XenevaOS



An Operating System project with its own kernel written from scratch for XR device and time critical devices adding general purpose and spatial purpose usage.

Introducing Xeneva

How we began?



Manas Kamal Choudhary



Ayushmaan Bora

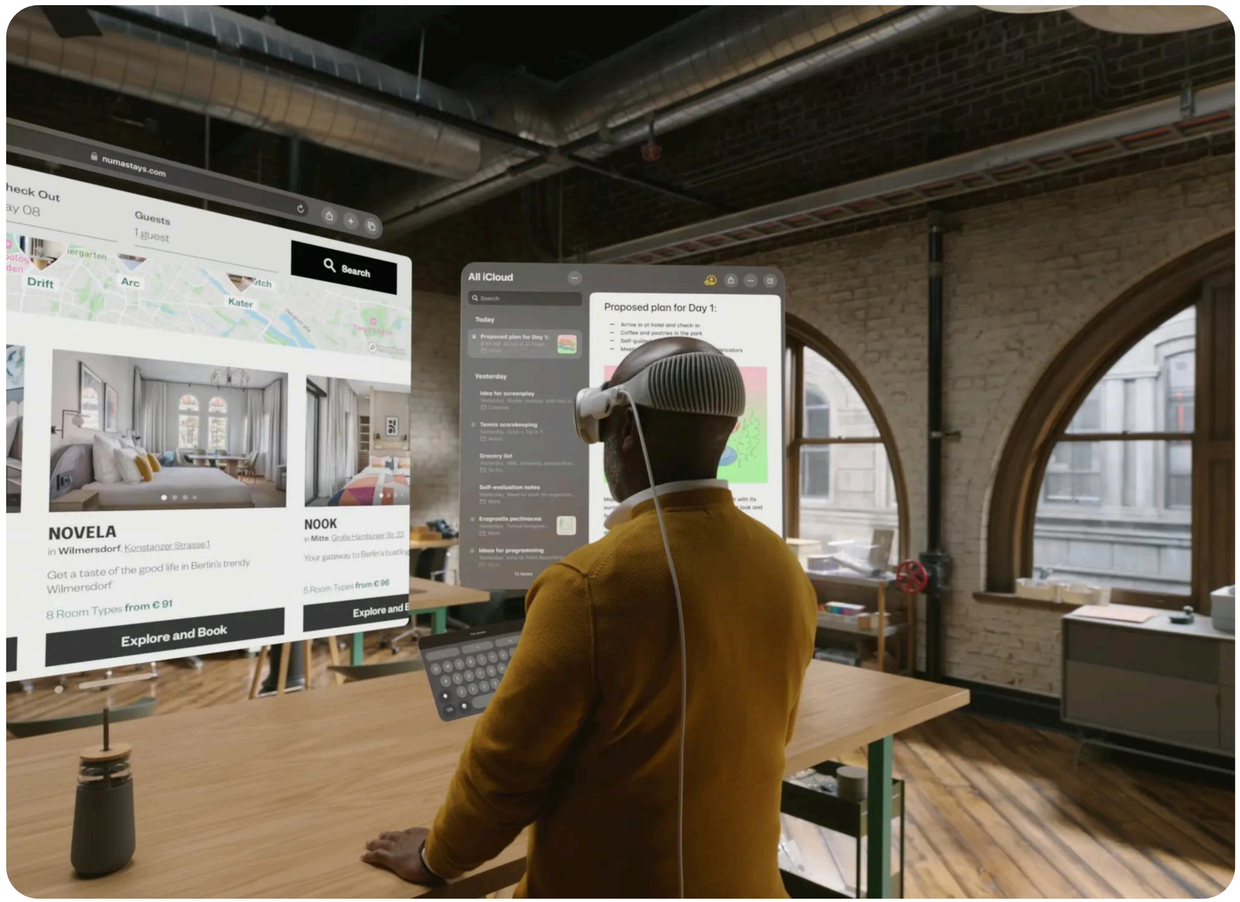
At our core, an OS company: XenevaOS, built on our own in-house Aurora kernel

Started in 2015-16 as Manas's personal, self-taught OS-development project

Ayushmaan joined the project through Reddit, and gave it a new direction: Extended Reality

Grew up alongside this very community. Xeneva was born out of IndiaFOSS

The Future is Spatial.



