EMTERIA

Second chance: Upgrading devices from Android 9 to Android 14

A real case of using open-source software to save thousands of old devices from being disposed and replaced

Igor Kalkov-Streitz

FOSDEM'25 / Brussels

Igor Kalkov-Streitz

AOSP developer,
PhD in embedded and real-time systems,
co-maintainer of Android for RPi,
CEO at emteria.

igor.kalkov@emteria.com
linkedin.com/in/kalkov

Android for RPi

Public AOSP repositories for Raspberry Pi

https://github.com/RTAndroid
https://github.com/android-rpi
https://github.com/emteria

emteria GmbH

- Providing Android support for SBCs and custom devices
- Validating and performing security audits for Android OS
- Offering over-the-air updates & device management services

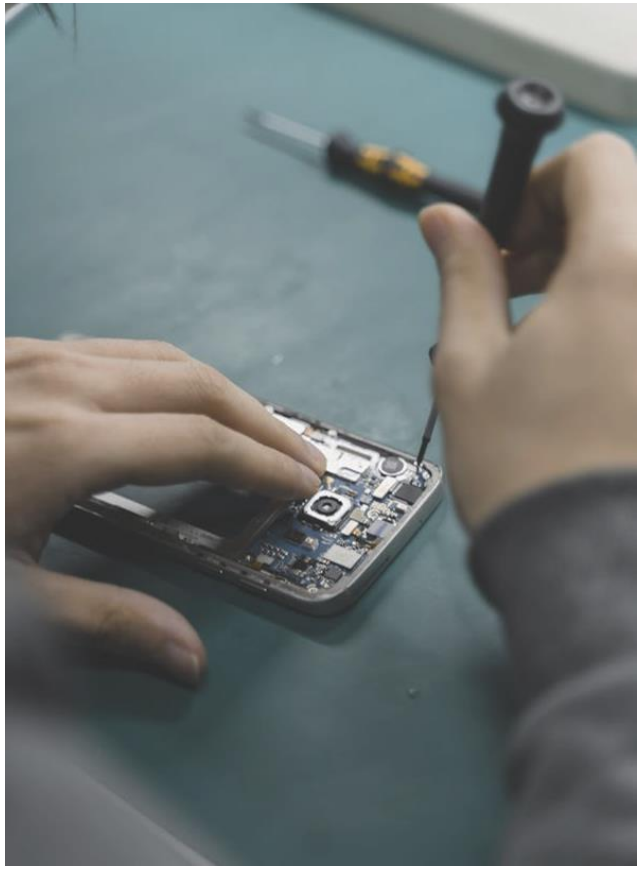
Software Obsolescence is a Problem

Why upgrading sofware on existing devices is important

- Hardware manufacturers make money by selling (new) devices
- Rapid Android release cycles → shorter device support lifespans

- Security vulnerabilities in unmaintained Android releases
- EU regulations & compliance for device longevity
- Economic & environmental benefits of upgrading instead of replacing

- Main project goals:
  - Upgrade an existing device from Android 9 to something newer
  - Reduce dependencies from the original device manufacturer
  - Use open-source components where possible



