# **Smart Thermostat System Using Raspberry Pi**

## **Objective**

The primary aim of this module was to develop a smart thermostat system based on Raspberry Pi with other real-time components such as the LCD display, LED indicators, input buttons, and temperature and humidity sensor (AHT20). The solution is a basic heating-cooling controller, enhanced with visual feedback and user interaction by a series of buttons.

## **System Overview**

The whole thermostat setup is designed using the Raspberry Pi and a 16x2 character LCD which can display date, time, temperature, and system state. Heating is represented with a red LED, while a blue LED is used to indicate cooling. The three push-button switches allow the user to toggle system modes (OFF, HEAT, COOL) and set the set point temperature. The sensors used to record the room ambient temperature at the startup are called AHT20; this temperature will be used as a reference throughout the operation of the system

For a Raspberry Pi thermostat lab project with two status LEDs indicating heating or cooling and output to the thermostat serial port (UART), here are some potential project artifacts you might create or utilize:

**Hardware Artifacts:**

Raspberry Pi: The core of the project, running the thermostat logic.

Temperature Sensor: Such as a DHT22 or DS18B20, to provide temperature readings to the Raspberry Pi.

Two LEDs: One for heating status (red) and one for cooling status (blue or green).

Resistors: Appropriate resistors for the LEDs to limit current and protect the Raspberry Pi GPIO pins.

Relay Board: To control the heating and cooling system

Jumper Wires and Breadboard: For connecting the components.

Power Supply: For the Raspberry Pi and potentially the relay board.

Connectors: For connecting to the thermostat serial port (UART).

Software Artifacts:

**Raspberry Pi Operating System to run the thermostat program.**

**Python Scripts:**

Temperature Reading Script: To interface with the temperature sensor and get temperature data.

Thermostat Control Script: To implement the thermostat logic, compare current temperature with the setpoint, and determine if heating or cooling is needed.

GPIO Control Script: To control the LEDs (turning them on/off) and potentially the relay board based on the thermostat control script's output.

Serial Communication Script: To send status or data to the thermostat's serial port (UART).

Configuration Files: For setting up the Raspberry Pi's GPIO pins, enabling UART, and potentially setting thermostat parameters.

Documentation/Visual Artifacts:

Wiring Diagram/Schematic: Illustrating how the components are connected to the Raspberry Pi.

Flowchart or Pseudocode: Detailing the thermostat control logic.

Code Documentation: Explaining the functionality of the Python scripts..

Photographs or Videos: Of the hardware setup and the system in operation.

UART Communication Artifacts:

UART Configuration: Details on the baud rate, data bits, parity, and stop bits for the serial port communication.

.

## **Technical Implementation**

**Key components and features:**

* AHT20 Sensor (I2C): Reads room temperature straight away when the system is initialized.
* System States: Controlled through a button (GPIO 21) with OFF, HEAT, and COOL modes.
* Temperature Control: Two buttons (GPIO 22 and GPIO 27) can be used for simulating room temperature (for testing purposes).

LED Feedback:

A circuit board with wires

AI-generated content may be incorrect.

* HEAT mode: the Red LED is turned on but will fade as room temperature is lower than the set point, becoming solid once above or equal to it.
* COOL mode: the Blue LED behaves similarly, indicating that it is cooling.

LCD display: First row displays real-time clock while the second-row toggles between temperature and system state with the set point.

UART logging: Logs the system to state room temperature and set point every 30 seconds for monitoring purposes.

A circuit board with wires

AI-generated content may be incorrect.

## **Reflections**

This project entailed a lot of direct engagement in combining digital and PWM inputs and outputs, time-basis logic, and I2C communications through Python with Raspberry Pi.

Using real-time conditions (such as reading sensors and simulating heating/cooling states) reinforced state machine concepts and embedded control logic. Interaction with buttons and LED transitions greatly aided in visualizing system behavior. Furthermore, modular function usage alongside condition-based logic is good programming practice for embedded systems.

## **Aspects done particularly well in implementing this feature would likely include:**

## Clear Indication: Using distinct colors (e.g., red for heating, blue for cooling) to clearly differentiate the system's current state.

## Prompt Response: The LEDs activating or deactivating quickly and accurately reflect changes in the heating or cooling mode, showing that the software logic is working correctly.

## Simple Implementation: Using readily available components like LEDs and resistors, and connecting them to the Raspberry Pi's GPIO pins, is a straightforward approach for a lab project.

## Effective Code Integration: The code controlling the LEDs is integrated seamlessly with the thermostat logic to ensure the LEDs accurately reflect the system's operational status

## **Where could I improve:**

## Improved Functionality:

## Automatic Heat-Cool Switchover: Add logic to automatically switch between heating and cooling based on the target temperature and current room temperature.

