3. Design Approach

This document outlines the design options, system overview, and subsystem descriptions for NetCommand, a network switch configuration system intended to streamline switch configurations in the field with the convenience of a mobile device. The most critical components guiding this design are the user interface, real-time command execution via a terminal emulator, and reliable backend automation using Ansible. Design constraints include limited RAM and processing power on the Raspberry Pi, minimal latency for command execution and feedback, and support for commonly used network protocols as well as serial communication standards. The selected approach was guided by cost-efficiency, simplicity, system responsiveness, and security mindfulness.

Design Options

3.1.1 Design Option 1: Standalone Laptop Interface with Python GUI

One proposed solution was to use a portable laptop running a custom Python GUI connected to the switch through a USB-to-Serial adapter. This design would allow full access to system resources, including native terminal emulation and automation tools. The interface could be developed using Tkinter or PyQt, and automation would be implemented through local execution of Python scripts or Ansible playbooks.

Advantages of this approach include more available resources like processing power and RAM, native CLI support, and ease of development due to the laptop's full operating system. However, drawbacks include the dependency on a laptop, which contradicts the project's goal of eliminating bulky equipment in field deployments. It also introduces potential boot-up delays, reduced battery efficiency, and unnecessary overhead for simple tasks.

3.1.2 Design Option 2: Raspberry Pi-Based Web App with Flask + React (Selected)

The chosen design leverages a Raspberry Pi to host a Flask backend and a React frontend, accessible from mobile devices via USB-C. This setup includes REST endpoints for automated Ansible tasks (e.g., creating VLANs) and WebSockets to support a real-time terminal input.

This option meets the portability and cost-effectiveness requirements, with the Raspberry Pi serving as an always-on, low-power appliance. It allows the user to interact via phone or tablet, streamlining the workflow for field engineers. The team selected this option because it offers the best balance between user accessibility, real-time interaction, and automation integration.

3.2 System Overview

At its core, NetCommand allows users to input network configuration settings through a webpage. These inputs are handled by a Raspberry Pi running backend and frontend logic, which communicates directly with a network switch using a console cable and USB 3.0 cable. As shown in **Figure 3-1**, the NetCommand system receives user configuration inputs from the mobile device and forwards the request to the backend which communicates to the switch and sends feedback to the user device.

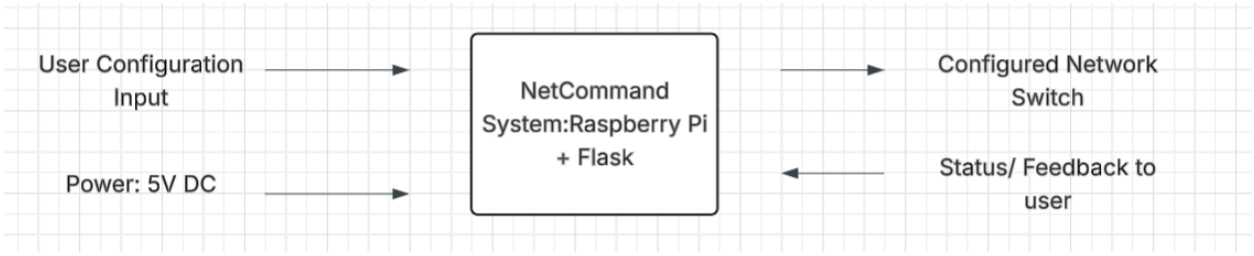
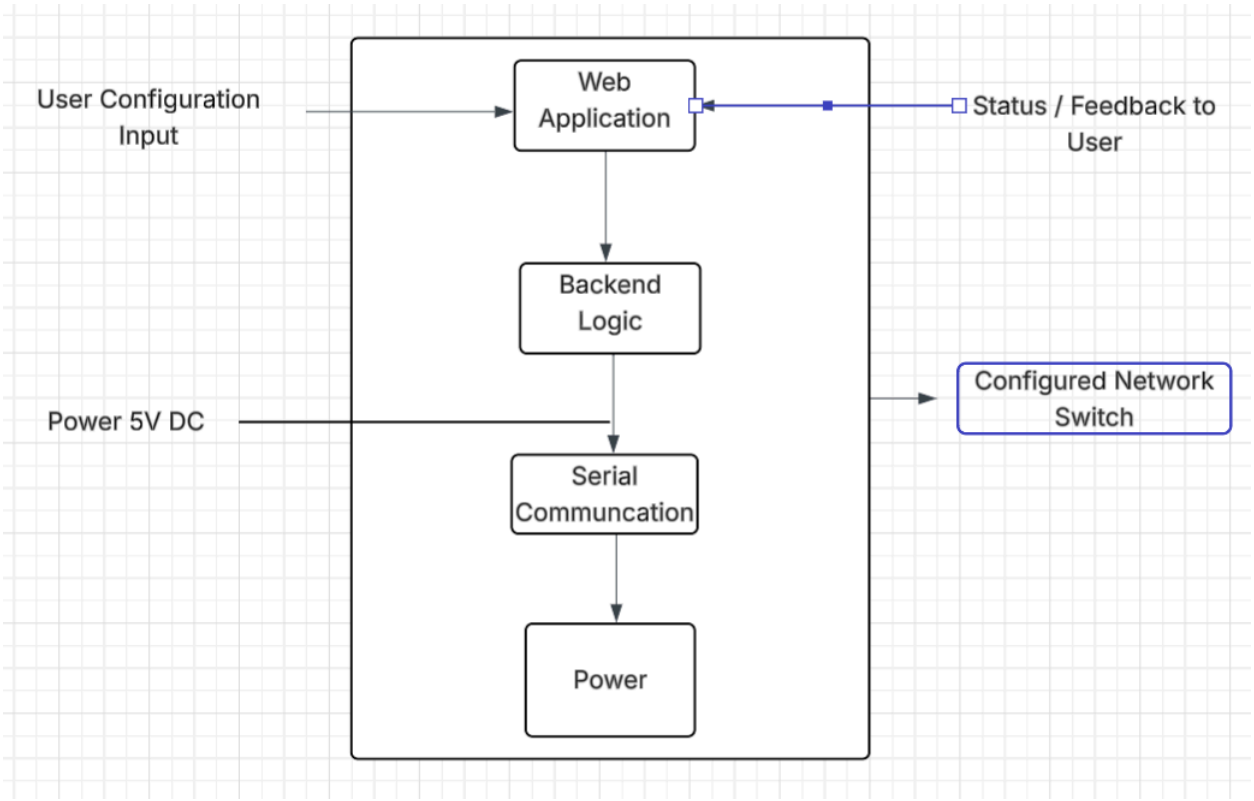