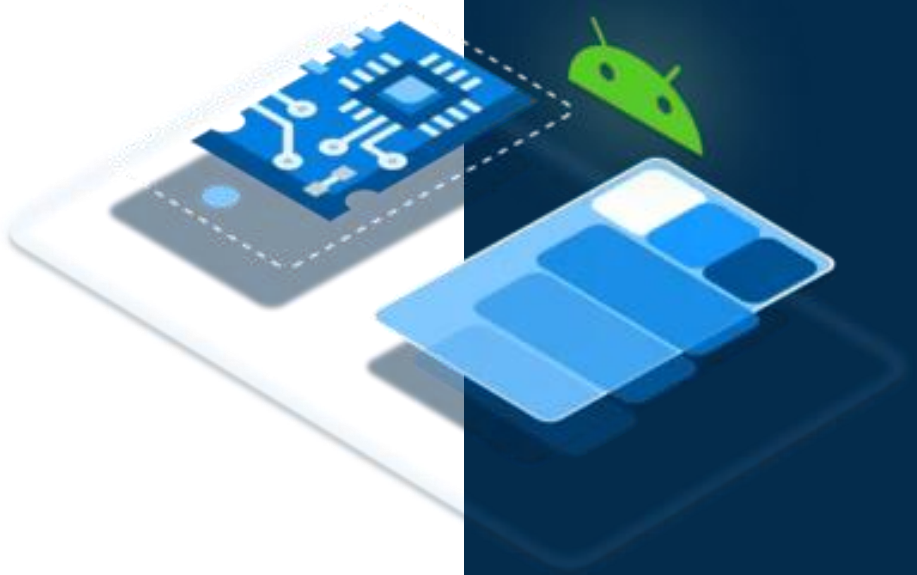
Hardware Specs

Specs of the device and usable interfaces

- Test device available for our experiments
  - No schematics, no JTAG, but UART is available
  - Unlockable bootloader, working fastboot

A good start!

- Popular Qualcomm Snapdragon SoC from 2018
- Lots of peripherals
  - Media: multiple cameras, audio
  - Wireless: WIFI, BT, NFC, radio
  - Extras: fingerprint reader, sensors



Software Specs

Contents of the original Board Support Package

- Original Android version: 9
- Original Kernel version: 4.9

- Proprietary BSP in a single ZIP file
  - Repo manifest with all projects: no
  - Git history for old changes: no
  - Technical support from ODM: no
  - Support from client's in-house expert: yes

A huge win!

Research of Open-Source BSPs

Gathering information about similar devices and BSPs

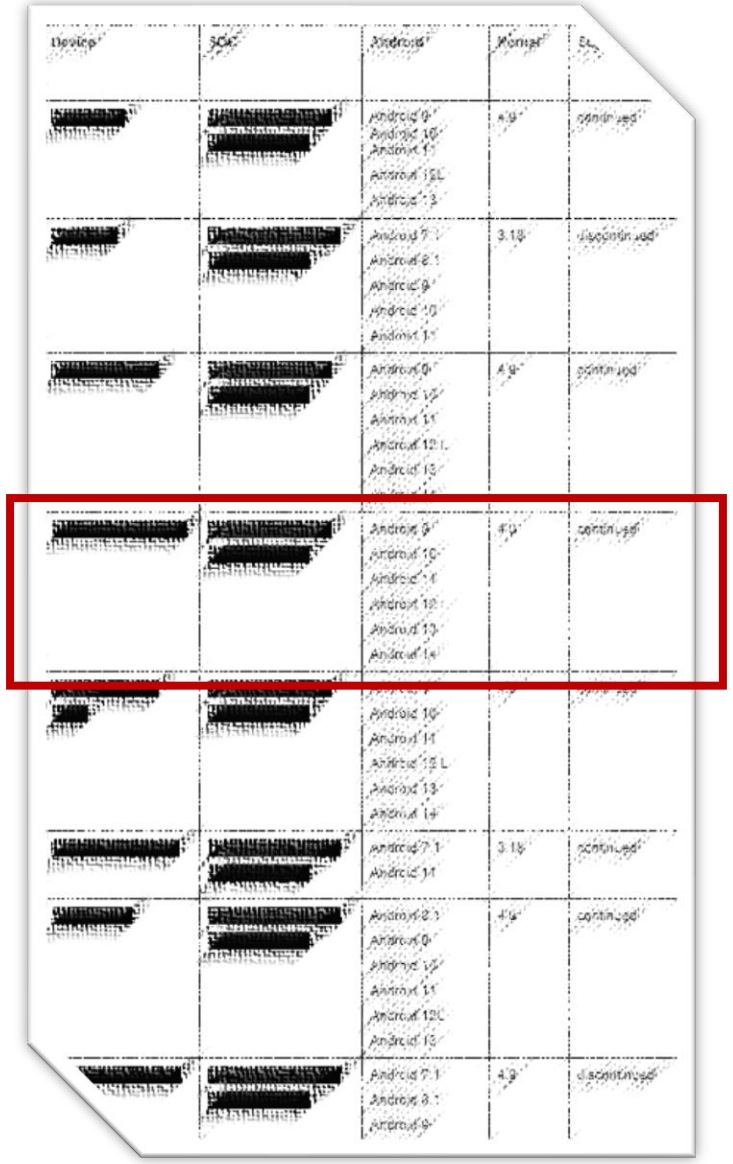
- Search for compatible devices and BSPs online
  - Same or pin-compatible SoC
  - Similar kernel version

