KDAB

EMBEDDED ENTWICKLUNG LEBENSZYKLUS

Was alles zur Embedded Programmierung gehört

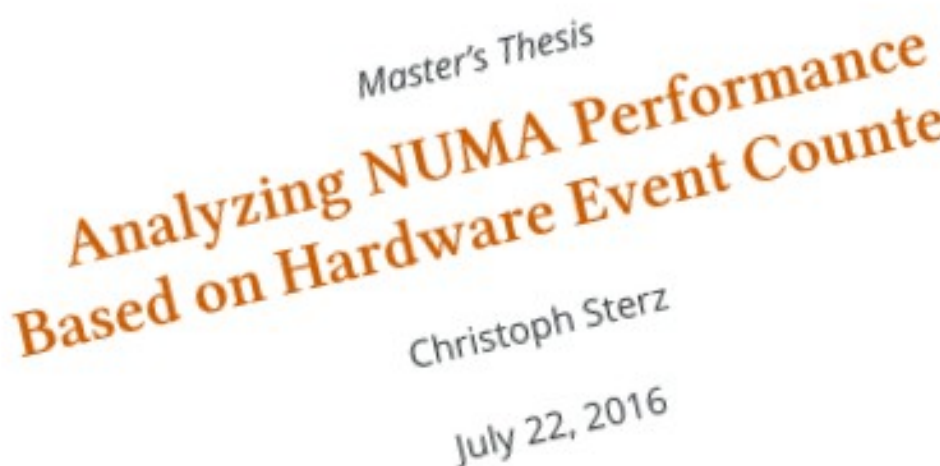
Christoph **Sterz**
christoph.sterz@kdab.com

Hi
Ich bin Christoph

About Me

- Bachelor & Master am **HPI** (zwischendrin **Cern**, Genf)
- Masterarbeit am OSM Lehrstuhl: ~ Linux perf auf NUMA
- Währenddessen schon bei **KDAB** gearbeitet
  - C++/Qt Firma in Berlin
  - >70% unserer Mitarbeiter sind OpenSource Devs
- Heute mach ich Linux perf(ormance) auf Embedded*

  * und mehr :)



Warum halte ich heute diesen Vortrag?

- Embedded Entwicklung & Consulting ist mein Beruf geworden.
  - Consulting = (sehr schnell sehr viel Code lesen || Gerätesoftware verstehen);
  - Consulting != Viel Reden
- Ich denke, dass ich einen guten Einblick habe:
  - **Infotainment:** 3 Auto SW-Platformen, Yachts für Scheichs, Flugzeuge
  - **Medizingeräte:** (Beatmungsgeräte, Mikroskope, Autoklaven, Zellzähler)
  - **Industrie:** (SCADA-PLCs, Finnische Sägewerke, Weltklasse-Ampere-Eichgeräte)
  - *Beispiel heute:* **Braumaschinen** von Hobby mit Stecker (10L/20L)
    bis zum Ratskeller (1000L, 60KW CAN Heizungen & Frequenzgeber)

KDAB

The Qt, OpenGL and C++ experts

Warum halte ich heute diesen Vortrag?

- Oft dreht es um Produkte / Produktlinien die **realweltliche Anforderungen** haben*****
- **Embedded Operating Systeme** und das Gesamte System werden darauf hin angepasst und bietet der eigentlichen Software notwendige Funktionen an
  - Klappt manchmal gut, oft muss man “anbauen”.

***** Wir reden heute viel darüber was die Industrie alles für sie wichtiges “erreichen will/muss” und was man dafür tun muss.

Disclaimer

- Das ist alles meine eigene kleine Weltsicht
- Kein Vollständigkeitsanspruch
- Es werden Produkt und Firmennamen fallen

  – Es existieren andere Produkte und Firmen die auch gut oder besser sind :)

Es gibt Bonuspunkte für gute Zwischenfragen!!

KDAB

Produktlebenszyklus
Entwicklungs-Phase 0: Bevor es losgeht

Zuallererst: Zertifizierungen

- Bevor alles beginnt: Produkte müssen legal sein
- Zertifizierungen und ISOs bestimmen die Industrie in ihren Planungen

  Es können gute und schlechte System-Architekturen entworfen werden aber an der Zulassung kommt niemand vorbei.

- ISO 26262 “FuSi” (Sicherheitslevel ASIL A-D)
- ISO 13458 *Medical Devices – Quality management systems*
- Für Hardware: EMV Prüfung, Flugzeug HW ~ 10G Beschleunigungstests

Hardwareplanung first, Software second

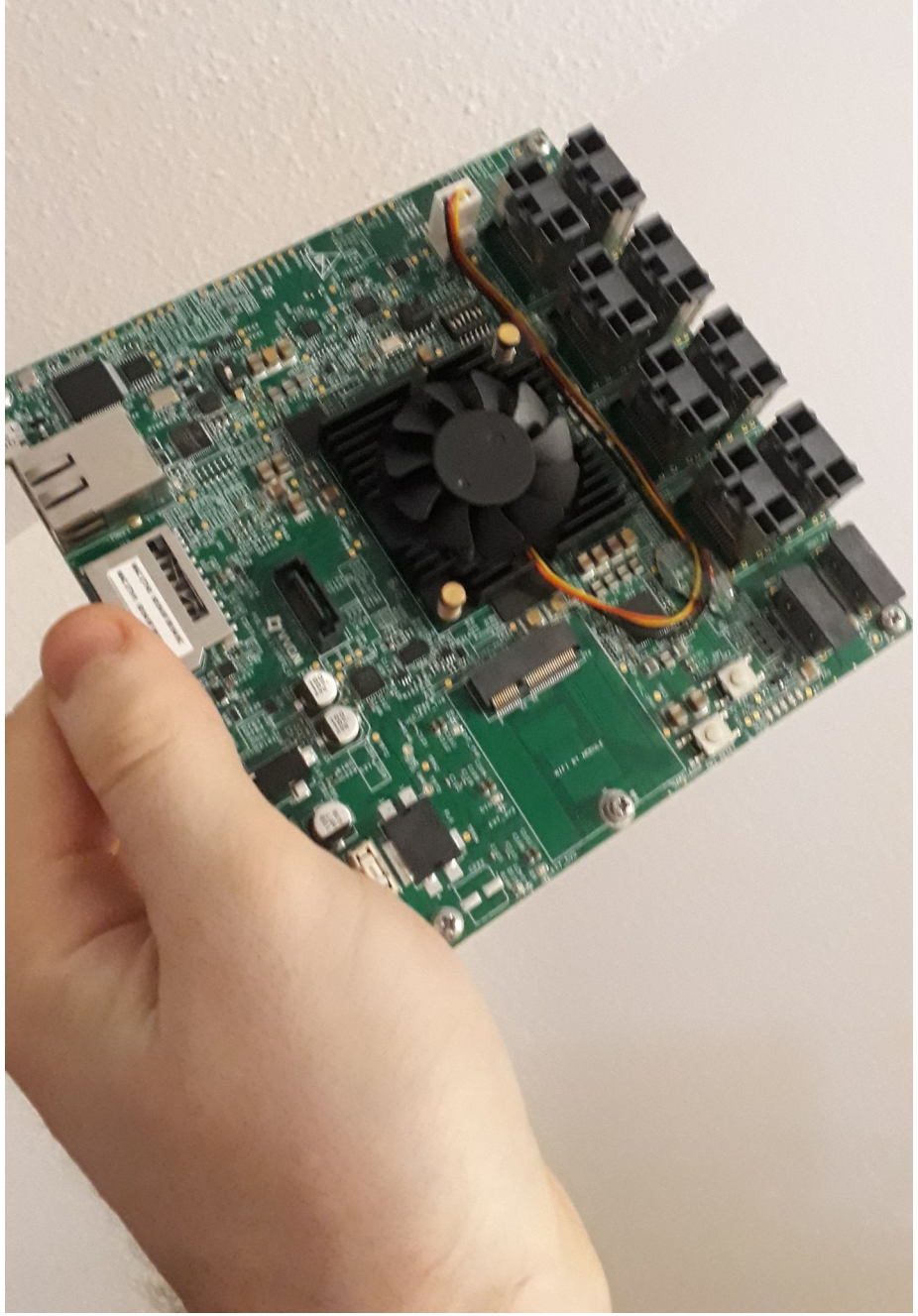
- Hardware ist sehr oft preisbestimmend
- Software folgt dem, was vorhanden ist
- Automotive HeadUnit+InstrumentCluster: 4-8 Cores ARM + GPU
  - Höchste Stückzahlen
- Industriegerät: 1–2 Cores ARM, GPU optional
  - Hohe Stückzahlen
  - Workhorse der Industrie NXP iMX.6/iMX8 (“Raspberry Pi der Industrie”)
- Medical Gerät: Ab einem gewissen Preislevel COTS x86 (Intel/AMD)
  - Oft niedrige Stückzahlen