Fig 3-1: Basic System Overview for the NetCommand Device (Level 0)

Figure 3-2 provides a detailed breakdown of the NetCommand system, outlining its subsystems and how they interface with the system's inputs, outputs, and the Raspberry Pi acting as the core controller.

Fig. 3-2: Functional Overview of the NetCommand System (Level 1)



The system is designed to be compact, field-deployable, and easily accessible through a smartphone browser. It accepts terminal commands from users in real-time and allows them to perform actions like setting hostnames, assigning IP addresses, and creating VLANs through automated tasks.

3.2.1

Microcontroller Selection

Multiple single-board computer options were evaluated for the system to ensure they could support the required functionality. The selected board needed to efficiently handle backend operations, serial communication with network switches, and frontend hosting. **Table 3-1** presents the specifications of the boards considered during this selection process.

Table 3-1: Microcontroller Options

Microcontroller	**RAM**	**CPU (GHz)**	**Networking**	**USB Ports**	**OS Support**	**Price**
Requirements	**2-6 GB**	**1-2 GHz**	**Ethernet + Wi-Fi**	**4**	**Linux**	**$40-65**
Raspberry Pi 4B	4GB	1.5 GHz	Ethernet + Wi-Fi	4	Raspbian/Linux	$62
Arduino UNO WiFi Rev2	6KB SRAM	0.020 GHZ	Wi-Fi	1	None	$45

Justification:

The Raspberry Pi 4B was selected due to its overall ideal fit for NetCommand. Its Raspberry Pi Operating System (OS) Lite, Linux-based, facilitates the installation of Flask, Node.js (used for loading React front-end), and Ansible. It supports multiple USB devices and wireless connectivity, both of which are essential for the mobile accessibility NetCommand relies upon and setting up the serial interfacing with the switch. The Arduino is not as good an option as the

Raspberry Pi because it does not natively support hosting an OS, meaning you cannot run any programming languages other than C/C++.

Figure 3-3: Raspberry Pi 4GB microcontroller

The Raspberry Pi 4 was selected for the NetCommand system because it meets all engineering requirements while providing a cost-effective and reliable solution. It supports a web-based user interface for VLAN creation and port assignments, as well as a command-line interface (CLI) with SSH and Serial communication for real-time switch configuration. The Pi 4 enables secure network changes through AES-256 encryption and multi-factor authentication (MFA). Additionally, it supports Ansible automation for consistent VLAN deployment and includes a USB-C interface for direct local access, allowing the system to function in secure or offline environments

---

3.3 Subsystems

NetCommand is composed of four main subsystems, each is responsible for an aspect of the system's functionality and integration.