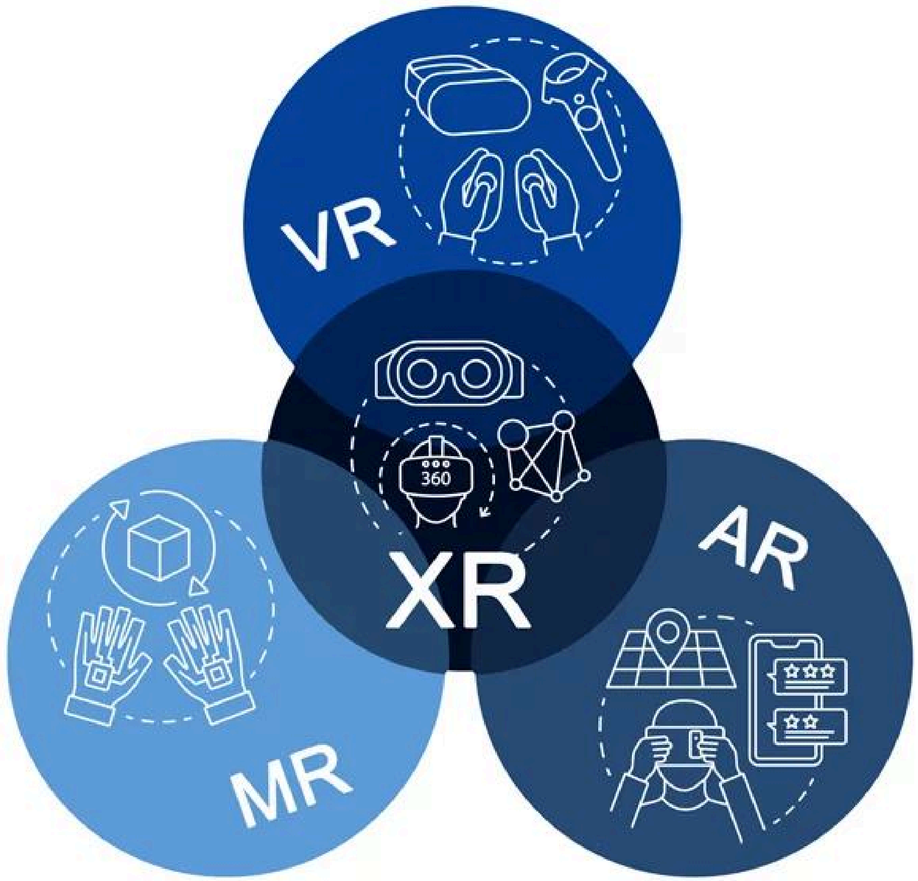
Computing started on the desk:
we went to it

It moved into our pocket:
we carried it

It's now moving into the space around us :
we wear it

XR collapses the screen back into the real world, and computing becomes ambient, contextual and hands-free

What is XR?



XR (extended reality) is an umbrella term for AR + VR + MR

Virtual Reality (VR) replaces the perception of the real world with a virtual environment, for example a virtual training or flight simulator

Augmented Reality (AR) renders digital information on top of the real world, for example a map displayed through a phone camera

Mixed Reality (MR) goes a step further than simple AR, with much more complex virtual objects rendered on top of, and anchored to, the real world

Why XR?



Screens fragment attention: heads down, eyes on a rectangle, disconnected from the room

XR keeps information contextual, overlaid on reality rather than replacing it

Hands stay free, as gesture, gaze and voice replace taps and swipes

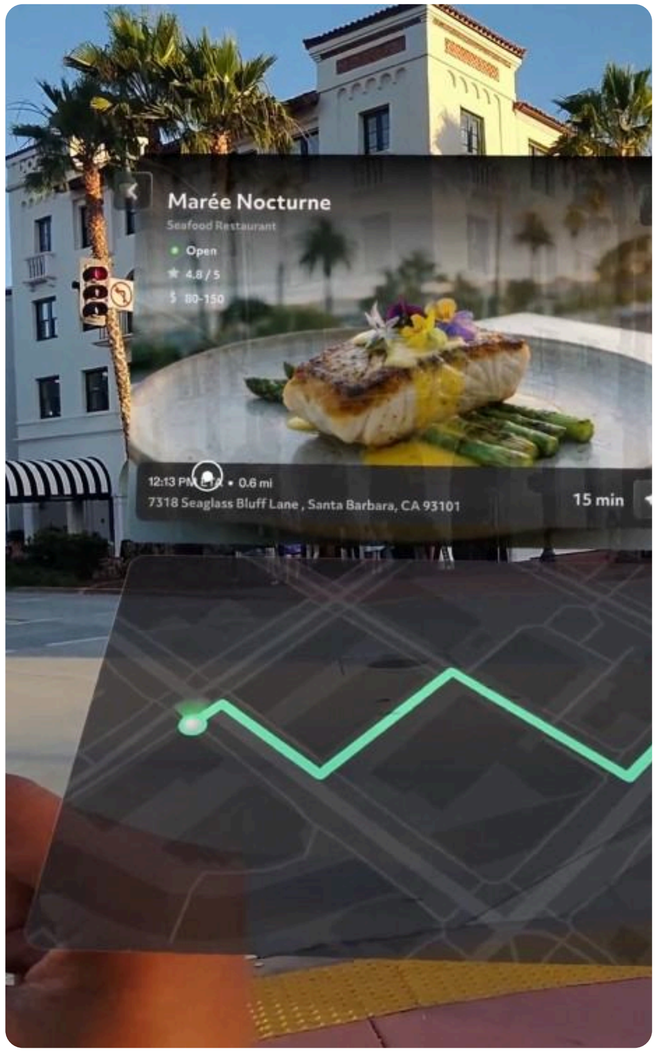
What we grew up seeing in sci-fi, AR glasses as everyday computers, is now buildable

Adoption of XR

Enterprise & industrial: training, remote assistance, design review, defense and field operations, ahead of the curve

Consumer: still early, held back by form factor, comfort, battery/thermal limits and price

