June 22, 2025

open nexus OS – A Unified, Open Ecosystem for Seamless Device Integration

Jenning Schäfer
jenningschaefer@gmx.de

Abstract

The project is developing a secure, modular operating system based on the Redox OS microkernel and the Rust programming language. The aim is to create a modern, open alternative to Android for RISC-V-based tablets and devices. The architecture combines formal security, strict process isolation and multi-platform capability. The graphical user interface is based on an extended COSMIC desktop with touch and gesture support. In addition to POSIX compatibility, the execution of Android apps is also planned, supplemented by a native app ecosystem with a QML-like UI language and Swift backend. The development plan follows a clearly structured roadmap with demonstrators, ports and developer tools. The project is published under the Apache 2.0 license, combines openness with commercial usability and focuses on community building, partnerships and targeted public relations. Financial support for infrastructure, full-time development, community support and marketing is being sought through grants, sponsors and crowdfunding. The project addresses the growing demand for trustworthy, free software in the RISC-V ecosystem and lays the foundation for a sustainable alternative mobile operating system.

Contents

1 Introduction and Motivation

1.1 The Vision of an Open, Unified Ecosystem

In today's digital world, intelligent devices are becoming increasingly important. However, the "intelligence" of these devices does not stem solely from raw processing power or the integration of large language models like ChatGPT. Instead, it lies in their ability to provide context-aware, automated functionality—without requiring manual configuration by the user.

A truly intelligent system recognizes which information and controls are needed in a given context, thereby reducing complexity. Intelligence also manifests in seamless device interoperability—ideally without explicit pairing or user intervention. Such interaction requires a comprehensive, interoperable ecosystem. A classic example is the automatic unlocking of a MacBook when wearing an Apple Watch—a feature enabled by Apple's tightly integrated ecosystem.

1.2 Analysis of Existing Ecosystems

1.2.1 Apple

Apple currently offers the most integrated digital ecosystem. Automatic device pairing (e.g., AirPods, Apple Watch) and deep integration between macOS, iOS, iPadOS, and watchOS deliver a cohesive user experience. However, the ecosystem's proprietary, walled-garden architecture is a critical drawback. Third-party developers have limited opportunities to innovate due to a lack of open interfaces. While user-friendly, Apple's ecosystem is neither scalable nor future-proof as an open technology standard.

1.2.2 Microsoft

Microsoft takes a desktop-centric approach. Synchronization mechanisms exist within the Windows ecosystem, particularly via Microsoft accounts, Office 365, and Azure services. Enterprises can define device policies and centralized management tools via cloud services. However, mobile integration remains fragmented, often requiring third-party solutions. Building secure mobile environments—for education or small businesses—demands significant development effort.

1.2.3 Android

Android suffers from two structural issues:

1. **Fragmentation**: The system is based on a minimal open-source core (AOSP), leading to divergent implementations by manufacturers. A consistent experience is typically limited to specific device families (e.g., Samsung).
2. **Google Dependency**: Heavy reliance on Google services conflicts with privacy requirements and the vision of open, controllable software infrastructures.

1.2.4 Linux

The Linux ecosystem faces fragmentation and limited mobile integration. While dominant in servers and embedded systems due to its stability, security, and flexibility, it lacks a standardized platform for end-user devices. Inconsistent package formats, interfaces, and GUI guidelines hinder commercial software adoption. Additionally, robust ARM-based desktop support is lacking, complicating cross-platform ecosystem development.

1.3 Goal: A Unified, Open-Source Ecosystem

A future-proof "single-device experience" requires seamless integration of distributed and virtual resources within an open, interoperable ecosystem. All interfaces must be open and documented to unlock innovation potential for developers, businesses, and institutions.

Our goal is to develop an open-source, modular platform based on an Android-like architecture but enabling full integration across device categories—from smart home components and wearables to mobile devices, desktops, and vehicles. To prevent fragmentation, the system will provide all core functionalities needed for a connected experience. Commercial vendors can build upon this foundation without compromising system integrity.