1. The **Development Subsystem** handles the communication between the querying device and the switch.
2. The **Automation Subsystem** manages the execution of quick configuration tasks using Ansible and the Linux command-line.
3. The **Firmware Subsystem** is responsible for interfacing with the physical networking equipment.
4. The **Hardware Subsystem** focuses on the physical design and integration of the system in a standard network rack.

These four subsystems work together to create the desired functionality of NetCommand.

3.3.1 Development

The development subsystem provides users with a responsive interface to configure and manage network switches with or without CLI expertise. It makes use of React.js for the frontend and Flask for the backend, which are both hosted on a Raspberry Pi. The system processes user requests and delivers feedback through a web page accessible from any tablet or mobile device.

This subsystem is comprised of multiple functional components that form and produce its overall functionality:

- Frontend: Built with React.js, it manages and renders the user interface, enabling intuitive interaction with network controls and feedback mechanisms.
- Backend: Developed with Flask and Flask-SocketIO, the backend processes API requests from the frontend, manages real-time communication to send commands and receive responses, and executes Ansible playbooks for automating network configurations. Flask-SocketIO enables live feedback and status updates to be pushed to the frontend without requiring page refreshes. This backend runs locally on the Raspberry Pi, acting as the central logic layer between the user interface and the network hardware.

Table 3-2 shows the development stack options considered for the web-based interface of NetCommand. The selected framework needed to support serial communication, Ansible automation, and real-time feedback, while being lightweight enough to run on a Raspberry Pi.

Table 3-2: Frontend and Backend Stack Options

Frontend Framework	**Backend Technology**	**Real-Time Support**	**Raspberry Pi Compatibility**	**Ease of Use**	**Price**
Requirements	**Flask**	**Websockets**	**Excellent**	**Moderate**	**Free**
Vanilla JS/HTML/ CSS	Flask (no SocketIO)	Polling only	Excellent	Easy	Free
React.js	Flask + Flask-SocketIO	WebSockets	Excellent	Moderate	Free

Figure 3-4 shows the selected development stack of React.js + Flask + Flask-SocketIO. This stack provided strong real-time communication features, Python-native compatibility with

Ansible and serial libraries, and mobile-friendly web access. It offered the best balance of performance, modularity, and lightweight deployment on the Raspberry Pi.

