Embedded Systems Intern Assignment - upliance.ai

Name: 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.

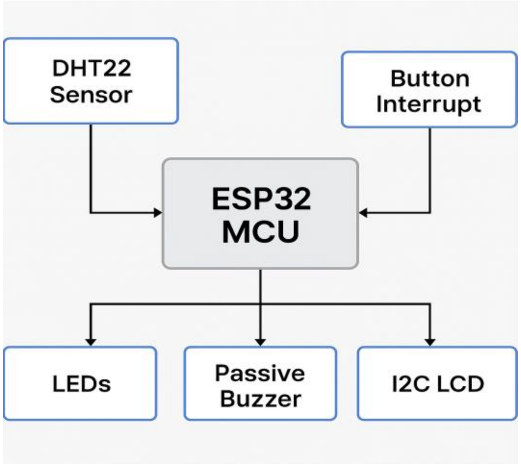
Project Overview:

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.

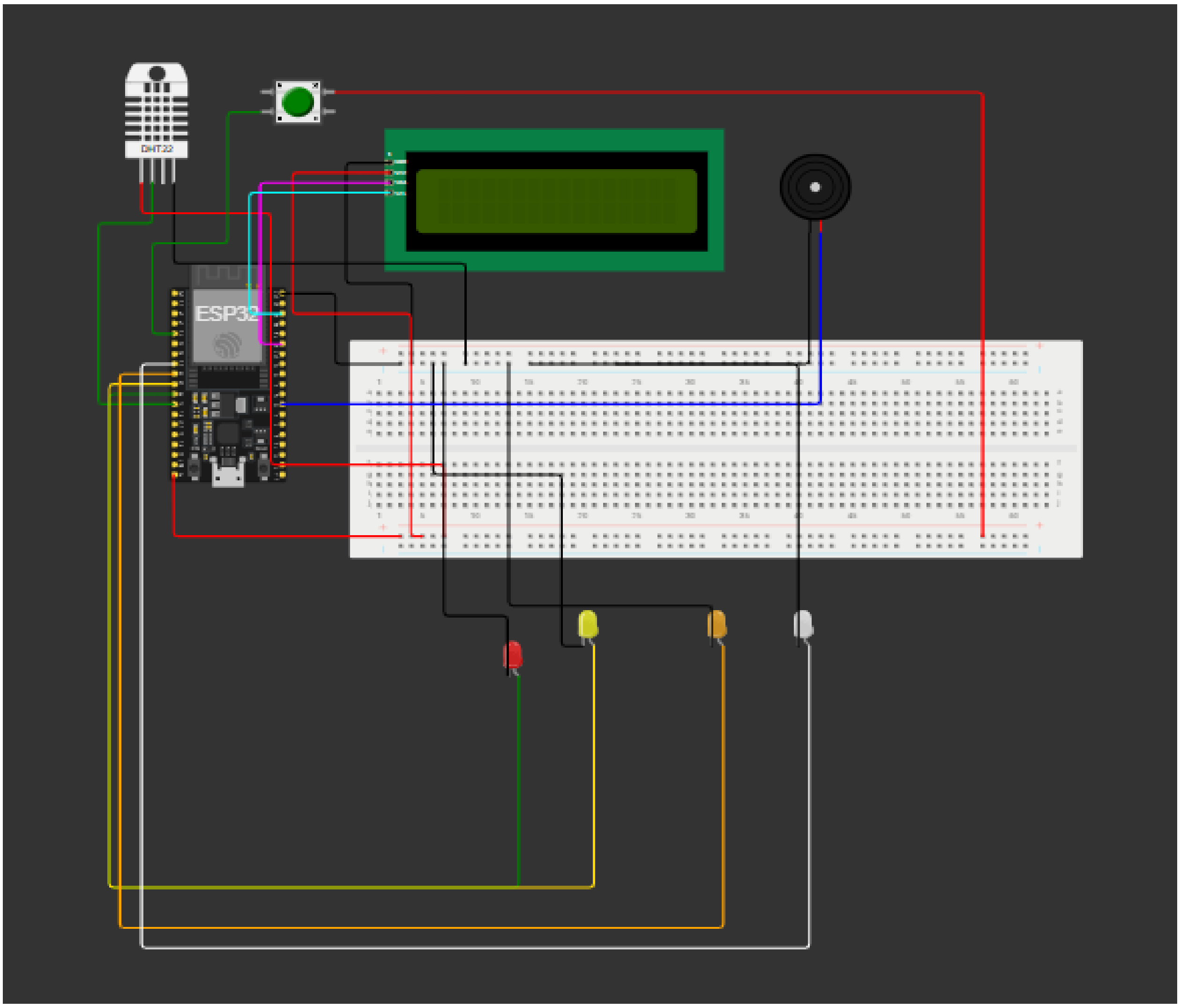
Hardware Components:

- 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 ON	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 are logs will be display in 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:
  - o A flag Button_Mode is set to true for 10 seconds.
  - o All LEDs are turned off.
  - o The buzzer is disabled.
  - o The LCD displays the message: Mode: Cooling the system.
- ➢ After 10 seconds, the system exits Interrupt Mode and resumes normal operation.

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.