Realistische Zahlen für Projekte

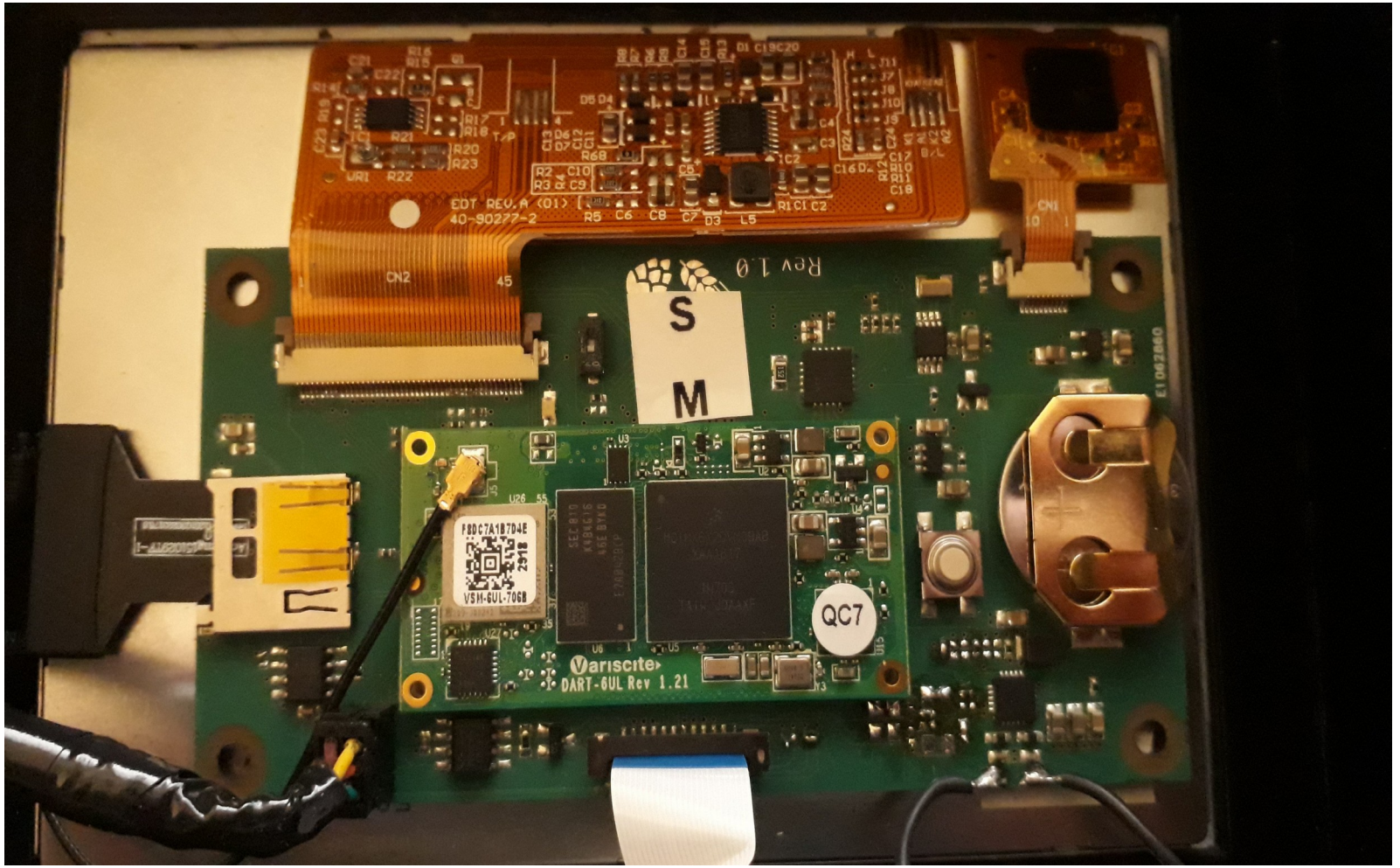
- Hausgeräte: bis 30Mio.
- Autos: bis 40 Mio.
- Industrie-Appliances: 20.000
- Industrie Kleinserien: 3.000
- Medizingeräte: 500, 5.000, Patientenbezogen mehr möglich

1GB mehr Ram kosten ~10$ – Auswirkungen auf die Beispiele

Eval Board



SOM – System on a Module



Kundenanforderungen

- Always On
- Personalization
- Full Internationalization (I18n)
  - Nicht nur live Übersetzungen, Left-To-Right, Right-ToLeft Oriented Design
- Updates
  - Over The Air
  - Targeted Updates / Canary Updates
- Reliability
  - A/B Images, Fallbacks, Werkseinstellungen…
- Smartphone ist der Standard gegen den verglichen wird.

