Alexandria Marbury

Professor Asadi

October 17, 2025

Smart Thermostat Prototype Report

This project demonstrates the development of a smart thermostat prototype that satisfies the functional and technical requirements outlined by SysTec’s engineering department. The prototype was created using a Raspberry Pi solderless bread board with various peripheral components, including LEDs, buttons, an LCD display, a temperature sensor (AHT20), and a UART interface.

The goal of this prototype is to simulate low-level thermostat functionality before integrating Wi-Fi and cloud connectivity in a future development phase. The thermostat automatically manages heating and cooling behaviors, responds to user input, displays system data, and communicates status information via serial output.

The thermostat operates around a state machine that transitions between three modes including off, heat, and cool. These states determine how the system reacts to sensor input and user interaction. The default temperature set point is 72°F. The thermostat continuously reads the room temperature using the AHT20 sensor and compares it to the set point to decide whether to heat or cool. LEDs, an LED display, and serial communication are used to indicate and log system activity. The display shows the current date and time, state, and setpoint. On another screen, it depicts the date and time and the current temperature.

Multiple hardware components were composed together. The Raspberry Pi acts as the central controller and runs the thermostat logic as well as it houses the terminal to execute the code for the system. To read the temperature, the AHT20 Temperature Sensor is used as an inter-integrated circuit. The red and blue LED are used to indicate the heating and cooling state. If the system is actively heating or cooling, their respective LED blinks. Otherwise, the LEDs stay solid. The push buttons work to control the system’s state and temperature set point. If the set point is set above or below the current temperature, depending on the state, this is the factor that causes the respective LEDs to blink or stay solid. The LED display shows the time, date, temperature, state, and set point. The UART Serial Output is a hardware component as well that handles serial communications. It outputs system state data every 30 seconds for logging and analysis.

The program starts by initializing the LEDs, buttons, LCD, UART, and AHT20 sensor. It sets the default temperature to 72°F. It enters a loop that continuously reads the current temperature of the AHT20 sensor, compares the temperature to the set point, adjusts the LEDs and displays output according to the input, and transmits data over the UART every 30 seconds. The user sets input by toggling the state to heat, cool, and off using the green button, as well as increase or decrease the set point by 1 degree using the red and blue buttons respectively. For the heat mode, if the temperature is less than the set point, the red LED blinks, otherwise it stays solid. For the cool mode, it the temperature is greater than the set point, the blue LED blinks, and otherwise stays solid. For the off mode, both LEDs are off. The LCD display shows the date and time in the first line and the second line alternates between the current temperature and the thermostat’s state and the set point. Every thirty seconds, the thermostat sends a comma-delimited string that includes the state, current temperature, and the set point.

The next development stage involves enabling Wi-Fi connectivity so that the thermostat can send temperature data and state updates to SysTec’s analytics servers in real time. For the production version, the Microchip SAMW25 or ESP32-based module would be ideal due to its integrated Wi-Fi, low power usage, and compatibility with the peripherals used in the prototype (Microchip Technology Inc., 2024). It offers sufficient Flash and RAM to support network communication and cloud integration while maintaining cost efficiency. While the Raspberry Pi is easy for prototyping, it is best for the developmental stages since it consumes more power, and due to its unnecessary complexity of a full operating system. As for the Freescale/NXP, while it has high performance compared to the others, it’s more complex and requires more extensive integrations for Wi-Fi (NXP Semiconductors, 2024).

References

Microchip Technology Inc. (2024). *SAMW25 Smart Connect Module: Product overview and technical documentation.* Retrieved from <https://www.microchip.com>

NXP Semiconductors. (2024). *Kinetis microcontrollers and connectivity solutions overview.* Retrieved from https://www.nxp.com