Project 2

Swiss Army Knife Network Sniffer

Course of study	Bachelor of Science in Computer Science
Author	Frank Gauss (gausf1) and Lukas von Allmen (vonal3)
Advisor	Wenger Hansjürg

Version 1.0 of October 5, 2025

► Technik und Informatik

Abstract

One-paragraph summary of the entire study – typically no more than 250 words in length (and in many cases it is well shorter than that), the Abstract provides an overview of the study.

Contents

1 First Thesis Chapter

1.1 Introduction

What is the topic and why is it worth studying? – the first major section of text in the paper, the Introduction commonly describes the topic under investigation, summarizes or discusses relevant prior research (for related details, please see the Writing Literature Reviews section of this website), identifies unresolved issues that the current research will address, and provides an overview of the research that is to be described in greater detail in the sections to follow.

1.2 Specification

1.2.1 System Delimitation

System Environment

2 Hardware Selection

2.1 Requirements

The following requirements were defined for the hardware platform used in this project:

- At least two native Ethernet interfaces for inline packet sniffing
- Support for 2.5 GbE or higher
- Onboard Wi-Fi with access point (AP) and monitor mode support
- Low power consumption suitable for 24/7 operation
- Compact form factor for laboratory and prototype setups
- Strong community and software support
- Affordable cost (below 150 CHF)

2.2 Evaluated Boards

Several boards were considered as potential variants. Their main specifications relevant to the project are listed in Table 2.1.

Table 2.1: Comparison of Board Variants

Board	SoC / CPU	RAM / Storage	Ethernet Ports	Power (typ.)	Wireless (onboard)
Banana Pi R3 Mini	MT7986A, Quad-core ARM Cortex-A53 @ 1.3 GHz	2 GB DDR4, 8 GB eMMC, microSD	2 × 2.5 GbE	5–7 W	MT7976C, Wi-Fi 6 (AP/Client/Monitor)
Banana Pi R3	MT7986A, Quad-core ARM Cortex-A53 @ 1.3 GHz	2–4 GB DDR4, eMMC, microSD	1 × 1 GbE, 2 × 2.5 GbE, 4 × 1 GbE	7–10 W	MT7976C, Wi-Fi 6
Banana Pi R4	MT7988A, Quad-core ARM Cortex-A73	4 GB DDR4, NVMe option	4 × 2.5 GbE, 2 × 10 GbE (SFP+)	10–15 W	None (M.2 Wi-Fi module required)
Banana Pi R5	MT7988B, Quad-core ARM Cortex-A73	4 GB DDR4, NVMe option	2 × 10 GbE, 2 × 2.5 GbE	12–18 W	None (M.2 Wi-Fi module required)
Raspberry Pi 4	BCM2711, Quad-core ARM Cortex-A72 @ 1.5 GHz	2–8 GB LPDDR4, microSD	1 × 1 GbE (second via USB dongle)	6–8 W	Wi-Fi 5 (AP/-Client only)
Raspberry Pi 5	BCM2712, Quad-core ARM Cortex-A76 @ 2.4 GHz	4–8 GB LPDDR4X, microSD	1 × 1 GbE (second via PCIe card)	8–12 W	Wi-Fi 5 (AP/-Client only)
NanoPi R76S	Rockchip RK3588S, Octa-core (4× Cortex-A76 @ 2.4 GHz + 4× Cortex-A55 @ 1.8 GHz)	16 GB LPDDR4X / LPDDR5, NVMe (option via M.2)	3 × 2.5 GbE (RJ45)	10–15 W	None (M.2 Wi-Fi 6E module recommended)

Table 2.2: Requirements Fulfillment by Candidate Boards

Requirement	R3 Mini	R3	R4	R5	RPi 4	RPi 5	NanoPi R76S
≥ 2 native Ethernet interfaces	✓	✓	✓	✓	✗	✗	✓
RAM > 4GB	✗	✗	✓	✓	✗	✓	✓
≥ 2.5 GbE support	✓ (2×)	✓ (2×)	✓ ✓ (4×)	✓ (2×)	✗	✗	✓
Onboard Wi-Fi with AP & Monitor mode	✓	✓	✗	✗	✗	✗	✗
Low power consumption (<10 W)	✓	✓/▲	✗	✗	✓	▲	✓
Compact form factor	✓	✗	✗	✗	✓	✓	✓
Strong community & software support	✓	✓	▲	▲	✓ (general)	✓ (general)	✓
Suitable for inline packet sniffing	✓	✓ (overkill)	▲ (overkill)	▲ (expensive)	✗	✗	✓

Legend: ✓ = Requirement fulfilled, ✗ = Requirement not fulfilled, ▲ = Partially fulfilled / limited

2.3 Decision

Based on the defined requirements and the evaluation of alternatives, the **Banana Pi R4** and the**NanoPI R76S** are most suitable hardware platform for this prototype implementation.

The Banana Pi R4 offers two native 2.5 GbE interfaces for inline sniffing the board is compact, affordable, and supported by a strong community. In Addition, the two 10 GbE SFP+ ports provide flexibility for extensions as fiber-based packet capturing. A drawback of the R4 is the weaker CPU and a larger size compared to the NanoPI R76S

The NanoPi R76S is more compact and provides up to 16GB of RAM, which is advantageous for memory intensive processing and buffering tasks. While it lacks built-in Wi-fi, it can be expanded via the M.2 Wi-Fi 6E module. It can not host both a Wi-Fi card and NVMe SSD simultaneously. Consequently, data storage must be provided via microSD card or external USB SSD

Alternative boards such as the Banana Pi R3 Mini, R3 are limited overall performance. Rasberry PI 4 or 5 offer higher single core performance but were ultimately discarded because they provide only a single native Ethernet interface, requiring external adapters that reduce performance for inline sniffing scenarios.