Key Advantages:

- **User Convenience**: Seamless cross-device interaction.
- **Cost Efficiency**: Reusable components reduce costs for businesses.
- **Transparency & Security**: Open-source standards ensure trust.
- **Innovation**: Community-driven development with open interfaces.

1.4 RISC-V as the Hardware Foundation

A unified hardware architecture is critical for an open ecosystem. Currently, desktops (x86) and mobile devices (ARM) are architecturally divided. While ARM-based SoCs (e.g., Apple's M-series) demonstrate impressive efficiency gains, open drivers and standardized interfaces are lacking for broad open-source adoption.

RISC-V offers a promising alternative:

- **Open & Modular**: Extensible architecture with growing adoption.
- **GPU Progress**: Open drivers (e.g., Imagination Technologies) enable a unified RISC-V platform.

Market-ready RISC-V devices are expected to align with our development timeline.

1.5 Huawei's OpenHarmony – A Flawed Alternative

OpenHarmony presents an existing holistic OS solution with a modular kernel, open licensing, and cross-device compatibility. However, key limitations remain:

1. The open-source version is feature-limited, fostering fragmentation.
2. It lacks enterprise-grade management tools.
3. Geopolitical dependence on a single vendor poses risks, especially for Europe's digital sovereignty.

Thus, we propose a fully independent, open-source project—designed from the ground up to be open, controllable, and extensible.

2 Technical Architecture

2.1 Microkernel Design: Redox OS as the Foundation

Our OS kernel is based on Redox OS, a modern microkernel written in Rust, ensuring memory safety and maintainability. Rust's ownership model, strict typing, and compile-time checks eliminate entire classes of security vulnerabilities (e.g., memory leaks, buffer overflows).

Redox combines proven microkernel concepts:

- **Formal Verification**: Inspired by seL4.
- **Modularity**: Adopted from Minix 3.

- **Minimal Kernelspace**: Handles only scheduling, memory management, and IPC. Drivers, filesystems, and services run in userspace for fault isolation.

Redox is architecture-agnostic (supports x86_64, ARM; RISC-V port underway) and includes a full **libc** implementation for C compatibility.

2.2 Hardware Support: Focus on Open Platforms

Initial Target Device: PineTab V (RISC-V, Allwinner D1 SoC).

Why? Hybrid use-case (desktop/mobile), open hardware docs, active developer community.

Long-Term: Platform-agnostic design for portability to other RISC-V systems (e.g., StarFive VisionFive, SiFive HiFive).

2.3 User Interface: Modularity with COSMIC

We fork **System76's COSMIC desktop environment** (written in Rust, Wayland-native) for our GUI, extending it with:

- Touch optimization
- Adaptive layouts (desktop/tablet switching)
- Energy-saving modes

2.4 Software Compatibility & App Ecosystem

- **POSIX Compliance**: GNU/Linux software runs natively.
- **Android APK Support**: Via a custom Hardware Abstraction Layer (HAL).
- **Native App Ecosystem**: A modern IDE with:
  - **Swift** for app logic (LLVM interoperability with Rust).
  - **QML-like UI language** with enhanced safety (typing, sandboxing).

3 Project Roadmap

The aim is to create a quick functional prototype to appeal to developers, funding bodies and potential partners - while at the same time taking long-term maintainability, portability and scalability into account. Prioritization is based on the principle of "proof of concept first, platform sustainability second".

3.1 Phase 1 (0–6 Months): QEMU-Based Prototype

Objective: Creation of an executable development environment on QEMU with a functional graphical user interface and rudimentary touch integration.

Milestones:

- **RISC-V porting of the Redox kernel**
  - Adaptation and testing of the HAL (Hardware Abstraction Layer) for RISC-V, based on existing ARM/x86 architecture
  - Functional tests of basic device drivers (RAM, timer, framebuffer graphics) in the emulator
- **Integration of the COSMIC desktop**
  - Compilation for RISC-V target architecture
  - Adaptation of Wayland protocol support for emulation environments
  - Basic functions: Window display, mouse/keyboard input
- **Tablet mode & UI switching**
  - Introduction of a responsive UI layout with touch optimizations
  - Implementation of dynamic mode switching between desktop and tablet interface (e.g. by gestures or screen size)

Result: ➔ A fully emulated system prototype that demonstrates basic user interaction and UI switching - as a technical proof-of-concept for stakeholders.

