Southern New Hampshire University  
CS 350 - Emerging Systems Architecture & Technology  
Mohamed Elhassan

[**Overview 3**](#_6wn37mmreuxz)

[**Peripheral Support 4**](#_sr17uncb1ums)

[**Cloud Connectivity via Wi-Fi 4**](#_xr8iiprr7mys)

[**Architecture Analysis and Recommendation 4**](#_s99xlhq2h333)

[**Architecture Capabilities 5**](#_fynz52u35660)

[**Conclusion 6**](#_4e270xri2qg9)

**Thermostat System Report**

### **Overview**

This project is all about getting a thermostat system up and running using a Raspberry Pi. It has three states—Off, Heat, and Cool—and works with different peripherals like an I2C temperature sensor, GPIO-controlled LEDs, and an LCD display. On top of that, it simulates sending data to a server using UART.

### 

### **Peripheral Support**

The Raspberry Pi handles temperature readings via I2C from the AHT20 sensor, controls LEDs through GPIO to show if the system is heating or cooling, and uses buttons to cycle through states and adjust the set temperature. It also updates an LCD display in real-time and sends temperature and state data over UART.

If you swap out the Raspberry Pi for a Microchip microcontroller (like PIC or AVR series), it can still manage the same tasks. It supports UART for serial communication and can handle I2C and GPIO control, though it needs additional libraries to function similarly.

Freescale (NXP) microcontrollers offer the same basic features as Microchip but with better power efficiency. They come with built-in peripherals for I2C, GPIO, and UART, making them a solid option for low-power embedded applications.

### **Cloud Connectivity via Wi-Fi**

The Raspberry Pi has built-in Wi-Fi, so it can push UART data to a server over HTTP or MQTT with a simple Python script. Microchip and Freescale architectures, however, need an external Wi-Fi module, like ESP8266 or ESP32, to do the same. Once set up, they can send data at intervals, allowing for remote monitoring of the thermostat's status.

### **Architecture Analysis and Recommendation**

For the next phase of this project, the thermostat needs to support all the peripherals used in the prototype, connect to the cloud via Wi-Fi, and have enough Flash and RAM to run the code efficiently. Raspberry Pi is the strongest candidate since it already integrates all required features, including Wi-Fi, without additional hardware. It also has ample memory to handle the software, making it an ideal choice for scalability.

Microchip and Freescale architectures could work but require external Wi-Fi modules, increasing complexity and potential integration issues. Additionally, some models may have memory constraints that could impact future updates or enhancements to the system. While they are more power-efficient, this comes at the cost of additional hardware and software adjustments.

Based on these factors, Raspberry Pi is the recommended architecture for production due to its ease of integration, built-in Wi-Fi, full peripheral support, and sufficient memory for future improvements.

### 

### **Architecture Capabilities**

All three architectures—Raspberry Pi, Microchip, and Freescale—support I2C temperature sensors, GPIO LED control, and UART for sending data to the cloud. Raspberry Pi has full support for an LCD display, whereas Microchip and Freescale have more limited options. While the Raspberry Pi comes with built-in Wi-Fi, Microchip and Freescale require an external module. In terms of power efficiency, the Raspberry Pi is decent, but Microchip is better, and Freescale is the most efficient.

### 

### **Conclusion**

The Raspberry Pi is the easiest and most flexible option for running this thermostat system. That said, if power efficiency is a bigger concern, Microchip and Freescale are good alternatives, but they need extra components to match the Pi’s functionality. Given the business requirements for cloud connectivity, peripheral support, and sufficient memory, Raspberry Pi is the best choice for production.