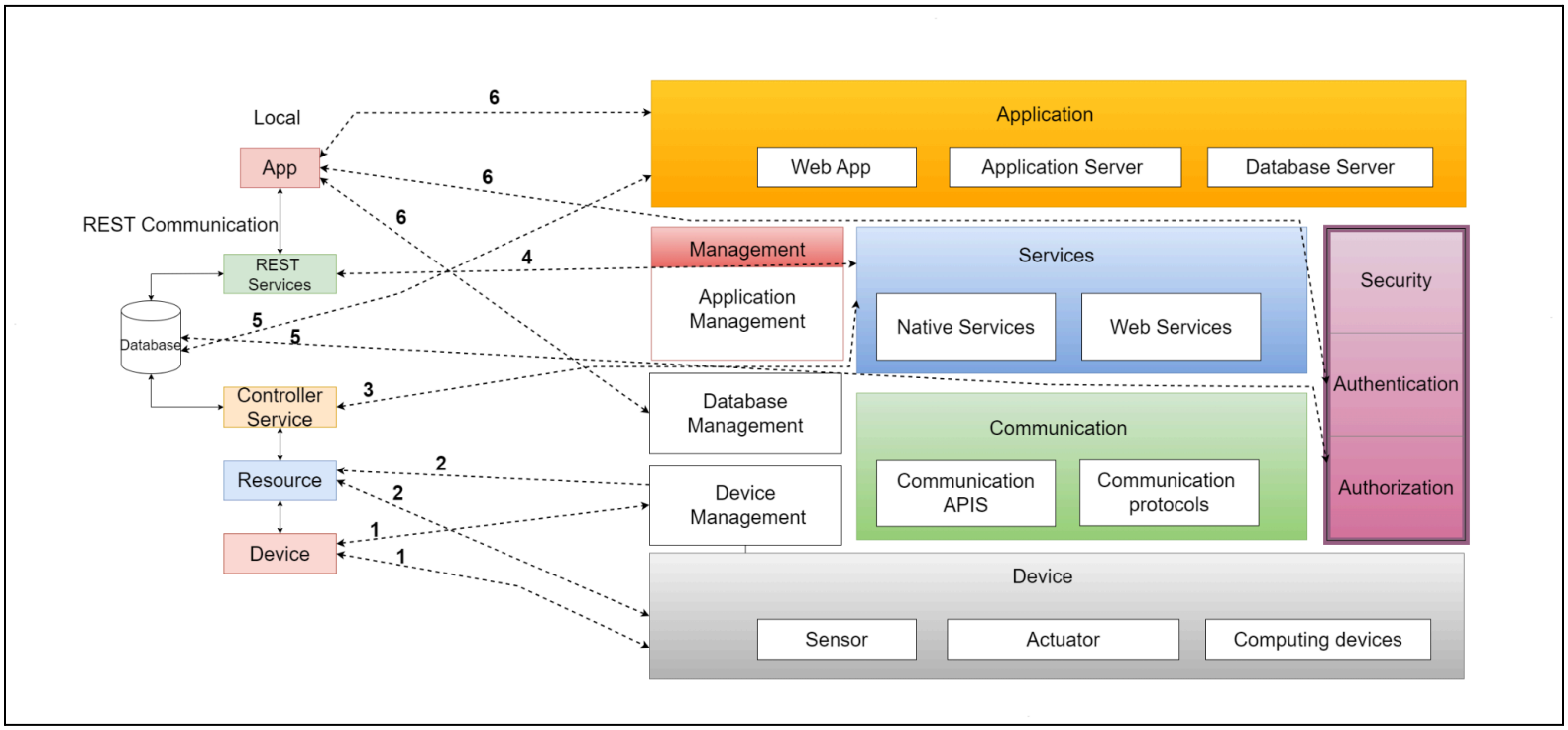

Fig. 6 Functional View Specification

- Shows functional groups within the system, such as sensing, decision-making, and user interaction.
- Details how each function contributes to overall temperature control.
- Focuses on key functions like temperature monitoring, device control, and notifications.

8. OPERATIONAL VIEW SPECIFICATION

8.1. Storage Options :

- The Smart Temperature Control System For Home system does not require a database for data storage. Instead, the device operates in real-time, triggering Buzzer, LED immediately when Temperature exceeds predefined thresholds And Turn on AC, Fan and Heater Based on Temperature. No historical data is stored.

8.2. Device Options :

- The Smart Temperature Control System For Home includes Temperature sensors for Measuring Temperature, a buzzer for audible alerts, and an LED for visual indication. The system can function.

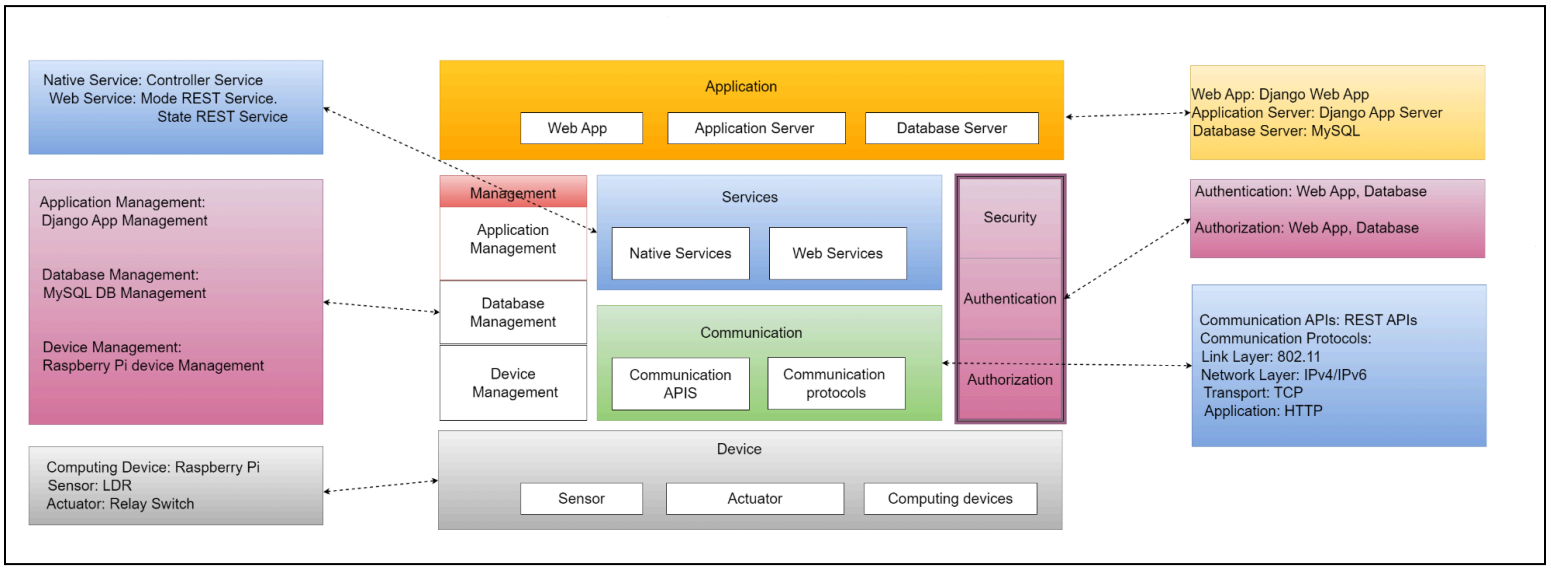

Fig. 7 Operational View Specification

- Outlines the operational setup, including hosting and device storage options.
- Specifies how services are hosted (locally or on the cloud).
- Describes communication between devices and system management interfaces.

