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

Cade Bray

Cade.Bray@snhu.edu

Southern New Hampshire University

Sina Asadi, Ph.D.

October 14th, 2025

For this assignment I’ve created a working low-level prototype of a thermostat. This prototype includes an LCD and three input buttons. The display will always provide the user with the current date and time while simultaneously providing a second line of the current temperature alternating with the set temperature every five seconds. One button cycles the state machine states for off, heating, and cooling. The other two buttons on the prototype are for managing the temperature set point. Additionally, this prototype provides an output state through a serialized string over USB medium. It should be noted that this prototype will need a higher voltage relay for controlling an HVAC system. A typical HVAC system requires 12-24v output to signal (Callahan, 2025). Implementing this relay should be trivial and was not included in the prototype at this time.

The second phase of this project is to connect the thermostat to a cloud platform. This requires design considerations for the hardware we utilize. The thermostat hardware architecture must support our current peripherals which are the three buttons, single LCD, and serialized output. Additionally, the architecture must support a wi-fi connection for its future product phase of connecting to the chosen cloud platform. Finally, the architecture must be minimal in nature while meeting the minimum Flash and RAM requirements to support any operating system and product code. I will make a recommendation after reviewing the proposed hardware architectures from ‘Raspberry Pi’, ‘Microchip’, and ‘Freescale’.

Touching on the cloud architecture briefly we can make some assumptions. The program running on the device will make GET/POST requests to a configured API managed on a cloud platform or hosted on company provided servers. This could be achieved in prototyping or production with a node.js backend for handling an API while providing a single page application (SPA) with Angular, React, or Vue for the end user to interact with the device. If the device checks for regular configuration updates from the node server instead of receiving API calls, we can avoid making the thermostat a publicly accessible device. This design consideration limits the attack surface of the project.

Considering our implementation for disk usage we can see that all packages utilized amount to 568MB of disk space with the command ‘du -sh /usr/lib/python3/dist-packages/’. The python interpreter is 55MB of disk space found with the command ‘du -sh /usr/lib/python3.12/’. The thermostat.py file itself is 16KB of disk space. Considering our implementation, we would need to deploy our product on a lightweight operating system such as Alpine Linux because we’re using operating system features such as threading libraries and sensor reading libraries over I2C. This light-weight operating system will require a single board computer (SBC) instead of a micro-controller. Alpine Linux requires a minimum of 1GB of flash memory for the operating system (Alpine Linux, n.d.). This means that I can recommend an SBC with an estimated minimum flash memory of 1624MB. We could rework the prototype in C/C++ to run on a micro-controller utilizing a real-time operating system (RTOS) if needed.

During program profiling tests I observed a maximum RAM utilization of 81.4MB. Alpine Linux Requires a minimum RAM installation of 256MB (Alpine Linux, n.d.). I can make the recommendation of a minimum RAM installation of 337.4MB. The likelihood of finding or manufacturing an SBC that is perfectly suited to these requirements isn’t likely, as operating system and package requirements may change in the future, so marginal spare overhead is desirable. After reviewing commonly supported devices from the previously mentioned manufacturers I can recommend the most cost-effective SBC as the Raspberry Pi Zero 2 W with 512MB of RAM, Micro-SD cart slot for our own installation, 2.4GHz 802.11b/g/n wi-fi enabled, and a quad-core processor for our multi-threaded application at a low retail cost of fifteen dollars

(Raspberry Pi, n.d.). Finally, The Raspberry Pi Zero 2 W has a full set of GPIO pins needed for our project integration and serialization of output data.

References

Alpine Linux. (n.d.). *Requirements*. Retrieved from Alpine Linux: https://wiki.alpinelinux.org/wiki/Requirements

Callahan, M. (2025, September 1). *Why HVAC Systems Use 24 Volts for Control Circuits (Explained Simply)*. Retrieved from The Furnace Outlet: https://thefurnaceoutlet.com/blogs/videos/why-hvac-systems-use-24-volts-for-control-circuits-explained-simply

Raspberry Pi. (n.d.). *Raspberry Pi Zero 2 W*. Retrieved from Raspberry Pi: https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/