# EOL-less Quintus

First steps towards bringing mainline Linux to the Volla Phone Quintus

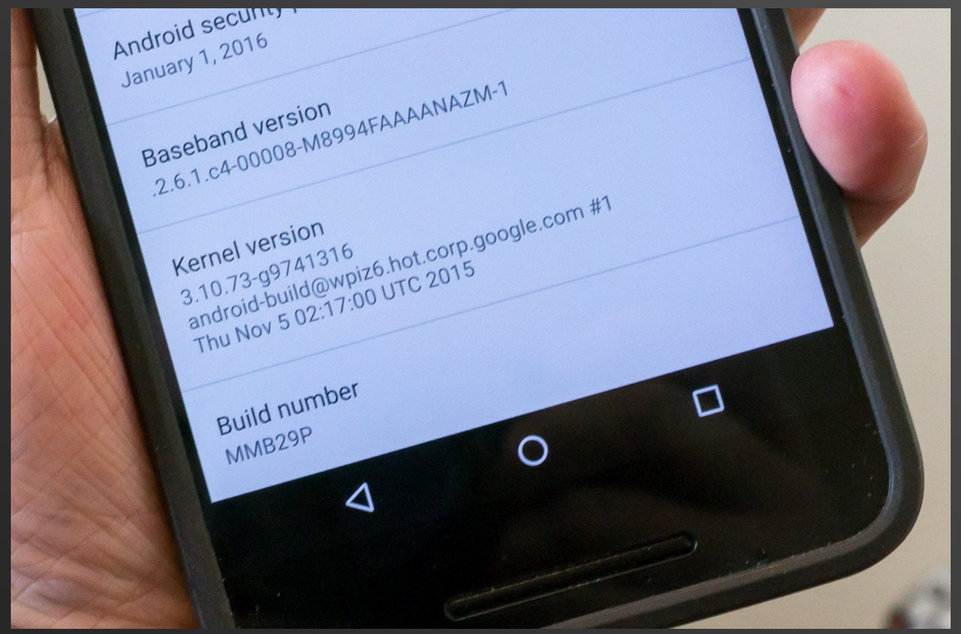
Ivaylo Ivanov, Muhammad Asif

# Table of contents

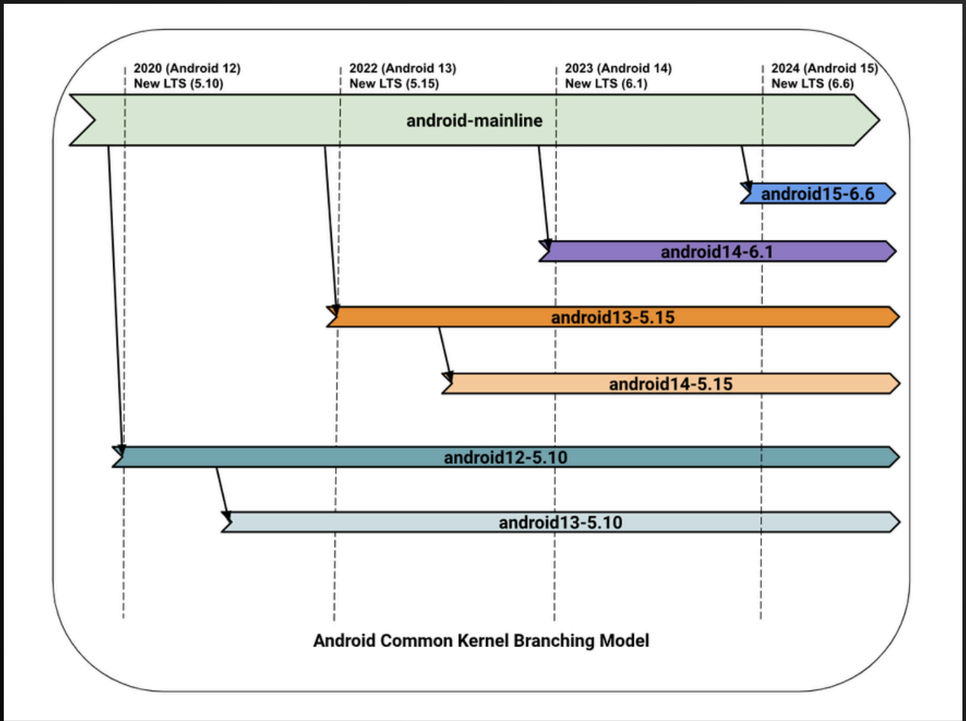
# 1. Introduction to the problem with mass-production phones

# The problem

- Most phones nowadays ship with either iOS or Android
- The ones that ship with Android make use of the Linux kernel, unlike iOS phones
- Due to the nature of Android, it requires extensive patches to the Linux kernel, which are organized by Google