9. DEVICES & COMPONENT INTEGRATION

9.1. Devices & Component Used

Component	Use	Real World Name
Arduino Uno R3	Acts as the control unit for various electronic components.	Arduino Uno R3
LCD 16 x 2	Displays data or messages, typically sensor readings.	Digital display screen
Temperature Sensor[TMP36]	measures temperature in The Environment	Analog Temperature Sensor
Potentiometer	Used to adjust resistance in a circuit, allowing for variable control of voltage, such as in volume knobs or dimmer switches.	Knob or dial
9V Battery	electric battery with a nominal voltage of 9 volts.	PP3 Battery
Relay SPDT	A relay is an electrically controlled switch that can manage high currents, with SPDT (Single Pole Double Throw) indicating a switch that has two output positions.	Electric switch
Piezo	Produces sound or alarm signals when triggered.	Buzzer or beeper
Light bulb	A light bulb emits light by heating a metal filament until it glows, producing illumination when electricity passes through it.	Bulb or lamp
DC Motor	Converts electrical energy into mechanical movement.	Electric motor
Breadboard Small	Solderless breadboards are for prototyping circuits quickly and easily.	Circuit board

9.2. DEVICE & COMPONENT INTEGRATION DIAGRAM

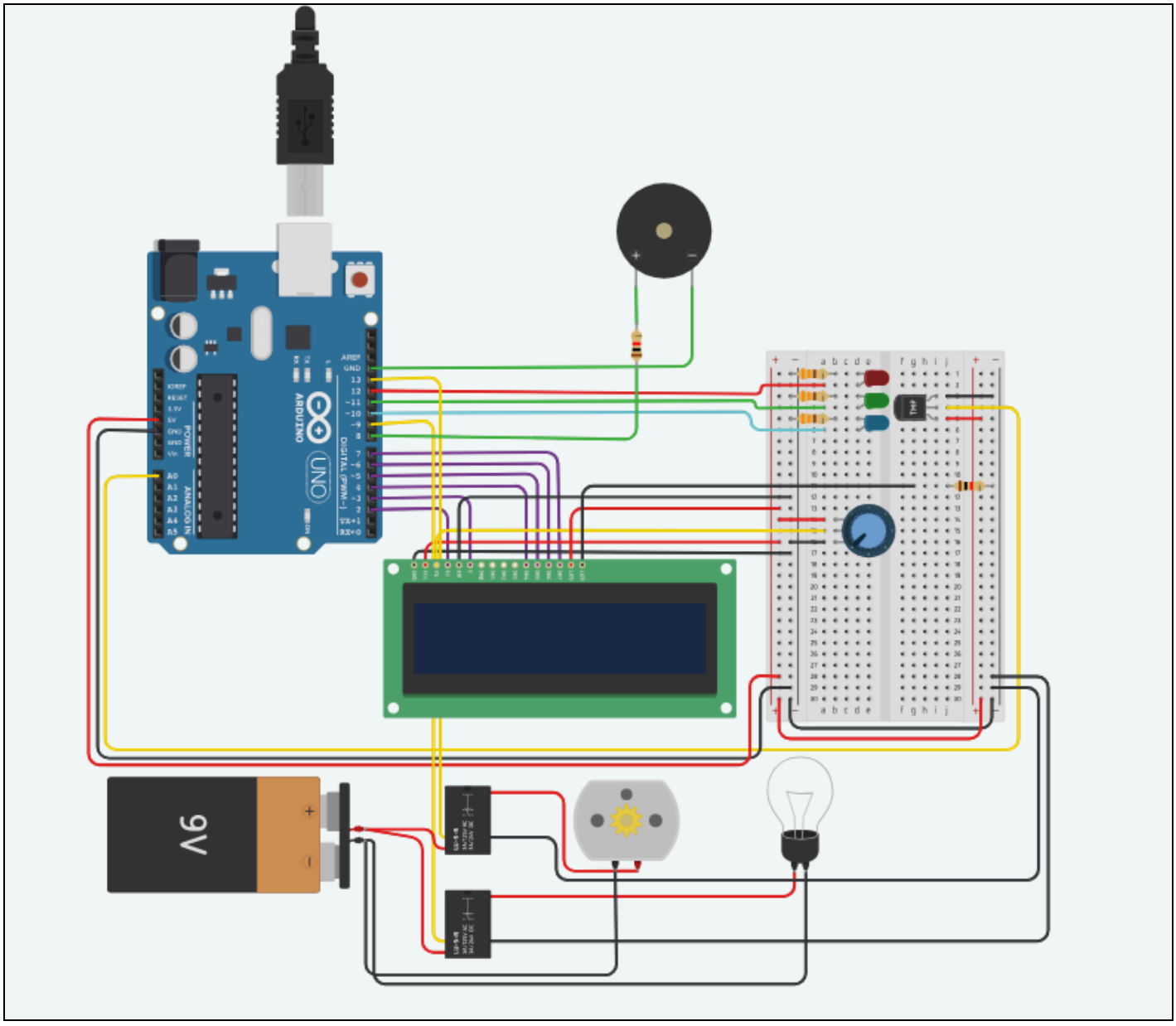

Fig. 8(a) Device & Component Integration Diagram

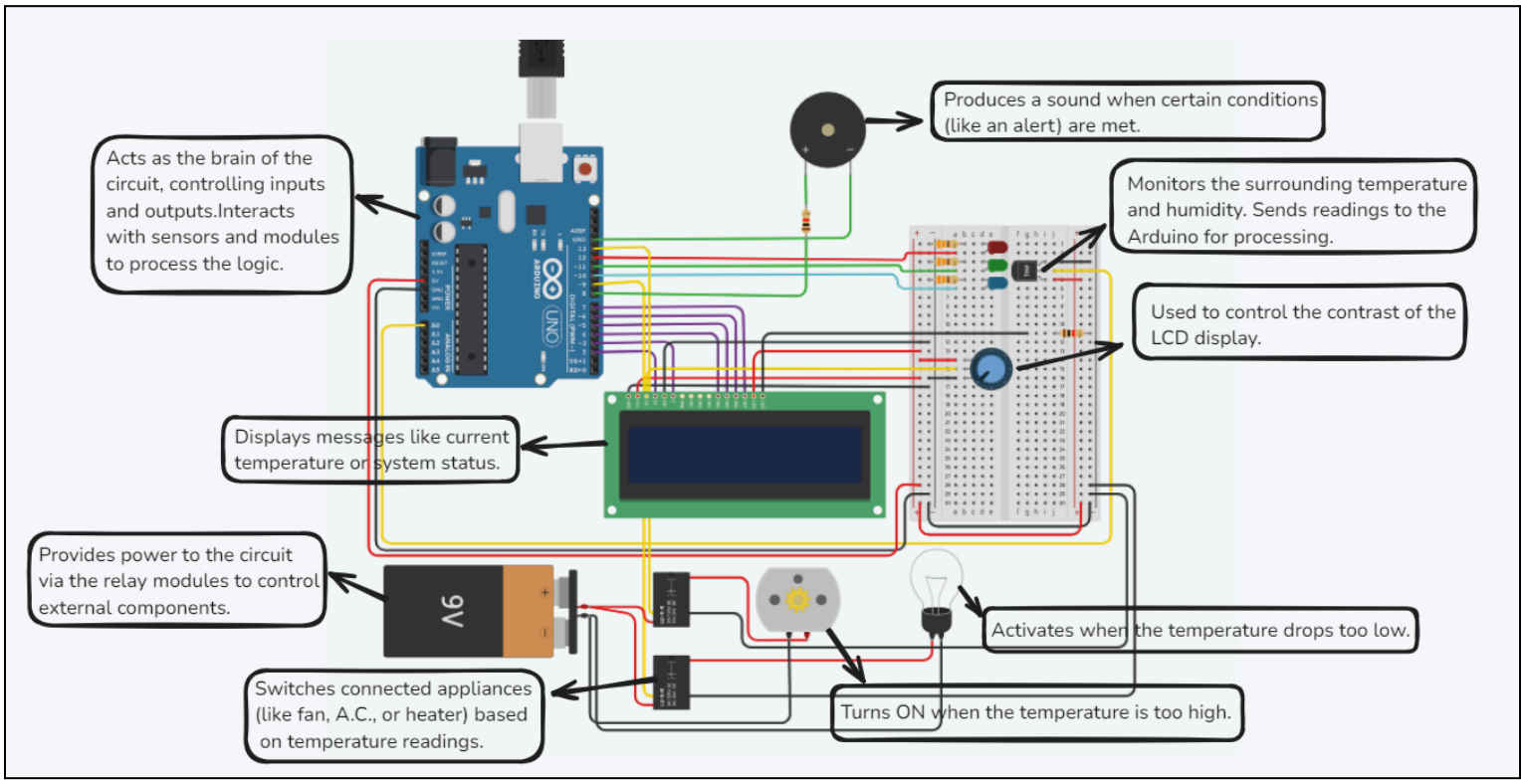

Fig. 8(b) Device & Component Integration Diagram With Details

10. APPLICATION DEVELOPMENT

10.1. Experiments Block Diagram

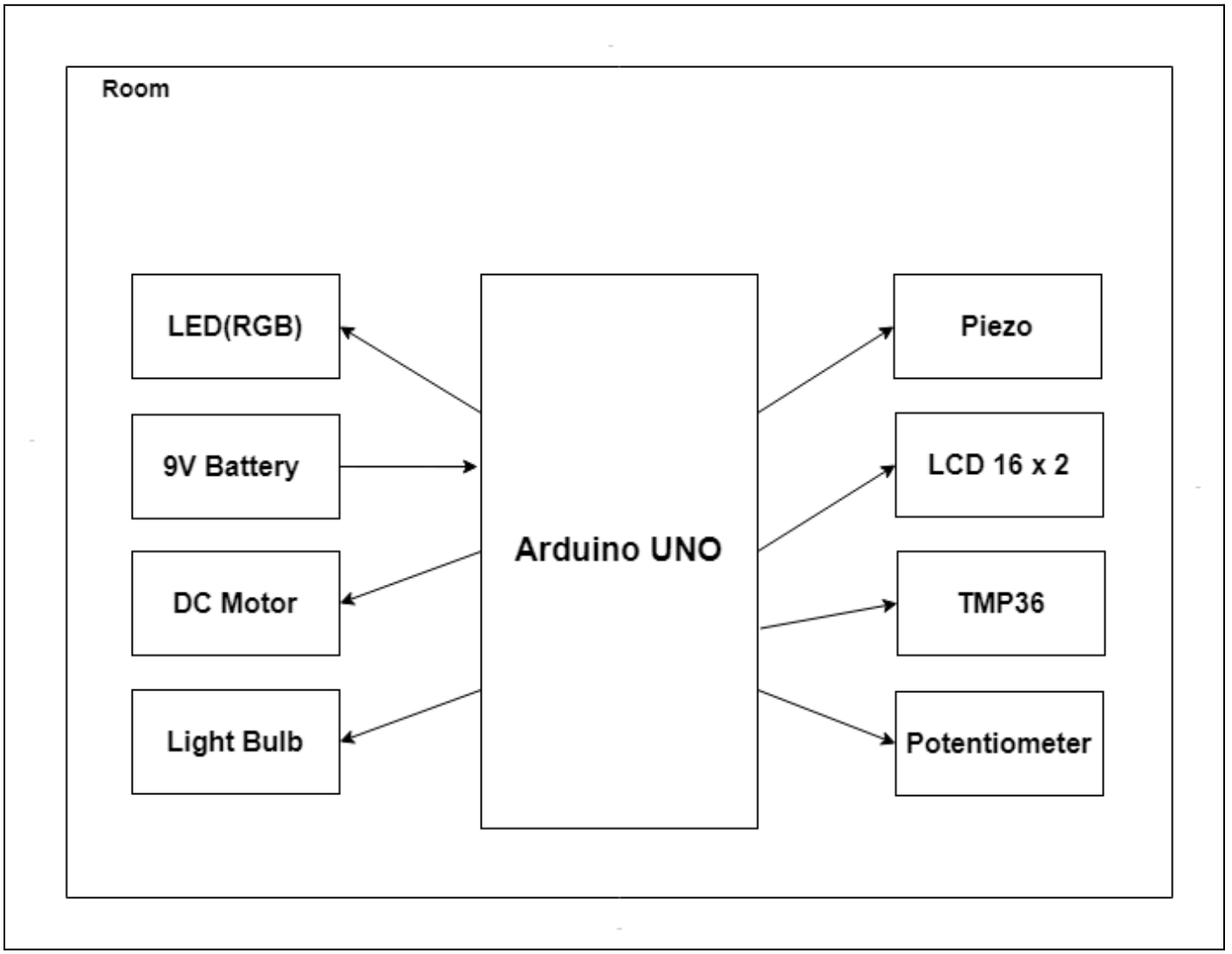

Fig. 9 Experiment Block Diagram

10.2. Experiments procedures

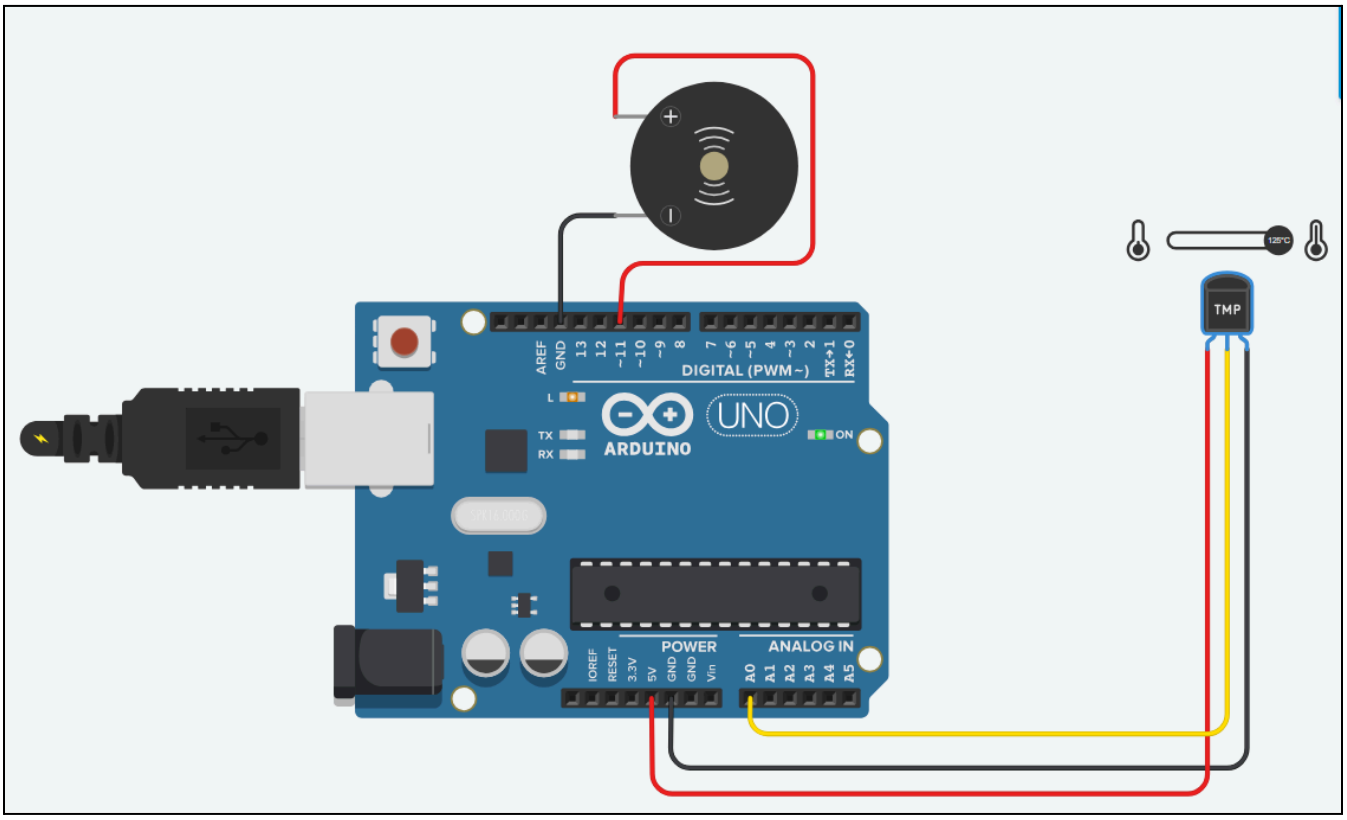

Fig. 10 Experiment Procedure

10.3. CODE (Arduino IDE / Python on Raspberry Pi)

```
#include <LiquidCrystal.h>

// Declare constants and pins
const int LM35 = A0; // Temperature sensor pin
