Nicholas Wyrwas

CS-350-13290-M01

Mr. Eric Gregori

08/19/2025

7-1 Final Project Report

The Raspberry Pi-based thermostat prototype was created as a way to showcase SysTec the capabilities of the low-level embedded system solution. The code for the prototype was implemented using the Python language and was bootstrapped into the Raspberry Pi 4B. The device and its peripherals consist of a temperature and humidity sensor, three push buttons, two LEDs, and an LCD. The thermostat itself is a state machine with three states: off, heat, and cool. The buttons are used to either transition between the thermostat states (button 1) or to raise or lower the temperature setpoint by one degree Fahrenheit (buttons 2 and 3). The user experience and design of the device were dictated by the need for the user to easily interact with the thermostat and be able to see both its state and environmental conditions in real time.

The temperature is read using the AHT2 connected through the I²C bus. The values from the sensor are converted to F and compared to the setpoint to determine if the thermostat should call for heat or cool. The LEDs are used to indicate the system state: red for heat, blue for cool. The indicator lights for heating and cooling both fade in and out while the thermostat is actively calling for heating or cooling, and are on solid when the desired setpoint has been reached. The LEDs are both disabled when the thermostat is in the off state. The LCD provides dynamic feedback to the user. Line 1 of the display shows the date and time in a format with month, day, and the value of the internal clock. Line 2 shows alternating information between the current room temperature and the active state of the thermostat, including the setpoint value. These two lines are designed to alternate to make both the environmental conditions and the state of the thermostat visible to the user at all times.

The Raspberry Pi UART interface is used to periodically provide an update string to a server to integrate data from the thermostat with SysTec’s back-end systems. The output string is a simple comma-delimited set of values, including state, the current room temperature in F, and the setpoint value. This output design was chosen for its simplicity and as a foundation to easily integrate the thermostat with a cloud-based analytics solution in the future. Python was chosen for the implementation language, and several libraries were used to support hardware access and integration of the peripherals. The gpiozero library was used to interface with the buttons and LEDs. The adafruit\_character\_lcd and adafruit\_ahtx0 libraries were used to support the LCD display and AHT2 sensor, respectively. Threads were added to support display updating in parallel with the main thermostat logic.

The architecture of the device is entirely Python-based and is very modular, supporting both responsiveness and user interaction. The device does, however, require Raspberry Pi for execution. The Raspberry Pi is a powerful and highly flexible platform, but it is not the best solution to the problem long term. The main downside to using a Raspberry Pi is that it is both high powered for a thermostat device and has a high unit cost. The preferred alternatives to the Raspberry Pi are a Microchip or Freescale (NXP) microcontroller. These platforms both have built-in Wi-Fi support, low energy requirements, and have adequate Flash and RAM to support the code for the thermostat. The Microchip microcontrollers are the preferred option in this case. These have a history of supporting embedded solutions and have solid support for Wi-Fi. The performance characteristics of the Microchip offerings are also solid, providing all of the hardware necessary to support the thermostat code at a lower cost than the other options.

The results of this project show a functioning thermostat system with temperature sensing, state management and control, user feedback through LEDs and LCD, and output through UART. This validates the design, confirms that it is a solid solution for the requirements SysTec, and shows that the design choices made throughout have yielded working and reliable results. The Raspberry Pi was a solid choice for a prototyping platform, but for commercial production, a Microchip microcontroller is the best choice based on its power efficiency, cost benefits, and built-in Wi-Fi.