# **CS 350 7-1: Final Project Thermostat Lab Report**

# **Austin Donaubauer**

# **Southern New Hampshire University**

# **CS-350 Emerging System Architecture & Technology**

# **Eric Gregori**

# **August 15th, 2025**

# **CS 350 7-1: Final Project Thermostat Lab Report**

I developed the thermostat prototype with a raspberry pi 4 device. Using a breadboard to wire up the different components like the LCD screen, the buttons, a couple of LED lights, and many wires to connect them. In this lab I was able to add two additional buttons and wire them up to the GPIO pins. With the new buttons added, I began implementing the functionality withing the python script. After adding the code, I began testing it by running the script and clicking the buttons to change the state of the thermostat to each of the three different states which are off, heat, and cool. I also tested that the other two buttons increased the set temperature value and decreased it accordingly. For the heat state, the red LED is lit up, and if the set temperature is higher than the current temperature which is read in from the temperature sensor, then the red LED light pulses. But if the set temperature point is equal or lower than the current temperature, then the red LED is just lit fully. Similarly, with the cool state, if the set temperature is lower than the current temperature than the blue LED pulses. If the set current temperature is equal to or lower than the set temperature point, then the blue LED is lit fully. For the off state, both red and blue LEDs are shut off. The LCD also is involved with displaying information to the user. The information displayed is on separate lines of the LCD due to its configuration of two lines and sixteen characters. The first line is configured to display the current date and time. The second line is configured to alternate its message between the current temperature and then display the current state and the set temperature value. These messages alternate every 6 seconds to change the message. The script also utilized debug messages and displaying information to the console for the user to see. The information displayed to the user is a counter, and this counter counts up to thirty seconds and loops back one. Every ten seconds it outputs the thermostat current state, the current temperature from the sensor, and the set temperature values. When a user increases or decreases the set temperature, both the LCD screen output and the console output are updated to display the updated value to the user.

The raspberry pi can support the different peripheral devices like the temperature sensor, LCD screen, LED lights, and buttons using the GPIO pins available. This architecture allows for different operating systems like Ubuntu as I used for this prototype, which allows for command line control of the system. This system can run python scripts that utilize the GPIO python library and Adafruit which allows for control over those peripherals in code to get data and turn on LED lights as needed. Using Microchip and Freescale would require use of a coding language like C or C++ to control the peripherals. These also are very basic and do not run operating systems like a raspberry pi can. They may offer a developer direct control of the components, but it involves more complex code.

The next step of the prototype is to implement a connection to the cloud. Using the raspberry pi device is great because it already comes WI-FI capable. So, adding the necessary code to use the WI-FI connection of the raspberry pi would just be to add in the HTTP python library which can then utilize the serial connection to upload the info to the cloud. This is similar to the ThermostatServer-Integration script which simulates the information passed over the serial connection to display the thermostat information to the console. Using other microcontrollers like Microchip and Freescale, this would involve a much different system. Unlike the raspberry pi, these microcontrollers do not have built in WI-FI capability. This means you would have to add a WI-FI sensor to the system and code that to communicate with the sensor.

Using the raspberry pi allows for the use of multi-threading like in my prototype. One thread is the main thread that handles the buttons and LEDs logic, and the other thread manages the LCD screen and logic. It also allows the user of python which is capable of handling multi-threading, along with a state machine library and others to help simplify the code and keep it clean and efficient. For the Microchip and Freescale architecture, they are able to only handle a single continuous thread. Something like my prototype on these architectures would require code in C or C++ and the use of a singular while loop and additional code to handle the components is more complex. It could be used to develop a thermostat prototype, and it would offer more power efficiency, making it so a simple device could run longer without much need for monitoring.

Developing this thermostat was a great project, and I impressed myself with how well I handled the hardware and getting everything wired up and working. Having little experience with this hardware made this a challenge in the beginning, but this final project was a great way to finish off this and show how much I learned and experienced.