const int fan = 13; // Fan pin
const int heater = 9; // Heater pin
const int RL = 12; // Red LED pin
const int GL = 11; // Green LED pin
const int BL = 10; // Blue LED pin
const int buzzer = 8; // Buzzer pin
// Temperature thresholds
const float HIGH_TEMP = 30.0;
const float LOW_TEMP = 15.0;

// Initialize the LCD library
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
// Global variables
float lastTemp = -1000; // Store last temperature to avoid redundant updates
bool buzzerTriggered = false; // Track if buzzer has buzzed

// Function to display and scroll messages on the LCD
void displayAndScrollMessage(const char* message) {
  lcd.setCursor(0, 1);
  lcd.print("                "); // Clear the entire bottom row
  int msgLength = strlen(message);
  if (msgLength <= 16) {
    lcd.print(message);
  } else {
    for (int position = 0; position <= msgLength; position++) {
```

```
            lcd.setCursor(0, 1);
            for (int i = 0; i < 16; i++) {
                lcd.print(position + i < msgLength ? message[position + i] : ' ');
            }
            delay(300); // Shorter delay for scrolling
        }
    }
}
// Function to handle devices and display messages
void updateState(bool fanState, bool heaterState, int redLED, int greenLED, int blueLED,
const char* message, bool shouldBuzz) {
    digitalWrite(fan, fanState);
    digitalWrite(heater, heaterState);
    digitalWrite(RL, redLED);
    digitalWrite(GL, greenLED);
    digitalWrite(BL, blueLED);

    // Buzzer logic
    if (shouldBuzz) {
        if (!buzzerTriggered) { // Only buzz if not already buzzing
            buzzerTriggered = true; // Mark as triggered
            for (int i = 0; i < 3; i++) { // Beep 3 times
                tone(buzzer, 1000); // Set buzzer frequency
