NAS Build – Spring 2025

- Vision statement:

The vision for this project is fairly straight forward. A NAS (network attached storage) is effectively a method to build a personal cloud storage network. At the lowest level, it allows a user to upload data to a storage device via a network. In terms of use cases, there are countless. The explosion of cloud computing in recent decades is evidence of this, including simple cloud storage products (Google Drive, Dropbox, etc..), the migration of many SAAS products to the cloud in the form of remote fulfilment (AWS, nearly all Microsoft and Apple software products), and the advent of gen AI which uses remote server banks to process queries.

The development required to build a NAS are far reaching, yet attainable. Ideally, I will need to develop a 'bare metal' OS to interphase with the hardware (the CPU, memory, storage, and auxiliaries) of a Raspberry Pi 5 unit. On top of coding in assembly to initiate the boot sequence, I will also need to write the code for the kernel, and device drivers in C. There will likely be additions to the tech stack as I progress through the process. Over all, this project will deepen my knowledge of OS design, kernel-hardware interactions, memory allocation, multi-threading in system design, and general low-level software development. All of which are highly desired skills in a wide range of software development fields at all major employers.

- Motivation:

My motivation for this project in particular follows a couple tenets, both long standing and more recent. I have been interested in embedded systems design since before I started in the CSPB program. I have been using the Arduino IDE to program a few of my own robotics and hobbies, however not nearly in this level. I think that I have a reasonable knowledge base to write and compile software before uploading to a stand-alone computing device.

I also believe that developing true low-level coding skills will be invaluable in the coming decades. Even before the advent of gen-AI, in particular the models that are focused on code generation, the act of software development has been gradually moving towards high level coding environments, where the projects, languages, and workplaces require less and less knowledge of fundamental computer systems, and systems design. As a result, new developers will likely lose out on real experience in these facets of software development, a trend that will only steepen for obvious reasons. This project is a great opportunity to deepen my experience in low level software development and computer systems design, which I hope will help me differentiate myself for competitive jobs. It's also an opportunity to develop some personal projects that are potentially marketable as well!

- Specific and measurable goals (learning objectives) for the project

Goals that will be able trackable and attainable are as follows:

- Download and install IDE, C and assembly code extensions, GCC cross-compiler, and any additional Pi OS expansions.

- Build the Pi Unit itself. This will consist of the Raspberry Pi, a M.2 hat, and a M.2 SSD
- Build the booter, in assembly, as a start. This code will initialize the kernel
- Build the source code which will run the OS (the kernel itself).
- Build or download drivers for the SSD
- Develop the networking code which will allow an off the shelf web application to talk to and upload to the completed NAS unit.
- Install and test software on the Raspberry Pi. This will be an ongoing task that will be revisited every time a feature is rolled out. I am expecting some level of trial and error the first time around.
- Produce the demo for final assessment.
- I am expecting each of these deliverables to take 1-2 weeks each, 3-5 hours per week to complete. This falls in line with the allotted time required for project completion.

• Risks to project completion: did they list a good set of risks?

There are a number of risks that are apparent to project completion.

- There is a possibility that I run out of time to bring the project to completion. The reasons for this could range from challenges in assembly, challenges keeping the kernel stable when implementing additional features, challenges in the handling of network communications (maintaining the server, finding the right port, etc.). Overall, I think this project would push right into 45-hour allocation regardless.
- Hardware issues could indeed arise, given that I would be dealing with embedded systems. I think the most likely point of failure is at the storage unit, I have read that Raspberry Pi units are quite robust.
- Version control with the Raspberry Pi unit. Most of the literature that I've found on building a stand-alone OS on a RPi unit is for the Raspberry pi 4. I have purchased the latest model, the Raspberry Pi 5. This could present some translation methods that I will have to navigate in terms of driver design and componentry. Thankfully, the RPi 5 still uses the ARM 64 architecture on the CPU.
- Scope issues could develop if not kept in check. I think this is already a challenging project for my existing skills, but increasing the feature set beyond what is originally defined can compound this.

• Mitigation Strategy for the risks listed: do the strategies mitigate the risks?

There are some easy ways to mitigate the risks presented above.

- Each component of the build could easily be replaced with an off the shelf component. The OS for instance can easily be replaced with PiOS, a Linux distribution, or even windows. If I get stumped on a particular component, I can defer to configuring and installing an off the shelf method.
- The networking component of this project feels like the most challenging. Building and hosting a server that can transmit data to the SSD will prove to be the most challenging, where open-source implementations exist to host the server. This could definitely wind up working better than my implementation, but we will have to see.
- As hardware goes, It is a prudent practice to vet any hardware before moving to any production state. This can easily be done with hardware detection tests.

- Generally sticking to the schedule of deliverables will be an effective way of making sure the project reaches completion by the deadline. Getting behind on certain features can have downstream effects on subsequent features.

• Project Assessments: did they provide a viable list of evaluation criteria for the project.

True Evaluation criteria could be challenging to provide given that I would be writing software for a specific device. It wouldn't be possible to give the OS cross-compatibility in the time allotted, so it will likely require a Raspberry Pi itself to function at all.

I will be making a presentation which will be a demo of the system working. I will walk through the code base from the boot sequence to networking, describing the tech used and the functionality of the code. I will then demo the working unit by uploading various files to the SSD.

• Project portfolio link:

I will be using a git repo to consolidate and maintain the code base. The repo can be found here:

https://github.com/roha1130/2025-NAS-build