3.2 Phase 2 (6–12 Months): PineTab V Port

Objective: Operation of the system on a real device to carry out practical functional tests and performance benchmarks.

Milestones:

- **Device-specific driver development (PineTab V, Allwinner D1 SoC)**
  - Display initialization (incl. 2D GPU support)
  - Touchscreen input
  - Energy management (battery status, standby modes, power management)
- **Performance optimization**
  - Reduction of input latency
  - Optimization of the graphics pipeline (frame buffer, double buffering, hardware acceleration)
- **Android compatibility (basic level)**
  - Creation of a minimal HAL-based emulation environment for APKs
  - Test operation of simple Android applications (browser, terminal, media playback)

Result: ➔ A functional tablet operating system on real hardware - ready for evaluation by developers, testers and early adopters.

3.3 Phase 3 (12–24 Months): Native App Ecosystem

Objective: Functional maturity towards an alternative for existing mobile platforms, with native app framework and growing developer base.

Milestones:

- **Kernel and security enhancements**
  - Introduction of secure application sandboxes
  - Improved energy management (active & passive energy-saving modes, adaptive CPU control)
- **App development environment**
  - Development of a native IDE for UI development with declarative language (QML-inspired)
  - Swift as backend language with connection to the system APIs via LLVM bindings
  - Example apps: Notes, web browser, media player
- **Community & infrastructure**
  - Developer documentation, API reference, sample code
  - Development of a decentralized, open-source app store concept

Result: ➔ A stable developer prototype with its own native app ecosystem - suitable for community building, pilot projects and further investments.

3.4 Long-Term (24+ Months): Scaling

- **Hardware expansion**
  - Support for further RISC-V platforms: Smartphones, laptops, mini PCs
- **Android compatibility as a transition path**
  - Completion of the Android API compatibility layer (incl. Google Services functionality optional)
- **Independent ecosystem**
  - Platform with full native app support (Rust/Swift)

- Security model for digital sovereignty and data protection
- Integration into alternative distribution channels (e.g. public administration, education, open source communities)

4 Open Source Strategy

Our open source strategy aims to build a vibrant, sustainable, and business-friendly ecosystem. At its core are a well-thought-out licensing model, transparent development processes, active community cultivation, and strategic partnerships.

4.1 Licensing Model: Apache 2.0 for Maximum Flexibility and Legal Certainty

The kernel is released under the Apache 2.0 license. This choice offers several key advantages over traditional open-source licenses:

- **Legal Protection via Patent Clause** The Apache License actively shields all contributors and users from patent litigation — a critical factor for enterprise adoption.
- **Business-Friendly and Permissive** Unlike copyleft licenses (e.g., GPL), Apache 2.0 allows commercial partners to keep proprietary modifications closed — a crucial incentive for hardware vendors, system integrators, and OEMs.
- **GPLv3 Compatibility** Legal compatibility with GPLv3 is preserved to enable future interoperability with other free software projects.

This licensing strategy fosters broad adoption — from individual developers to industrial partners with integration interests.

4.2 Community Building & Development Infrastructure

A successful open source platform thrives on its community. Our goal is a transparent, welcoming, and collaborative development environment:

- **Dual Repository Infrastructure**
  - **GitHub** will serve as the public portal for source code, documentation, and issue tracking.
  - **GitLab** will support internal development workflows, CI/CD pipelines, and roadmap management.
- **Targeted Recruitment of Experienced Contributors**
  - Focus on developers from the Redox OS, Rust, and RISC-V communities
  - Priority on low-level systems expertise, driver architecture, and graphical toolkit development
- **Mentorship and Onboarding Programs**
  - Introduction of "Good First Issues", detailed CONTRIBUTING guidelines, and targeted code bounties to lower the barrier to entry

