**SNHU: CS-350 Emerging System Architectures and Technologies**

Module 7– Final Project

Prototype Reflection and Recommendations

Matthew Bandyk

## Table of Contents

1. Introduction
   1. Project Overview
   2. Purpose and Scope
   3. Project Requirements
2. Code Overview
   1. Description of the Implemented Code
   2. Task Scheduler Algorithm
   3. Inputs and Outputs
   4. Expected Results
3. Next Phase: Connecting the Thermostat to the Cloud
   1. Business Requirements
   2. Hardware Architecture Comparison
   3. Recommendation and Justification

# **1) Introduction**

**Project Overview**

This project aimed to develop a smart thermostat capable of monitoring room temperature, adjusting setpoints, and controlling a heating system while simulating sending data to a server. The project prototype utilizes the Texas Instruments (TI) SimpleLink platform, specifically the CC3220SF microcontroller, to implement the thermostat's hardware and software components.

**Purpose and Scope**

The primary purpose of this report is to discuss how the implemented code meets the project requirements. It also outlines the technical and operational documentation for the task scheduler used to fulfill the project's functionality. Furthermore, this report delves into the next phase of the project, which involves connecting the thermostat to the cloud.

**Project Requirements**

The project requirements are outlined as follows:

* Develop a thermostat capable of monitoring room temperature.
  + Implement a task scheduler for periodic operations.
  + Control a heating system based on temperature and setpoints.
  + Provide appropriate documentation for the task scheduler.
  + Recommend hardware architecture to utilize and connect the thermostat to the cloud via Wi-Fi.

**2) Code Overview**  
**Description of the Implemented Code**

The implemented code consists of firmware that runs on the TI microcontroller. It initializes various hardware components, including GPIO pins, UART for communication, I2C for temperature sensing, and a timer for task scheduling. The GPIO is utilized to capture button presses to set the desired temperature. The UART is utilized to display the current temperature reading, the current temperature set point, the status of the heater, and the number of seconds that have passed since the thermostat has been reset. It utilizes a state machine-driven task scheduler to manage periodic tasks using the timer that was initialized. To provide additional details, a state machine diagram was created and can be viewed to see the functionality of the task scheduler.

**Task Scheduler Algorithm**

The task scheduler is a state-based scheduler with five main states: Initialization, Idle, Button Press Check, Temperature Reading, and Report and LED Update. Each state is triggered by a timer interrupt. There is an interrupt every 100ms created by the timer which allows the system to successfully run the different states at 200ms, 500ms, and 1000ms. The scheduler manages the execution of tasks, including checking button presses, reading temperature, updating the LED, and reporting data over UART to simulate being sent to a server based on the time that has passed, executing each state utilizing the planned state periods.

**Inputs and Outputs**

The task scheduler takes inputs from external sources, including button presses and temperature readings. Outputs include controlling the heating system, updating LEDs, and reporting data via UART. The scheduler's outputs meet the project's requirements by providing real-time information about room temperature, setpoints, heating status, and time since the board was reset.

**Expected Results**

The code is expected to check for button presses every 200ms, provide accurate temperature readings every 500ms, control the heating system effectively based on temperature setpoint, and report data at 1-second intervals. The task scheduler ensures that each task is executed at its specified time interval, meeting the project's operational requirements.

# **3) Next Phase: Connecting the Thermostat to the Cloud**

**Business Requirements**

The next phase of the project involves connecting the thermostat to the cloud via Wi-Fi. This phase has the following business requirements:

* The thermostat must support the peripherals used in the project.
* The thermostat must connect to the cloud via Wi-Fi.
* The chosen architecture must have sufficient Flash and RAM to support the code.

**Hardware Architecture Comparison**

To meet the business requirements, I have evaluated three hardware architectures: TI, Microchip, and Freescale (now part of NXP). Each of these options offers a range of microcontroller/ microprocessor options suitable for the project's needs. I have reviewed the catalog of products that each has to offer to identify the best solution for this project. Reference to each is provided below and the pros and cons are provided as well.

***Texas Instruments (TI)***

Texas Instruments (TI) offers a diverse range of microcontrollers and microprocessors known for their versatility and compatibility with various peripherals which can be seen at their website at https://www.ti.com.