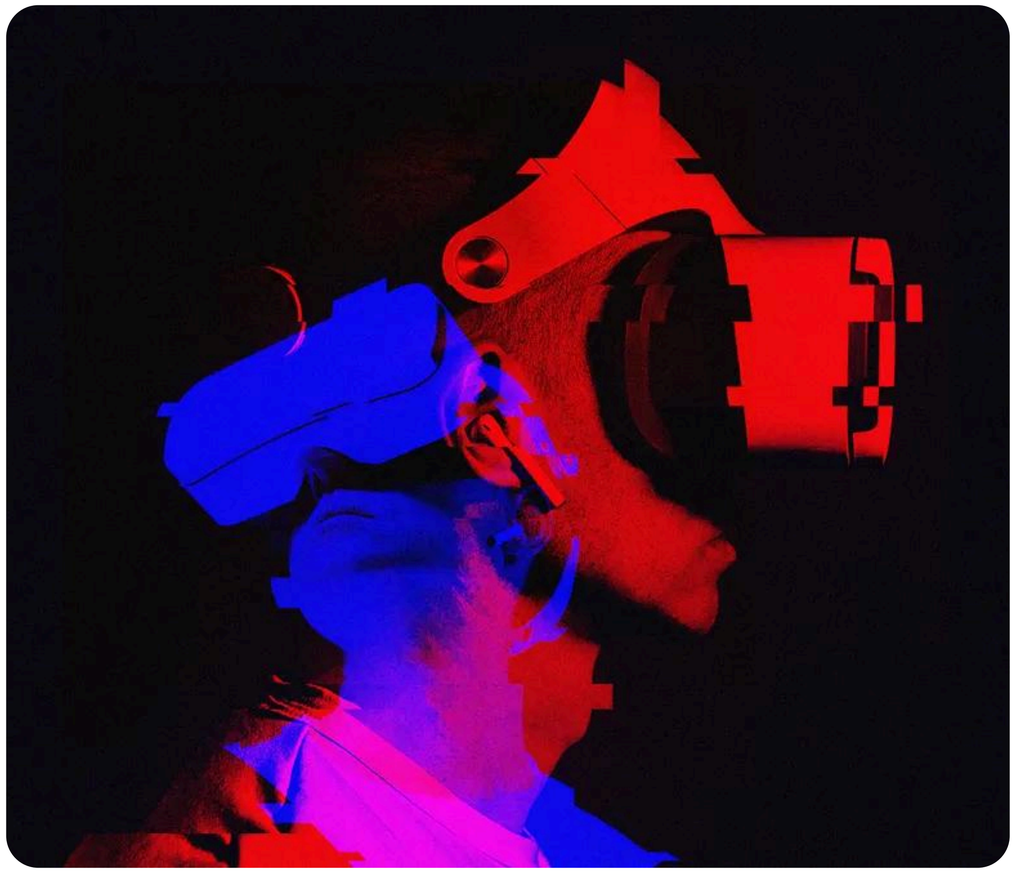
The consumer ecosystem is still maturing around all of this







Issues in the World of XR



Latency & motion sickness: motion-to-photon lag breaks presence, and can make people physically ill

Battery & thermal limits: real-time rendering and tracking are power-hungry on a device worn on your face

Form factor & comfort: weight and heat still stand between a headset and a pair of glasses

Fragmented hardware: drivers, SDKs and toolchains differ across nearly every device

No purpose-built operating system: most XR devices run a mobile or desktop OS adapted after the fact and has no basis on its ecosystem, making it seem unadaptable

Xeneva's Role in XR

Every issue on the last slide traces back to one root cause: *XR hardware is running software that was never designed for XR*

We are building XenevaOS from the kernel up, for the timing, spatial and power demands of XR, not adapted onto it afterward

Mature Kernels and Their Intent of Development

Unix – solving the problem of large, expensive computers being difficult to use

BSD – solving the problem of turning Unix into a powerful research and networking platform

Linux – solving the problem of having a free Unix-like kernel for everyone to use and develop

QNX – solving the problem of deterministic embedded computing

VxWorks – solving hard real time control problem

Android – solving the smartphone application ecosystem problem

ChromeOS – solving the web-centric computing problem

How our Kernel Solves Existing Problems.

Scheduling algorithms such as EDF (Earliest deadline first) algorithms for XR scheduler