                delay(200);         // Beep duration
                noTone(buzzer);     // Stop sound
                delay(100);         // Pause between beeps
            }
        }
    } else {
        buzzerTriggered = false; // Reset flag if no buzzing needed
    }
```

```
    displayAndScrollMessage(message);
}

void setup() {
    Serial.begin(9600);
    lcd.begin(16, 2);
    lcd.print("Temp = ");
    delay(2000);

    pinMode(fan, OUTPUT);
    pinMode(heater, OUTPUT);
    pinMode(RL, OUTPUT);
    pinMode(GL, OUTPUT);
    pinMode(BL, OUTPUT);
    pinMode(buzzer, OUTPUT);
}

void loop() {
    // Read temperature from LM35
    float Temperature = analogRead(LM35) * 500.0 / 1023.0;
    Serial.print("Current Temperature: ");
    Serial.println(Temperature);

    // Update temperature display if changed significantly
    if (fabs(Temperature - lastTemp) > 0.1) {
        lcd.setCursor(6, 0);
        lcd.print(Temperature, 1);
        lcd.print(" ");
        lastTemp = Temperature;
    }
```

```
    bool fanState = false;
    bool heaterState = false;
    const char* message = "Comfortable Temp";
    bool shouldBuzz = false;

    // Determine fan and heater state based on temperature
    if (Temperature > HIGH_TEMP) {
        fanState = true;
        message = "Too Hot! Fan & AC ON!";
        shouldBuzz = true; // Activate buzzer for high temp
    } else if (Temperature < LOW_TEMP) {
        heaterState = true;
        message = "Too Cold! Heater ON";
        shouldBuzz = true; // Activate buzzer for low temp
    }

     Serial.print("Should buzz: ");
    Serial.println(shouldBuzz);

    // Update state and display message
     updateState(fanState, heaterState, fanState ? HIGH : LOW, heaterState ? LOW : HIGH,
LOW, message, shouldBuzz);

    delay(300); // Shorter delay for more responsiveness
}
```

