Android Embedded

Smart Hubs als Schaltzentrale des IoT

Dominik Helleberg
Dominik Helleberg

Mobile Development
Android / Embedded
Tools

http://dominik-helleberg.de/+
Android as we know + love it
### Android as we know + love it

<table>
<thead>
<tr>
<th>Period</th>
<th>Android</th>
<th>iOS</th>
<th>Windows Phone</th>
<th>BlackBerry OS</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2 2014</td>
<td>84.7%</td>
<td>11.7%</td>
<td>2.5%</td>
<td>0.5%</td>
<td>0.7%</td>
</tr>
<tr>
<td>Q2 2013</td>
<td>79.6%</td>
<td>13.0%</td>
<td>3.4%</td>
<td>2.8%</td>
<td>1.2%</td>
</tr>
<tr>
<td>Q2 2012</td>
<td>69.3%</td>
<td>16.6%</td>
<td>3.1%</td>
<td>4.9%</td>
<td>6.1%</td>
</tr>
<tr>
<td>Q2 2011</td>
<td>36.1%</td>
<td>18.3%</td>
<td>1.2%</td>
<td>13.6%</td>
<td>30.8%</td>
</tr>
</tbody>
</table>

Source: IDC, 2014 Q2
Android everywhere?
Android everywhere?

Google Glass  http://www.google.com/glass/start/
Android everywhere?
But there’s even more...

Andy Rubin
@Arubin

the definition of open: "mkdir android; cd android; repo init -u
git://android.git.kernel.org/platform/manifest.git; repo sync; make"

https://twitter.com/Arubin/status/27808662429
Android Embedded

Amazon
Samsung
Boing 787
LG Kühlenschrank
Motorola MC3200
Honda
NCR

http://www.amazon.de/Kindle-Fire-Zoll-HDX-Display-WLAN/dp/B00D695H6O
http://www.samsung.com/de/promotions/galaxycamera/
http://www.boeing.com/boeing/commercial/787family/
http://www.lg.com/us/refrigerators/lg-LFX31995ST-french-3-door-refrigerator
http://goo.gl/5dOVL7
http://response.ncr.com/NCR-Kalpana
How does it relate to IoT?
Manage local „Things“

- Z-Wave
- ZigBee
- Thread
- WiFi
- Bluetooth (LE)
- ANT(+)
- BidCos
- KNX
- CAN Bus
- AUTOSAR
- BoardR-Reach
Smart Hubs

**Secure** local „Things“

- Encryption
- Secure Connection
- Distribute Updates
- Firewall / Gateway
Smart Hubs

**Control** local „Things“

- Centralized Logic
- Data Aggregation
- Status Reports
- Local HMI

Smart Hub
Smart Hubs are embedded Systems

- Robust
- Secure
- Easy to update
- Embedded friendly
- Rich Graphics
- Easy to build upon
Is there anything new?

We do have

• Windows Embedded
• QNX
• Embedded Linux(s)

....
Smart Hubs with Android

- Performance + HMI
- „App“ like Build-cycle
- Stable + Proven Architecture
- Pluggable Build System
- Robust + Secure OTA Update
- Protocols + Hardware integration
Smart Hubs with Android

- Performance + HMI
- „App“ like Build-cycle
- Stable + Proven Architecture
- Pluggable Build System
- Robust + Secure OTA Update
- Protocols + Hardware integration
Smart Hubs with Android

HMI Evolution
Smart Hubs with Android

Hardware Acceleration “builtin”

Application → Views → Canvas API → libHWUI → SKIA → OpenGL ES → SurfaceFlinger
Smart Hubs with Android
Smart Hubs with Android

Google „Power“

http://www.google.com/design/spec/material-design/introduction.html#
https://www.youtube.com/watch?v=EBITzQsUoOw
http://arstechnica.com/gadgets/2014/07/examining-project-volta-we-put-android-l-through-our-battery-test/
Smart Hubs with Android

Google „Power“

Ian Rogers
Shared publicly - Nov 9, 2014

With ART being open source, and Android 5.0 hitting AOSP, its time for some openhub (aka ohloh) analysis. Some fun stats:
- "took an estimated 79 years of effort (COCOMO model)"
- "296,665 lines of code"
- I’ve changed 764,371 lines of code. That is, I’ve contributed 2.5 lines of code for every line of code that has so far stuck, most of which won’t be mine.
- I’ve contributed 18,849 lines of make file code.
- I’ve contributed 40,156 lines of assembly code. These numbers are high and remind me of the sort of thing I thought I’d have grown out of when I started programming in the ’90s (at least I don’t have a line count for pascal).