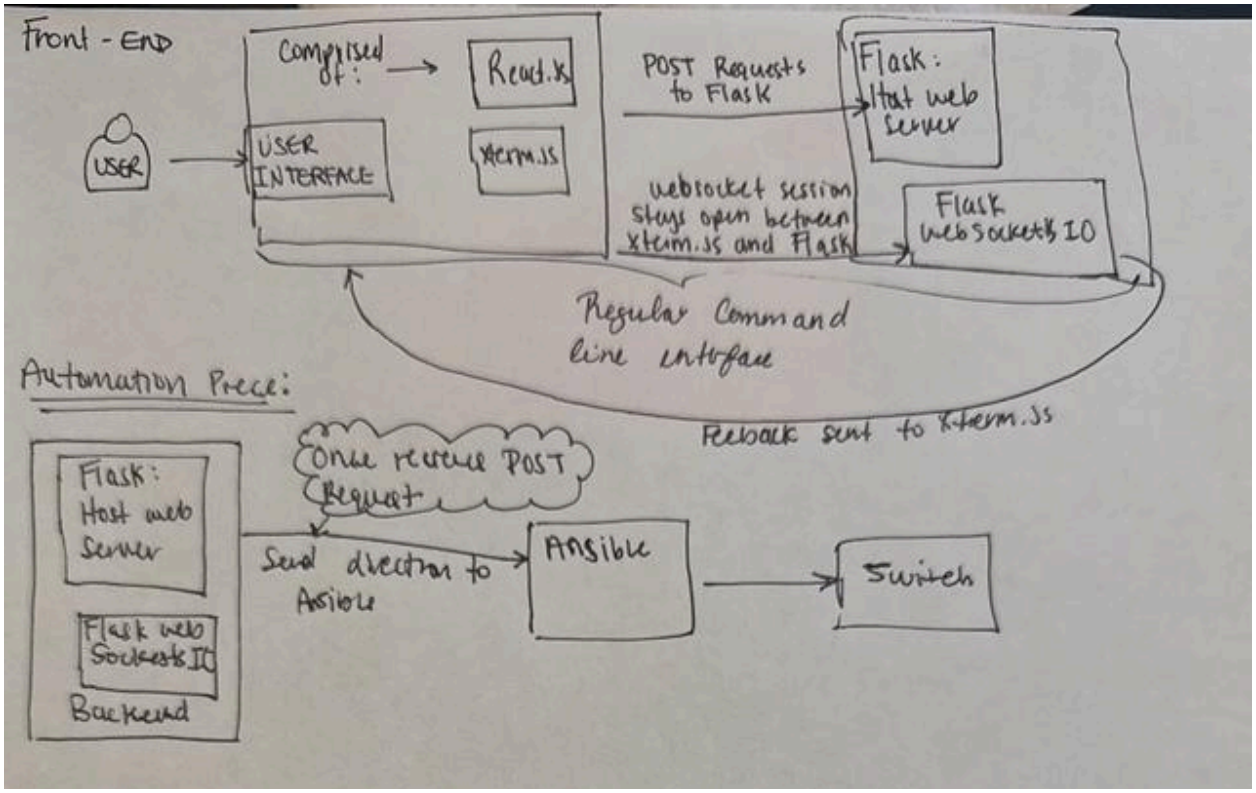


Figure 3-4: Selected React.js + Flask + Flask-SocketIO Stack for NetCommand

The development subsystem allows the user to control and configure network hardware with or without CLI knowledge, forming a bridge between user interaction and network automation. This software layer is essential for the portability and accessibility goals of NetCommand.

3.3.2 Automation

The Automation Subsystem is responsible for executing predefined network configuration tasks without requiring users to interact with the switch's command-line interface. It leverages Ansible, an open-source automation tool, to push configuration changes—such as VLAN creation and port assignments—to the network switch. This subsystem is initiated through the user-friendly frontend interface and is closely integrated with the backend, allowing for seamless, secure, and responsive automation of core networking functions. By reducing reliance on manual CLI commands, the automation layer demonstrates the capability of NetCommand to streamline common administrative tasks in real-world network environments.

Table 3-3 lists the playbooks considered for VLAN creation. Early test versions had limited functionality and hardcoded values. The selected vlan_creation.yml playbook accepts full input from the user interface and includes error handling to support reliable automation.

Playbook File	**Function**	**Required Inputs**	**Description**
Requirement	**VLAN creation**	**VLAN ID, Name, Description**	**Ability to create VLAN on switch with configuration parameters**
vlan_creation.yml	Create a VLAN	VLAN ID, Name, Description	Creates a new VLAN on the switch using the provided configuration parameters.
vlan_creation_test.yml	Create VLAN (test version)	VLAN ID only	Early prototype with limited functionality and no error handling.

Table 3-3: VLAN Creation Playbook Options

Table 3-4 lists the playbooks considered for port assignment. While the basic version allows single-port assignment, the selected port_assignment.yml playbook supports multiple port entries and improves efficiency by reducing playbook executions.

Playbook File	**Function**	**Required Inputs**	**Description**
Requirements	**Assign Ports to VLAN**	**VLAN ID, List of Ports**	**Batch port enabling with execution result**
basic_port_assign.yml	Assign Ports (test)	VLAN ID, Single Port	Prototype used during development testing; limited to one port at a time.
port_assignment.yml	Assign Ports to VLAN	VLAN ID, List of Ports	Selected playbook enables batch port assignment and returns execution result.

Table 3-4: Port Assignment Playbook Options

Figure 3-5 illustrates the step-by-step communication flow within the automation subsystem. A user initiates a configuration task via the frontend interface, which triggers a POST request to the Flask backend. The backend then executes the appropriate Ansible playbook and returns a result to the user interface in the form of a success or error message.

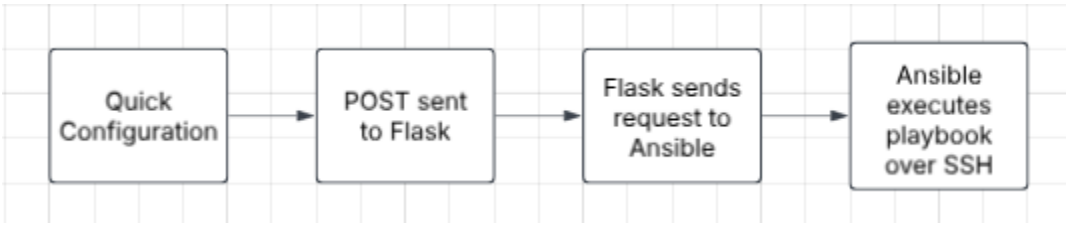


Figure 3-5: Automation Request Flow

Figure 3-6 shows the user-facing form used to initiate VLAN creation. The form collects the necessary configuration details (e.g., VLAN ID, name, and description) and sends them to the backend as a RESTful request. This frontend element serves as the entry point for network automation tasks.

