Upliance.ai

Embedded Systems Intern Assignment

Submitted by: Vase Anil
Phone no: +91 9550691198
Email: anilvasy9930@gmail.com

Linkedin: https://www.linkedin.com/in/anil-vase-2362b5244/

Resume: https://drive.google.com/file/d/1OtswgTTKlDU

Wokwi link: https://wokwi.com/projects/436618122468225025

GitHub link: https://github.com/Anilvasy/Heater_control_system

Aim: To Build a Basic Heater Control System.

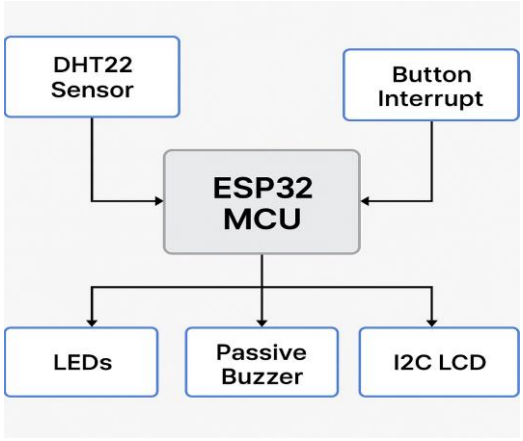
Project Overview:

This embedded system monitors temperature and humidity using a DHT22 sensor, indicates current heating status via LEDs and passive buzzer, and displays readings on an I2C LCD. It supports both AUTO mode and MANUAL mode with a physical button interrupt.

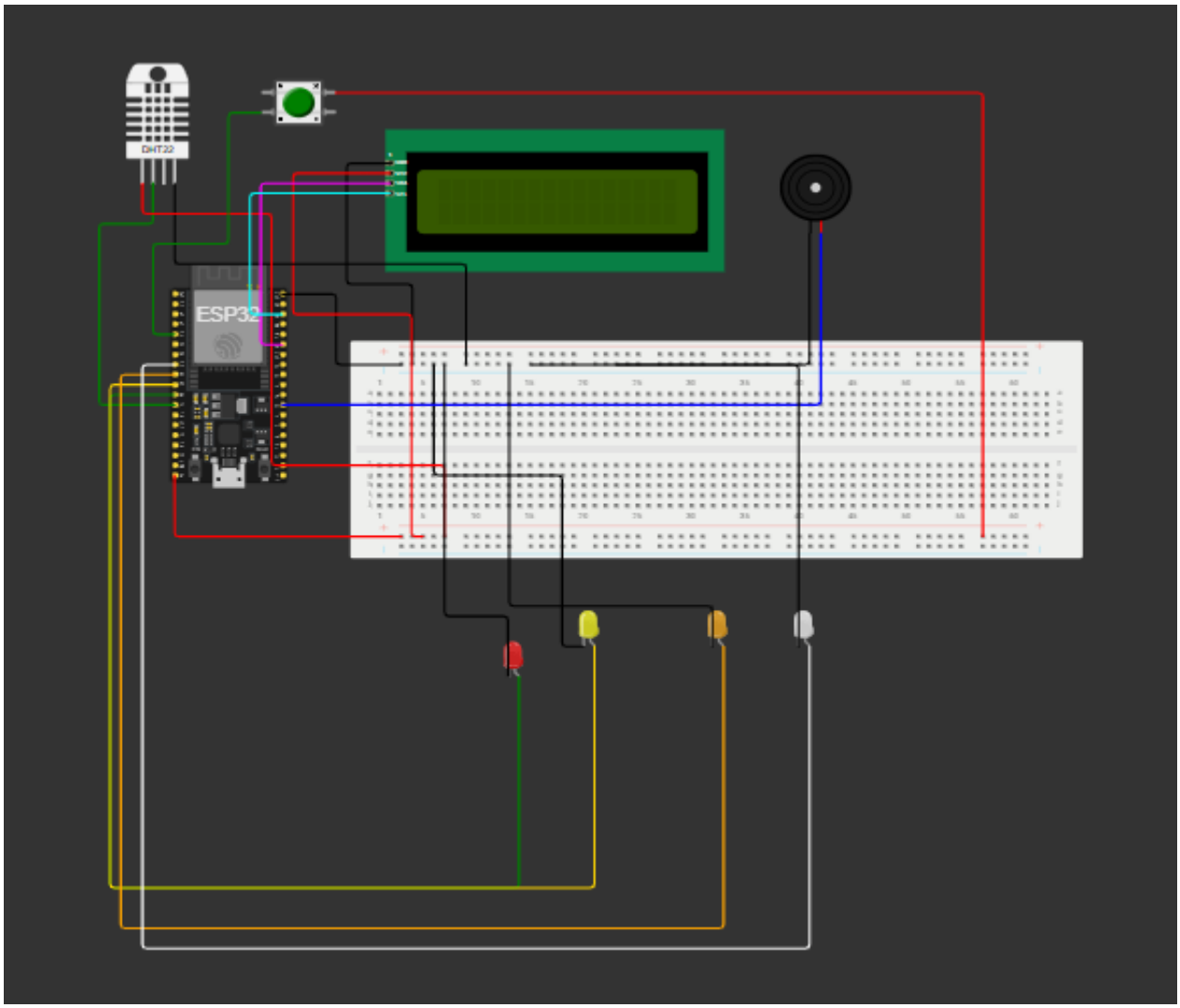
Hardware Components:

- ESP32 Microcontroller
- DHT22 (for monitors Temperature & Humidity Sensor)
- 16x2 I2C LCD Display (address: `0x27` and size: 16x2)
- 4 LEDs (used to display the instant heater's status.)
- Passive Buzzer
- GPIO Button (for interrupt)
- Jumper wires, breadboard

Block Diagram:



Circuit Diagram:



System Overview:

- The system continuously reads temperature and humidity from the DHT22 sensor.
- If temperature exceeds the overheat threshold (70°C), the system activates an audible alert via the buzzer and turns on the RED LED.
- The user can press a button to temporarily disable the alert, which activates manual cooling mode for 10 seconds.
- After 10 seconds, the system reverts back to auto mode, continuing normal monitoring.
- The current temperature, humidity, and system mode (AUTO/MANUAL) are displayed on the LCD.

Default Mode:

(I mean the system when it running without an interrupt)

Condition	State	LED	Buzzer
temp < 30°C	Idle	LED4	OFF
30°C ≤ temp < 50°C	Heating	LED3	OFF
50°C ≤ temp < 70°C	Target Reached	LED2	OFF
temp ≥ 70°C	Overheat	LED1	ON

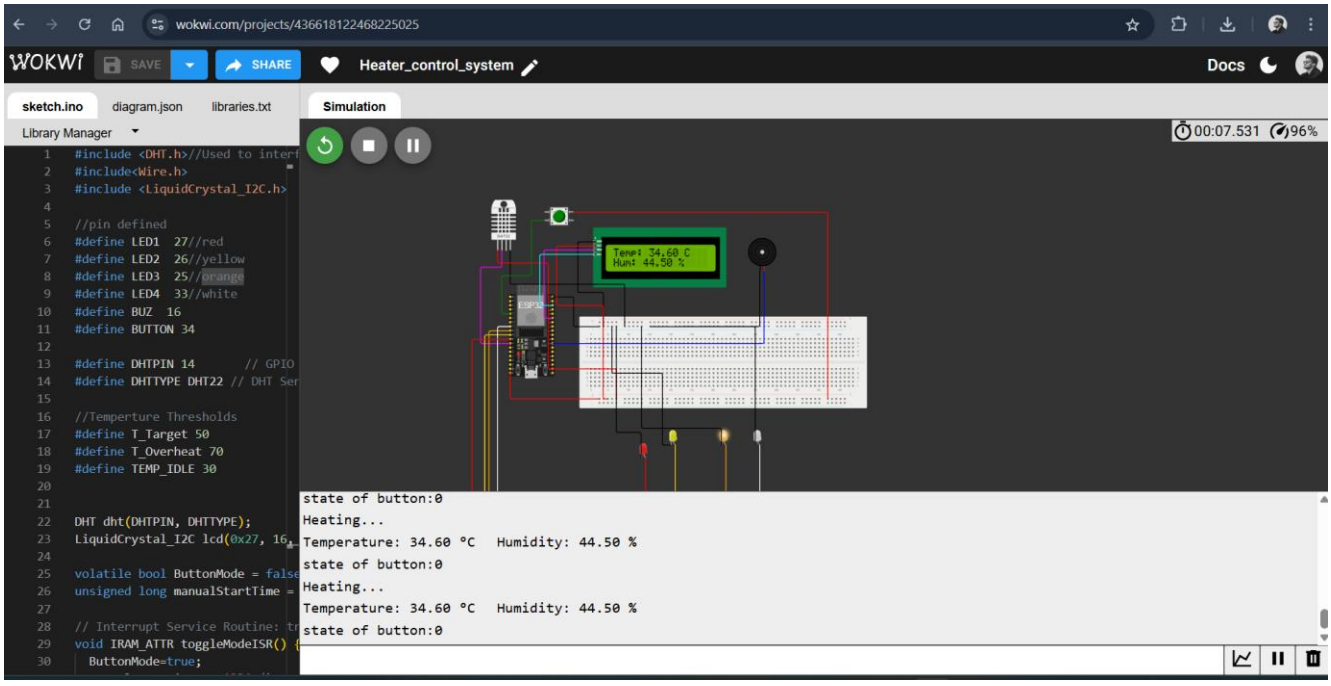
- LCD displays live temperature & humidity values.
- Buzzer beeps at 1khz when overheating to alert the user.
- All transitions and logs will be displayed in the Serial Monitor.