# The problem

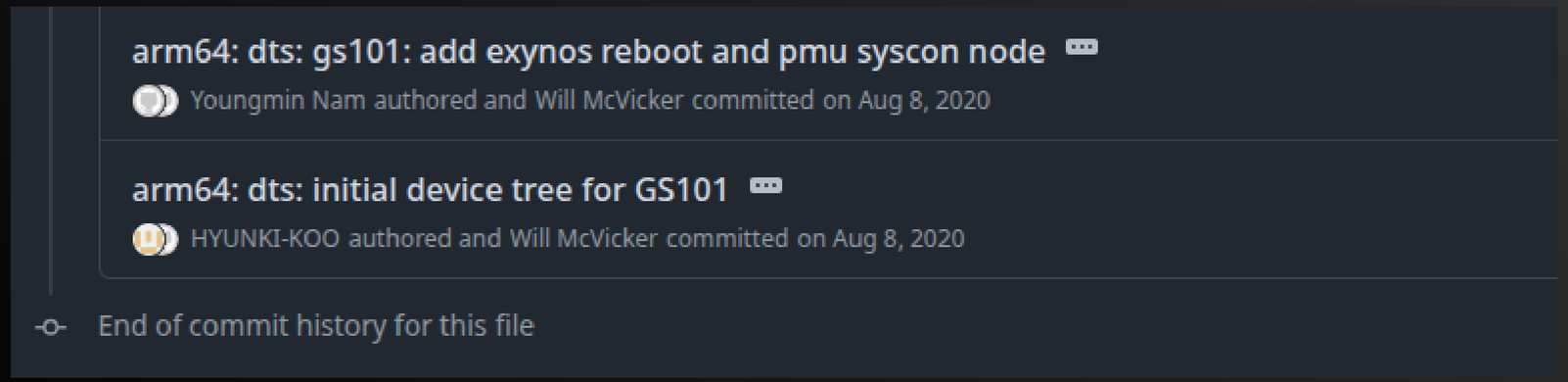



Lifetime of Android kernels

- Phone OEMs use BSPs provided by the SoC vendor, which include stuff like late-stages of bootloader, kernel, TZ, userspace trees, etc
- SoC vendors fork the linux kernel with android patches from Google (officially called ACK), which is forked from torvalds/linux

# The problem

- SoC manufacturers have to go through an extensive amount of development stages, which may take multiple years and then hand-off the already-old codebase to OEMs
- Due to Google's relaxed requirements for kernel versions in the past, SoC vendors never bothered to catch up with newer Linux versions




When did Pixel 6 come out?

October 28, 2021

# The problem

```
1    # SPDX-License-Identifier: GPL-2.0
2    VERSION = 4
3    PATCHLEVEL = 19
4    SUBLEVEL = 191
5    EXTRAVERSION =
6    NAME = "People's Front"
7
8    # *DOCUMENTATION*
```

Kernel version of Quintus;
Phone was released in October 2024
Kernel was released in October 2018
We are at 6.16rc1

- Although that has changed with Google forcing much newer kernel versions for SoC vendors, it still does not fix the dependencies on SoC vendors for drivers
- For example, a kernel upgrade for Qualcomm or MediaTek chipset costs a lot of money that could've been spent on R&D

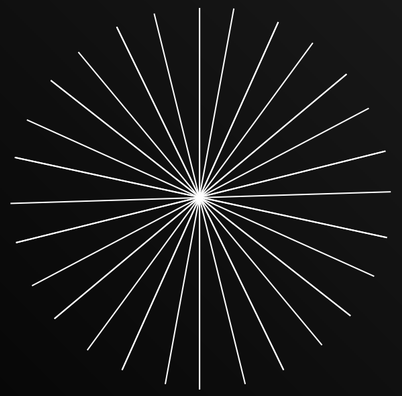
# The problem

- This is not a sustainable situation, as it leaves the SoC vendor in charge of upgrading the kernel source
- SoC vendors also don't entertain B-tier and C-tier OEMs/ODMs, because from the vendor's point of view, they are not as much of a money source as A-tier OEMs.

# 2. What can we do about it?

“If you want a thing done well, do it yourself”

~ Napoleon Bonaparte

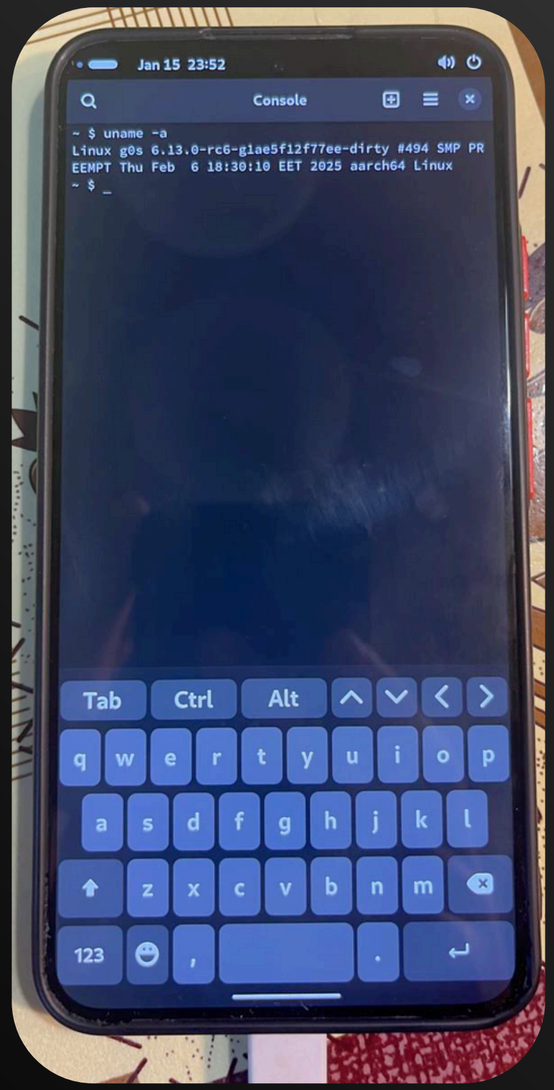


# What can we do

- As normal users who are not affiliated with companies, there is no easy route to achieving a longer lifespan.
- Community projects that exist to try and solve the EOL problem:

1. PostmarketOS
2. Ubuntu Touch
3. Droidian/Mobian

- Most of the devices supported by these projects use the stock OEM kernel (downstream), which relies on extensive prebuilt blobs linked to the Android userspace and libc implementation (bionic)
- Hybris is another example of a project that aims to run GNU/Linux with Android blobs, but it's effectiveness steadily decreases as Linux programs use newer features only available in recent kernels




Galaxy S22+ running mainline Linux with framebuffer, usb and touchscreen

# What can we do

- There are really only two options:

1. Use the vendor-provided kernel with a Linux userspace, which will extend the life of the device for a few more years (until the kernel becomes too outdated)
2. Spend a lot of time and effort getting the mainline Linux kernel to work on the device.

# 3. First steps towards becoming EOL-less

# The Mainline Way!

```
[net-next v1] net: ethernet: mtk_eth_soc:
support named IRQs
 2025-06-14 17:31 UTC  (2+ messages)

[PATCH 00/28] iio: zero init stack with { }
instead of memset()
 2025-06-14 14:18 UTC  (46+ messages)
` [PATCH 01/28] iio: accel: adxl372: use = "
` [PATCH 02/28] iio: accel: msa311: "
` [PATCH 03/28] iio: adc: dln2-adc: "
` [PATCH 04/28] iio: adc: mt6360-adc: "
` [PATCH 05/28] iio: adc: rockchip_saradc: "
` [PATCH 06/28] iio: adc: rtq6056: "
` [PATCH 07/28] iio: adc: stm32-adc: "
` [PATCH 08/28] iio: adc: ti-ads1015: "
` [PATCH 09/28] iio: adc: ti-ads1119: "
` [PATCH 10/28] iio: adc: ti-lmp92064: "
` [PATCH 11/28] iio: adc: ti-tsc2046: "
```

- The mainline kernel is the basis for all SoC vendor kernels, albeit forked to oblivion.
- The most sustainable solution to the EOL problem is to implement and upstream support for the device straight to the mainline Linux kernel and other userspace firmware like MESA. From there on, distributions like Ubuntu, paired with proper mobile interface (ex. Phosh), can be used.

# The Mainline Way!

- It takes a lot of time to get non-QCOM hardware in upstream due the to sheer amount of work required. But it’s worth it!