10.4. Screenshots of input/output

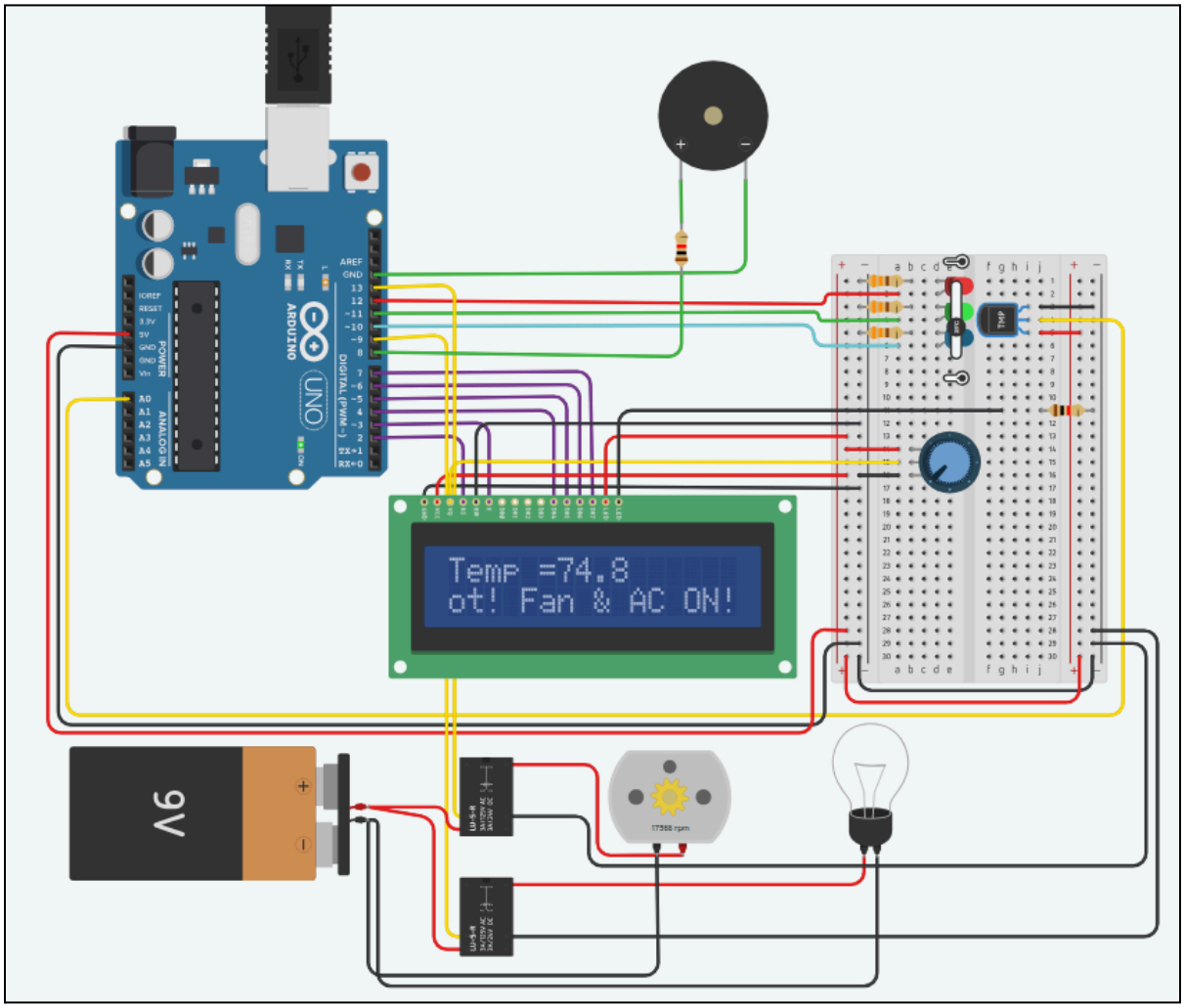

Fig. 11(a.1) Input(Temperature(75 °C))