- Popular open-source Android distros (in alphabetical order):
  - AOSPA: https://github.com/AOSPA
  - CalyxOS: https://gitlab.com/CalyxOS
  - /e/OS: https://gitlab.e.foundation/e
  - GrapheneOS: https://github.com/GrapheneOS
  - LineageOS: https://github.com/LineageOS
  - XDA Forums: https://xdaforums.com

Result Evaluation

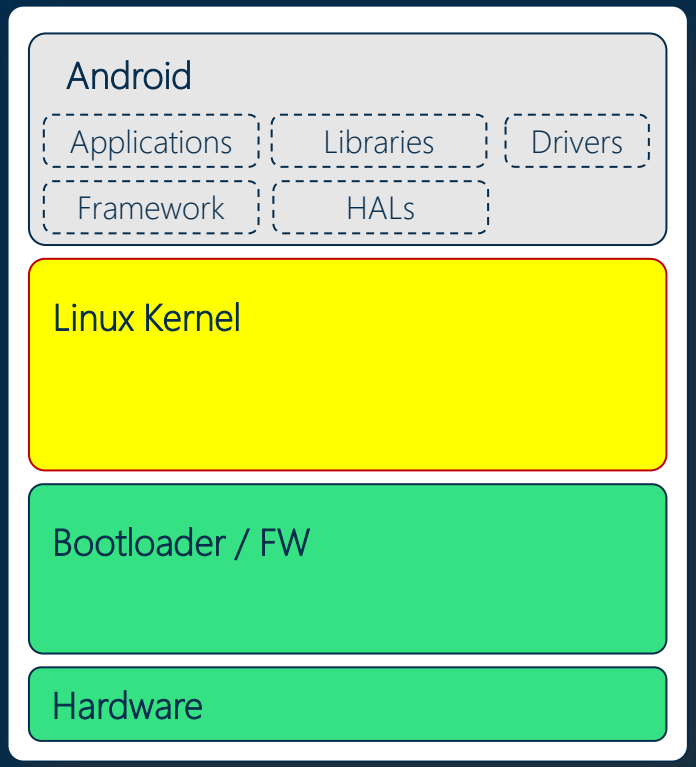
Deciding which BSP is the best match

- Create a list of maybe-compatible open-source BSPs
  - Device name and SOC information
  - Kernel and Android version(s)
  - Development status
- Pick the most promising one
  - Closest match for the Linux kernel: same version!
  - Newest Android version: 14!
  - Similar peripherals: more or less
  - Officially supported by LineageOS: yes
  - Actively maintained: yes



Step #1: Linux Kernel

Preparing the device tree and the defconfig



- Goal: Make the original kernel compile with the new BSP
  - Possibility 1: fully replace LOS's kernel with original kernel
  - Possibility 2: migrate original DT & drivers into the LOS kernel

This was easier in our case.

- Difficulties compiling the original kernel
  - Dependencies on LOS' makefiles and defines
  - Different toolchains in original BSP and in new BSP

- Keep the device tree small

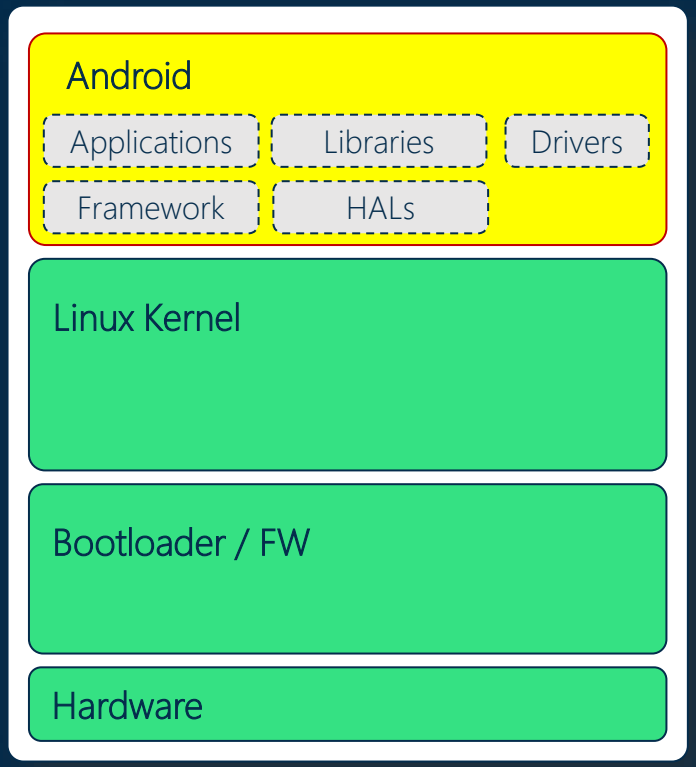
Mainly early partition mounts.