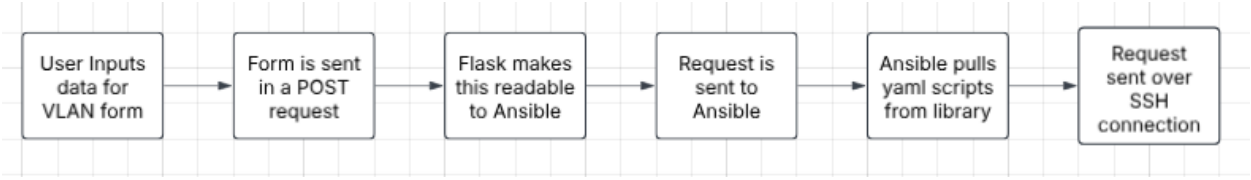


Figure 3-6: Example VLAN Creation Form (UI Screenshot)

The Automation Subsystem executes Ansible playbooks to configure the switch by creating VLANs and assigning ports. Triggered through the frontend's Quick Configuration menu, user inputs are sent to the Flask backend, which runs the appropriate playbook. Results are returned as success or failure messages to the user interface. The system currently supports one task at a time and may expand to include more network automation functions in future versions.

3.3.3 Firmware

The firmware subsystem of NetCommand handles all backend processing, local storage, and communication between the user interface and the network switch. The Raspberry Pi 4B serves as the core processing unit, running the backend Flask server and executing automated configuration tasks via Ansible. A 32GB SanDisk Ultra microSD card provides storage for the

operating system, backend application code, and network configuration files. Communication between the Raspberry Pi and the switch is established using a USB-to-RJ45 console cable, enabling serial command exchange. To maintain secure access, the firmware incorporates AES-256 encryption and multi-factor authentication (MFA). The Raspberry Pi is powered through a 5V USB-C power adapter, delivering consistent and reliable operation for system performance.

Table 3-5 shows the microSD card options considered for the firmware subsystem. The selected storage device needed to provide sufficient memory capacity, fast read/write speeds, and reliable performance to store the operating system, backend software, configuration files, and support the overall functionality of the Raspberry Pi 4.

Table 3-5: Comparison of microSD card options

MicroSD card	Capacity	Speed Class	Cost
Requirements	**32 GB**	**Class 10**	**$5 - $15**
SanDisk Ultra 32GB (Chosen)	32 GB	Class 10	$8.99
Samsung EVO Select 32GB	32 GB	Class 10 UHS-1	$9 - $11
Kingston Canvas Select 32GB	32 GB	Class 10	$7

Figure 3-7 shows the selected SanDisk Ultra 32GB microSD card used in the firmware subsystem. This microSD card offers reliable storage capacity and fast read/write speeds at an affordable price, making it the best choice for supporting the Raspberry Pi 4's operating system and backend software.

Figure 3-7: SanDisk Ultra 32 GB[14]

Table 3-6 shows the USB-to-Serial console cable options considered for the firmware subsystem. The selected console cable needed to provide reliable serial communication between the Raspberry Pi 4 and the network switch, with compatibility for standard RJ45 console ports used in networking equipment.

Table 3-6: Comparison of console cables

Cable	Connector Type	Compatibility	Cost
Requirements	**USB-A to RJ45**	**Switches/Routers**	**$10 - $20**
USB to RJ45 Console Cable (Chosen)	USB-A to RJ45	Cisco Routers/Switches, Windows/Mac/Linux	$10.00
TRENDnet USB to RS232	USB-A to RS232	Requires RS232-to-RJ45 adapter	$12 - $14
UGREEN USB to RJ45 Console Cable	USB-A to RJ45	Cisco/Networking Equipment	$13 - $16

Figure 3-8 shows the selected USB to RJ45 console cable used in the firmware subsystem. This cable provides a reliable and direct serial communication link between the Raspberry

Pi 4 and the network switch, making it the ideal choice for transmitting configuration commands within the NetCommand system.