One common XR chained scheduler for both CPU/GPU

Ready-to-use kernel resources similar to Real Time Behaviour

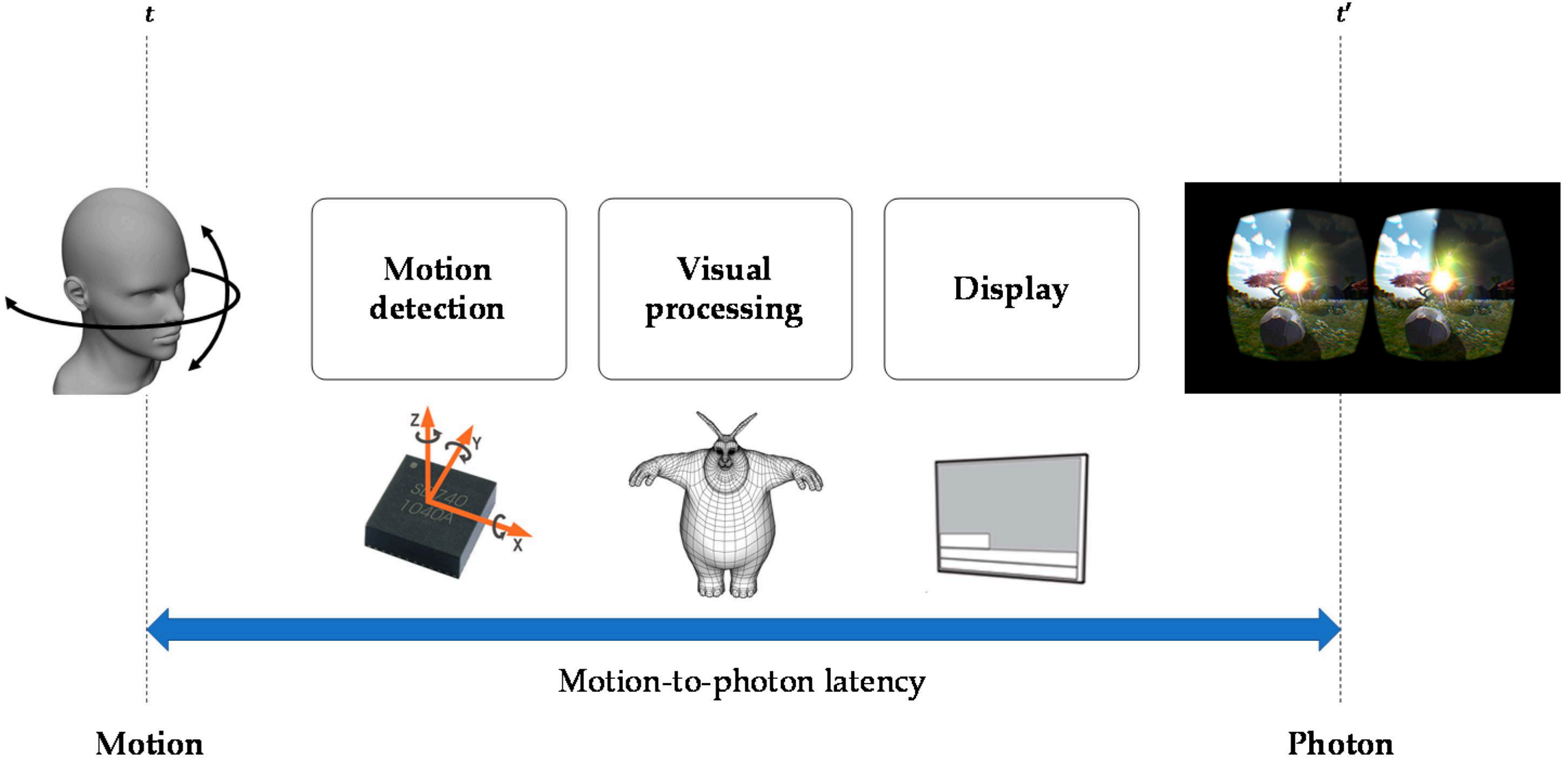
On-demand hardware power on/off/pause/resume behaviour

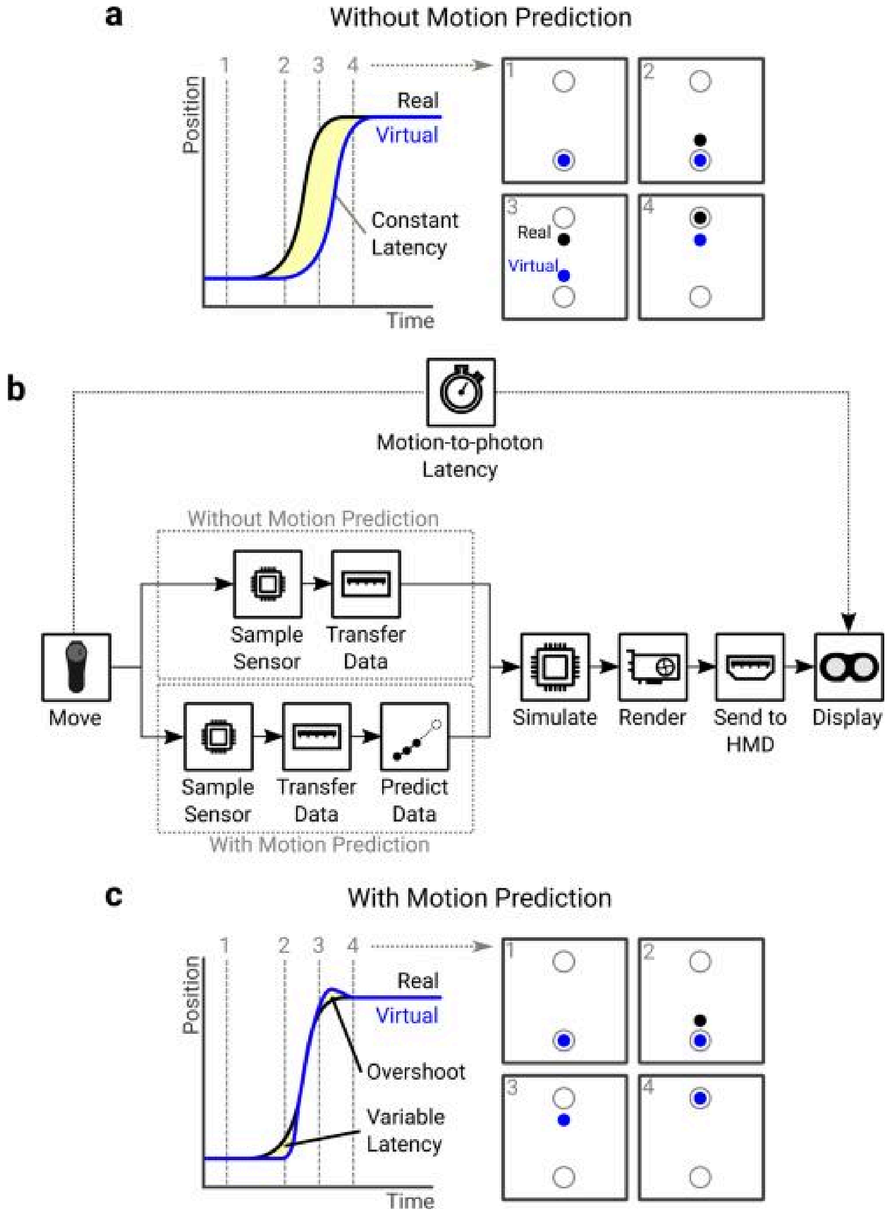
Mature Kernel like Linux has deadline and real time scheduling but mostly struggles with GPU scheduling for XR

