Connor Bailey

CS-350: Emerging Systems Architecture & Technology

Professor Morales

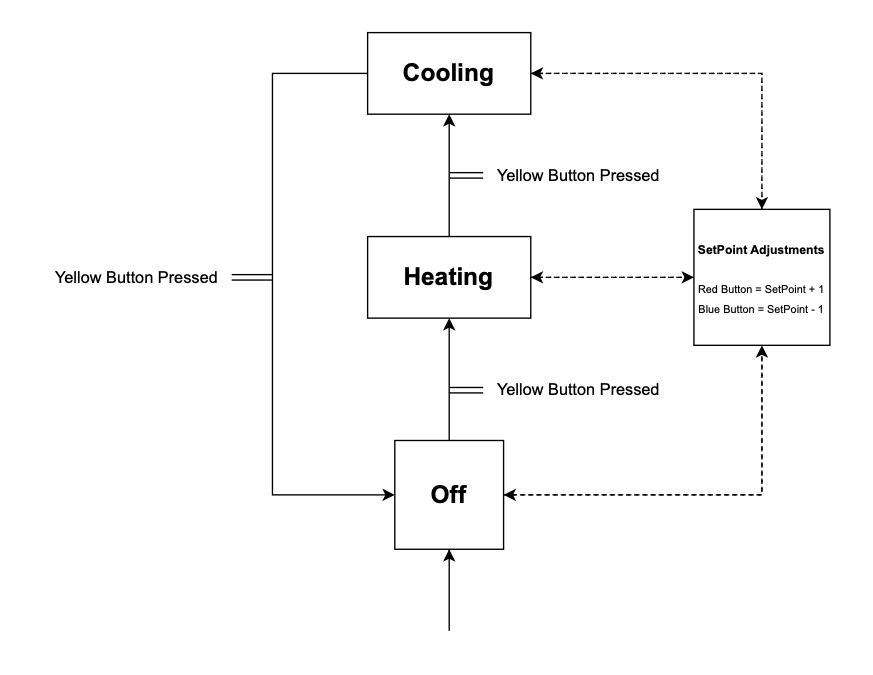
21 June 2025

Final Project Report

**Overview**

This smart thermostat prototype created for SysTec is designed around the Raspberry Pi 4B. This prototype features an AHT2 temperature and humidity sensor, a 16x2 character LCD, two status LEDs, and three physical push buttons for toggling the state and adjusting the set temperature. The LCD, LEDs, and buttons are all connected to the Raspberry Pi via the GPIO header pins, and the AHT2 sensor is connected via the I2C bus. The Raspberry Pi transmits the sensor data via UHART and receives the UHART transmissions in this prof of concept to simulate a cloud connected server. With all of these connected peripherals, we were successfully able to meet all of the business requirements set out by SysTec in the design of this IoT Thermostat.

**State Diagram**

****

**Platform: Raspberry Pi**  
 The Raspberry Pi 4B has a 40-pin GPIO header that offers multiple dedicated pins for I2C and UART. This flexibility allows the system to interface with all of the physical components in the system without requiring additional hardware for expansion. PWM (Pulse Width Modulation) in Python via the *gpiozero* library allows our embedded system to support different LED behavior based on the system’s state and temperature readings. The Broadcom BCM2711 processor is quite capable of supporting our software stack, and the integrated 802.11ac wireless with support for both 2.4 GHz and 5.0 GHz frequencies allows for seamless internet connectivity. The Raspberry Pi does require a full operating system in order to run, which comes with increased overhead and security considerations for our IoT thermostat. This system runs on expandable microSD card storage, which can be configured as small as 8GB to reliably allow the OS and our code to run properly.

**Platform: Microchip**

Microchip offers different microcontrollers that offer peripheral support that is well suited to this project’s needs. These devices are specifically designed for embedded IoT applications and include native support for I2C, UART, and 60 GPIO pins, which would support all of our hardware from this prototype. The AHT2 sensor would utilize the I2C bus, the LEDs and the LCD would have multiple GPIO pin options to connect to the system, and the system will be able to utilize UHART per SysTec’s specifications. Certain offerings from Microchip also contain 802.11 b/g/n Wi-Fi with support for the 2.4 GHz frequency. While the lack of 5 GHz connectivity is notable, this is connectivity typically would not be required for an edge IoT device like the one we are designing. Products from Microchip typically contain only a few MB of storage; however, the system does not require a full operating system to interface with the peripheral hardware. This means that our current implementation in Python would not be possible, as the Python interpreter itself would utilize all the available space on the device *(Python.org)*. Our project would need to be refactored into a compiled language like C or MPLAB to run on this device. If the development time was taken to accomplish this, the product would be lightweight and much more secure than maintaining an operating system around our project.

**Platform: Freescale**

Freescale, now a part of NXP Semiconductors offer a broad portfolio of microcontrollers and SoCs intended for embedded systems in the automotive space. In general, the offerings include sufficient GPIO and hardware support for I2C and PWM, making it possible to integrate our AHT2 sensor, LCD display, LEDs, and buttons. However, there is noticeable limitation in the Wi-Fi support in these devices. Unlike the Raspberry Pi and Microchip offerings, most Freescale/NXP devices do not include integrated Wi-Fi connectivity, and instead either use the ZigBee protocol or communicate via the CAN bus, which is common in the automotive industry. In theory, Wi-Fi could be added with the addition of an external module, but this would add increased complexity and cost to our design. We would also be responsible for managing the drivers and interfacing with this added module, which are currently beyond the scope of our current implementation. Another key limitation of the products from Freescale/NXP is the flash storage limitations. The amount of storage in these systems scales from 0 to 4 MB, meaning that some systems do not contain any on board flash storage. In our current implementation, the embedded system must contain some kind of on-board storage to store and run our code, which means any of the offerings available here would not suit our needs.

**Proposed Platform Decision** After examining these three architectures, I believe Microchip is the preferred hardware platform to produce the smart thermostat for SysTec. Microchip offers support for all of our required peripherals, and unlike Freescale’s solutions, includes a built in 802.11 b/g/n Wi-Fi radio. While the Raspberry Pi provides excellent peripheral support and was easy to develop and prototype on, its higher power consumption and use of a full Linux operating system makes it less ideal for this embedded systems deployment. Microchips microcontrollers are specifically optimized for IoT applications and provide the right balance of processing power and flash storage, making them the right choice for this project. While migrating to this architecture will require refactoring and migrating our code to a compiled language, the final product will be low-power, efficient, and secure IoT product.

**Work Cited**

“Python 3.12.0.” Python.org, Python Software Foundation, https://www.python.org/downloads/release/python-3120/. Accessed 21 June 2025.