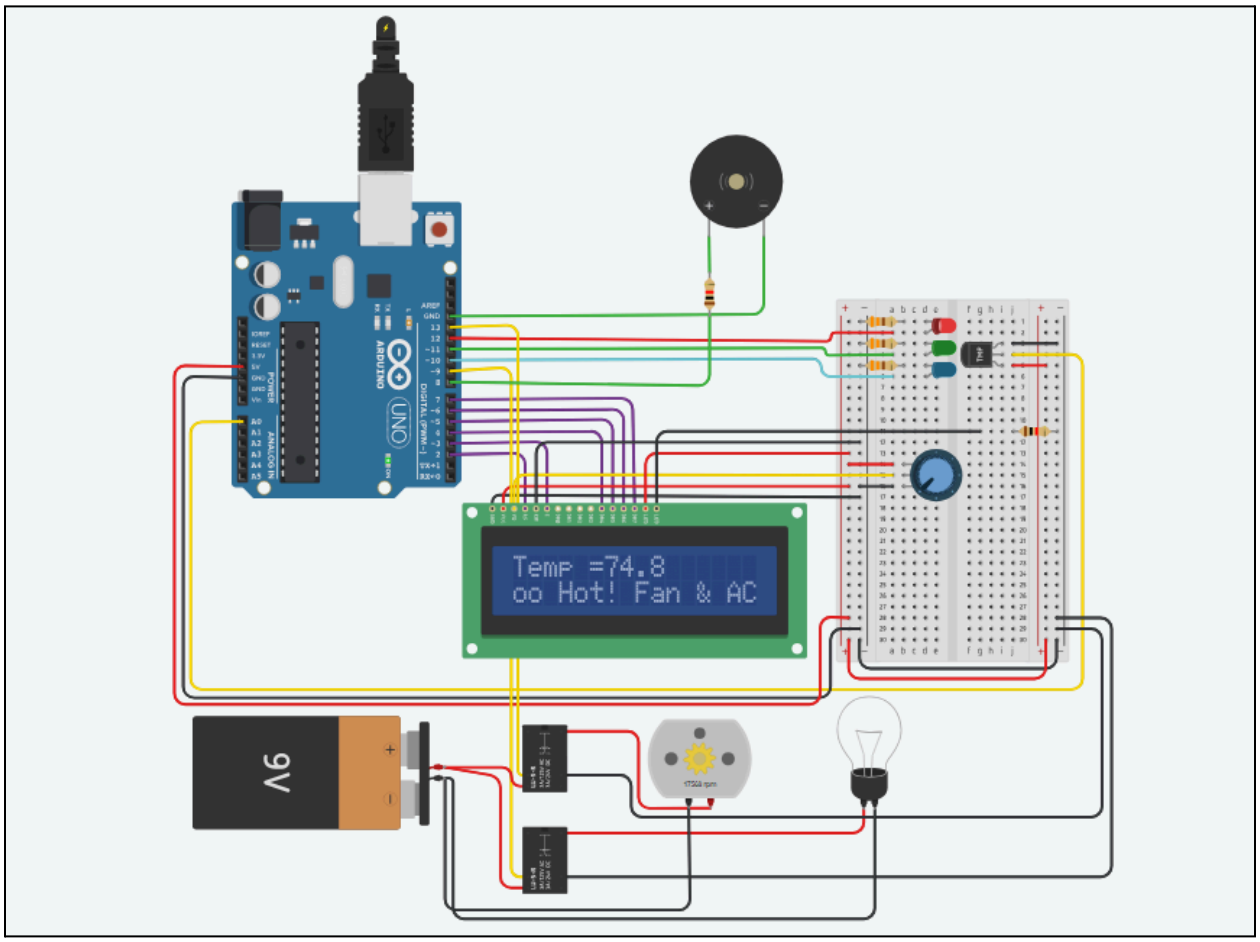

Fig. 11(a.2) Output(When Temperature is too high)

When the temperature becomes too high in this system, the DC motor, representing the fan and AC, will turn on to cool the environment. Simultaneously, the buzzer will activate, providing an audible alert to indicate that the temperature has exceeded the threshold. Along with this, the red LED will illuminate, giving a visual warning of the high temperature condition. The Arduino controls all these actions, ensuring that when the temperature crosses a set limit, cooling mechanisms and alerts are triggered effectively.

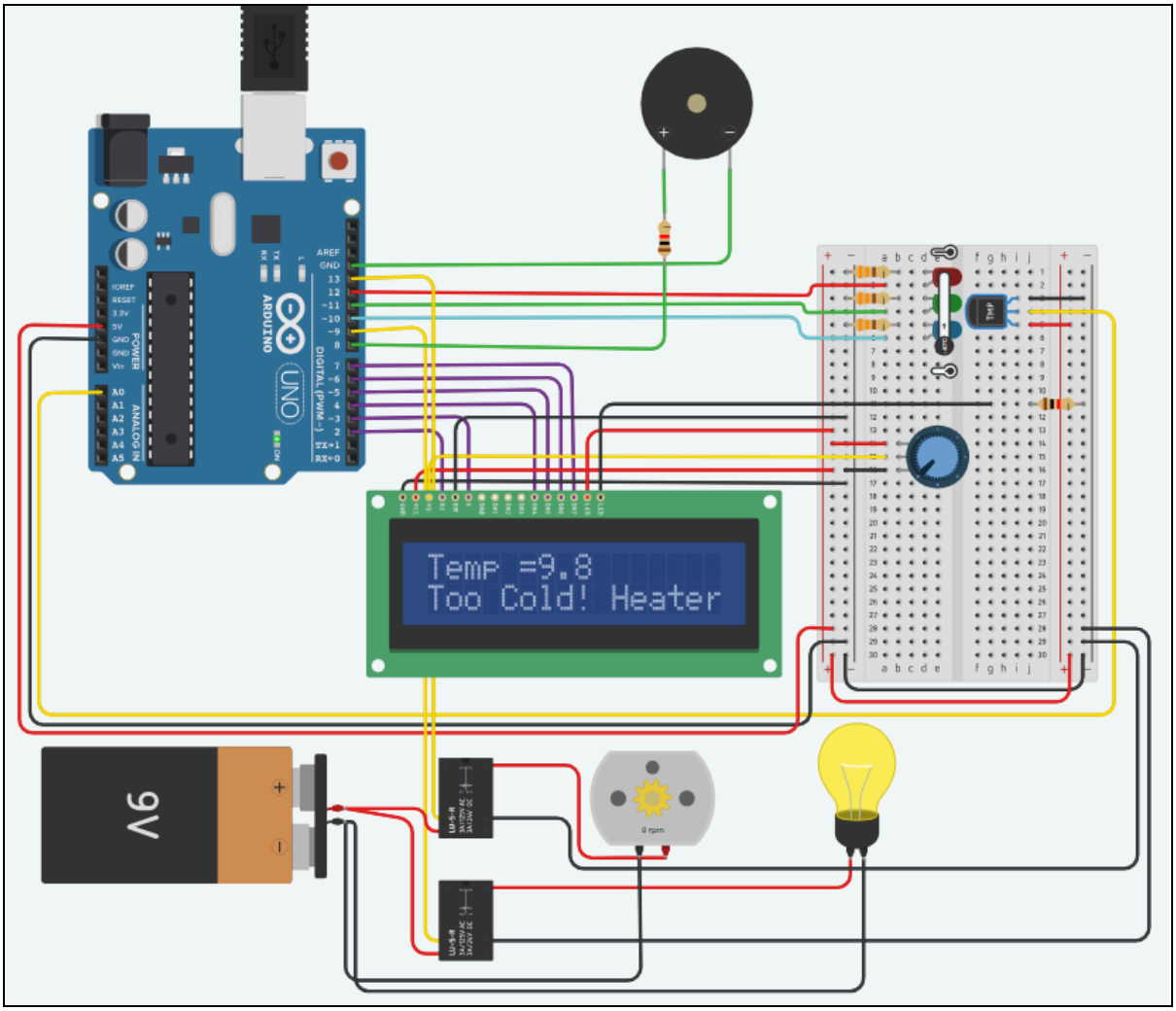

Fig. 11(b.1) Input(Temperature(9.8 °C))