KDAB

Produktlebenszyklus Entwicklungs-Phase1: Zu wenig Prototypen

Eine Tatsache. Viele Auswirkungen…

- In meiner Erfahrung gibt es zuerst keine dann wenig Hardware.
  - Automotive-Entwicklungszyklen können 15 Jahre dauern
  - Mein letztes Auto-Projekt war mit 5 Jahren extrem schnell
  - Beta-Chips für Entwicklung, damit aktuell im SOP (Start of Production)
- Performance: Desktop/Simulatoren führen zu Täuschung
- Verfügbarkeit: 2-Klassen Entwickler mit und ohne Zugang zu Hardware
- Wenig Testing / Wenig Autotesting
- Viel Kupferlackdraht auf den wenigen Prototypen

Opa erzählt vom Krieg (I)

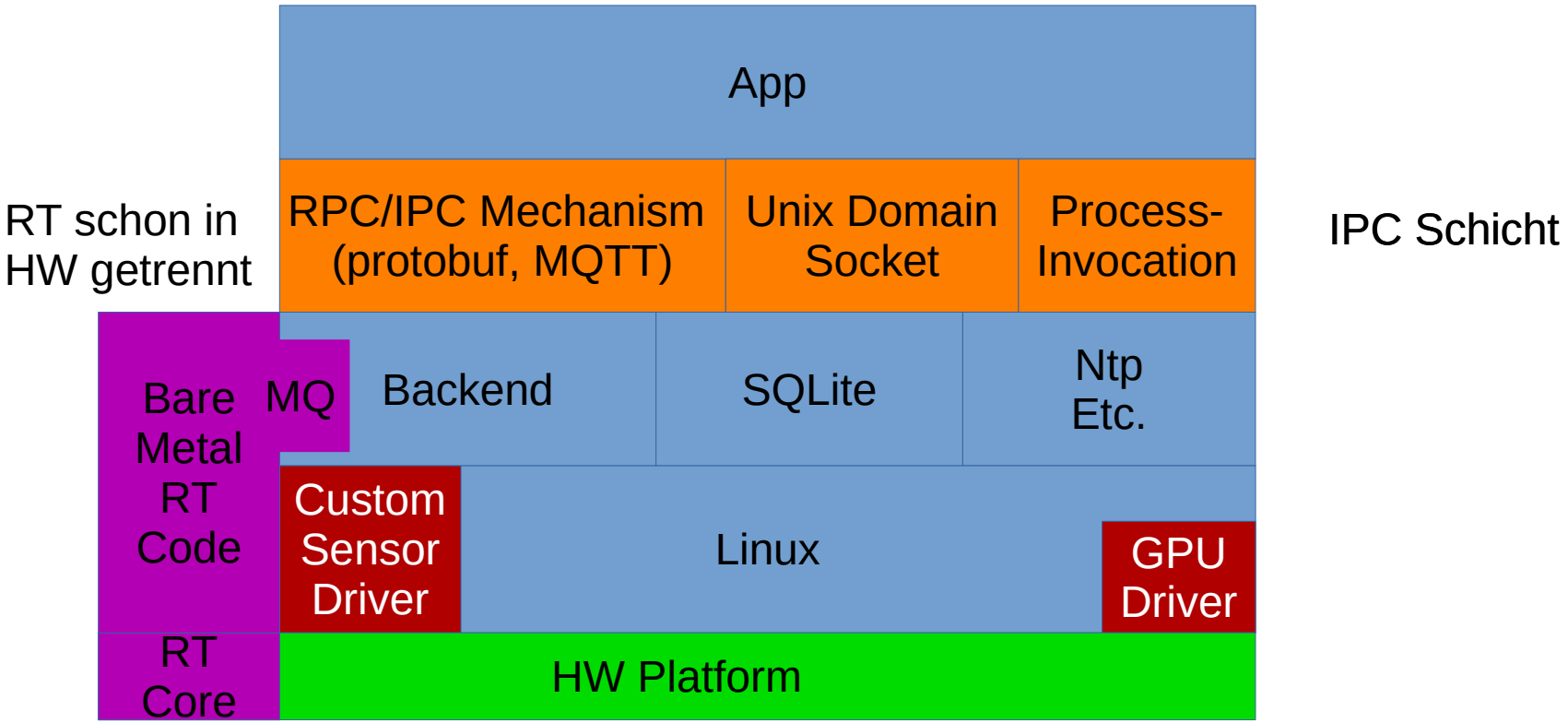
- Autoprojekt, >10Mio Autos, A-Silicon (=CPU-Beta)
- Random Crashes in App immer in Function Calls.
- Nachgeschaut: Crashes auch in allen möglichen anderen Systemdiensten
- 2 Monate Debugging
  - Ist es libc?
  - Ist es der Crashhandler, der falsches reportet?
  - Wem kann ich noch trauen?
  - Soll ich vielleicht irgendetwas mit Holz machen, statt mit Software?

- Dann: Crashdumps: Immer nach einer gewissen Instruction (STP)
  - Hardware Bug, Thermal Issue in ARM ISA StorePaired, µ-code Patch nötig

