**Smart Thermostat Hardware Recommendation**

Now that the prototype for the smart thermostat is working, the next step is figuring out what hardware we’d actually use if this was going into production. Since this thermostat needs to support several peripherals and eventually connect to the cloud over Wi-Fi, the hardware we choose has to check all the right boxes.

I looked into three different options; Raspberry Pi, Microchip, and Freescale. I’m recommending we stick with the Raspberry Pi. It’s what we used for the prototype, and it’s the most practical option moving forward.

**What the Thermostat Needs**

Here’s what the system has to support:

* I2C for the AHT20 temp sensor
* GPIO with interrupts for three buttons
* PWM for two LEDs (heating/cooling indicators)
* 16x2 LCD display support
* UART for serial output to a server
* Built-in or add-on Wi-Fi for future updates
* Enough RAM and storage to run the code without issues

**Comparing the Options**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Raspberry Pi 4B** | **Microchip MCU** | **Freescale** |
| **I2C, UART, GPIO, PWM** | Fully supported | Fully supported | Fully supported |
| **RAM** | 1 GB – 8 GB | 32 KB – 256 KB | 512 KB – 1 MB |
| **Flash / Storage** | microSD / USB | 256KB – 1MB | 2MB – 4MB |
| **LCD Support** | Easy w/ Python Libraries | More complex, low level | Driver setup needed |
| **Wi-Fi** | Built-in | Some chips have it built-in | Some models have it onboard |
| **OS Support** | Linux | Bare metal / RTOS only | Linux on high-end models |
| **Programming** | Python / C++ / Shell | Mostly C only | C but heavier setup |
| **Community / Docs** | Huge and helpful | Decent, more technical | Split between tools & docs |
| **Price per unit** | ~ $35-55 | ~ $2 - 10 | ~ $10 - 25 |

**Why I Recommend Raspberry Pi**

Even though the Pi has more power than we need for something simple like reading sensors and blinking LEDs, it’s the easiest and most complete solution for right now.

Here’s why I think it’s the right choice:

* We’re already using it for the prototype, so no need to rewrite everything.
* It supports all the interfaces we used: GPIO, I2C, UART, LCD, etc.
* Python support makes it easier to develop, test, and troubleshoot.
* It runs Linux, so connecting to a server or cloud service later will be much smoother.
* There’s a huge community, so it’s easy to find answers if something breaks or we want to upgrade features.
* If we want to shrink it down for production, we can move to a Raspberry Pi Compute Module (CM4) version.

Microchip and Freescale both have solid options too, especially if this was a commercial product going into mass production. But they’d take a lot more time and experience with embedded firmware to get working right. For where we are right now, the Raspberry Pi makes everything way simpler. It's reliable, powerful, and flexible enough to scale with our needs. For now, I believe it’s the most logical choice.

**References**

* Raspberry Pi Foundation. (n.d.). Raspberry Pi 4 Model B specifications. Raspberry Pi. <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>
* Microchip Technology Inc. (n.d.). ATSAMD21G18A datasheet. [https://www.microchip.com/](https://ww1.microchip.com/downloads/en/DeviceDoc/SAM-D21DA1-Family-Datasheet-DS40001882G.pdf)
* NXP Semiconductors. (n.d.). i.MX RT1050 crossover processor. <https://www.nxp.com/products/i.MX-RT1050>