Lee Kitchen

CS-350  
14OCT2024

7-1 Project Report

The thermostat system operates through a state machine that handles various tasks and transitions between different operational states. Initially, the system enters the idle state, where it waits for a timer interrupt to trigger the next action. Once the timer is triggered, the system transitions into the button check state, where it checks the status of the button flags to determine if the set-point temperature needs to be updated. From there, the system moves to the temperature read state, where the temperature sensor is read, and the heater status and LED are updated accordingly. Next, the system enters the UART output state, where it communicates the current system status, including room temperature, set-point, heater status, and uptime, to the server via UART. After completing the UART output, the system returns to the idle state and awaits the next timer interrupt. The expected results of this system include specific behaviors for button handling, temperature reading, LED control, and UART output. When a button is pressed, the set-point temperature should increase or decrease by 1 degree every 200ms. The temperature sensor should be read every 500ms, and this data is used to control the heater. If the room temperature is below the set-point, the heater turns on, and the LED lights up; if the temperature is above the set-point, the heater turns off, and the LED is off. Every second, the system sends a report to the server, detailing the current room temperature, set-point, heater status, and uptime in the format <AA,BB,S,CCCC>, where AA is the room temperature, BB is the set-point, S represents the heater status, and CCCC is the system uptime.

The thermostat project supports multiple peripherals for its operation. The I2C peripheral is initialized and used to communicate with the TMP006 temperature sensor, which updates the system with temperature data every 500ms. The GPIO handles the LED and buttons, with the LED indicating the heater's state and the buttons used to adjust the set-point temperature. The UART peripheral is responsible for communicating with the server by sending status reports every second, while the timer peripheral drives the task scheduler, executing tasks at 200ms, 500ms, and 1000ms intervals.

When comparing different microcontroller architectures, Texas Instruments (TI), Microchip, and Freescale (now NXP) offer various strengths. TI’s CC3220S microcontroller provides 256KB of RAM and 1MB of flash memory, sufficient for embedded applications like this thermostat. It also includes built-in support for I2C, UART, GPIO, and timers, making it a strong choice for IoT applications. In contrast, Microchip’s PIC32MX series offers 512KB to 2MB of flash memory and 64KB to 512KB of RAM, which is suitable for small to medium-sized embedded projects, though its peripherals may require more manual configuration. Freescale’s Kinetis series offers up to 1MB of flash and 256KB of RAM, making it comparable to TI’s architecture, with low-power options ideal for IoT applications.

For cloud connectivity, the thermostat relies on Wi-Fi, with the TI CC3220S providing built-in Wi-Fi capabilities, including a network processor and TLS encryption support. This makes cloud integration seamless, requiring minimal external components. Microchip offers Wi-Fi solutions through separate modules that integrate with their MCUs, while Freescale’s Kinetis series often uses external Wi-Fi modules like the ESP8266 to provide cloud connectivity.

In terms of flash and RAM usage, TI’s CC3220S uses 1MB of flash memory to store the application code, Wi-Fi drivers, and cloud communication logic, while its 256KB RAM handles the task scheduler, sensor data buffers, and Wi-Fi stack. Microchip’s PIC32 series offers flash sizes ranging from 512KB to 2MB, which is used for application code and communication stacks, while the RAM handles application variables, stacks, and buffers. Similarly, Freescale’s Kinetis series provides up to 1MB of flash for code storage and 256KB of RAM for real-time data processing and communication stacks.

References

CC3220S. CC3220S data sheet, product information and support | TI.com. (n.d.). https://www.ti.com/product/CC3220S#tech-docs

K series cortex-M4. NXP Semiconductors. (n.d.). https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/general-purpose-mcus/k-series-arm-cortex-m4:KINETIS\_K\_SERIES

PIC32MX family of Microcontrollers | Microchip Technology. (n.d.). https://www.microchip.com/en-us/products/microcontrollers-and-microprocessors/32-bit-mcus/pic32-32-bit-mcus/pic32mx