arendi

Android SoC auf custom Hardware
Ein Erfahrungsbericht

Simon Egli

r&d

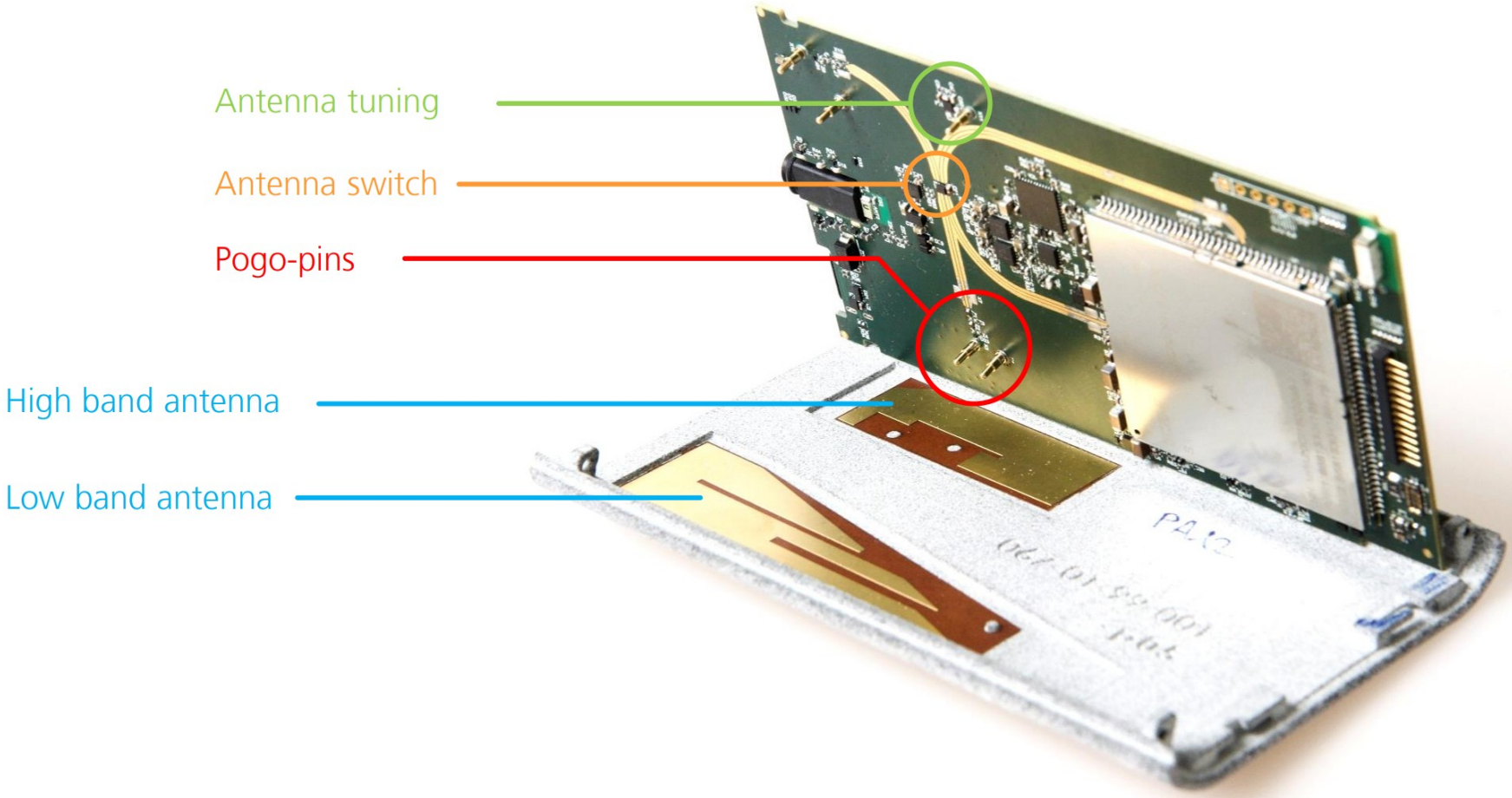
arendi

Gerätebeschreibung

- Android 8.1 (kein neueres Android verfügbar)
- Custom Hardware und Antenne
- Kein Display, eingabe über Hardwaretasten
- Ausgabe über TTS-Engine
- Co-Prozessor für Low-Level Hardware Interaktion
  - Serielle Schnittstelle zur Kommunikation
  - Antennenansteuerung
- 10h Betriebszeit mit aktivem Anruf