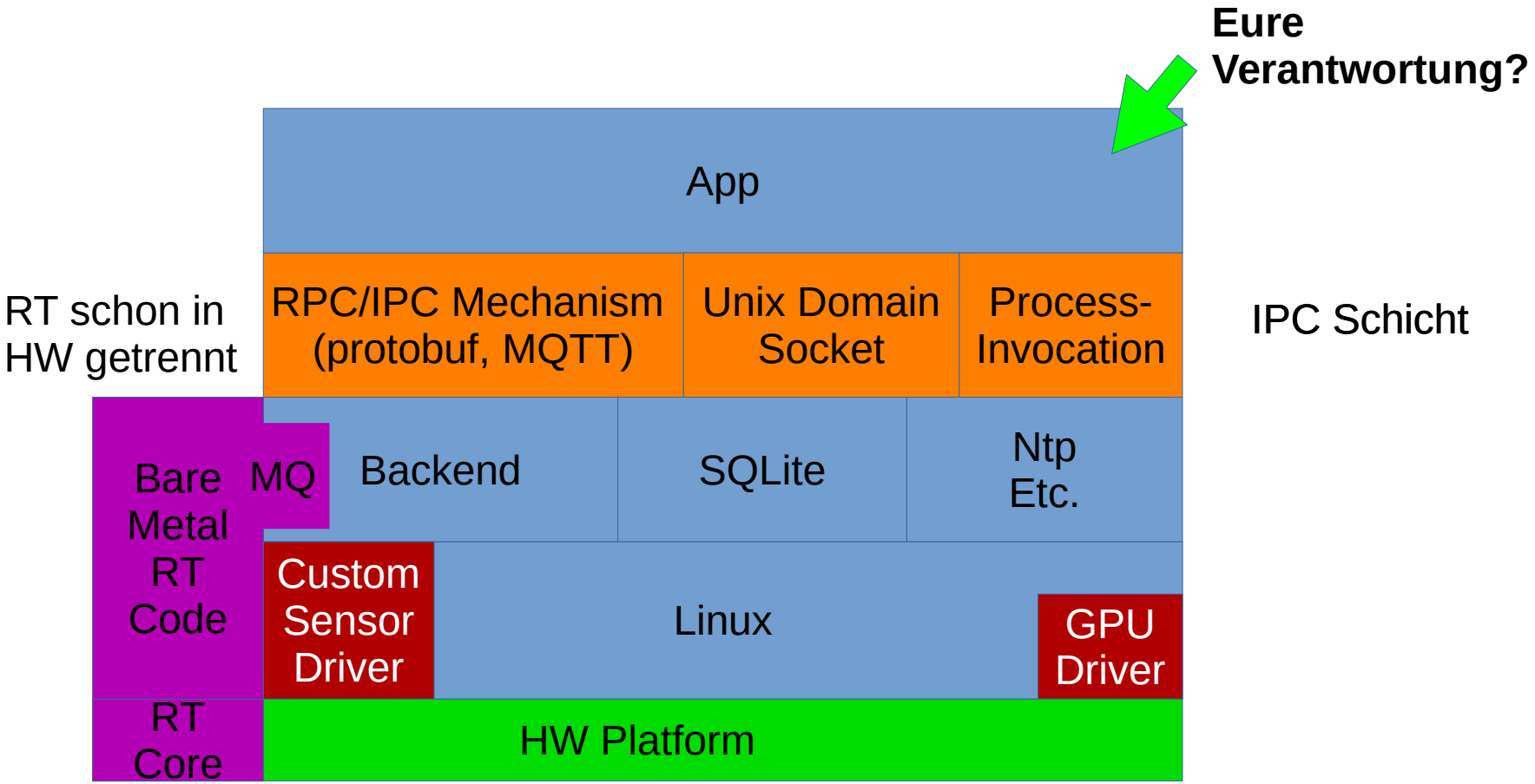
Software-Stack Typische Architektur

- **Linux+Android**: 70%, Bare-Metal: 10%, Windows: 10%, QNX:5%, VXWorks/Integrity: 3%, iOS...
  - Windows stirbt aus, EndOfLife für WinCE lang vorbei, kein RT mehr
- Linux: **Yocto**/Buildroot: 70%, Distro (oft Debian-*): 30%
- Linux: **Systemd**: 80%, SystemV: 20%
- Linux: **DBus** Support: >50%
- Code: Treiber: C, Business Logic **C++**, Python, C#
  - Bisher wenig Java außerhalb von Android gesehen
- GUI: **Qt**: >60% ← Observation Bias!
- IOT everywhere: Hersteller-API
- (schlimm) Altes Webkit mit HTML Usermanual

Architekturen “Der Klassiker”

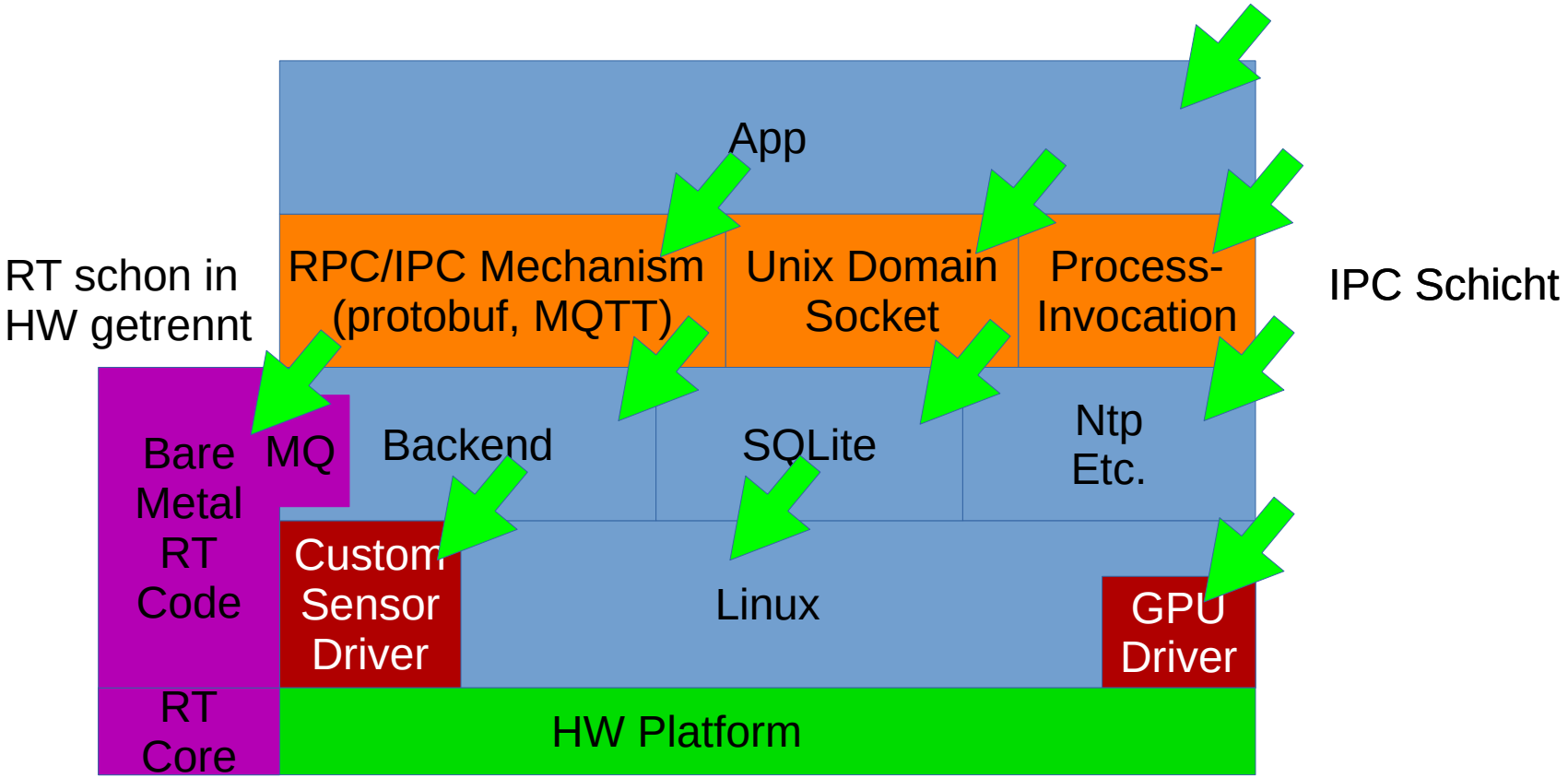


Architekturen “Der Klassiker”



Von der Hardware bis zum eigenen Code

Architekturen “Der Klassiker”



Opa erzählt vom Krieg (II)

- Prototypen laufen super
- Vorserienproduktion: 150 Devices, viel hellere Displays werden verbaut
- **Anruf**: alle Displays bleiben Schwarz
- **Kunde->Panik**

- SW Fehler: Linux GPIO Driver setzt **GPIO-direction** auf **”out”**
  - **!?** intial auf **LOW**
- Nächste Code-Zeile setzt sie auf **HIGH**
- Neue Displays hatten anderes timing, 100ns haben den Unterschied gemacht
- Lösung: SW-Patch zur korrekten *Direction* **“high”** (es gibt: in, out, low, high)