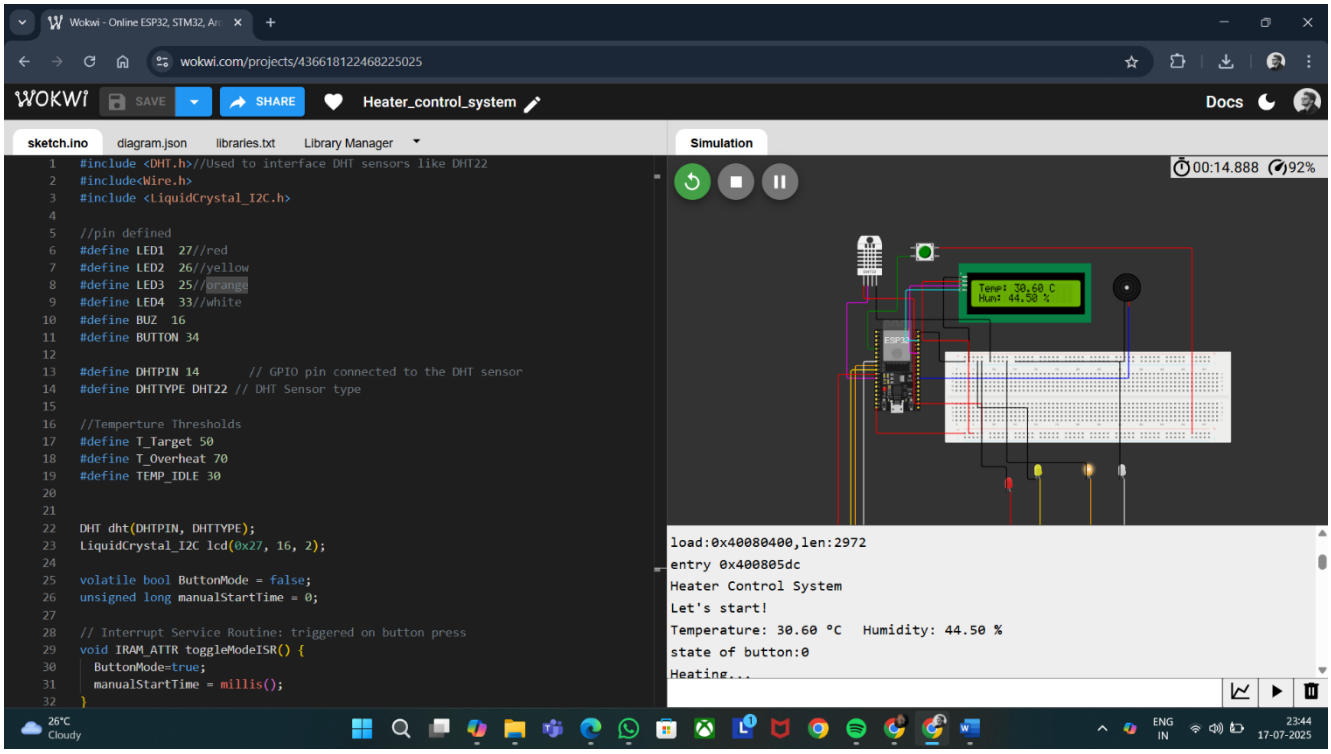
Interrupt Mode:

(When the button is pressed, this interrupt mode is triggered to manually cool down the heater, especially if it's in an Overheat state.)

- A button is connected to GPIO pin 34.
- When the button is pressed (detected by a falling edge interrupt), it activates Interrupt Mode.
- In Interrupt Mode:
    - A flag Button_Mode is set to true for 10 seconds.
    - All LEDs are turned off.
    - The buzzer is disabled.
    - The LCD displays the message: Mode: Cooling the system.
- After 10 seconds, the system exits Interrupt Mode and resumes normal operation.

Output Images:





Video link:
https://drive.google.com/file/d/18OcdW_Zzb6vchuQTLbv9jZY5_czGPiH/view?usp=sharing

Future Expansion Ideas:

- **Broadcast Data via BLE**
  Use Bluetooth Low Energy (BLE) advertising to share real-time temperature and operating mode with nearby devices.
- **Task Separation with Free RTOS**
  Implement independent tasks for sensor reading, display updates, and buzzer control using FreeRTOS, ensuring smoother multitasking.
- **Custom Heating Profiles**
  Add selectable heating modes like Eco, Fast Heat, or Comfort, to provide energy-efficient or high-performance options based on user needs.

