**7-1 Project**

Justin Phillips

Southern New Hampshire University

CS-350 Emerging System Architectures

Joseph Martinez

June 22, 2025

**7-1 Project**

**Peripherals**

This prototype thermostat uses a Raspberry Pi as a foundation and several hard-wired components for user friendly interfacing and input/output. Of these components, there are three buttons, a red LED light, a blue LED light, a 16x2 LCD display, and an Adafruit temperature and humidity sensor. From left to right, each button has a standard real-time thermostat function; Cycling the state machine (Off -> Heating -> Cooling), raising the temperature, and lowering the temperature, respectively. As the thermostat runs, real time data is reported on the LCD display, such as the current time and temperature in all states. Real-time temperature data is read from the sensor throughout runtime. Once the thermostat leaves the off state and enters either the heating or cooling state, the LCD display alternates every ten seconds between displaying the real-time data and the state of the machine (heating or cooling). As well, if the machine is either actively heating or cooling, it will blink the corresponding red or blue LED. If it is set to heat or cool but is not actively adjusting the temperature through its connected heater or air conditioner unit, it displays a solid LED to indicate the temperature adjustment is done. These elements facilitate industry-standard user interaction as well as visual feedback from the thermostat in real-time, allowing any end-user to visualize the control logic at work.

**Thermostat Cloud Connection**

This thermostat is configured to produce serial communications to a server in the form of a string with the state ID, the current temperature Fahrenheit, and the target temperature Fahrenheit. While it can fulfill the low-level system requirements of a thermostat, it requires more work and planning for the production hardware to be capable of interfacing with the peripherals required (three buttons, a display, two LEDs, and a temperature sensor) and communicating through integrated WiFi. There have been three principle suggestions of an architecture; Raspberry Pi, Microchip, and NXP Semiconductors (architectures formerly provided by Freescale).

**Hardware Architectures**

The software currently employed for running the low-level thermostat functionality and low-information telemetry server sends is a 16kb file which can run within less than 32kb of RAM. The future ambitions of this product are to include an automatic heating/cooling mode, as well as superior telemetry which can be used for us to gather real-time data as well as possibly display data to the end user. There are currently robust offerings between all three architectures, rooted in MicroPython supporting options with limited flash storage and RAM capacities, and offering low power consumption as well as low costs. The advantage of MicroPython is that it is a low level Python 3 code interpreter for embedded systems, which allows dropping a main.py script directly into the embedded systems storage to run as the operating system. This allows for rapid and easy updating, along with allowing re-use of the prototype thermostat’s software, saving work hour investments.

For the Raspberry Pi Pico W ($6ea), the processor is a configurable clock speed processor offering up to 133MHz, a 802.11n 2.4Ghz single band integrated network interface, 264kb of RAM and 2Mb of storage, a 26-pin GPIO header, 2xSPI, 2xI2C, 2xUART, and 16xPWM channels. These specifications are gratuitous, but do not need to be fully utilized, and provide overhead for product growth (*Raspberry Pi Pico W*). Its sleep power draw is also 1-2mA, making it fairly low power consumption when not in use.

Microchip offers more succinct options which are more power efficient. WFI32E01PC is a promising contender which features low power draw and some better hardware specifications. It features a faster 200MHz processor, comparable 2Mb storage, single band 802.11bgn 2.4Ghz WiFi, 1x I2C for the temperature gauge, 37 GPIO pins, 320kb RAM, and 3 UART serial connections (*WFI32E01PC*). It also offers a MicroPython loader sourced on GitHub (*PIC32MZW1\_MicroPython*). It is $5.12ea. Its limitations are that its MicroPython support architecture is less mature than the Raspberry Pi Pico, and it has no native PWM, which requires adjusting the pulse rate manually for blinking LEDs consuming more power and compute than otherwise needed. However, it offers 10 microampere power consumption in deep sleep, therefore significantly less power draw when not in use. It also has no real-time clock built in, which means it may rely on the network for time reads, while the Pico has an RTC built-in. In comparison to Microchip’s offerings, the Raspberry Pi Pico W is superior to the WFI32E01PC.

NXP offers a wide range of microcontrollers, though the MIMXRT1021CAF4B in particular offers a MicroPython integration environment and a competitive quantity based pricing model (100+ of these units being a mere $3.67, while singular units are still only $4.34). They offer a rapid compute speed (up to 400MHz), 256kb of RAM, 2 PWM, 1 USB, 8 UART, 4 I2C, 57 GPIO, and 10 microampere sleeping power consumption. An advantage of the high clock speed is that it can process instructions much more quickly than the competing options, and therefore return to a sleeping state much faster. It also does feature a real-time clock, but it does not feature on-board WiFi, meaning it requires an extra peripheral to access the inter net for telemetry use. I2C WiFi modules can vary in pricing depending on quality and band access, but I was able to find I2C WiFi adapters as low as $3 each. That said, this and the power consumption of the faster processor could make it more expensive than the Pico W, so while it is a strong contender, it is not the best option (*i.MX RT1020*).

.

**REFERENCES**

*i.MX RT1020. (n.d.). Nxp.Com. Retrieved June 22, 2025, from https://www.nxp.com/products/processors-and-microcontrollers/arm-microcontrollers/i-mx-rt-crossover-mcus/i-mx-rt1020-crossover-mcu-with-arm-cortex-m7:i.MX-RT1020*

*PIC32MZW1\_MicroPython: “IoT Made Easy!” - support micropython on pic32mzw1 (wfi32). (n.d.).*

*Raspberry Pi Pico W*. (n.d.). CanaKit. Retrieved June 22, 2025, from <https://www.canakit.com/raspberry-pi-pico-w.html>

*WFI32E01PC*. (n.d.). Microchip.Com. Retrieved June 22, 2025, from https://www.microchip.com/en-us/product/WFI32E01PC