r&d

Gerätebeschreibung



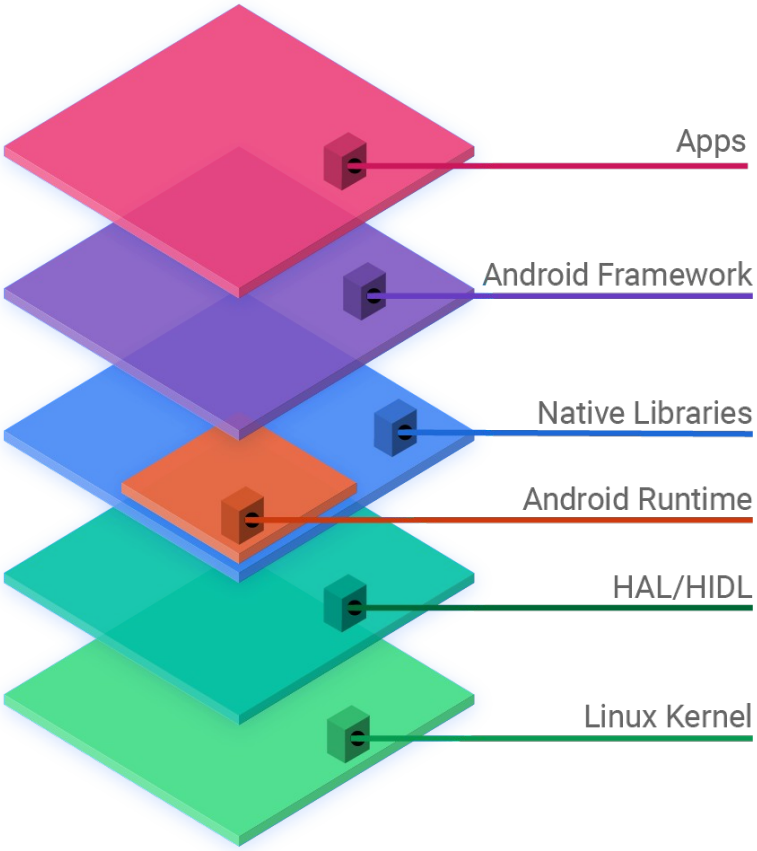
arendi

Leitfaden durch den Android-Dschungel

- System ist nur ein stark modifiziertes Linux
- Hat immer noch einen Kernel
- Hat immer noch Bootloader
- Hat verschiedene Bootstages
- Hat App Berechtigungsschutz via SELinux
- Sourcecode vorhanden

r&d

Leitfaden durch den Android-Dschungel



https://source.android.com

arendi

Wieso Android?

- Komplettes Betriebssystem für verschiedene Hardware
- Telefon-/Smartphone-Funktionen
- Vertraute Basis für Programmierer
  - Libraries
  - Funktionen
  - Vorhandenes- und Internet-Wissen
- Audio-Features
- Einfache Integration von Telefon-Zubehör

r&d

Android Build Umgebung

- Meist “Repo” basiert
  - Mehrere GB an Daten
  - Anleitungen des Herstellers beachten
- Mehrere Git-Repositories
- Linux-basierte Build-Umgebung
  - source /android-data/build/envsetup.sh
    lunch myProduct-user
    m -j8

arendi

Android und Continuous Integration/Delivery

- Docker Container für einheitliches Environment
- Jenkins als Buildserver-Umgebung
- Dedizierter Build-Node mit über USB angeschlossenem Prototyp
- Android Unit-/UI-Tests (Junit, Espresso)
- NSIS für Windows Utilities
- Kurzer Feedback-Cycle für Entwickler
- Automatisch generierte Update-Tools für Endnutzer