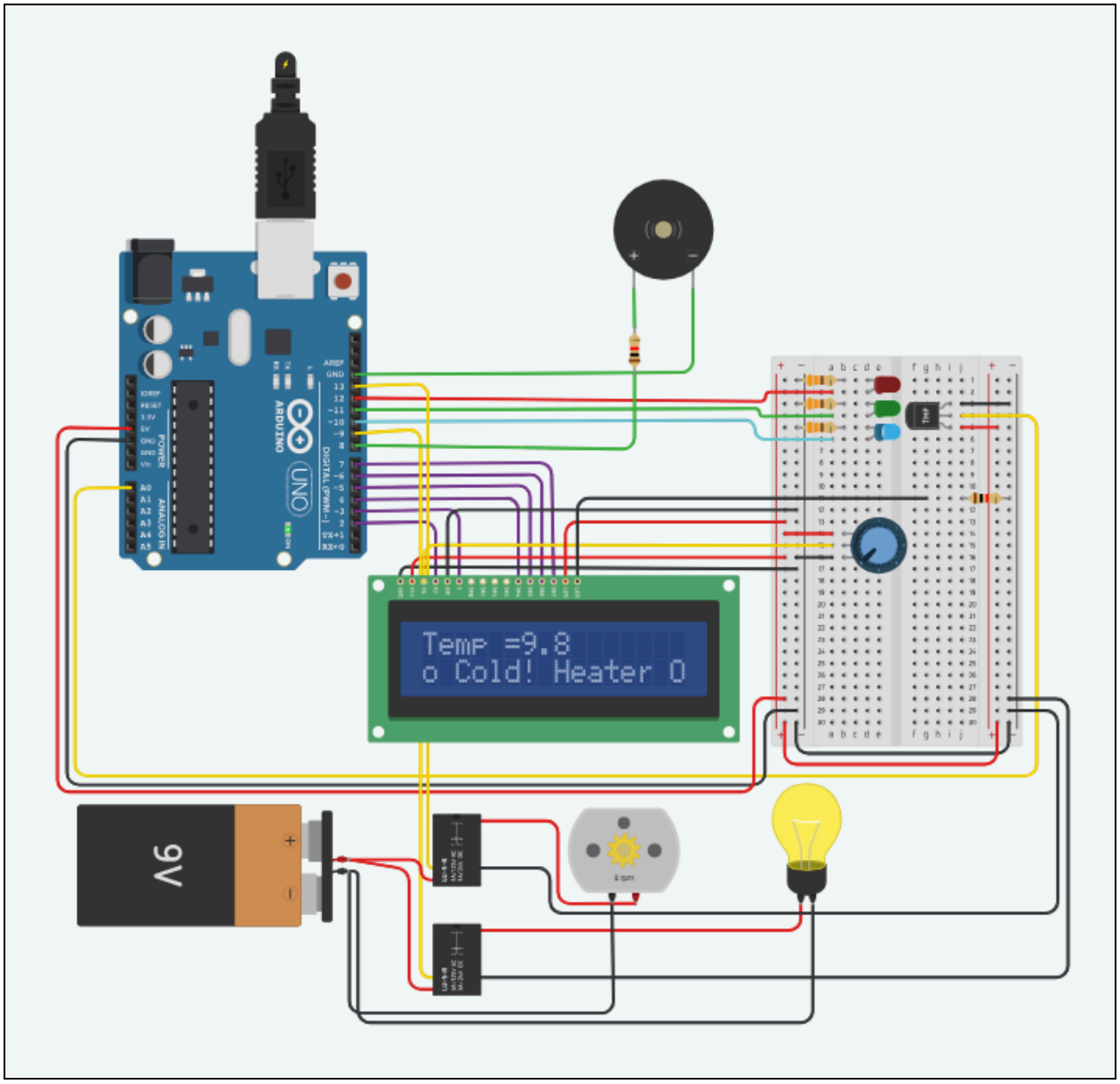

Fig. 11(b.2) Output(When Temperature is too Low)

When the temperature drops too low, the yellow light represents the heater. When the temperature falls below a set threshold, the yellow light (heater) turns on to indicate that the heating process has started. At the same time, the buzzer sounds, giving an audible alert for the low temperature condition. The LCD shows the current temperature (e.g., -9.8°C) along with a message stating that it's cold and the heater is on ("Too Cold! Heater ON"). This setup ensures that when it gets too cold, the heater (represented by the yellow light) is activated to warm the environment.

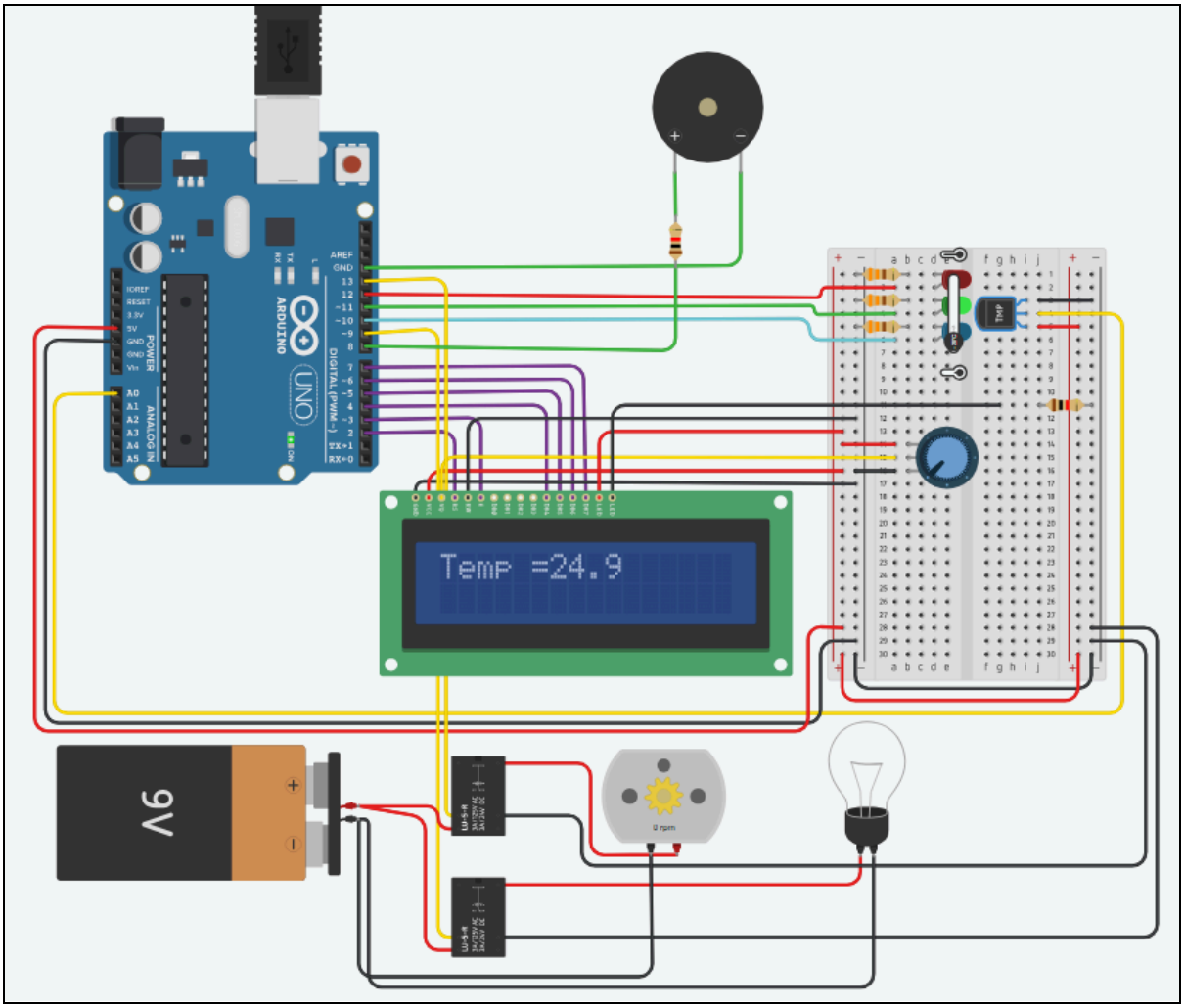

Fig. 11(c.1) Input(Temperature(25 °C))