Predictable and guaranteed timeline between software components

Motion to Photon Latency

Warburton, M.,Mon-Williams,M.,Mushtaq,F., &Morehead, J.R. (2023). “Measuring motion-to-photon latency for sensorimotor experiments with virtual reality systems.”,





MTP existing research measurement

As per research on devices like HTC Vive, Oculus Rift, Oculus Rift S, Valve Index, sudden movement's MTP pipeline latency ~ 21-42ms

---

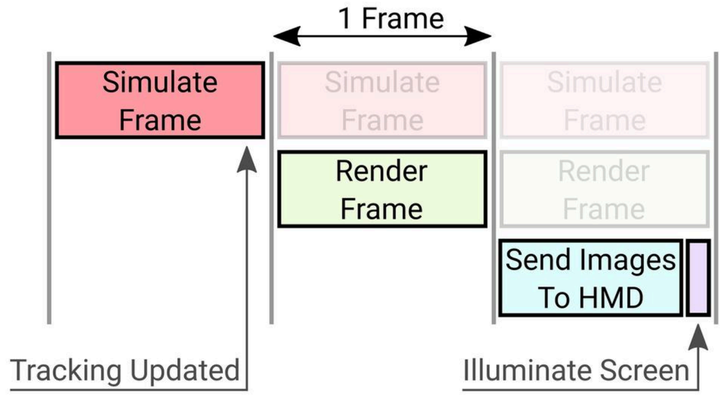
After Prediction algorithm effective : 2-13ms

Transition occurred approx. 25-58 ms after movement onset.

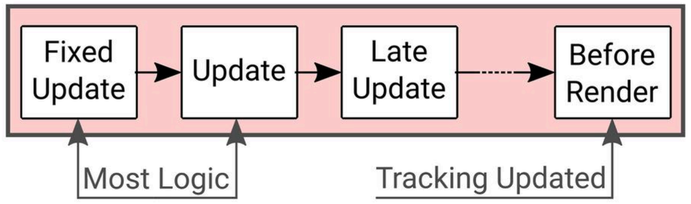
Warburton, M.,Mon-Williams,M.,Mushtaq,F., &Morehead, J.R.(2023). "Measuring motion-to-photon latency for sensorimotor experiments with virtual reality systems.",

Measurements under rendering load

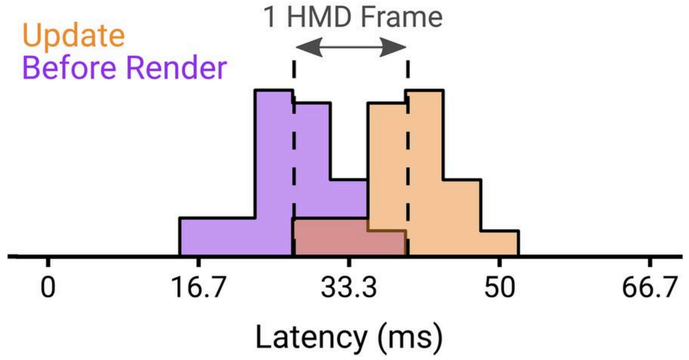
a Frame Timeline



b Simulation Flow



c Effect of Loop Used



As per "Time Sequential MTP latency measurements for Virtual Reality Head-Mounted Displays"

Rendering Workload	Average MTP
Low	46.55ms
Workload 1	63.22ms
Workload 2	101.29ms
Workload 3	154.63ms

hoi,S.-W,Lee,S.,Seo, M.-W.,& Kang, S.-J.(2018). "Time Sequential Motion-to-Photon Latency Measurement System for Virtual Reality Head-Mounted Displays." Elctronics, 7(9),171

Roadmap of R&D for Xeneva's Kernel

Uni-kernel approach with pico-process model

Optimized Software pipelines

Micro-optimization in Kernel

Reduced Software abstraction layers as much as possible

Running the Kernel in Hypervisor layer and providing the OS as a Library to pico-process

Xeneva From a Developers Perspective



Xeneva aims to have an easy build environment.

Currently the project is multi-platform aware, i.e. it supports MSVC, Clang/LLVM and GCC.

Default ISO C11 standard is used in MSVC and GNU17 standard is used in Clang/LLVM.

The Project uses multi-licensing model. Core Project uses BSD-2 Clause License while DCL Layer uses GPLv2

Kernel could follow some of the traditional operating system philosophy with a modern approach.

For example, Unix file philosophy + Capability based security of Object based system.

Basically Having Unix Signal mechanism with better user control.

Xeneva's Struggles as an Open Source Project.



Writing a new Operating System kernel from scratch – and trying to take it beyond a hobby project – is challenging in today's ecosystem.

The development struggle of XenevaOS is broadly divided into two software phases.

Tackling OS Dev Struggles Phase 1 *sigh

Predictable memory allocation and deallocation.

Deadline based scheduling even with SMP.