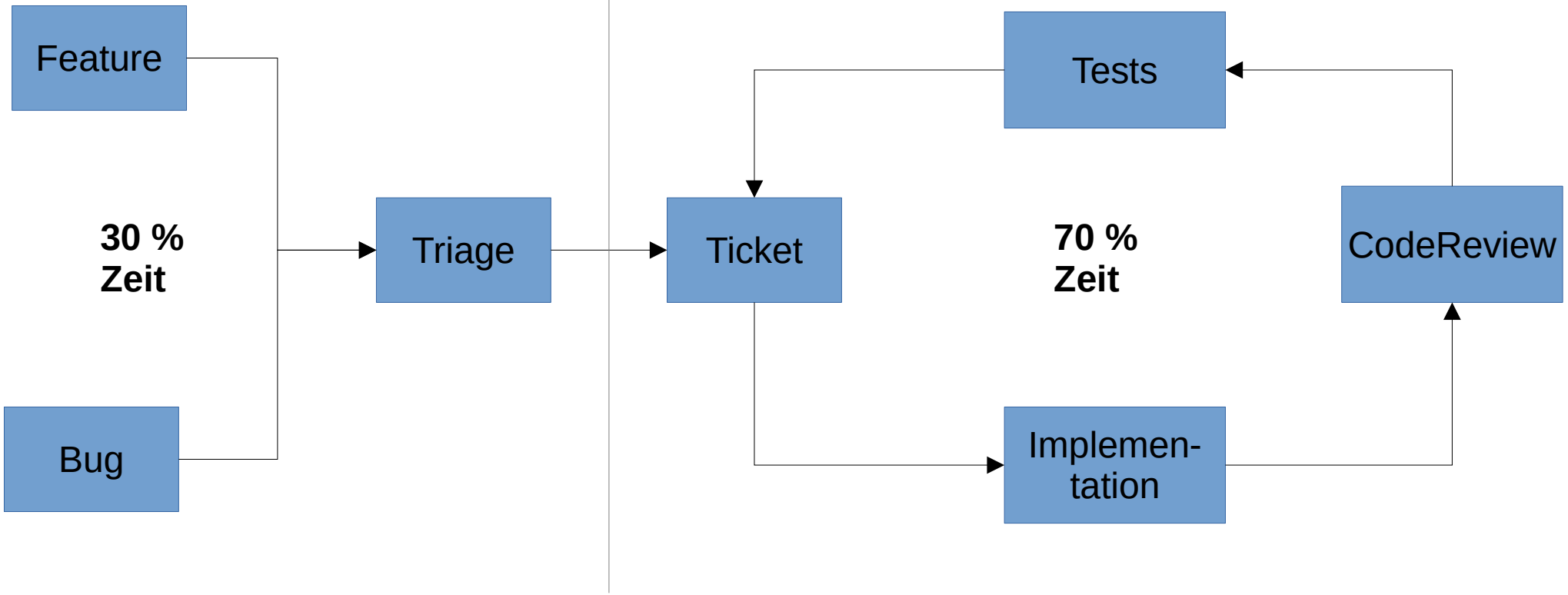
KDAB

Produktlebenszyklus Entwicklungs-Phase2: Kunde sieht Produkt zum 1. Mal

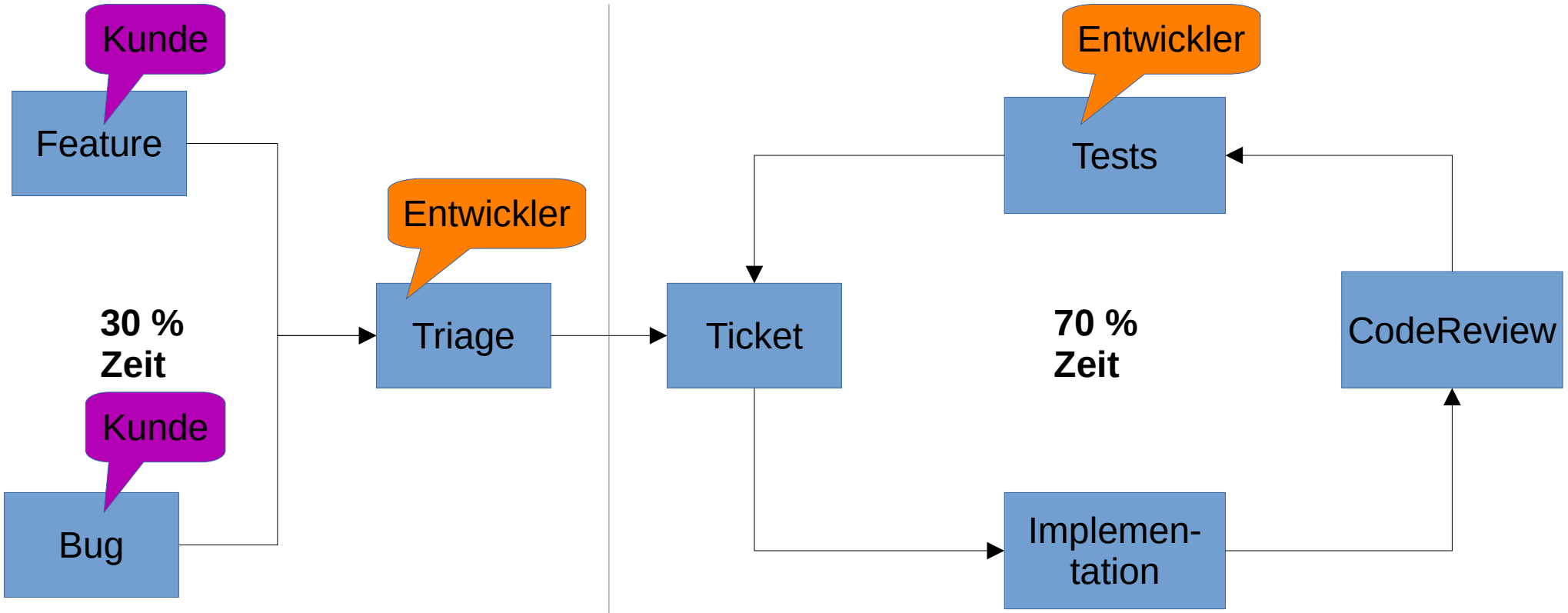
Alles ändert sich

- Ungeachtet des Entwicklungsmodells (selbst bei Wasserfall) kommt jetzt ein harter Bruch
  - Kunde testet und sieht Wünsche erfüllt / nicht erfüllt.
  - Designänderungen, UX Änderungen, Business Logic Änderungen
  - Definitionen werden klar, weil konkret
  - Kunde wird vor harte Entscheidungen gestellt, die biser vage blieben (out of Spec)
- UI/UX: Die Expectations sind immer Smartphone-Grade
  - Selbst in der Industrie
  - **Frage:** Wieviele Entwickler haben bei Nokia an der virtuellen Tastatur gearbeitet?
  - **Frage:** Wieviele Entwickler arbeiten an der virtuellen Tastatur für euren Kunden?

Entwicklungsalltag



Entwicklungsalltag



Entwicklungsalltag OS: Yocto Project

- Distro-Erstellungs Baukasten
- Basierend auf HW-Board-Support-Packages
- Sich überlagernde Schichten von Rezept-Layern
  - Basisrezept
  - Erweiterung
  - Eigene Erweiterung
- Kompilliert erst den Cross-Compiler,
  dann Linux Images,
  dann SDK und Updates.

```
# for coredumps and better debugging
#INHIBIT_PACKAGE_STRIP = "1"
#INHIBIT_PACKAGE_DEBUG_SPLIT= "1"


DEPENDS += " \
        qtbase \
        qtserialbus \
        qtdeclarative \
        qtmultimedia \
        qtconnectivity \
"

RDEPENDS_app += " \
        iw \
        wpa-supplicant \
        i2c-tools \
        mosquitto \
        alsa-utils \
        alsa-state \
        qtmultimedia \
        qtmultimedia-plugins \
        iproute2 \
        can-utils \
        tzdata \
        fonts \
        canopensocket \
"

FILES_${PN} += " \
  /opt/app/bin/app \
  /opt/backend/bin/backend \
  /opt/wifidaemon/bin/wifidaemon \
  /opt/mosquitto/ \
  /opt/mosquitto/access.acl \
  /opt/mosquitto/passwords \
  /opt/wifidaemon/wpa_backup.conf \
  /opt/app/sounds/click.wav \
  /opt/app/sounds/notify.wav \
  /opt/app/sounds/notify_boosted10db.wav \
  /opt/WebBridge/bin/WebBridge \
  /opt/testrig/bin/testrig \
```