Figure 3-8: USB to RJ45 Console Cable [15]

Table 3-7 shows the RJ-45 to RJ-45 Ethernet cable options considered for the firmware subsystem. The selected Ethernet cable needed to provide reliable network connectivity between the Raspberry Pi 4 and the network switch, while supporting the T568B wiring standard commonly used in networking equipment.

Table 3-7: RJ-45 to RJ-45 Ethernet Cable Comparison (T568B Standard)

Cable Option	Length	Category	Speed Rating	Cost
Requirements	**6ft**	**Cat5e**	**Up to 1 Gbps**	**$2 - $10**
GEARIT Cat5e Ethernet Patch Cable (Chosen)	6ft	Cat5e	Up to 1 Gbps	$2 - $4
Amazon Basics Cat6 Ethernet Cable	6ft	Cat6	Up to 1 Gbps	$4 - $6
Monoprice Cat6 Ethernet Cable	6ft	Cat6	Up to 1 Gbps	$3 - $5

Figure 3-9 shows the selected GEARIT Cat5e Ethernet Patch Cable used in the firmware subsystem to connect the Raspberry Pi 4 to the network switch. This cable supports the T568B standard and provides reliable communication for remote management and system updates.

Figure 3-9: GEARIT Cat5e Ethernet Patch Cable [16]

Table 3-8 shows the power supply options considered for the firmware subsystem. The selected power supply needed to provide a stable 5V output with sufficient current through a USB-C connection to reliably power the Raspberry Pi 4 during continuous operation.

Table 3-8: Power Supply Comparison for Raspberry Pi

Power Supply	Output	Connector Type	Cost
Requirements	**5V 3A**	**USB - C**	**$10 - $20**
Official Raspberry Pi 4 USB-C Power Supply (Chosen)	5V 3A	USB - C	$16.99
CanaKit USB-C Power Supply	5V 3.5A	USB - C	$9 - $11
iUniker USB-C Power Supply	5V 3A	USB - C	$7 - $9

Figure 3-10: Raspberry Pi 4 USB-C Power Supply used in the firmware subsystem to provide stable 5V 3A power to the Raspberry Pi 4 for reliable operation and backend processing.

Figure 3-10: Raspberry Pi 4 USB-C Power Supply [17]

3.3.4 Patch Panel

The Patch Panel features a 3D-designed enclosure that organizes and separates internal wiring and system components. This design provides dedicated sections for power, networking, and backend processing, improving cable management and system maintenance. The enclosure will function as a patch panel, allowing for clean and accessible Ethernet and console connections to network devices.

Table 3-9: 3D Printing Filament Comparison

Filament	**Printing Accuracy**	**Nozzle temp**	**Cost**
Requirements	**+/-0.02mm**	**190-230C**	**$15-40**
OverTure PLA 1kg filament	+/-0.02mm	200-220C	$14.99
Creality 2kg Black & White PLA	+/-0.03mm	190-230C	$28.49
SUNLU PLA 3D Filament 1kg	+/-0.05mm	200-230C	$13.99

Table 3-9 above helps to show Overture PLA filament offers a very affordable price for the quality you receive. For consistent printing, minimal stringing, and reliable layer adhesion, it

ends up making OverTure the best fit. Plus, it's easy to work with on most printers, making it perfect for getting good results on the prototypes without having to constantly tweak it.

Figure 3-11: OverTure PLA 1kg filament[18]

Above in **Figure 3-11**: OverTure PLA Filament was chosen for the Patch Panel 3D printing subsystem to produce durable and precise prototype components, offering consistent extrusion and minimal warping for reliable printing.