2.4 Module

ID	Module Name	Description
M1	Hardware Evaluation & Setup	Evaluation and selection of suitable embedded hardware (e.g., Banana Pi R4, NanoPi R6S). Setup of OS, drivers, and base configuration.
M2	System Configuration	Installation and configuration of the operating system, networking stack, and essential system packages.
M3	Network Capture Engine	Implementation of inline network sniffing using pcap/tcpdump, bridge configuration, and throughput validation.
M4	Traffic Filtering & Manipulation	Integration of firewall, packet filtering, and traffic injection capabilities (iptables/nftables).
M5	Performance Measurement	Development of measurement tools for throughput, latency, and jitter (iperf3, netperf, ping tests).
M6	Attack Simulation	Controlled execution of network attacks such as MITM, downgrade, or ARP spoofing for testing purposes.
M7	Switching & Routing (SDN)	Implementation of dynamic routing or bridging functions, optional integration with SDN controllers.
M8	VPN Endpoint	Configuration of a secure VPN endpoint (WireGuard or OpenVPN) supporting host-to-network and network-to-network connections.
M9	Wi-Fi Client Mode	Setup of Wi-Fi module as a client endpoint for wireless measurements and connectivity.
M10	Wi-Fi Access Point Mode	Setup of hostapd to run the board as a wireless access point, including channel and signal analysis.
M11	Mobile Network (Tethering)	Integration of LTE/5G USB modem for mobile uplink or hotspot functionality.
M12	Backend Communication	Secure communication with a remote backend or control server via REST API or MQTT.
M13	User Interface (CLI / Web)	Development of a local command-line and/or web-based configuration interface for module control.
M14	System Security & Hardening	Implementation of SSH key management, firewall, certificates, and user authentication for secure operation.
M15	Monitoring & Logging	Collection of system logs, traffic statistics, and implementation of rotation and data persistence.
M16	Documentation & Reporting	Continuous documentation of development progress, project report, and final presentation materials.

Table 2.3: Overview of project modules for the *Network Swiss Army Knife* prototype.

2.5 Diagram

2.5.1 Variant 1 Inline-Sniffing

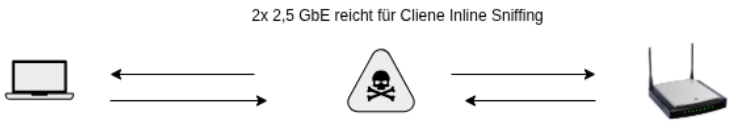

Figure 2.1

2.5.2 Variant 2

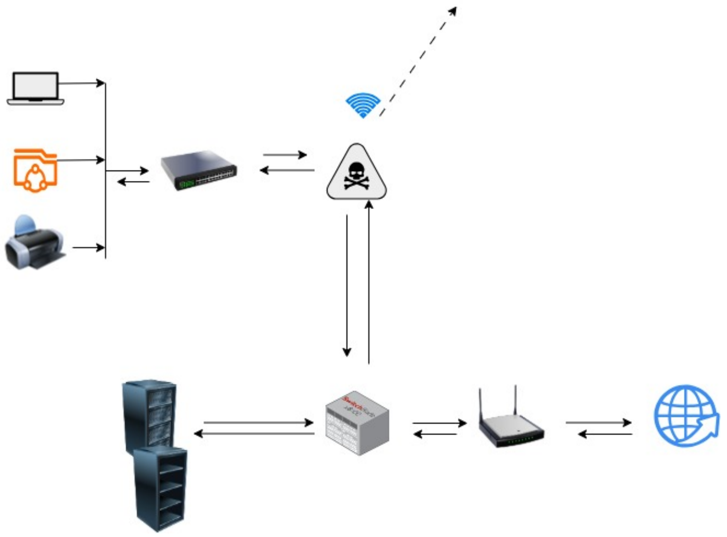

Figure 2.2

2.5.3 Variant 3

2.5.4 Variant 4

2.5.5 Variant 5

3 Second Thesis Chapter

3.1 Implementation

3.1.1 Architecture

Declaration of Authorship

I hereby declare that I have written this thesis independently and have not used any sources or aids other than those acknowledged.

All statements taken from other writings, either literally or in essence, have been marked as such.

I hereby agree that the present work may be reviewed in electronic form using appropriate software.

October 5, 2025

Frank Gauss (gausf1) and Lukas von Allmen (vonal3)

Bibliography

List of Figures

List of Tables

Listings

Glossary

This document is incomplete. The external file associated with the glossary 'main' (which should be called `documentation.gls`) hasn't been created.

Check the contents of the file `documentation.glo`. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like `\gls` or `\glsadd`) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

If you don't want this glossary, add `nomain` to your package option list when you load `glossaries-extra.sty`. For example:

```
\usepackage[nomain]{glossaries-extra}
```

Try one of the following:

- Add `automake` to your package option list when you load `glossaries-extra.sty`. For example:

  ```
  \usepackage[automake]{glossaries-extra}
  ```

- Run the external (Lua) application:

  ```
  makeglossaries-lite.lua "documentation"
  ```

- Run the external (Perl) application:

  ```
  makeglossaries "documentation"
  ```

Then rerun LaTeX on this document.

This message will be removed once the problem has been fixed.

.1 First Appendix Chapter

.1.1 Project 2 Proposal