**Thermostat Prototype Project Report**

**Author:** Victor Udeh  
**Course:** CS350 - Embedded Systems  
**Date:** 16th Sept 2024

**Introduction**

The purpose of this project was to develop a prototype for a smart thermostat using the Texas Instruments CC3220S Launchpad, as part of SysTec’s initiative to enter the smart thermostat market. The thermostat prototype simulates low-level functionality, including reading the room temperature, adjusting the set-point temperature, and simulating communication with a server. The key components used in this project include a TMP006 temperature sensor, an LED to indicate heating status, GPIO buttons to adjust the set-point, and UART for server simulation.

**Functional Overview**

The system’s core functionality includes:

* **Temperature Reading (I2C):** The TMP006 temperature sensor was used to measure room temperature via the I2C protocol. The sensor readings were converted to Celsius and used in the thermostat control logic.
* **LED Output (GPIO):** A single red LED was used to represent the heating status of the thermostat. The LED turns on when the room temperature is below the set-point (indicating the "heater" is on) and turns off when the room temperature exceeds the set-point.
* **Button Interrupts (GPIO):** Two buttons were configured using GPIO interrupts to allow the user to increase or decrease the set-point temperature. Each button press adjusts the set-point by 1°C.
* **Server Simulation (UART2):** UART2 was used to simulate sending data to a server. The data is formatted as <AA,BB,S,CCCC>, where AA is the room temperature, BB is the set-point temperature, S is the heater status (0 or 1), and CCCC is the number of seconds since the system was powered on. This data is transmitted over UART2 every second.

**Task Scheduler**

The task scheduler ensures that the system functions at specific intervals:

* **Every 200 ms:** Check button inputs to adjust the set-point.
* **Every 500 ms:** Read room temperature via I2C.
* **Every 1 second:** Update LED status and transmit data to the simulated server via UART.

A diagram for the task scheduler was created in **draw.io** and represents the following flow: A diagram of a process

Description automatically generated

**Thermostat Peripheral Support**

The thermostat prototype successfully supports the following peripherals:

* **I2C for Temperature Sensor:** The TMP006 sensor is connected via I2C, allowing the thermostat to read real-time temperature data from the room.
* **GPIO for LED and Buttons:** The GPIO pins control the LED and read input from the two buttons, enabling user interaction and system output to indicate heater status.
* **UART for Server Simulation:** The UART peripheral simulates data transmission to the server by formatting and sending the thermostat’s status, including temperature and heater status, to a virtual server.

The integration of these peripherals enables the thermostat to function as a complete system that reads, processes, and outputs temperature data.

**Hardware Architectures: TI, Microchip, and Freescale**

For expanding this thermostat prototype to connect to the cloud via Wi-Fi, I evaluated three hardware architectures from Texas Instruments, Microchip, and Freescale, based on their ability to support Wi-Fi, peripherals, and required memory.

**Texas Instruments (TI) Architecture:**

* **Peripherals:** The TI CC3220S Launchpad supports all the required peripherals—GPIO, I2C, UART, and Wi-Fi connectivity via a dedicated network processor.
* **Cloud Connectivity:** The TI CC3220S has integrated Wi-Fi support with a network processor that manages Wi-Fi and internet protocols independently of the main application processor, allowing seamless cloud communication.
* **Flash and RAM:** It features 256KB of RAM and 1MB of Flash, which provides ample space to handle the thermostat's firmware, peripheral control, and Wi-Fi functionalities.

**Microchip Architecture:**

* **Peripherals:** Microchip’s ATWINC1510 wireless module can handle Wi-Fi, while the ATSAM family of microcontrollers provides peripheral support for I2C, UART, and GPIO.
* **Cloud Connectivity:** The ATWINC1510 module has built-in support for Wi-Fi, enabling a secure and reliable connection to the cloud.
* **Flash and RAM:** The ATSAM family typically offers 128KB to 256KB of Flash and 64KB of RAM, which could support basic thermostat functionalities but may be constrained when implementing more complex features like Wi-Fi security.

**Freescale (NXP) Architecture:**

* **Peripherals:** Freescale’s KW41Z family of microcontrollers supports I2C, GPIO, UART, and Bluetooth connectivity. However, Wi-Fi would require additional modules.
* **Cloud Connectivity:** Freescale doesn’t offer an integrated Wi-Fi solution. Cloud connectivity would require external modules, which could complicate the design.
* **Flash and RAM:** The KW41Z family provides up to 512KB of Flash and 128KB of RAM, which is sufficient for peripheral control but lacks integrated support for Wi-Fi.

**Flash and RAM Comparison:**

* **TI CC3220S**: 1MB of Flash, 256KB of RAM—more than sufficient for handling both the thermostat's local control and Wi-Fi communication.
* **Microchip ATWINC1510 + ATSAM**: The combined solution offers enough Flash and RAM but may be limited when complex Wi-Fi protocols or security features are required.
* **Freescale KW41Z**: Freescale offers more Flash and RAM than Microchip but lacks integrated Wi-Fi, which limits its suitability for the project.

**Conclusion**

This project demonstrated the development of a fully functional smart thermostat prototype using the TI CC3220S Launchpad. The system met all specified requirements for low-level functionality, including temperature reading, button inputs, LED output, and server simulation via UART. By overcoming challenges related to UART2 and ensuring proper task scheduling, I was able to deliver a working solution that supports peripherals, Wi-Fi connectivity, and sufficient memory for future cloud-based expansions.