Pros:

* Peripheral Compatibility: TI microcontrollers offer a wide range of peripherals, including GPIO, UART, I2C, and timers, which align with the project's existing requirements.
* Native Wi-Fi Support: Many TI microcontrollers come with built-in Wi-Fi connectivity options, simplifying the implementation of cloud connectivity.
* Resource Availability: TI microcontrollers typically provide ample Flash and RAM resources, accommodating both the existing codebase and future enhancements.

Cons:

* Cost: Some TI microcontrollers can be relatively expensive, which could impact the project's budget.
* Complexity: The abundance of features and options in TI microcontrollers may lead to a steeper learning curve for development.

***Microchip***

Microchip offers a comprehensive portfolio of microcontrollers known for their diversity and competitive pricing which can be found on their website at https://www.microchipdirect.com.

Pros:

* Peripheral Compatibility: Microchip microcontrollers often include a broad set of peripherals, meeting the project's hardware requirements.
* Variety of Options: Microchip offers a wide range of microcontrollers, allowing for flexibility in choosing the right fit for the project's needs.
* Competitive Pricing: Microchip microcontrollers are competitively priced, which may be advantageous for budget-conscious projects.

Cons:

* Wi-Fi Integration: While Microchip provides connectivity solutions, including Wi-Fi modules, adding Wi-Fi functionality may require additional components and complexity.
* Resource Constraints: Some lower-end Microchip microcontrollers may have limitations in Flash and RAM, potentially requiring compromises in code optimization.

***Freescale***

Freescale, which is now part of NXP, specializes in microcontrollers known for their security features and energy efficiency which can be found on their website at https://www.nxp.com.

Pros:

* Peripheral Compatibility: Freescale microcontrollers offer a range of peripherals, similar to the project's requirements.
* Security Features: Freescale microcontrollers often come with robust security features, which can be advantageous for cloud-connected devices.
* Energy Efficiency: Freescale microcontrollers tend to excel in low-power applications, potentially extending battery life for wireless devices.

Cons:

* Limited Wi-Fi Integration: Unlike TI, Freescale does not offer native Wi-Fi support in its microcontrollers, potentially necessitating additional Wi-Fi modules.
* Resource Constraints: Some Freescale microcontrollers may have limited Flash and RAM resources, requiring careful code optimization.

**Recommendation and Justification**

Considering the business requirements and the pros and cons of each architecture, I would recommend utilizing the Texas Instruments (TI) architecture for the next phase of connecting the thermostat to the cloud. This choice is justified by the following factors:

* Peripheral Compatibility: TI microcontrollers provide seamless compatibility with the project's expected peripherals.
* Native Wi-Fi Support: TI microcontrollers offer built-in Wi-Fi connectivity options, simplifying cloud integration.
* Resource Availability: TI microcontrollers typically offer sufficient Flash and RAM resources, accommodating both the current codebase and future enhancements.

By selecting the TI architecture, we ensure a smooth transition to cloud connectivity while minimizing development complexity. This would utilize a higher-priced solution but would include systems with strongly incorporated Wi-Fi modules, reducing the complexity and possible resources and money that would be required to add the needed modules in the other architectures. It should be noted that this could change as all three of the different hardware architectures, TI, Microchip, and NXP advance and provide new offerings and this should be reviewed again before in the future to ensure the best solution is utilized for the project.

References

Wikipedia contributors. (2023c). Microchip Technology. *Wikipedia*.

<https://en.wikipedia.org/wiki/Microchip_Technology>

Parai, M., Das, B., & Das, G. (2013). An Overview of Microcontroller Unit: from Proper Selection to Specific Application. *International Journal of Soft Computing and Engineering*, *2*(6), ISSN: 2231-2307.

<https://www.ijsce.org/wp-content/uploads/papers/v2i6/F1161112612.pdf>

*Microcontrollers (MCUs) & processors | TI.com*. (n.d.).

<https://www.ti.com/microcontrollers-mcus-processors/overview.html>

*Welcome to MicrochipDirect*. (n.d.).

<https://www.microchipdirect.com/mcu-mpu-overview>

*Processors and microcontrollers*. (n.d.). NXP Semiconductors.

<https://www.nxp.com/products/processors-and-microcontrollers:MICROCONTROLLERS-AND-PROCESSORS#/>