| | | | | |
|---|---|---|---|---|
| 2025-02-05 | arm64: dts: exynos8895-dreamlte: enable support for the touchscreen | Ivaylo Ivanov | 1 | -0/+40 |
| 2025-02-05 | arm64: dts: exynos8895-dreamlte: enable support for microSD storage | Ivaylo Ivanov | 1 | -0/+32 |
| 2025-02-05 | arm64: dts: exynos8895: add a node for mmc | Ivaylo Ivanov | 1 | -0/+16 |
| 2025-02-05 | arm64: dts: exynos8895: define all usi nodes | Ivaylo Ivanov | 1 | -0/+868 |
| 2025-02-05 | arm64: dts: exynos8895: add syscon nodes for peric0/1 and fsys0/1 | Ivaylo Ivanov | 1 | -0/+24 |
| 2025-02-05 | soc: samsung: usi: implement support for USIv1 and exynos8895 | Ivaylo Ivanov | 1 | -13/+58 |
| 2025-02-05 | soc: samsung: usi: add a routine for unconfiguring the ip | Ivaylo Ivanov | 1 | -0/+28 |
| 2025-02-05 | dt-bindings: soc: samsung: usi: add USIv1 and samsung,exynos8895-usi | Ivaylo Ivanov | 2 | -37/+79 |
| 2025-01-13 | dt-bindings: mmc: samsung,exynos-dw-mshc: add specific compatible for exynos8895 | Ivaylo Ivanov | 1 | -0/+1 |
| 2025-01-07 | i2c: exynos5: Add support for Exynos8895 SoC | Ivaylo Ivanov | 1 | -4/+31 |
| 2025-01-07 | dt-bindings: i2c: exynos5: Add samsung,exynos8895-hsi2c compatible | Ivaylo Ivanov | 1 | -0/+1 |
| 2025-01-05 | dt-bindings: soc: samsung: exynos-sysreg: add sysreg compatibles for exynos8895 | Ivaylo Ivanov | 1 | -0/+8 |
| 2024-12-30 | arm64: dts: exynos8895: Add camera hsi2c nodes | Ivaylo Ivanov | 1 | -0/+44 |
| 2024-12-14 | arm64: dts: exynos8895: Add a PMU node for the second cluster | Ivaylo Ivanov | 1 | -1/+11 |
| 2024-12-02 | arm64: dts: exynos8895: Add serial_0/1 nodes | Ivaylo Ivanov | 1 | -0/+26 |
| 2024-11-14 | dt-bindings: clock: actions,owl-cmu: convert to YAML | Ivaylo Ivanov | 3 | -53/+61 |
| 2024-11-13 | dt-bindings: timer: actions,owl-timer: convert to YAML | Ivaylo Ivanov | 3 | -22/+108 |
| 2024-11-04 | tty: serial: samsung: Add Exynos8895 compatible | Ivaylo Ivanov | 1 | -0/+13 |
| 2024-11-04 | dt-bindings: serial: samsung: Add samsung,exynos8895-uart compatible | Ivaylo Ivanov | 1 | -2/+12 |
| 2024-10-26 | arm64: dts: exynos8895: Add spi_0/1 nodes | Ivaylo Ivanov | 1 | -0/+30 |
| 2024-10-26 | arm64: dts: exynos8895: Add Multi Core Timer (MCT) node | Ivaylo Ivanov | 1 | -0/+20 |
| 2024-10-26 | arm64: dts: exynos8895: Add clock management unit nodes | Ivaylo Ivanov | 1 | -0/+85 |
| 2024-10-26 | dt-bindings: timer: exynos4210-mct: Add samsung,exynos8895-mct compatible | Ivaylo Ivanov | 1 | -0/+2 |
| 2024-10-26 | clk: samsung: Introduce Exynos8895 clock driver | Ivaylo Ivanov | 2 | -0/+2804 |
| 2024-10-26 | clk: samsung: clk-pll: Add support for pll_{1051x,1052x} | Ivaylo Ivanov | 2 | -0/+4 |
| 2024-10-26 | dt-bindings: clock: samsung: Add Exynos8895 SoC | Ivaylo Ivanov | 2 | -0/+692 |
| 2024-10-21 | spi: dt-bindings: samsung: Add a compatible for samsung,exynos8895-spi | Ivaylo Ivanov | 1 | -0/+4 |
| 2024-10-02 | soc: samsung: exynos-chipid: add exynos8895 SoC support | Ivaylo Ivanov | 1 | -0/+1 |
| 2024-10-02 | dt-bindings: hwinfo: samsung,exynos-chipid: add exynos8895 compatible | Ivaylo Ivanov | 1 | -0/+1 |
| 2024-10-02 | arm64: dts: exynos: Add initial support for Samsung Galaxy S8 | Ivaylo Ivanov | 2 | -0/+127 |
| 2024-10-02 | arm64: dts: exynos: Add initial support for exynos8895 SoC | Ivaylo Ivanov | 2 | -0/+1345 |
| 2024-10-02 | dt-bindings: soc: samsung: exynos-pmu: Add exynos8895 compatible | Ivaylo Ivanov | 1 | -0/+1 |
| 2024-10-02 | dt-bindings: arm: samsung: Document dreamlte board binding | Ivaylo Ivanov | 1 | -0/+6 |
| 2024-10-02 | pinctrl: samsung: Add exynos8895 SoC pinctrl configuration | Ivaylo Ivanov | 4 | -0/+150 |
| 2024-10-02 | dt-bindings: pinctrl: samsung: add exynos8895-wakeup-eint compatible | Ivaylo Ivanov | 1 | -0/+1 |
| 2024-10-02 | dt-bindings: pinctrl: samsung: Add compatible for Exynos8895 SoC | Ivaylo Ivanov | 1 | -0/+1 |
| 2024-10-02 | dt-bindings: arm: cpus: Add Samsung Mongoose M2 | Ivaylo Ivanov | 1 | -0/+1 |

# First steps towards becoming EOL-less

- Coming back to the topic, our target of interest is Volla’s flagship phone, the **Quintus**!
- It all started from us liking Volla’s openness and Linux and privacy orientation
- They were kind enough to donate us a phone in March that we could start developing on.