r&d

arendi

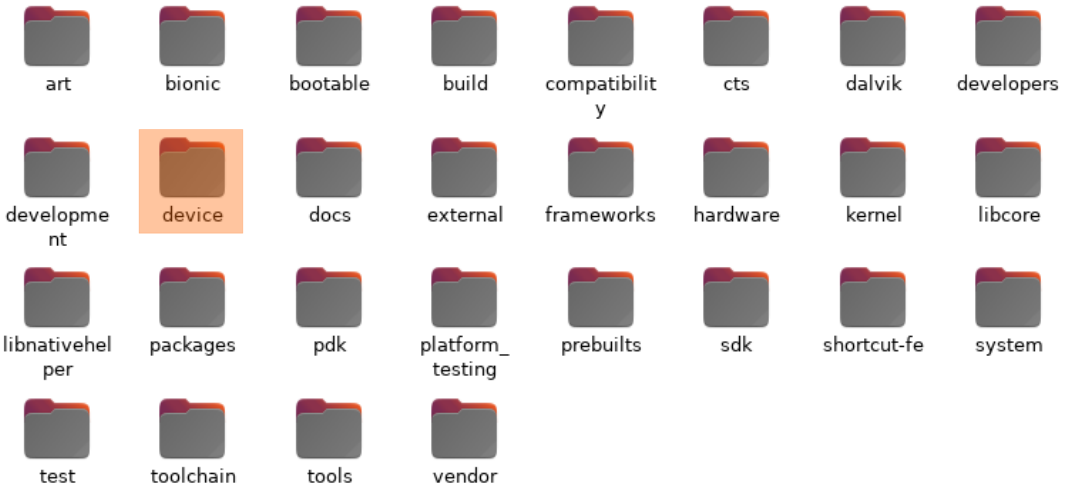
Android Entwicklung

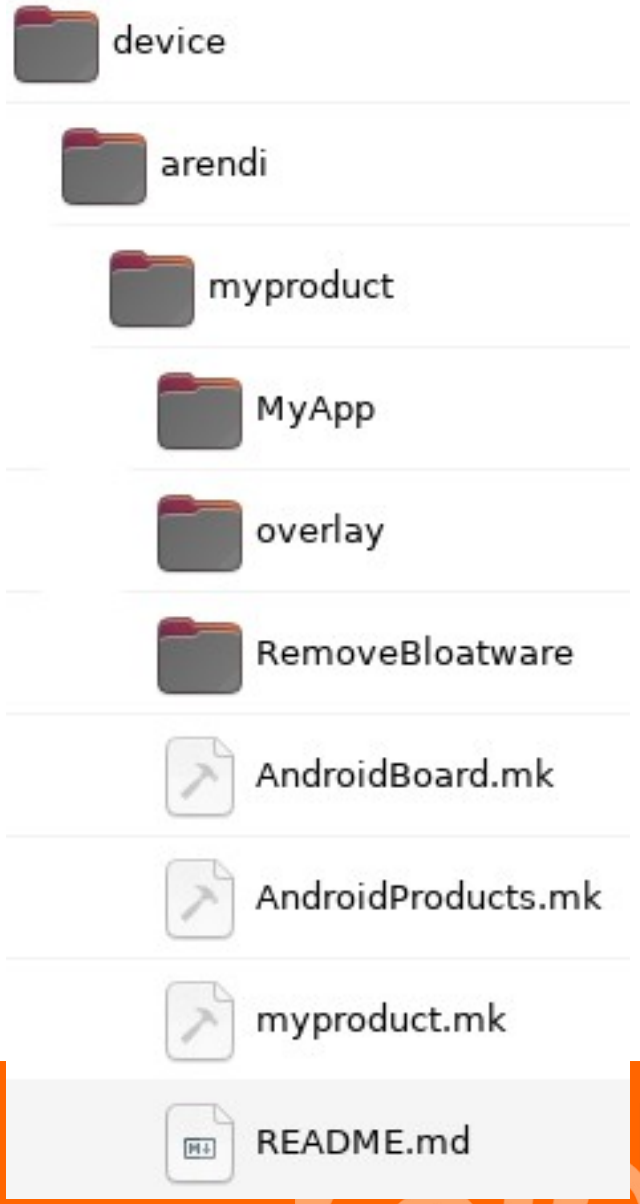
- Makefile basiert, “.mk” Endung
  - Neuere Versionen sind JSON orientiert (Soong)
- Verschiedene “Recipes” die global selektiert werden können
- Verschiedene Android-Varianten
  - eng, userdebug, user
- Ok Dokumentation, Details nicht abgedeckt