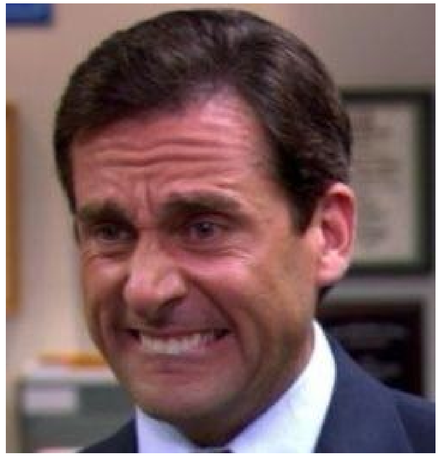
Selection of suitable data structures.

Low Latency kernel pipelines.

Micro optimization in kernel.

Research on pico-process and uni-kernel approach while having multiple apps running.

The Chicken–Egg Problem (Phase 2)



Hardware vendors generally have little incentive to invest engineering resources into writing and maintaining drivers for an early-stage operating system.

At the same time, an Operating system cannot become usable on Bare Metal without hardware support.

That’s the chicken-and-egg problem.

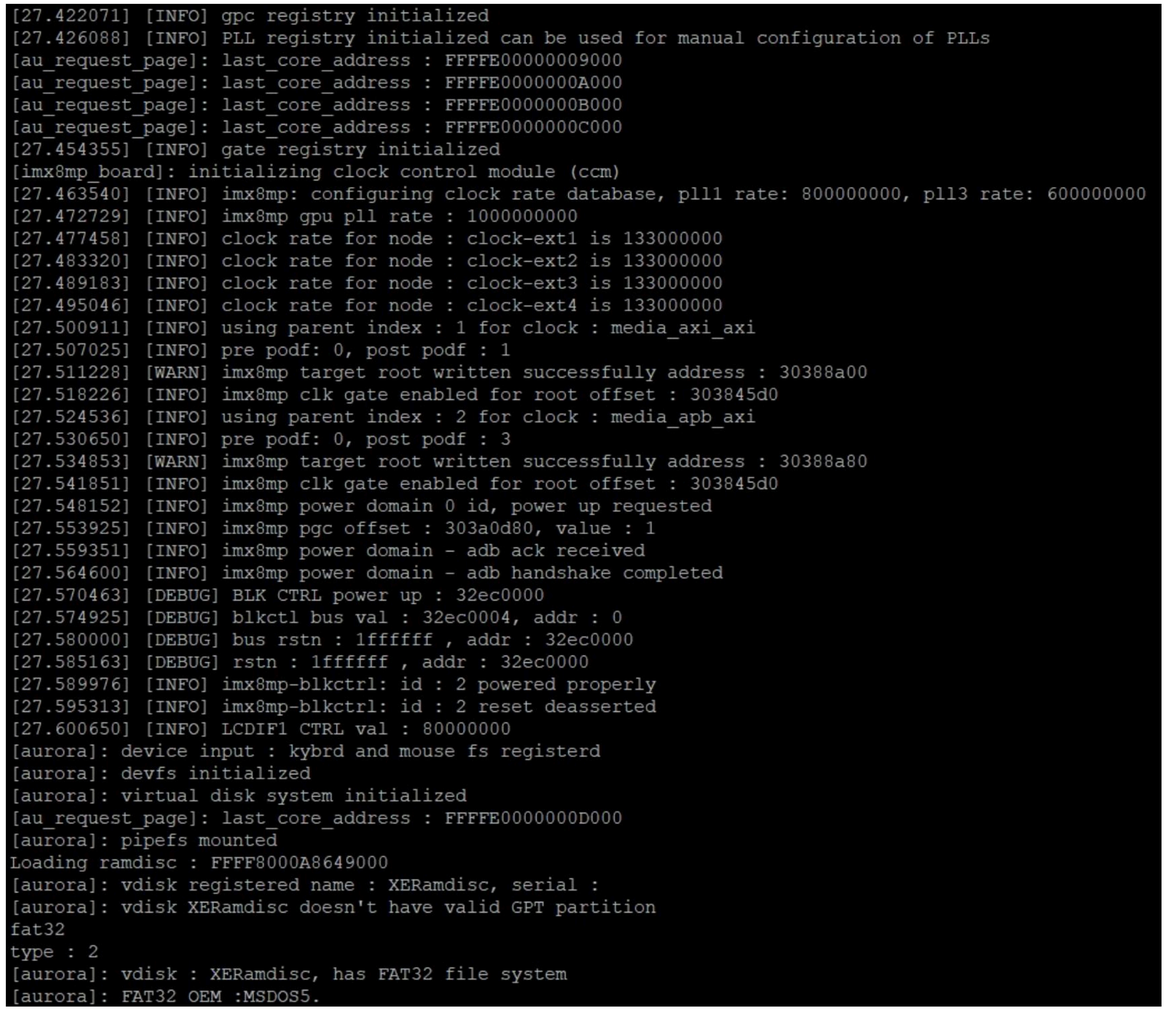
Approaches to this Chicken-Egg problem (Phase 2)

Rewriting or porting hardware drivers is an important part of Xeneva's R&D roadmap – especially GPU drivers for XR hardware.