KDAB

The Qt, OpenGL and C++ experts

KDAB

Produktlebenszyklus Entwicklungs-Phase3: Start of Production SOP

SOP naht – Die Stunde der Performanceprobleme

- Jetzt kulminieren die Features
- Große, separat entwickelte Teile kommen auf der Hardware zusammen
- Es gibt endlich genug Prototypen, es wird auf echter Hardware getestet
- Zeit wird knapp
- Auch Stunde der Quick & Dirty Hacks

Erfahrung: 5 Jahre Performance Consulting

- Premature Optimization is the Root of all Evil

Bevor man versucht besonders smart zu optimieren, sollte man aufhören an anderer Stelle Performance zu verschwenden

- Die Probleme sind oft an unerwarteten Stellen
- Hardware verhält sich auf dem Gerät anders
- “Death by 1000 Cuts”
- Computer sind schnell!

Die Klassiker

- “Warum ist da ein sleep?”
- Helfen, Datenmengen einschätzen zu lernen
  - Hochoptimierte Suchfunktion für 200 Listeneinträge à 1KB
  - Aber falsches ImageFormat im 12MB Background-Bild
  - Pfad der Daten(Mengen) folgen, Kopien beseitigen
- Jeden hochfrequenten Timer In Frage Stellen
- Heap-Allozierte Objekte in einer tight Loop
  - Falls nicht tcmalloc (etc.) verwendet wird: Der Heap ist Singleton
  - Global Heap Lock.

Das Werkzeug #1: Linux Perf & Flamegraphs

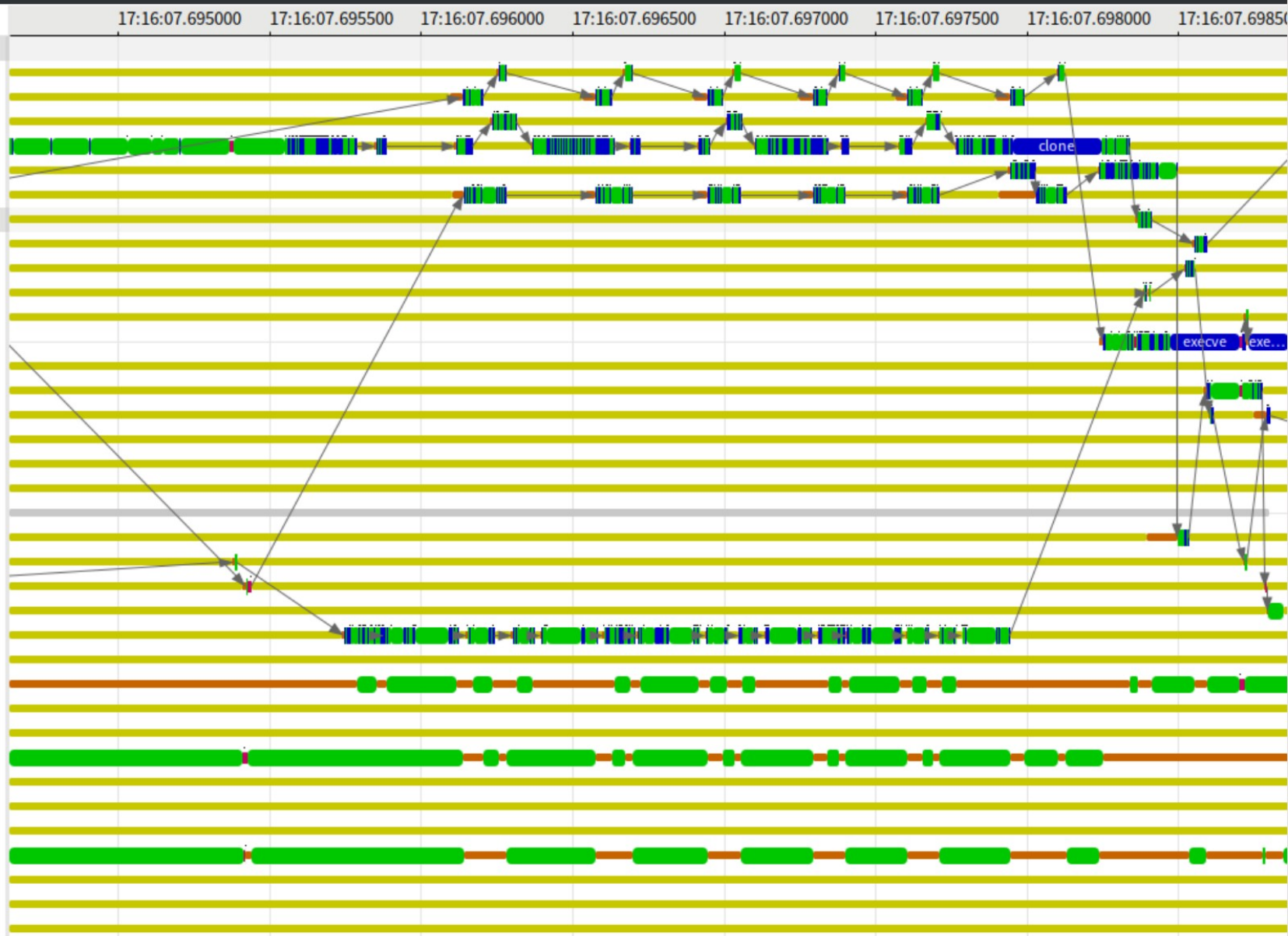
- Erwähnte ich Linux perf :) ?
  - Nutzt PerformanceMeasurementUnit der CPU (PMU) , low overhead
  - Sampling Profiler
  - Stack Unwinding auf dem Desktoprechner möglich

