Heater Control System Design

Aim - To design and simulate a Heater Control System that turns the heater “on” or “off” based on the temperature thresholds.

Components -

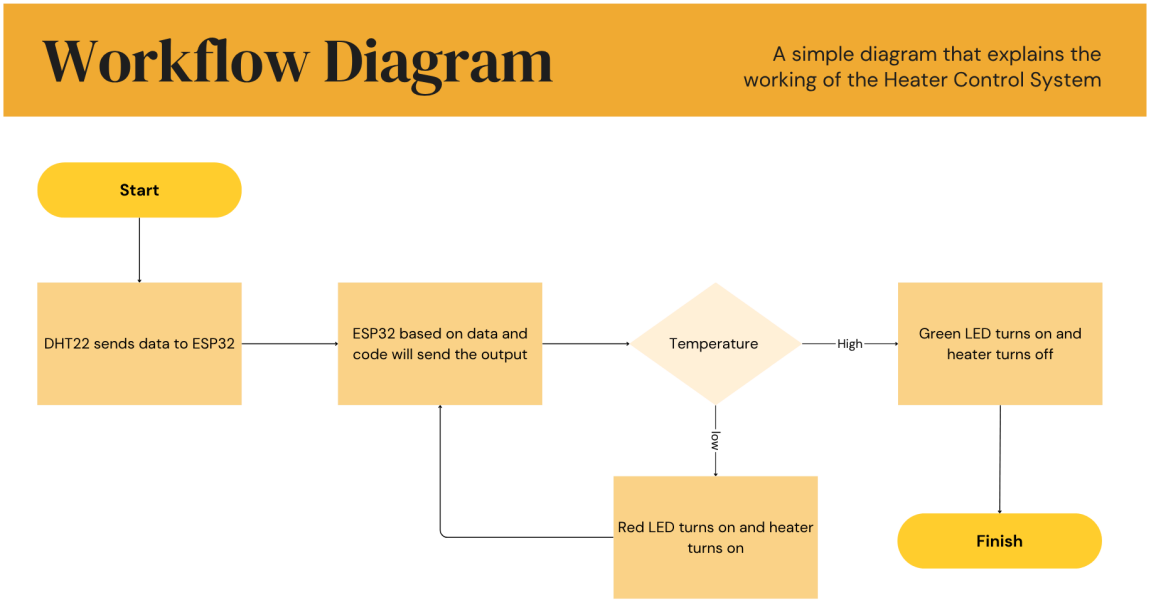
- ESP32
- DHT22 (Temperature sensor)
- LED
- Buzzer

Design -

This Embedded System Design involves no actual components but a real-time simulation using Wokwi. The idea behind the system is basic , the ESP32 is the brain of the system , it reads the data from DHT22 sensor (Temperature Sensor). The buzzer and LED acts as visual and acoustic feedback from the system signalling possible conditions and warnings.

Clarifications -

1. Identify the minimum sensors required for heating detection and control
    a. DHT22 acts as the temperature sensor for the system.
    b. For control , ESP32 microcontroller is being used as further updation can be done easily as ESP32 supports both WIFI and Bluetooth.
    c. One sensor would be enough but adding another sensor gives us a chance to identify possible errors or even smoothen the system’s performance.
2. Recommend a communication protocol and justify the choice
    a. For this system a Digital Single-Wire Protocol would be suitable
    b. This protocol is being chosen as -
        i. It is simple to implement, no complex wiring
        ii. Has a good resistance to noise
        iii. No ADC required as the sensor sends pre-processed signals
        iv. Using digital protocol gives us more accuracy without calibration
3. Provide a block diagram showing key modules-



4. Outline a future roadmap: How the system could evolve to support overheating protection, multiple heating profiles.
    a. This system is really basic and simple in nature but it works overall
    b. The system can be improved in multiple aspects
        i. Software Aspect-
            1. Handle missing data without returning the whole loop
            2. Build an app or webpage that supports the control system, can make use of ESP32's WIFI and Bluetooth properties for sending data to the cloud for storage and further processing.
            3. Allow different target temperatures based on context.That is , use different temperature standards for different times of the day and different days of the year.(Multiple Heating Profile)
            4. Build an alert system that would send the error log to the cloud during the alert period making sure that all the internal process are being logged for future reference
        ii. Hardware Aspect-
            1. Critical temperature cut-off, that is if temperature exceeds , the system shuts down
            2. Use OLED display to show real-time info of the system
            3. Make use of WIFI and Bluetooth for IoT applications and send alerts through SMS and other methods.
            4. Backup sensor (NTC) for redundancy
            5. Hardware cutoff: physical relay disconnected above a critical temp.