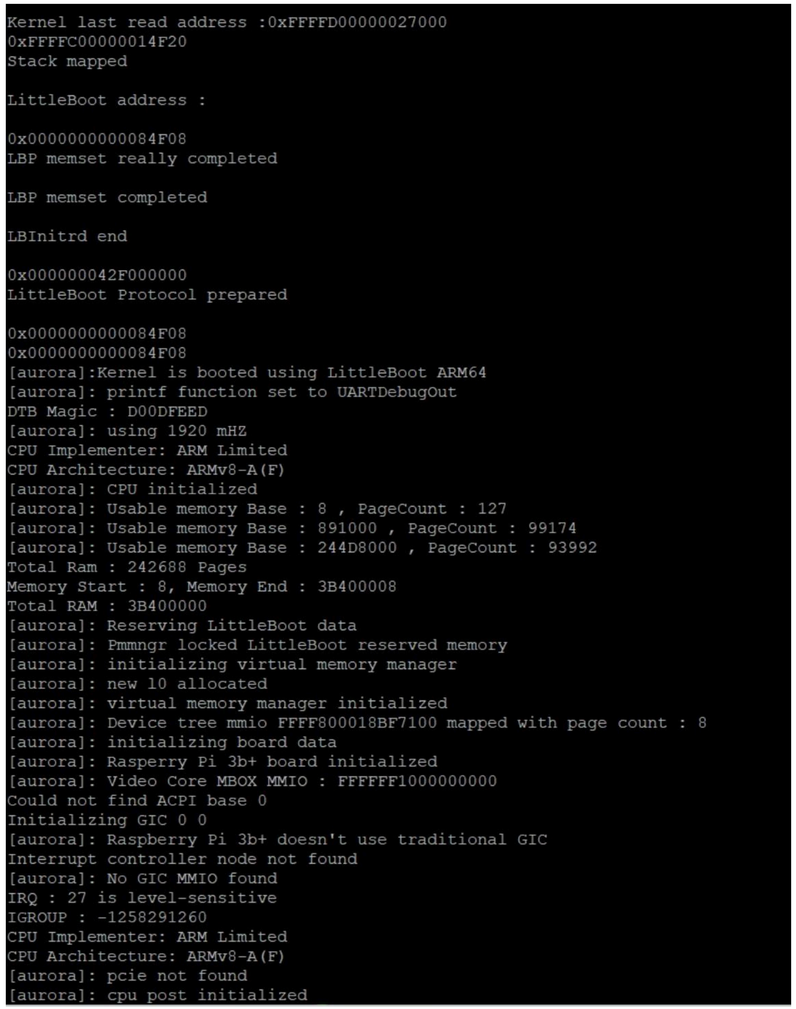
XenevaOS has its own driver framework designed to support integration between upstream driver resources and downstream kernel callbacks.

Rely on Open and Widely supported hardware standards such as: VirtIO (QEMU/Virtualization), USB, NVMe and AHCI.

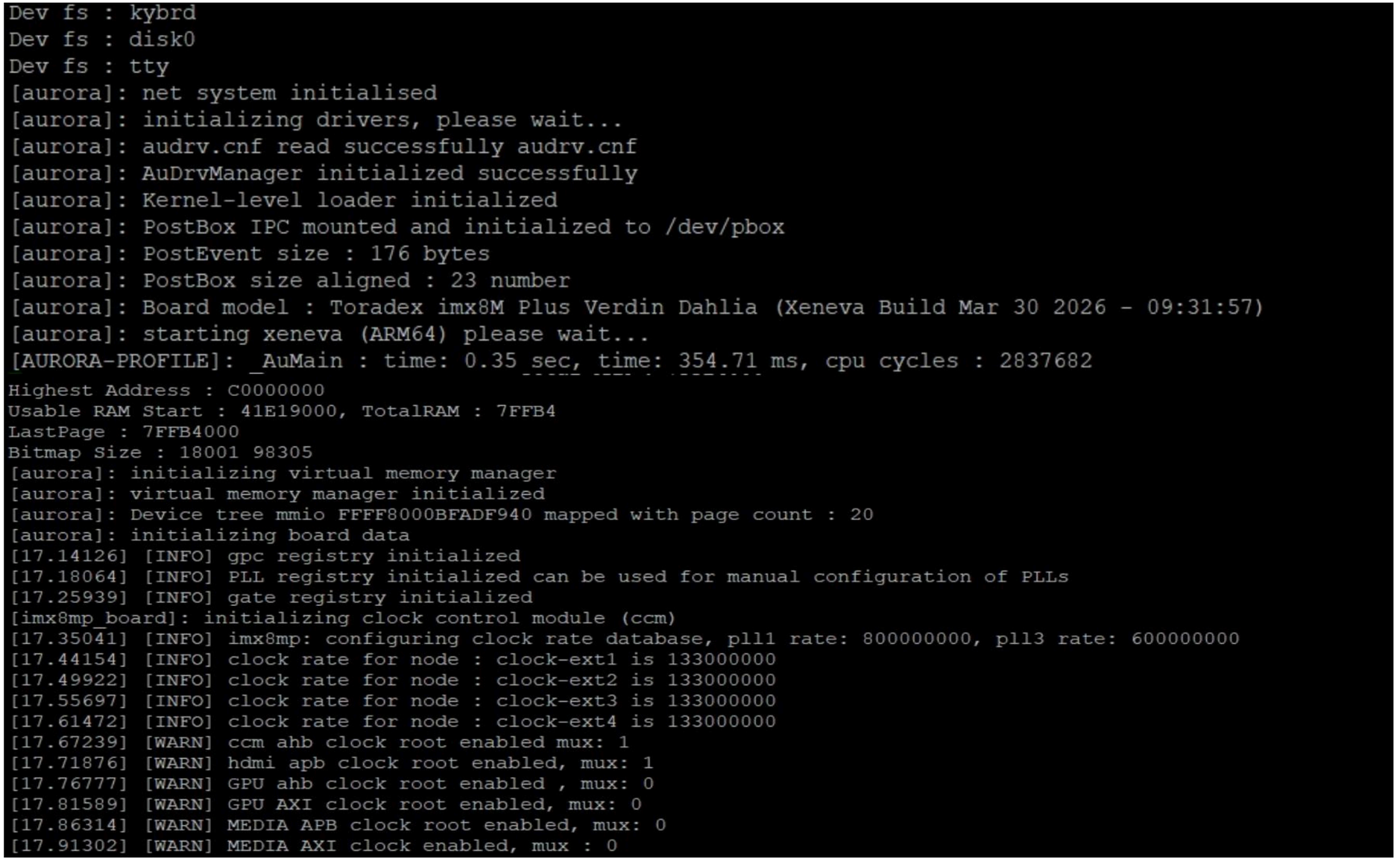
Research is also underway into porting existing Linux Drivers through the Driver Compatibility Layer (DCL) project.