r&d

arendi

Android Entwicklung





arendi

Produktedefinition in AndroidProducts.mk

```
# Inherit from mk8909
$(call inherit-product, device/qcom/msm8909/msm8909.mk)

PRODUCT_NAME := myProduct
PRODUCT_VENDOR := Arendi
TARGET_VENDOR := arendi
PRODUCT_BRAND := Arendi
PRODUCT_MODEL := myProduct on SC20
PRODUCT_MANUFACTURER := Arendi

PRODUCT_PROPERTY_OVERRIDES += arendi.imageVersion=20

PRODUCT_PACKAGE_OVERLAYS := device/arendi/myproduct/overlay

PRODUCT_PACKAGES += RemoveBloatware
PRODUCT_PACKAGES += MyApp
```


r&d

arendi

Eigene Module via Android.mk

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)

LOCAL_MODULE_TAGS := optional

LOCAL_PACKAGE_NAME := MyApp
LOCAL_MODULE := MyApp
LOCAL_CERTIFICATE := platform
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
LOCAL_MODULE_CLASS := APPS
LOCAL_PRIVILEGED_MODULE := true

PACKAGES.$(LOCAL_MODULE).OVERRIDES:= $(LOCAL_OVERRIDES_PACKAGES)
TARGET_OUT_DATA_APPS_PRIVILEGED := $(TARGET_OUT_DATA)/priv-app

LOCAL_SRC_FILES := $(LOCAL_MODULE).apk
include $(BUILD_PREBUILT)
```


r&d

Hersteller Bloatware entfernen

```
#This is a fake module to get rid of pre-installed packages
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)

LOCAL_MODULE_TAGS := optional
LOCAL_MODULE := RemoveBloatware
LOCAL_PACKAGE_NAME := RemoveBloatware
LOCAL_MODULE_CLASS := APPS
LOCAL_PRIVILEGED_MODULE := true
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
TARGET_OUT_DATA_APPS_PRIVILEGED := $(TARGET_OUT_DATA)/priv-app
TARGET_OUT_FAKE_PRIVILEGED := $(TARGET_OUT)/fake

# overrrides packages to avoid their addition to the image
LOCAL_OVERRIDES_PACKAGES := \
    SnapdragonMusic \
    SnapdragonGallery \
    SnapdragonCamera \
PACKAGES.$(LOCAL_MODULE).OVERRIDES:= $(LOCAL_OVERRIDES_PACKAGES)
PRODUCT_PACKAGES -= $(LOCAL_OVERRIDES_PACKAGES)

include $(BUILD_PHONY_PACKAGE)
```

Android Konfiguration via overlay/conf.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- Control the behavior when the user long presses the power button.
            0 - Nothing
            1 - Global actions menu
            2 - Power off (with confirmation)
            3 - Power off (without confirmation)
    -->
    <integer name="config_longPressOnPowerBehavior">3</integer>

    <!-- Flag specifying whether VoLTE is available on device -->
    <bool name="config_device_volte_available">true</bool>

    <!--Thresholds for LTE dbm in status bar-->
    <integer-array translatable="false" name="config_lteDbmThresholds">
        <item>-140</item>    <!-- SIGNAL_STRENGTH_NONE_OR_UNKNOWN -->
        <item>-128</item>    <!-- SIGNAL_STRENGTH_POOR -->
        <item>-118</item>    <!-- SIGNAL_STRENGTH_MODERATE -->
        <item>-108</item>    <!-- SIGNAL_STRENGTH_GOOD -->
        <item>-98</item>     <!-- SIGNAL_STRENGTH_GREAT -->
        <item>-44</item>
    </integer-array>
</resources>
```

Android Kompilation

- Mehrere Stunden
- Mehrere GB an Daten werden erzeugt

- Android flashen via Android Recovery – “fastboot”
- Qualcomm Chip Flashing via “QFIL” in speziellem Boot-Modus

