7-1 Project Submission

Rui Costa

Southern New Hampshire University

CS-350-H7977 Emerging Sys Arch & Tech 23EW6

Professor Roland Morales

My Reflections on Thermostat Functionality and Hardware Architecture

Introduction

In this report, I will present my reflections on developing and analyzing a thermostat system. The goal was to create a system that reads room temperature, controls an LED indicator, allows temperature adjustments using buttons, and simulates data transmission to a cloud server. I developed the code using Code Composer Studio (CCS) IDE for this project. The project's main components included Timer, Interrupt, I2C, GPIO, and UART peripherals.

Code Functionality

I am delighted to report that the code I implemented flawlessly achieved all the specified functionality. Utilizing the I2C communication protocol, I could accurately read the room temperature and display the results via the UART. The LED indicator, representing the heating status, was expertly controlled by the GPIO peripheral, while the buttons allowed for easy adjustment of the desired set temperature. Thanks to the Timer, I efficiently handled all periodic tasks, ensuring the thermostat system operated seamlessly.

Timer

I set up the Timer to run continuously with a 1-second interval, which made it possible for the task scheduler to execute the temperature control and button handling tasks at their designated times. This timing was essential for the thermostat system to work correctly.

Interrupt

It was a fulfilling task to set up interrupts to recognize button presses. I enabled interrupt capabilities on the GPIO and established two callback functions named "gpioButtonFxn0" and "gpioButtonFxn1" to activate upon button presses. These functions successfully modified the "buttonUp" and "buttonDown" variables, enabling me to alter the desired temperature accordingly.

I2C Peripheral

I was able to initialize the I2C peripheral and utilize it to obtain temperature sensor data. The "readTemp()" function facilitated the required I2C transfer to retrieve the temperature from the sensor. The reading was then analyzed and exhibited through the UART.

GPIO Peripheral

I initialized and utilized the GPIO peripheral for controlling the LED and receiving input from a button. To represent the heating status, I set the LED pin as an output, while the button pins were configured as inputs with pull-up and interrupt on the falling edge.

UART Peripheral

I successfully initialized the UART peripheral and utilized it for transmitting data. By invoking the "initUART()" function, I established UART communication with particular settings, and then I employed the "DISPLAY()" macro to transmit information to the linked UART terminal. This enabled me to showcase the temperature, the designated temperature, heating status, and the duration since the beginning of the process through UART.

Task Scheduler Functionality

Implementing the task scheduler functionality was both challenging and fulfilling. By utilizing the "tasks" array of structures, I could schedule tasks based on their defined periods effectively. The "timerCallback()" function successfully executed tasks at their specified intervals. The temperature control and button handling tasks were performed sequentially and repeatedly as intended.

Best Practices

During the project, I ensured that I followed coding best practices by using clear and descriptive names for my variables and maintaining organized formatting.

Conclusion

To sum up, I am pleased with the code that has been implemented, as it meets all the functional requirements of the project. The thermostat system is running smoothly with temperature sensing, LED indicators, and the ability for users to set the temperature.