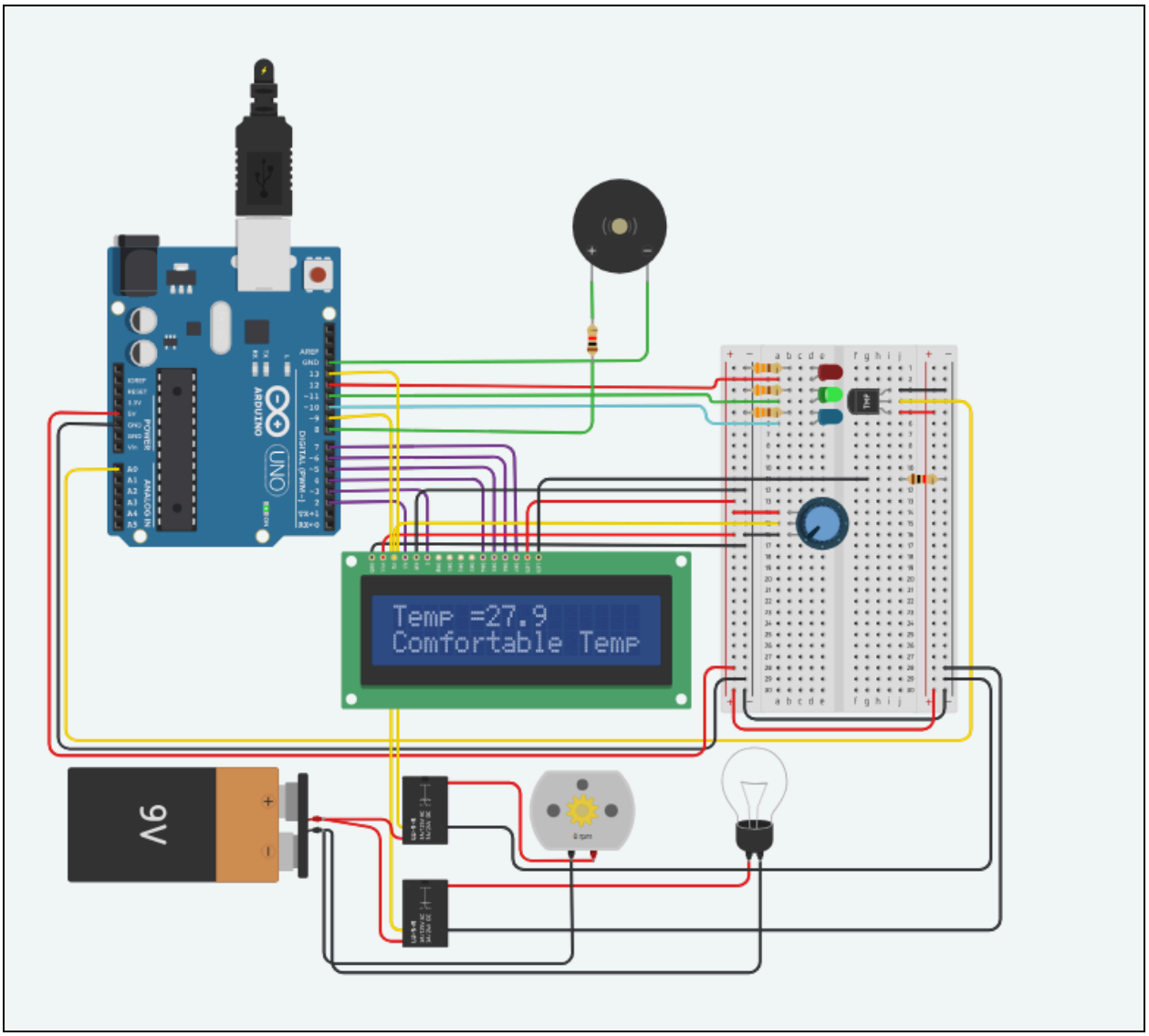

Fig. 11(c.2) Output(When Temperature is Normal)

When the temperature is within the normal or comfortable range, as indicated in the diagram (showing 27.9°C on the display with the message "Comfortable Temp"), the system deactivates the AC (fan), heater, and buzzer. The yellow light (heater) turns off, and the DC motor representing the AC and fan also stops. No alerts are triggered because the temperature is within the acceptable range, so both the red and yellow LEDs remain off, and the system remains idle until the temperature changes significantly.

10.5. Conclusion

The Smart Temperature Control System for Homes project uses IoT technology to automatically regulate home temperatures by monitoring internal and external conditions. It integrates components like sensors, microcontrollers (Arduino), and actuators (fans, heaters). Based on set thresholds, the system provides real-time feedback and automated control, ensuring a comfortable environment by triggering the appropriate actions like turning on fans or heaters when necessary. The system is also capable of notifying users via buzzers, LEDs, and displays, providing a comprehensive and efficient solution for temperature management.

10.6. Future Scope and Limitations

Future Scope :

- **Remote Control and Monitoring:** Integrating smartphone apps or web interfaces to allow users to control and monitor home temperature remotely.
- **AI Integration:** Using machine learning to predict user preferences and adjust temperatures based on historical data.
- **Energy Efficiency:** Incorporating solar power or energy-saving algorithms to optimize energy consumption.
- **Integration with Smart Home Ecosystems:** Expanding compatibility with other smart home devices like lights, locks, and security systems for a more connected living environment.

Limitations :

- **Dependency on Sensors:** The system's accuracy is limited by the precision of the temperature sensors used.
- **Local Deployment:** The current system is designed for local deployment, limiting its ability to be monitored or controlled remotely unless expanded.
- **Single Room Application:** The project is tailored for a single room, requiring further development to scale for multi-room or whole-house solutions.
- **Limited User Control:** Although it provides automated control, there is limited manual user override capability, which could be expanded to provide greater flexibility.

11. REFERENCE

- **TinkerCad : https://www.tinkercad.com/**
- **Original Project : https://www.tinkercad.com/things/34fnO1FzgiC-room-temperature**
- **Research Paper : https://www.researchgate.net/publication/348387507_Improving_the_Ambient_Temperature_Control_Performance_in_Smart_Homes_and_Buildings**