3.4.1 Level 2 Diagram

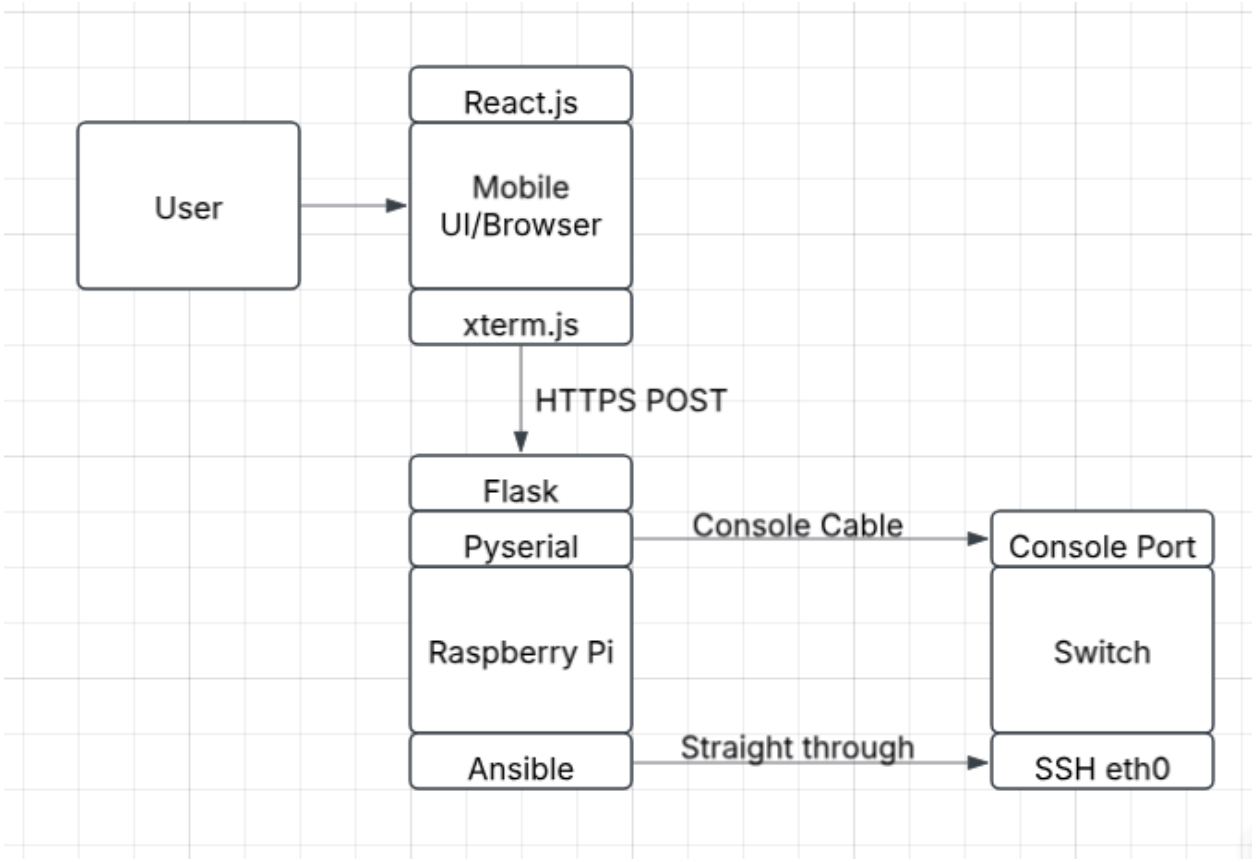


Figure 3-12: Diagram for NetCommand

Figure 3-12 shows the Level 2 system layout for the NetCommand device. The design illustrates connections between the frontend, backend, and the switch. The Raspberry Pi interfaces with both the user and the network device (via USB-to-Serial), providing a closed-loop configuration workflow. All modules run simultaneously and communicate through the physical interfaces. A web browser acts as the user interface, while the Raspberry Pi handles request processing, configuration automation, and serial command relay.

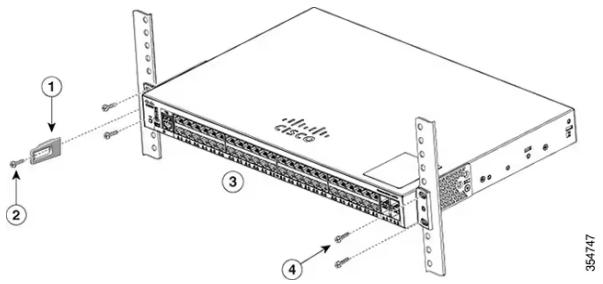


Figure 3-13: Diagram of Switch mount brackets [13]

Figure 3-13 Cisco switch mounts and how the compartment will be mounted to the same rack.

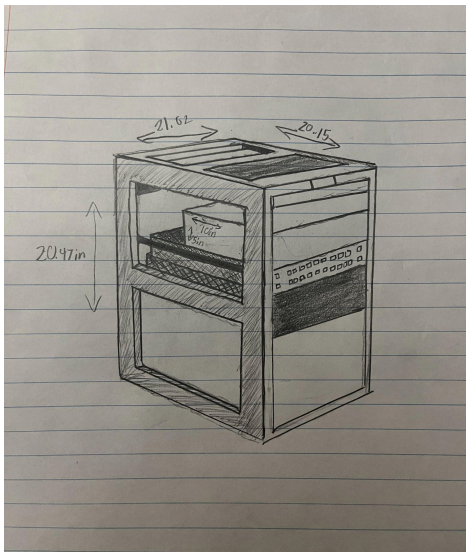


Figure 3-14: Illustration of planned design

Figure 3-14: The figure illustrates a custom drawing designed to house system components, including a Raspberry Pi 4. The structure measures 21.62 inches wide, 20.15 inches deep, and 20.47 inches tall, featuring compartmentalized sections for modular hardware integration and cable management.