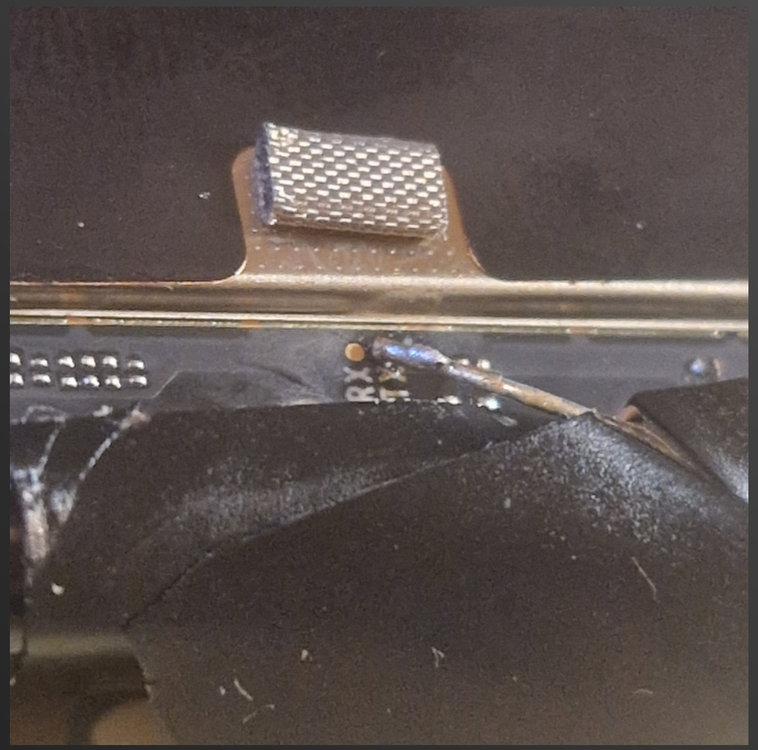
# First steps towards becoming EOL-less


Disassembled Volla Phone Quintus

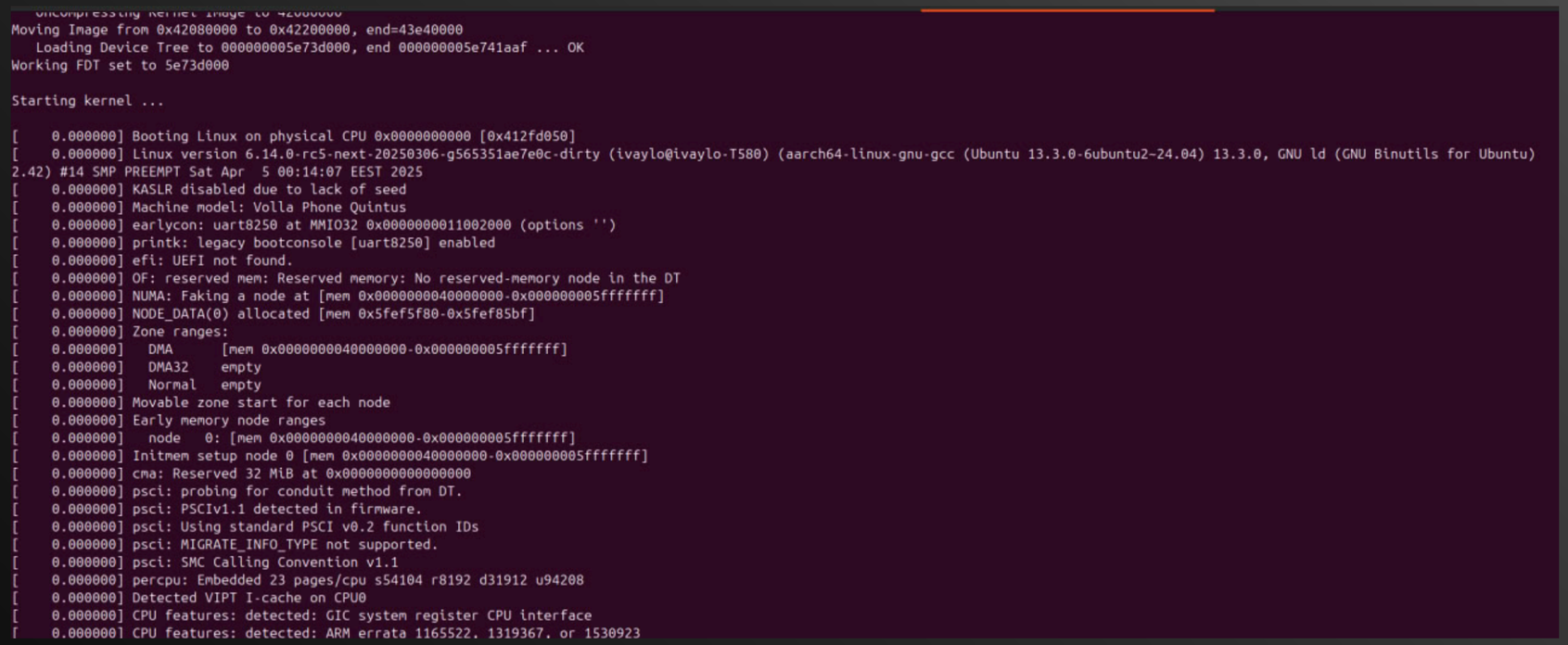
- So we began working on it!
- Getting basic mainline Linux with only 8 cores and some way of debugging was the priority
- Due to the ugliness of the stock MediaTek bootloader (LK), we had to add support for Quintus to some custom shim bootloader in order to avoid the forced DTB overlaying.

# First steps towards becoming EOL-less

- In April, support for Quintus to uniLoader was added [1]. We also have a fork of U-Boot working.
- The preferred way of debugging is using UART. The Quintus exposes small RX and TX pins on the left part of the motherboard.