## Real-Time Clock (RTC): Incorporate an RTC module with battery backup to ensure accurate timekeeping and scheduling even when the Raspberry Pi loses power.

## External Switch for Safety: Add a switch to disconnect the 3.3V supply to the solid-state relays, allowing you to disable HVAC control without powering off the Raspberry Pi.

## Units Selection: Allow users to choose between Fahrenheit and Celsius for temperature display.

**What tools and/or resources are you adding to your support network?**

When working on a Raspberry Pi project like this, your support network should include:

Online Communities and Forums:

Raspberry Pi Forums: A great place to ask questions and find solutions to common issues.

Dedicated project forums: Search for forums related to Raspberry Pi thermostat or HVAC control projects.

General electronics and maker forums: Sites like Stack Exchange and Reddit can be helpful for general questions.

Documentation and Tutorials:

Official Raspberry Pi Documentation: Essential for understanding the GPIO pins and programming.

Project-specific tutorials: Look for tutorials on building Raspberry Pi thermostats, especially those that include LEDs.

Datasheets for components: Make sure you have datasheets for your temperature sensor, relay module, and LEDs.

Software and Libraries:

RPi.GPIO library: The standard Python library for controlling GPIO pins.

Libraries for your sensors: Ensure you have the necessary libraries for reading data from your temperature sensor (e.g., Adafruit\_DHT for a DHT22).

Hardware and Tools:

Soldering iron and supplies: For making reliable connections.

Breadboard and jumper wires: For prototyping and testing circuits.

Multimeter: For testing voltage and continuity.

Open-Source Projects:

GitHub repositories: Explore projects shared on GitHub for inspiration and code examples.

**What skills will be particularly transferable to other projects and/or course work**

**GPIO Control and Interfacing:**

Understanding how to control hardware components (like LEDs) using the Raspberry Pi's GPIO pins: This is a fundamental skill for any project involving the Raspberry Pi or similar microcontrollers, where you need to interact with external sensors, actuators, or displays.

Applying basic electrical concepts: You'll learn about connecting components safely and correctly, including using resistors to limit current for LEDs.

2. Programming for Hardware:

Utilizing Python (or another relevant language) to control GPIO: This project will give you hands-on experience using libraries that provide access to Raspberry Pi's GPIO pins, a highly sought-after skill for electronics projects.

Developing logic to control system behavior based on inputs (e.g., temperature) and trigger outputs (e.g., LED state): This translates directly to any project involving sensor readings, decision-making, and controlling outputs.

3. Sensor Integration:

Interfacing with a digital thermometer (DS18B20): You'll learn how to connect and read data from a common temperature sensor, a skill applicable to a wide range of projects involving data acquisition and environmental monitoring.

**How did you make this project maintainable, readable, and adaptable?**

**Modular Code Design:**

* Separate Components: Structure the code into logical modules for different functionalities.
  + Sensor Handling: A module for reading temperature data from the sensor (e.g., using a library like Adafruit\_DHT).
  + Thermostat Logic: A module containing the core logic for deciding when to heat or cool based on the set point and current temperature.
    - Functions and Classes: Use functions and classes to encapsulate related code, making it reusable and easier to understand. For example, a function to turn the heating LED on or off, and a class to represent the thermostat with its current state and set point.

**Readable Code:**

* Meaningful Variable and Function Names: Use descriptive names for variables, functions, and classes that clearly indicate their purpose.

**Adaptable Architecture:**

* Configuration Files: Use a configuration file (e.g., JSON, INI) to store settings like the desired temperature range, hysteresis, and GPIO pin assignments. This allows easy customization without modifying the core code.
* Testing: Implement unit tests for individual modules and integration tests to verify the system's overall functionality.

**Utilizing Libraries and Tools:**

* Relevant Libraries: Use established libraries (e.g., RPi.GPIO for GPIO control) to simplify development and leverage existing solutions.

**Example Implementation Details**:

* Temperature Sensor: Using a DHT22 sensor with the Adafruit\_DHT library. Alternatively, utilize an Arduino board like a Pro Trinket to handle temperature readings and communicate via serial with the Raspberry Pi to offload the timing-critical task.
* Relay Module: Controlling a relay module connected to the heating/cooling system via GPIO pins.
* LEDs: Connecting LEDs to separate GPIO pins to indicate heating or cooling status.
* Configuration: Storing the set point, hysteresis, and other parameters in a JSON file that can be easily modified.

\*\*

## References

<https://forums.raspberrypi.com/viewtopic.php?t=191336>

<https://forums.raspberrypi.com/viewtopic.php?t=359930>

https://ve2zaz.net/RasTherm/RasTherm.htm