SELinux

- Zugriffskontrolle auf Ressourcen
- https://stopdisablingselinux.com/
- Bedeutet Einarbeitungszeit
- Android hat MCL und MCS security
  - type arendi_app, domain, mlstrustedsubject;
    app_domain(arendi_app);

https://people.redhat.com/duffy/selinux/selinux-coloring-book_A4-Stapled.pdf

arendi

SELinux

- Android in permissive Mode setzen
  - *adb shell setenforce 0*
- Logcat nach Zugriffsverstössen scannen
  - *adb shell su root dmesg | grep 'avc: '*
- Automatisch Zugriffsregeln aus Verstössen generieren
  - *adb logcat -b events -d | audit2allow -p policy*
- Neue Regeln zu policy hinzufügen
  - *allow arendi_app arendi_led:chr_file { setattr getattr ioctl open read write };*
- https://source.android.com/security/selinux/validate

r&d

Gerätekonfiguration via Properties

- Konfiguration nach Inbetriebnahme durch Hilfs-Programm
- Fleeting oder Persistent Konfiguration
- Via adb setprop/getprop
- MCL/MLS muss richtig gesetzt sein für App-Zugriff
- property_contexts File Properties hinzufügen

```
arendi.              u:object_r:arendi_prop:s0
persist.arendi.      u:object_r:arendi_prop:s0
```

arendi

Was nicht passt wird passend gemacht

- Verändern des Source-Codes und somit des Verhaltens
- Zum Teil Einarbeitungszeit in AOSP-Teil nötig
- Dokumentation und Beispiele meist nur spärlich vorhanden

```
--- a/frameworks/base/services/usb/java/com/android/server/usb/UsbDeviceManager.java
+++ b/frameworks/base/services/usb/java/com/android/server/usb/UsbDeviceManager.java
@@ -654,6 +656,23 @@ public class UsbDeviceManager {
                 functions = getDefaultFunctions();
             }
             functions = applyAdbFunction(functions);
+            String arendiStatus = SystemProperties.get(ARENDI_USB_PROPERTY, "unknown");
+            if(arendiStatus.equals("start")) {
+              functions = UsbManager.addFunction(functions, UsbManager.USB_FUNCTION_MTP);
+              functions = UsbManager.addFunction(functions, UsbManager.USB_FUNCTION_ADB);
+            }
+            if(arendiStatus.equals("stop")) {
+              functions = UsbManager.removeFunction(functions, UsbManager.USB_FUNCTION_MTP);
+              functions = UsbManager.removeFunction(functions, UsbManager.USB_FUNCTION_ADB);
+            }

             String oemFunctions = applyOemOverrideFunction(functions);
```


r&d

arendi

Probleme und Hindernisse - Projektablauf

- Dokumentation von Nicht-Mainstream Features
- Riesiger Source-Code
  - Gute Volltextsuche erforderlich
- Nur ein kleiner Fisch mit kleinen Stückzahlen
- Second/Third Level Support über China
  - Verzögerungszeiten inbegriffen

r&d

arendi

Probleme und Hindernisse - Android

- Unerwünschte Android Features
  - AudioRouting
  - Headset-Erkennung
  - Sicherheitsfeatures
- Android Fragmentierung
- AOSP ist riesiges Projekt mit ganz verschiedenen Teilen
- Low-Level Driver Blobs

r&d

Probleme und Hindernisse - Hardware

- Antennen haben grossen Einfluss auf das Benutzererlebnis
  - Off-the-shelf Antenne unbrauchbar
  - Custom aktive Antenne entwickelt
- Mobilfunk sehr komplexe und verworrene Materie
  - LTE-Funktion muss evtl. für Chips durch Betreiber (Swisscom, ...) freigeschaltet werden
- Chip-Dokumentation gross, übersetzt und nicht detailiert

arendi

Zusammenfassung

- Mit Android können in kurzer Zeit umfangreiche und komplexe Projekte realisiert werden
- Features lassen sich einfach implementieren oder wo nötig auch einkaufen
- Bietet gute Abstraktion der Hardware für Programmierer
- Kompliziert wird es sobald der “normale” Weg verlassen wird
- Hardware hat grossen Einfluss auf das Benutzererlebnis

r&d

arendi

Wir sind Ihre Lösung.

Arendi AG
Eichtalstrasse 55
8634 Hombrechtikon
Schweiz

Telefon +41 55 254 30 30
Fax +41 55 254 30 31
www.arendi.ch

r&d