  - Add Android-specific DT definitions
  - Disable all non-critical drivers in the defconfig

Like audio, touch, radio, networking.

Step #2: Android Boot

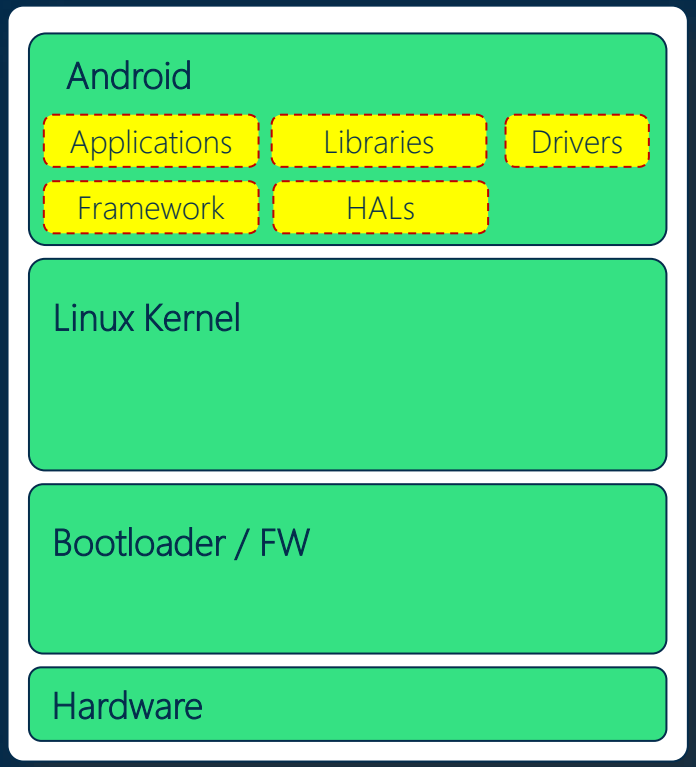
Let the device boot for the first time



- Goal: Get Android to boot to the main UI
  1. Make device config similar to the original one
     - Example: make the partition layout compatible.
  2. Make device config as small as possible
     - Example: disable all non-critical features and HALs.
  3. Make device config as permissive as possible
     - Example: disable security (AVB, SELinux, etc.)
- Repeat the cycle:
  - Compile, flash, boot, record logs
  - Fix the most critical issue which causes freezes or boot loops

Step #3: Userspace

Debugging remaining features step by step



- Goal: Bring back features and peripherals

  Start with Touch, USB, ADB, WIFI, …

- Continue debugging and fixing step by step
  - Record an error trace for a specific component
  - Figure out how it is configured in LOS
  - Figure out how it was configured in the original BSP
  - Re-enable kernel driver config and DT entries
  - Make corresponding adjustments or replace HALs/blobs

- Cleanup and finalize
  - Enable SELinux and extend product-specific policies
  - Replace signing certificates

Conclusion and Future Work

Key takeaways and next steps

- This project was successful, but it is not yet complete
- We will continue working with Lineage repos and contribute where possible

- Huge dependency on availability of code and knowledge
  - Would not be possible without open-source software → plays huge role for device longevity
  - Difficult without ODM support → we had help from an expert familiar with this kernel

- Call to action for everyone
  - Contribute to open-source projects
  - Upgrade existing devices

Dr. Igor Kalkov-Streitz

Managing Director

igor.kalkov@emteria.com

linkedin.com/in/kalkov