Project Summary:

- This project showcases a real-time heater control system with features like user interaction, mode switching, and safety mechanisms (e.g., overheat protection).
- Developed entirely in C++ using the Arduino Framework, it runs seamlessly on both the Wokwi simulator and actual ESP32 hardware.

Code:

#include <DHT.h> //To using temperature sensor like DHT22, DHT11

#include<Wire.h>//This is the I2C communication library provided by Arduino.

#include <LiquidCrystal_I2C.h> //This is a specialized library for controlling LCD displays with I2C interface.

```
//pin defined
#define LED1  27//red
#define LED2  26//yellow
#define LED3  25//orage
#define LED4  33//white
#define BUZ  16
#define BUTTON 34
#define DHTPIN 14       // GPIO pin connected to the DHT sensor
#define DHTTYPE DHT22 // DHT Sensor type

//Temperture Thresholds
#define T_Target 50
#define T_Overheat 70
#define TEMP_IDLE 30

DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);
volatile bool ButtonMode = false;
unsigned long manualStartTime = 0;
// Interrupt Service Routine: triggered on button press
void IRAM_ATTR toggleModeISR() {
  ButtonMode=true;
  manualStartTime = millis();
}
void setup() {
  Serial.begin(115200);
  dht.begin();
  Serial.println("Heater Control System");
  Serial.println("Let start.....!");
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
```

```
  pinMode(BUZ, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);
 // Attach interrupt on button pin (rising/falling)
 attachInterrupt(digitalPinToInterrupt(BUTTON), toggleModeISR, CHANGE);
// Initialize LCD
  lcd.init();
  lcd.backlight();
  lcd.clear();
  }
void loop() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();
 // Check for sensor read errors
  if (isnan(temp) || isnan(hum)) {
    Serial.println("Failed to read from DHT sensor!");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Sensor Error");
    return;
  }

  //Print readings on Serial Monitor
  Serial.print("Temperature: ");
  Serial.print(temp);
  Serial.print(" °C\tHumidity: ");
  Serial.print(hum);
  Serial.println(" %");
  Serial.print("state of button:");
  Serial.println(ButtonMode);
//Display readings on LCD
  lcd.setCursor(0, 0);
  lcd.print("Temp: ");
  lcd.print(temp);
  lcd.print(" C");
```

```
lcd.setCursor(0, 1);
 lcd.print("Hum: ");
 lcd.print(hum);
 lcd.print(" %");
 if (ButtonMode) {
  Serial.println("Manual Mode: Heater ON");
  lcd.clear();
  lcd.setCursor(1,0);
  lcd.print("System In");
  lcd.setCursor(1,1);
  lcd.print("Cooling...");
  // Turn everything OFF
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, LOW);
  digitalWrite(LED3, LOW);
  digitalWrite(LED4, LOW);
  noTone(BUZ);
  delay(500);
  if (temp<T_Overheat || millis() - manualStartTime > 10000) {
    lcd.clear();
    ButtonMode = false;
    Serial.println("Switched back to AUTO Mode");
  }
 }
 else {
  //normal mode
  if (temp >= T_Overheat) {
    // Overheat
    digitalWrite(LED1, HIGH);//red
    digitalWrite(LED2, LOW);
    digitalWrite(LED3, LOW);
    digitalWrite(LED4, LOW);
    tone(BUZ, 1000);
    delay(200);
```

```
  noTone(BUZ);
  delay(200);
  Serial.println("Overheat!");
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print(" Over heat!");
  lcd.setCursor(0,1);
  lcd.print("push the button");
  delay(200);
}
else if (temp >= T_Target && temp < T_Overheat) {
// Target Reached
digitalWrite(LED1, LOW);
digitalWrite(LED2, HIGH);//yellow
digitalWrite(LED3, LOW);
digitalWrite(LED4, LOW);
noTone(BUZ);
Serial.println("Target Reached");
}
else if (temp >= TEMP_IDLE && temp < T_Target) {
 // Heating
digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED3, HIGH);//orage
digitalWrite(LED4, LOW);
noTone(BUZ);
Serial.println("Heating...");
}
else {
// Idle
digitalWrite(LED1, LOW);
digitalWrite(LED2, LOW);
digitalWrite(LED3, LOW);
digitalWrite(LED4, HIGH);//white
```

```
    noTone(BUZ);
    Serial.println("Idle");
    }
  }
  delay(500);
}
```