This project aimed to build a prototype smart thermostat using the TI CC3220 LaunchPad, where the core functionality involved reading the room temperature, controlling a simulated heating system, and communicating relevant data via UART. The process began with setting up the TMP006 sensor to monitor the temperature and integrating buttons to adjust the set-point, representing the desired temperature, on the fly. The heating system was symbolized by an LED, which turned on when the room temperature fell below the set-point, and off when it exceeded it.

One of the most engaging aspects of this project was managing the peripherals. The I2C sensor posed an early challenge. Although simple in concept, the task of ensuring proper communication and accurate temperature readings required careful attention. The system was designed to scan multiple sensor addresses and select the correct one. Once this step was sorted, the sensor started providing reliable data, forming the foundation for the rest of the system.

GPIO buttons allow the user to increase or decrease the set-point temperature. It was essential that these button presses be detected promptly, without interfering with the other processes. The interrupt-driven approach worked well here, ensuring immediate response to the user’s input without disrupting the flow of temperature readings or data transmissions. The addition of the timer to control task intervals, specifically for reading temperature every 500 milliseconds, adjusting the heater state, and sending updates every second, was where the task scheduler became invaluable. Without it, managing all these tasks in a responsive and orderly manner would have been far more difficult.

The UART component of the project was another satisfying piece to develop. By sending data in a clear format, temperature, set-point, heater status, and time since startup, it simulated communication with a server, which would be a key feature in any real-world deployment. Each element of the system came together through careful timing and task management, ensuring that all functions worked in harmony.

However, there were inevitable challenges along the way. Fine-tuning the task scheduler took some effort, especially in ensuring that each process ran at its proper interval without delays or overlaps. It required constant adjustments, particularly when it came to balancing the I2C reads with the other tasks. Ensuring that no task blocked another while keeping the system responsive was a key lesson from this experience.

Looking back, there are several areas where this system could be expanded and improved. Connecting the thermostat to a real cloud-based server using Wi-Fi would take the project to the next level. This would open the door for remote control and monitoring, making the system more versatile and practical. Additionally, integrating a display to show the current temperature and set point directly on the device could enhance the user experience, providing more immediate feedback without relying solely on UART communication.

This project was an excellent exercise in embedded systems development, building up my experience with real-time data handling, peripheral communication, and task scheduling. Each challenge along the way offered valuable insights, from understanding the quirks of I2C sensors to ensuring that the system remained responsive under various conditions. As I reflect on the work completed, I’m satisfied with how the system met the key goals and demonstrated the practical application of embedded systems principles in a potential real-world scenario.