References

1. Office of Advocacy, “Frequently Asked Questions About Small Business 2023,” *SBA’s Office of Advocacy*, Mar. 07, 2023. https://advocacy.sba.gov/2023/03/07/frequently-asked-questions-about-small-business-2023/
2. T. Freed, “Top Challenges With Network Management,” *Prelude Services*, Aug. 19, 2019. https://www.preludeservices.com/blog/top-challenges-with-network-management/
3. “Network Administrator Job Description,” ECU Online Programs, [Online]. Available: https://onlineprograms.ecu.edu/blog/network-administrator-job-description/. [Accessed: 12-Feb-2025].
4. “Information Technology - Generalist,” Nova Scotia Community College (NSCC), [Online]. Available:https://www.nscc.ca/programs-and-courses/programs/plandescr.aspx?prg=ITGE&pln=ITGENERAL. [Accessed: 12-Feb-2025].
5. “Cisco Discovery Protocol Configuration Guide, Cisco IOS Release 15M&T - Secure Cisco Discovery Protocol [Support],” *Cisco*, Jun. 2016. https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cdp/configuration/15-mt/cdp-15-mt-book/nm-cdp-secure-cdp.html (accessed Mar. 16, 2025).
6. IEEE Standard for Ethernet, IEEE Standard 802.3-2018, IEEE, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8457469
7. IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks, IEEE Standard 802.1Q-2018, IEEE, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8403927
8. IEEE Standard for Local and Metropolitan Area Networks—Media Access Control (MAC) Bridges, IEEE Standard 802.1D-2004, IEEE, 2004. [Online]. Available: https://ieeexplore.ieee.org/document/1309630
9. IEEE Standard for Local and Metropolitan Area Networks—Port-Based Network Access Control, IEEE Standard 802.1X-2020, IEEE, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9018450
10. IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11-2020, IEEE, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9363693
11. J. Postel and J. Reynolds, “Telnet protocol specification,” RFC 854, Internet Engineering Task Force, May 1983. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc854
12. T. Ylonen and C. Lonvick, “The secure shell (SSH) transport layer protocol,” RFC 4253, Internet Engineering Task Force, Jan. 2006. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4253

13. Cisco Systems, "Cisco Catalyst 2960-L Series 24-Port and 48-Port Switch Hardware Installation Guide," May 17, 2017. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst2960l/hardware/installation/guide/b_c2960l_24_48_hig/b_c2960xr_hig_chapter_010.html#ID227.
14. SanDisk, “SanDisk Ultra 32GB microSDHC UHS-I Memory Card,” Amazon, 2024. [Online]. Available: https://www.amazon.com/dp/B073JWXGNT. [Accessed: Apr. 20, 2025].
15. OIKWAN, “OIKWAN USB to RJ45 Console Cable Compatible with Cisco Routers and Switches,” Amazon, 2024. [Online]. Available: https://www.amazon.com/OIKWAN-Compatible-Opengear-Aruba%EF%BC%8CJuniper-Switches/dp/B075V1RGQK. [Accessed: Apr. 20, 2025].
16. GearIt, “GearIT Cat5e Ethernet Patch Cable, 6 Feet, 24-Pack,” Amazon, 2024. [Online]. Available: https://www.amazon.com/GearIt-24-Pack-Cat5e-Ethernet-Patch/dp/B00X8FGY18. [Accessed: Apr. 20, 2025].
17. Security-01, “5V 3A USB-C Power Supply Adapter (2-Pack),” Amazon, 2024. [Online]. Available: https://www.amazon.com/Security-01-2-Pack-Supply-Adapter-Type-C/dp/B09JW2H7KK. [Accessed: Apr. 20, 2025].
18. OVERTURE, "OVERTURE PLA Filament 1.75mm, Dimensional Accuracy +/- 0.02 mm, 3D Printer Consumables," Amazon. [Online]. Available: https://www.amazon.com/OVERTURE-Filament-Consumables-Dimensional-Accuracy/dp/B07PGY2JP1.

The authors acknowledge the use of ChatGPT in the preparation of this assignment for phrasing and proofreading.