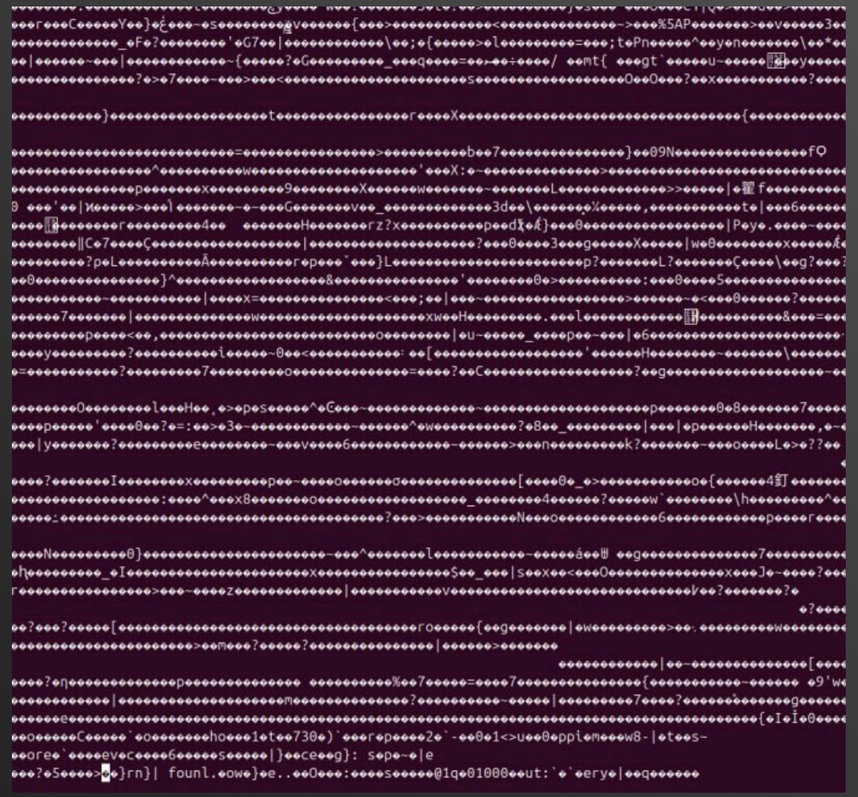
# First steps towards becoming EOL-less




After a few days of working on it, we had Linux v6.14-rc5 booting up to initramfs. More things are documented at ivoszbg's website [2]

# First steps towards becoming EOL-less

- Unfortunately, on the same night after getting mainline booting, something REALLY unfortunate happened. The TX pad of our unit lifted and we were no more able to get any logs and debug.
- We notified Volla and while waiting for a response, we started trying to figure out alternative ways to debug.



# First steps towards becoming EOL-less

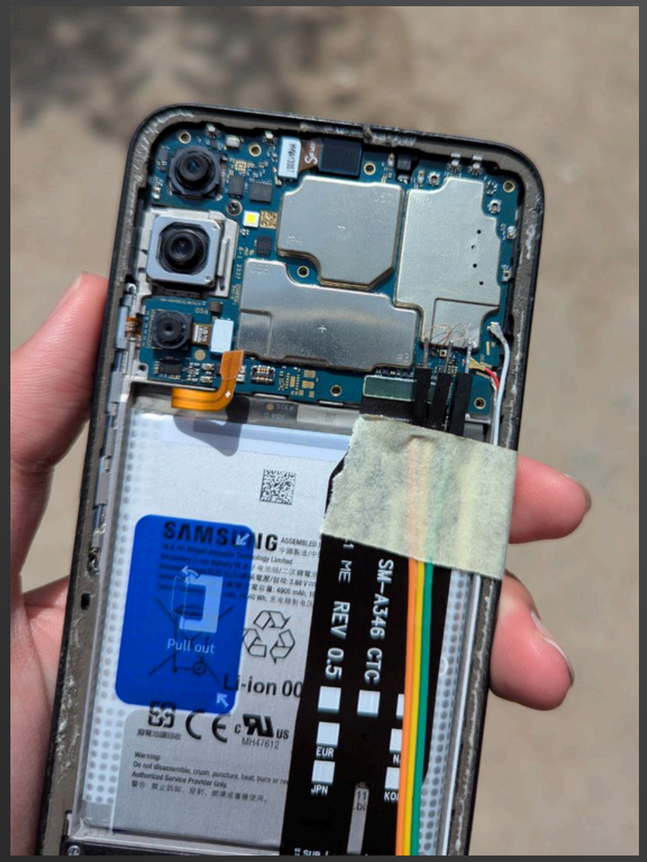
Framebuffer writes in LK bootloader

We had 2 ideas for an alternative:

- using pstore/ramoops
  - this turned out to not be possible as the DRAM gets wiped on every boot.
- framebuffer
  - this required fiddling with DSI after every pixel update, which I could not get working outside the LK bootloader quickly enough.

# First steps towards becoming EOL-less

- After struggling for over 3 months, we felt hopeless.
- We had gotten absolutely nowhere with the Volla Community Days coming up, so I decided to make a decision.
- I went out and bought a Galaxy A34, a device with the same SoC as the Quintus. We had no schematics for it initially, getting hands on them took a few days.
- On the 12th of June, I successfully got UART on the Galaxy A34.