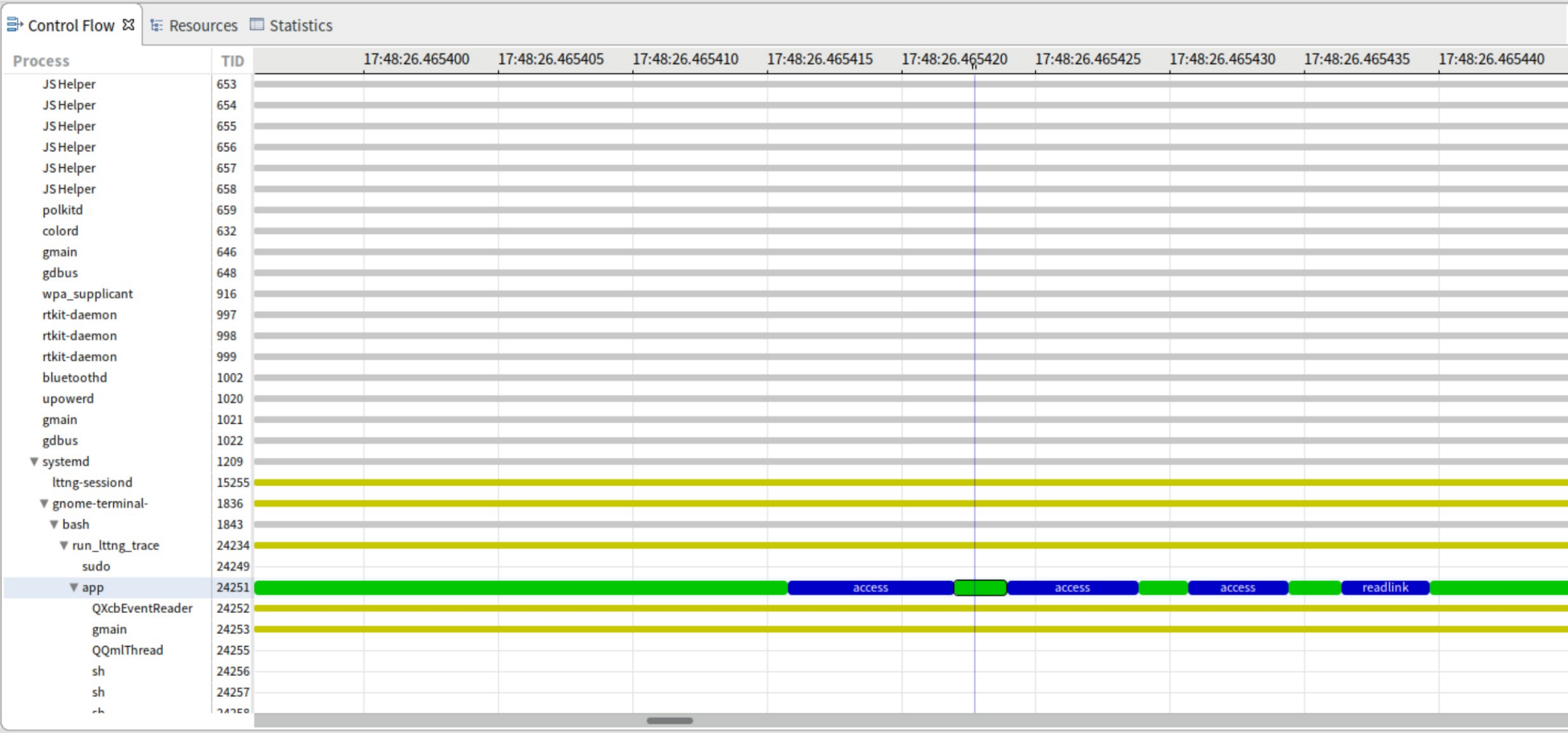
- **Beispiel** Hotspot (Free and Open Source perf viewer)
- Dann mit Entwicklern über die Stellen in der großen CodeBase Reden
  - “Ihr macht in eurer Renderloop DB-Anfragen, soll das so?”
  - “Jedes Logging Hat einen Timestamp, Jeder Timestamp fragt beim OS die Zeitzonen-Tabelle an, soll das so?”
  - “Ah! Nein, das war so nicht geplant.” Momente abwarten

Sonstige Alltägliche Werkzeuge

- Strace | heaptrack | die compiler sanitizer
- Linux Tracing Tools next Generation (LTTNG) für Systemweites tracing
  - Bei den anderen OS Herstellern gibts oft Herstellertools, die sind oft schlechter

Process	TID	PTID	Birth time
kernel(11)			
polkitd	624	1	17:16:06.625985267
Xorg	1224	1222	17:16:06.619666556
systemd	1		17:16:06.619703771
sudo	15297	15197	17:16:07.683345211
systemd-journal	225	1	17:16:06.625961338
gdbus	650	1	17:16:06.625986996
Softwar~cThread	1936	1272	17:16:06.614836870
Chrome_~dThread	1982	1886	17:16:06.614942060
Gecko_IOThread	1904	1272	17:16:06.614913257
BHMgr Monitor	1930	1272	17:16:06.614857889
rcu_preempt	10	2	17:16:06.612046349
lttng	15298	15297	17:16:07.698240556
kworker/3:1	60	2	17:16:06.619720160
Web Content	1980	1886	17:16:06.614983926
BHMgr Monitor	1994	1886	17:16:06.615000868
run_lttng_trace	15197	1843	17:16:06.626413446
ksoftirqd/2	29	2	17:16:06.608722166
rcuc/2	28	2	17:16:06.608716016
sleep	15296	15197	17:16:06.681389769
IPDL Background	1946	1272	17:16:06.614881136
rcu_sched	11	2	17:16:06.625916109
ksoftirqd/0	9	2	17:16:06.625914899
rcuc/0	13	2	17:16:06.625917311
dbus-daemon	482	1	17:16:06.625973936
lttng-consumerd	15274	15204	17:16:06.608443916
swapper/3	0		17:16:06.608443916
lttng-sessiond	15255	1209	17:16:06.608711373
rcuc/1	21	2	17:16:06.608711373
swapper/2	0		17:16:06.608716016
kworker/1:2	170	2	17:16:06.610839478
kworker/2:0	13576	2	17:16:06.610864409
kworker/0:1	43	2	17:16:06.610864411
swapper/0	0		17:16:06.610864411
ksoftirqd/3	36	2	17:16:06.612116581
rcuc/3	35	2	17:16:06.612123518
BHMgr Monitor	2100	1886	17:16:06.612757854