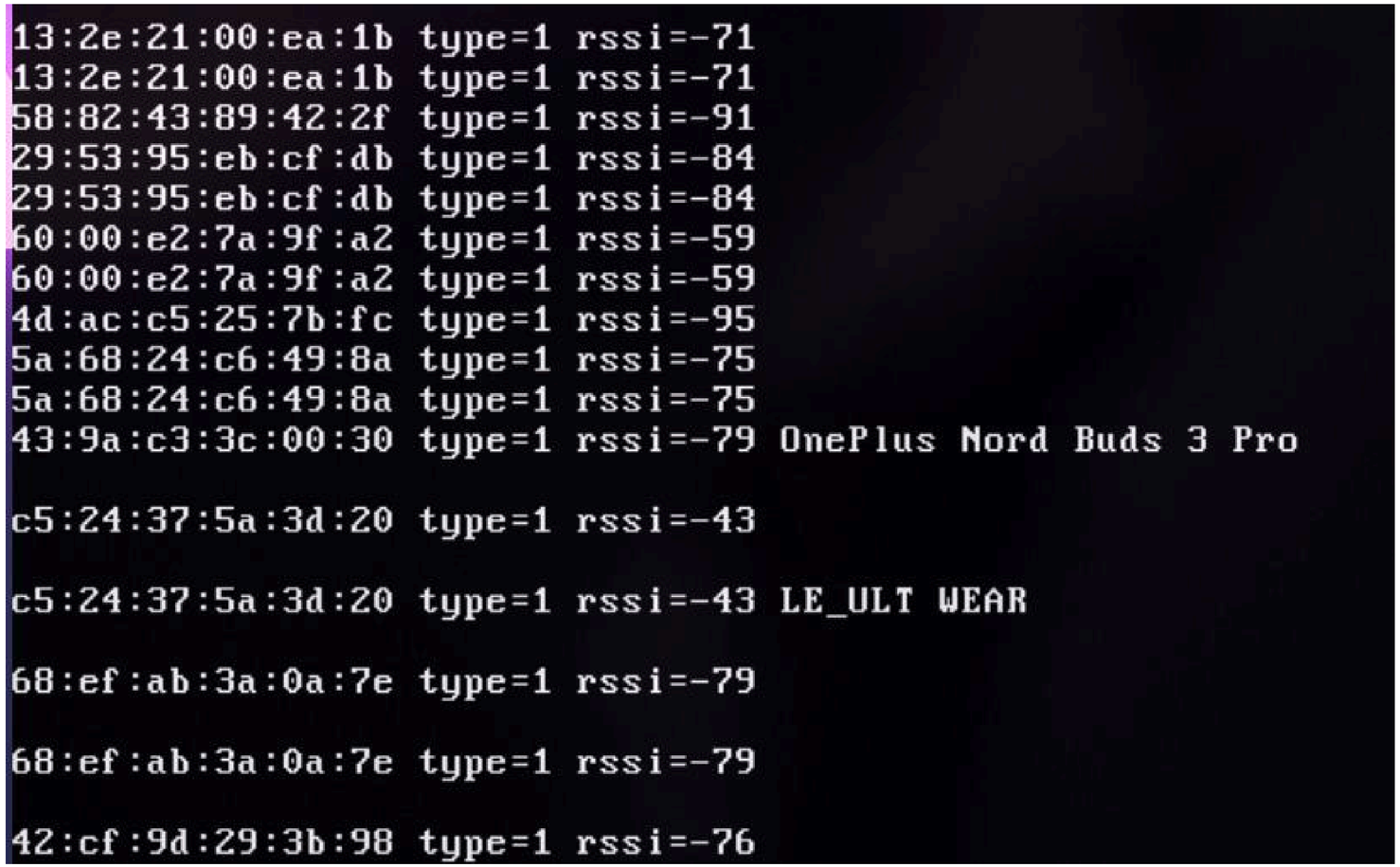
i.MX8MP SoC with CCM/GPC/BLKCTRL driver, ported from U-Boot/Linux source code using Xeneva's in-house driver framework.



Raspberry Pi 3B+ with DWC2 OTG driver ported from RPi 3B+ bare metal USPI Library



Early PoC to the Phase 1 Kernel and Software Abstraction



BLE Stack Implementation Demo

Thank You

Please Visit Our Booth for Live Demo of XenevaOS

GitHub | https://github.com/manaskamal/XenevaOS