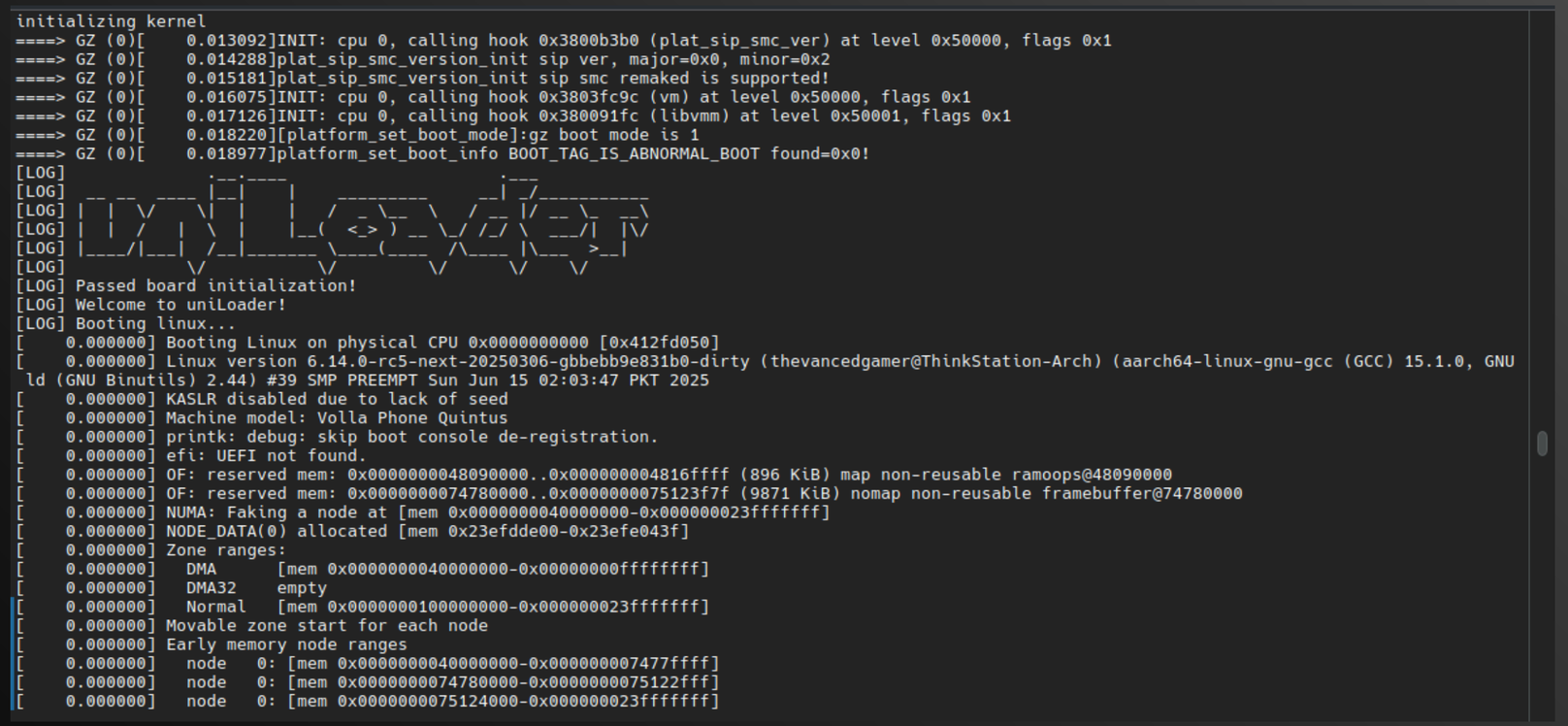



Galaxy A34 with UART on the board

# First steps towards becoming EOL-less

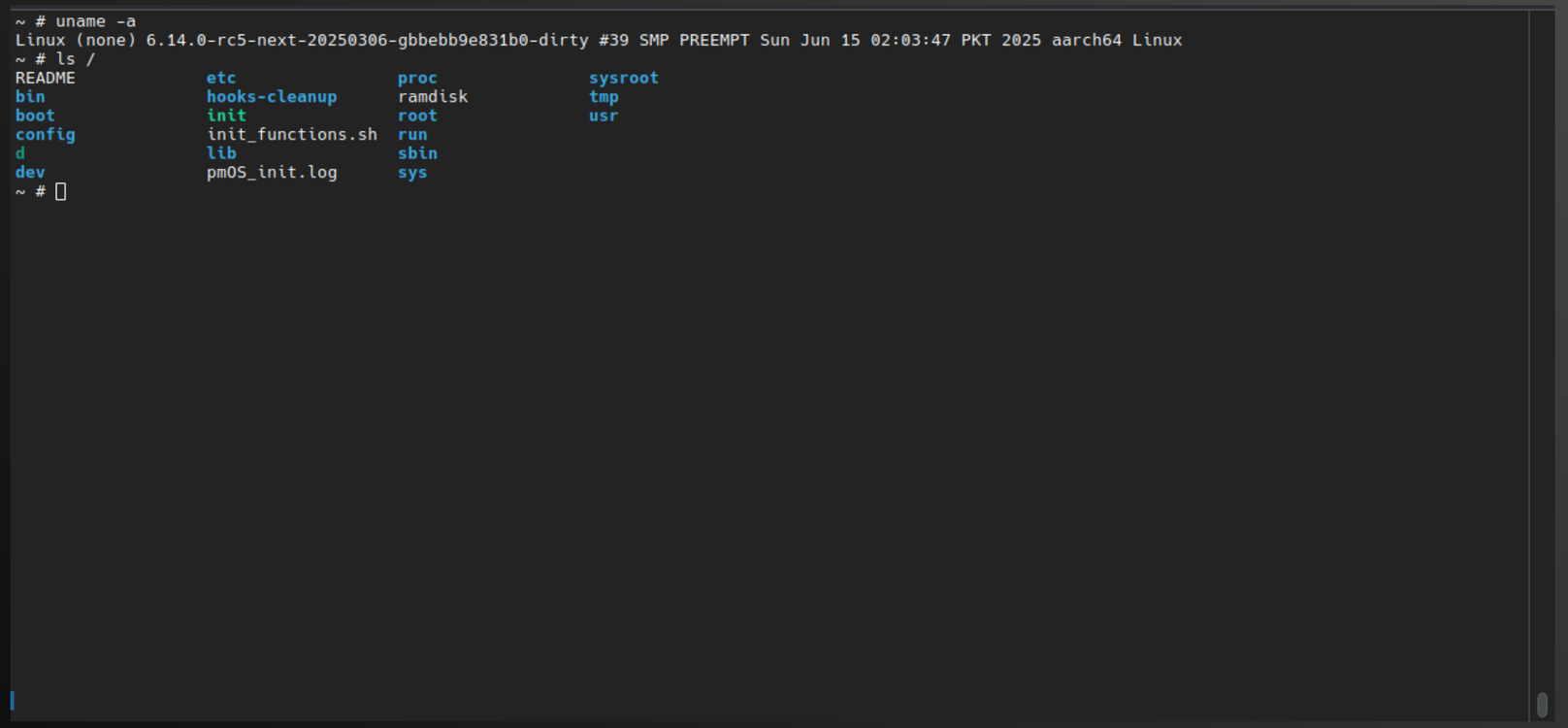
- Now that we had another device, it was all-or-nothing. I spent multiple nights getting all of the core peripherals up.
- After 3 brutal nights, I had gotten core subsystems up for interfacing with hardware like PMICs and IOMMUs.
- Now we had only one more goal, functional USB. Unfortunately due to the lack of time on our part, we only managed to get the USB controller and USB regulators working. But the USB port **electrocutes** me now!

# First steps towards becoming EOL-less




Mainline Linux kernel booting on the Galaxy A34

# First steps towards becoming EOL-less




PostmarketOS debug initramfs working on Galaxy A34

# First steps towards becoming EOL-less

```
~ # dmesg | grep mtu3
[    7.666584] mtu3 11201000.usb: supply vusb33 not found, using dummy regulator
[    7.668616] mtu3 11201000.usb: supply vbus not found, using dummy regulator
[    7.669368] mtu3 11201000.usb: dr_mode: 3, drd: auto
[    7.670051] mtu3 11201000.usb: u2p_dis_msk: 0, u3p_dis_msk: 0
[    7.674564] mtu3 11201000.usb: usb3-drd: 1
[    7.675641] mtu3 11201000.usb: irq 161
[    7.676266] mtu3 11201000.usb: IP version 0x1005(U3 IP)
[    7.677473] mtu3 11201000.usb: max_speed: super-speed-plus
[    7.678382] mtu3 11201000.usb: fifosz/epnum: Tx=0x2000/8, Rx=0x2000/8