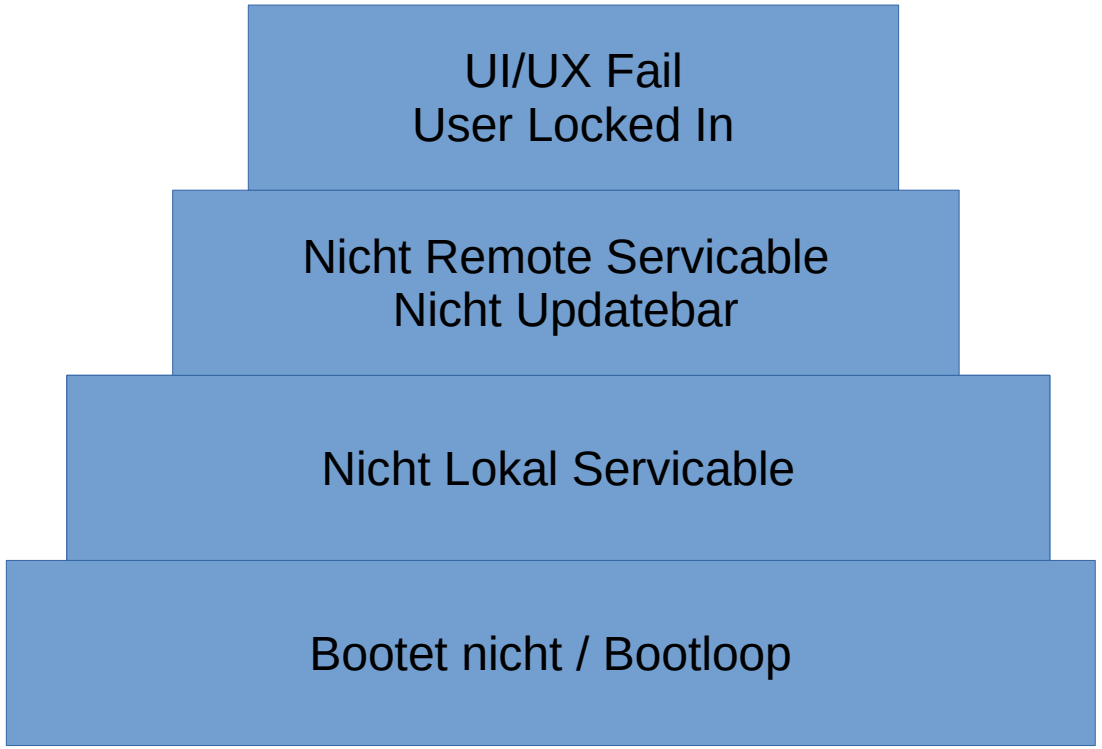

Control Flow
Resources
Statistics
Process
TID
17:48:26.465400
17:48:26.465405
17:48:26.465410
17:48:26.465415
17:48:26.465420
17:48:26.465425
17:48:26.465430
17:48:26.465435
17:48:26.465440
JS Helper 653
JS Helper 654
JS Helper 655
JS Helper 656
JS Helper 657
JS Helper 658
polkitd 659
colord 632
gmain 646
gdbus 648
wpa_supplicant 916
rtkit-daemon 997
rtkit-daemon 998
rtkit-daemon 999
bluetoothd 1002
upowerd 1020
gmain 1021
gdbus 1022
systemd 1209
lttng-sessiond 15255
gnome-terminal- 1836
bash 1843
run_lttng_trace 24234
sudo 24249
app 24251
QXcbEventReader 24252
gmain 24253
QQmlThread 24255
sh 24256
sh 24257
access
access
access
readlink

KDAB

Produktlebenszyklus Entwicklungs-Phase4: Erste Hardware Beim Endkunden

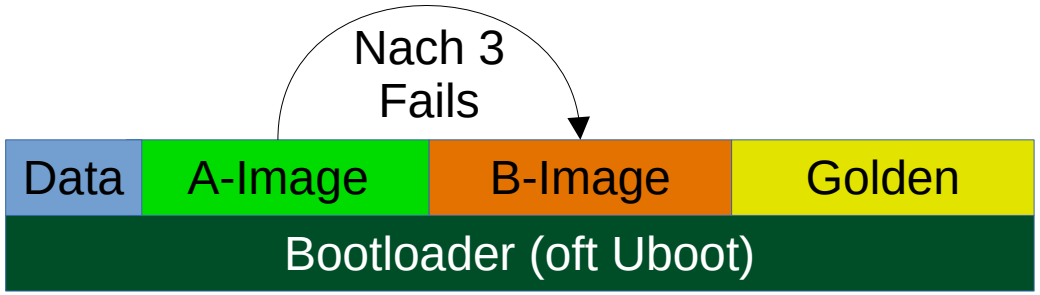
Ängste

Gerät wird zum “Brick”



Strategien gegen Bricks

- Zur Entwicklungszeit: Immer eine UART einfordern!
- A / B Image
- “Golden” Image



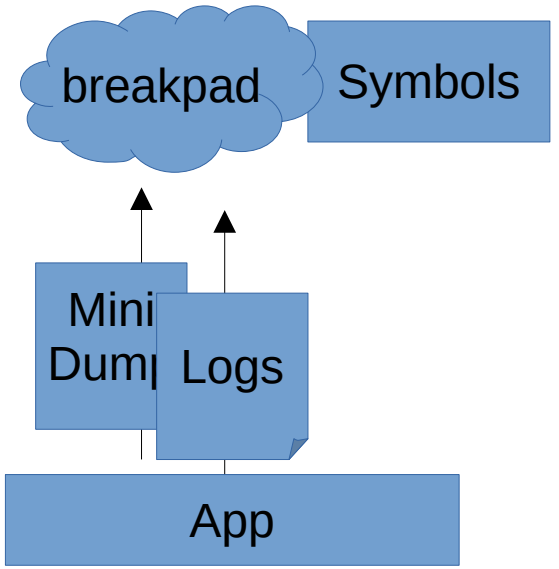
- Wifi /Connection Access Manager als separater Prozess!
- Werkseinstellungen
- Website-Flasher a la FritzBox

Updates

- Canary Updates:
  - Zielgerichtet auf TestNutzer / Normale Nutzer
  - Spezialversionen für einzelKunden
- 2 Grundlegende Ansätze:
  - SW-Update – Gesamtes Image wird geflashed
  - OSTree – Diff basiert “git fürs Filesystem”
- Datengröße: Anwendungsfälle sind verschieden
  - Meine Images: 50MB für Alles ← das ist viel!
  - Falls GSM als Kanal verwendet wird: <1MB

Crashes im Feld

- Endkunden haben kein besonderes Talent darin, Fehlerfälle gut zu beschreiben…
- Lösung: Schmale Crashdumps an einen Telemetrieserver schicken, der dann passende Debug-Symbole vorliegen hat



KDAB

Verschiedenes

Security: Das ‘S’ in IOT steht für Sicherheit

- Alle Updates immer Signieren!
- Alle Updates immer Verschlüsseln!
- Alle Protokolle ohne Crypto ablehnen!
- Selbst wenn User an die Serial-Konsole/UART kommen
  - Passwort (ab dann haben sie es meiner Meinung nach verdient)
- “Burner” SSH Schlüsselpaare auf den Geräten vorhalten
  - Die mit der Zeit abschalten wenn geleakt
  - Beobachtete Opsec: 10 Schlüssel ausdrucken und in den Firmen-Safe legen, Dem Notar geben, Dateien danach löschen

Datum und Zeit

- Großes Problem ab dem Punkt wo man mit externen Schnittstellen mit anderer Zeitvorstellung reden muss
  - RTC, Zeitzonen…
- Typisches Problem: Zertifikate
  - Let’s Encrypt Certs sind entweder nicht mehr,

    oder oft *noch nicht* gültig.

- Daumenregel: Sobald man “nach außen” kommmuniziert braucht man vernünftige Zeit!

Lizenzen

- Die Menge von bekannter und unbekannter Software erreicht Level in der die Entwickler über Lizenzen keine Aussagen mehr über installierte Software treffen können
  - Es gibt dedizierte Tools die im eigenen Code nach Lizenzen suchen
- Softwarelizenzen bringen Pflichten mit sich
  (zum Beispiel Lizenztext anzeigen)
- Wichtig ist: eine Bill Of Material (BOM) führen

- Für Interessierte: sucht bei modernen Autos die Lizenzliste

  → Ihr werdet sie finden :)

KDAB

End Of Life

End of Life

- Es gibt Gründe, alle Geräte einzusammeln
- Auch bei Industrieprodukten laufen Zeiten aus
- Es gibt explizit auf lange Laufzeiten ausgerichtete Hardware /Distributionen, Konferenzen nur darüber (30 Jahre+)
- Der Rest kann ein Sicherheitsrisiko werden
- Killswitch-Update

KDAB

End Of Vortrag!

Ich beantworte alle Fragen, AMA!