SPACEMIT
进迭时空

A User-Friendly and AI-Ready Desktop for RISC-V: Bianbu LXQt

Xiaogang Fan, SpacemiT

SPACEMIT
进迭时空

Why LXQt?

Conclusion & Future

01

02

03

04

Early experiments

Our solution

Early experiments

Comparing features of mainstream Linux desktop environments

1

GNOME

Mature ecosystem , but GTK4 apps have slow launch times

2

KDE

Windows-like, but some components are complex

3

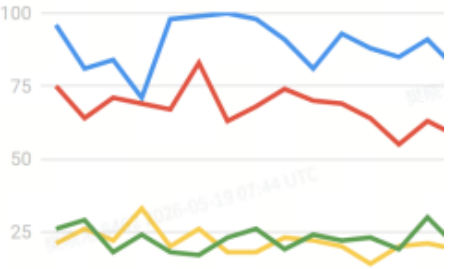
Cinnamon

Same as GNOME

4

MATE

Lightweight, but the interface is a bit outdated



Google Trend

GNOME has a slightly higher market share than KDE

Fragmented Hardware Support

Linux desktops struggle with diverse peripheral landscape.

1

Peripheral Support Gaps

Our SBC does not use standard X86 PC hardware.
Standard Linux desktop stacks lack support for SBC peripherals.

2

No Unified HAL

Unlike Android, Linux Desktops lacks a comprehensive Hardware Abstraction Layer (HAL), making integration efforts more complex and fragmented.

Handling AI architecture differences

SpacemiT CPU IME instruction set support

1 AI CPU Core

The SpacemiT processor integrates IME instructions to accelerate AI computing within the CPU core via instruction-level parallelism, distinct from GPUs and NPUs.

2 AI inference framework adaptation

We fully adapted the software stack, from low-level operator libraries to inference frameworks like llama.cpp and ONNX Runtime.

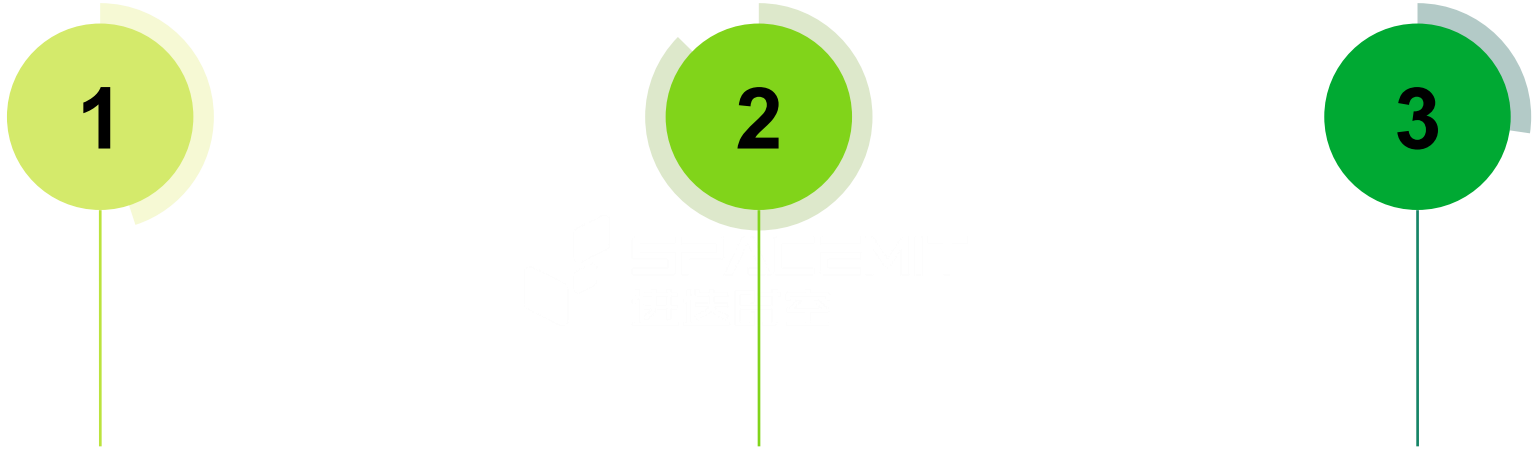
Why LXQt?

Considering both performance and user experience

Performance is the key bottleneck	Target User	Advantages of LXQt
On current RISC-V hardware, GNOME fails to deliver a good user experience.	Not just geeks, Also include industry users and beginners.	Lightweight, fast, resource-efficient, smoother desktop experience, balance of performance and usability.

New development approach

Continuously optimize based on real user feedback



User Driven

Continuously collect feedback from educational institutions, developers, and early adopters.

Focus on pain points

Iteratively fix key issues such as interface lag, app crashes, and complex configurations.

Final Goal

Ensure ordinary users can focus on creation and learning without worrying about system stability or manual tuning.

Full-Stack Development

A complete solution from hardware to applications

1

Hardware-software collaboration

Our work spans the entire chain, from IP design, SoC integration, and device manufacturing to OS optimization, applications, and introductory tutorials.

2

Out of the box

For entry-level users and educational scenarios, we strive for a smooth, beautiful, and clean experience.

Openness & AI

Lower the barrier to AI development

1

Support for mainstream frameworks

We have completed support for mainstream inference frameworks such as ONNX Runtime and llama.cpp.

2

Rich development resources

AI sample programs covering scenarios like image recognition and speech processing.

3

Feature a GUI

Many examples feature intuitive GUIs, making them much more accessible to beginners.

4

Getting Started Guide

Provide detailed tutorials to guide users in reproducing AI functionalities from scratch.

Conclusion & Future

It is meaningful to optimize the desktop experience and lower the barrier to AI development.

Lightweight office work

Continuously optimize key applications like teamwork tools and office to enhance overall productivity.

01

02

03

upstream collaboration

Work closely with the open-source community and upstream projects to standardize key improvements and contribute them back to the community.

AI-native platform

more tools and solutions to easily create their own AI Computers and Robots.