[    7.679223] mtu3 11201000.usb: dma mask: 36 bits
[   12.765346] mtu3 11201000.usb: xHCI platform device register success...
[   12.766667] mtu3 probe done successfully!
[   12.766771] starting mtu3!
[   12.784507] mtu3 11201000.usb: gadget (super-speed) pullup D+
~ #
```

MTU3 (dual role USB) working

# First steps towards becoming EOL-less: Overall Experience

- Overall, MediaTek sounds like an easy platform to work on, but in practicality, that's definitely not the case. There are unrelated driver changes scattered around everywhere that makes rewriting the driver much more difficult, as all of the logic is spread apart with no easy way to piece it together.
- However, with enough effort and time, it is entirely possible, as shown in our PoC.

# What works and what needs to be done

Working:

- Core subsystems
- Clocks
- Pinctrl
- USB
- PWRAP
- Main PMIC (MT6359)

TBD:

- Rest of the PMICs
- UFS
- Display (dispsys subsystem) and GPU
- ICCs, cpufreq, etc
- All phone-related features

# Planned roadmap

## Q2 2025

Initial bring up of basic core peripherals in a fork of ours

## Q3/Q4 2025

Upstreamed basic support for MT6877 and Quintus, work on complex hardware blocks

## Q4 2025 and onwards

If all goes well, more work on usability as a phone

# Resources

[1]: https://github.com/ivoszbg/uniLoader/commit/5ee9fd9c1a71168abfac4704e8bb8458a9cfb0dct
[2]: https://ivoszbg.xyz/blog/quintus-hacking-1/

# Q/A