4.3 Strategic Partnerships & Visibility

Early engagement with key organizations and multipliers is essential to scale trust, visibility, and long-term growth:

- **Partnerships with Key Institutions** RISC-V International, OSB Alliance, Free Software Foundation Europe Collaboration on standardization, interoperability, security, and educational initiatives
- **Events and Outreach** Hosting hackathons, community meetups, and open/virtual bounty programs Presence at major conferences such as RISC-V Summit, FOSDEM, CCC, RustConf

4.4 Funding & Support

- **Initial funding** through sponsors and in-kind contributions
- **Mid- to long-term** funding via foundation grants, EU programs (e.g., Horizon Europe), and focused crowdfunding campaigns

4.5 Developer Marketing & Outreach

Our developer engagement strategy is authentic, technically grounded, and built for the long term:

- **Central Project Website**: open-nexus-os.io Regular blog posts, technical updates, roadmap insights, and behind-the-scenes content
- **Social Media Presence** Using Mastodon, Twitter/X, and YouTube for release demos, dev interviews, and livestreams
- **Inclusive Contributor Culture** Emphasis on a respectful, inclusive community with clear guidelines and open governance. Support for new contributors through mentorship and active community help

4.6 Conclusion

Our open source strategy combines technical excellence with pragmatic licensing, focused community building, and strategic outreach. The result is an open, cooperative ecosystem that appeals to both idealistic developers and commercial partners — laying a solid foundation for the project's long-term success.

5 Funding Needs and Support

To successfully realize this project, targeted financial and organizational support is essential. These resources will enable the development of a powerful, open, secure, and license-free alternative in the mobile OS space.

5.1 Technical Infrastructure & Hardware

To establish a robust development environment, we require:

- **Domains, Servers & CI Systems** Setup and operation of Git servers, build systems, container registries, and documentation platforms
- **Test Hardware** Acquisition of devices such as the PineTab V, StarFive VisionFive 2, and other RISC-V platforms to ensure driver compatibility and real-world performance testing

5.2 Full-Time Development & Core Team

To drive continuous and focused development, we plan to build a dedicated core team:

- **Key Full-Time Roles**
  - Rust & OS developers for kernel, graphics stack, and hardware abstraction
  - UI/UX engineers for further development of the graphical interface (COSMIC fork)
  - Build/tooling specialists for CI/CD, package management, and cross-compilation
- **Contract Structure** Project-based, remote-friendly roles — ideal for open-source freelancers or research engineers

5.3 Community Programs

A thriving ecosystem emerges through active participation. To encourage this, we will implement:

- **Bounty Programms** Financial incentives for critical components (drivers, documentation, IDE features)
- **Workshops & Hackathons** Events to actively involve external developers — particularly from Rust, RISC-V, and embedded communities

5.4 Public Outreach & Visibility

Visibility is key to attracting users, developers, and partners:

- **Conference Participation** Talks and demos at events such as RISC-V Summit, FOSDEM, CCC, RustConf
- **Multimedia Marketing** Technical blog articles, video demos, and project showcases on platforms like Mastodon, YouTube, LinkedIn
- **Travel & Materials Budget** For in-person networking and outreach

5.5 Funding Strategy

We aim to build a diversified and stable funding base:

Funding Source	**Purpose**
Grant Programs	e.g. Prototype Fund, EU Horizon, ZIM
Corporate Sponsorships	Especially from RISC-V and embedded industry
Community-Crowdfunding	Early supporters, beta testers and open source fans
Long Term Foundation	Inspired by models like Apache Software Foundation

5.6 Why It's Worth the Investment

- **A Free, Secure Mobile OS for RISC-V Is Still Missing** The project fills a critical gap in the open-source ecosystem.
- **High Value for Industry Partners** Thanks to the Apache 2.0 license, the system is freely usable and easily adaptable — ideal for OEMs and research institutions.
- **Community-Driven Growth Potential** An open architecture and engaged developer base accelerate innovation and adoption.