SysTec is targeting a fast‑growing smart‑thermostat market, expected to reach nearly $9 billion by 2026. To get started, our team built a working prototype on a Raspberry Pi 4B. This prototype meets SysTec’s core requirements; it reads room temperature, lets users switch modes and adjust the set point, shows status on LEDs and an LCD, and simulates cloud data streaming over UART.

The heart of the system is an AHT20 temperature sensor connected via I²C. In software, we convert its Celsius reading to Fahrenheit for consistency with U.S. thermostats. Two PWM‑capable GPIO pins drive red and blue LEDs. When in heating mode and the room is cooler than the target, the red LED smoothly fades in and out, then stays solid when the set point is reached. Cooling mode works the same way with the blue LED. Three buttons tied to GPIO interrupts provide user control: one cycles through Off, Heat, and Cool, and the other two raise or lower the temperature by one degree. A separate display thread updates the 16×2 LCD once per second, showing the date and time on the first line. Every five seconds, it alternates the second line between the current temperature and the active mode with its set point. Every thirty seconds, the Raspberry Pi packs the current mode, set point, and measured temperature into a comma‑separated string and sends it over UART to simulate cloud data upload.

For production, we need integrated Wi‑Fi and a lower‑power, cost‑effective platform. We evaluated three options; continuing with the Raspberry Pi 4B, moving to a Microchip PIC32MZ2048EFH144 microcontroller, or using an NXP Kinetis MK66FX1M0VLQ18 MCU. The Pi 4B offers built‑in 802.11ac Wi‑Fi, abundant I²C and GPIO pins, and SD‑card storage (effectively unlimited flash) with 1–8 GB of RAM. Its downsides are higher power consumption (around 3–5 W) and the complexity of a full Linux OS. The PIC32MZ provides 2 MB of on‑chip Flash, 512 KB of RAM, multiple I²C and PWM channels, and real‑time determinism. This is enough headroom for our code plus future features. It needs an external Wi‑Fi module (such as Microchip’s MRF24WG0MA), which increases the bill of materials but keeps power draw under 100 mW. The Kinetis MCU has 1 MB Flash and 256 KB RAM, which is enough for core functionality but leaves less room for expansion, and likewise requires a separate Wi‑Fi transceiver.

All three architectures support our peripheral set: I²C for the AHT20, PWM for LEDs, GPIO interrupts for buttons, and UART for data output. In terms of flash and RAM, our state machine and display code occupy under 200 KB of flash and about 50 KB of RAM, fitting comfortably on the PIC32MZ and the Kinetis device, and trivially on the Pi’s SD card. Cloud connectivity is simplest on the Pi, since its Linux stack already includes TCP/IP and Wi‑Fi drivers. The MCU paths demand additional firmware layers for TCP/IP over SPI or SDIO to the Wi‑Fi chip, adding complexity and development time.

Given these trade‑offs, our recommendation is to continue prototyping on the Raspberry Pi 4B for rapid development and early testing. For the final product, we suggest a Microchip PIC32MZ series microcontroller paired with a compact, low‑power Wi‑Fi module. This combination offers real-time responsiveness and enough memory to support both current and future features. It also maintains low energy consumption, aligning with SysTec’s goals of reliable performance, cloud connectivity, and long-term competitiveness.

**Reference:**

Dinmohammadi, F., Farook, A. M., & Shafiee, M. (2025). Improving energy efficiency in buildings with an IoT-based smart monitoring system. *Energies, 18*(5), 1269. <https://doi.org/10.3390/en18051269>