In a Nutshell, platform_art...

... has had 17,034 commits made by 127 contributors representing 296,665 lines of code

... is mostly written in C++
  with an average number of source code comments

... has a codebase with a long source history
  maintained by a very large development team
  with increasing Y-O-Y commits

... took an estimated 79 years of effort (COCOMO model)
  starting with its first commit in June, 2011
  ending with its most recent commit 2 days ago

https://plus.google.com/u/0/104162817526798820937/posts/UVXbWmxWvBh
https://www.openhub.net/p/platform_art
Current „State of the art“
Current „State of the art“
Current „State of the art“
Current „State of the art“
Current „State of the art“
Android Graphics Performance
Media Framework

https://source.android.com/devices/media.html
Smart Hubs with Android

- Performance + HMI
- „App“ like Build-cycle
- Stable + Proven Architecture
- Pluggable Build System
- Robust + Secure OTA Update
- Protocols + Hardware integration
Smart Hubs with Android

Application

SystemService

Manager Interface

Manager Service

libhardware

driver.so

/dev/foo

Application logic

Java API / SDK / Documentation
Access to service over RPC

Binder Interface for the service (AIDL)

Logic, Security Checks, Native JNI calls

Native “Interface” (header file)

Device Dependent Driver

Kernel / Module
Smart Hubs with Android
Smart Hubs with Android

- Performance + HMI
- „App“ like Build-cycle
- **Stable + Proven Architecture**
- Pluggable Build System
- Robust + Secure OTA Update
- Protocols + Hardware integration
Smart Hubs with Android
shell@shamu:/ $ mount

rootfs / rootfs ro,seclabel,relatime 0 0
tmpfs /dev tmpfs rw,seclabel,nosuid,relatime,size=1505044k,nr_inodes=171672,mode=755 0 0
devpts /dev/pts devpts rw,seclabel,relatime,mode=600 0 0
proc /proc proc rw,relatime 0 0
sysfs /sys sysfs rw,seclabel,relatime 0 0
selinuxfs /sys/fs/selinux selinuxfs rw,relatime 0 0
debugfs /sys/kernel/debug debugfs rw,relatime 0 0
none /acct cgroup rw,relatime,cpuacct 0 0
none /sys/fs/cgroup tmpfs rw,seclabel,relatime,size=1505044k,nr_inodes=171672,mode=755,gid=1000 0 0
tmpfs /mnt/asec tmpfs rw,seclabel,relatime,size=1505044k,nr_inodes=171672,mode=755,gid=1000 0 0
tmpfs /mnt/obb tmpfs rw,seclabel,relatime,size=1505044k,nr_inodes=171672,mode=755,gid=1000 0 0
none /dev/cpuctl cgroup rw,relatime,cpu 0 0
pstore /sys/fs/pstore pstore rw,relatime 0 0
/dev/block/platform/msm_sdcc.1/by-name/system /system ext4 ro,seclabel,relatime,data=ordered 0 0
/dev/block/platform/msm_sdcc.1/by-name/cache /cache ext4 rw,seclabel,nosuid,nodev,noatime,data=ordered 0 0
/dev/block/platform/msm_sdcc.1/by-name/modem /firmware ext4
ro,context=u:object_r:firmware_file:s0,relatime,data=ordered 0 0
/dev/block/platform/msm_sdcc.1/by-name/persist /persist ext4
rw,seclabel,nosuid,nodev,relatime,data=ordered 0 0
/dev/block/platform/msm_sdcc.1/by-name/mdm1m9kefs3 /fsg ext4
ro,context=u:object_r:fsg_file:s0,nosuid,nodev,relatime 0 0
adb /dev/usb-ffs/adb functionfs rw,relatime 0 0
/dev/block/dm-0 /data ext4
rw,seclabel,nosuid,nodev,noatime,nodiratime,nobarrier,noauto_da_alloc,data=ordered 0 0
/dev/fuse /mnt/shell/emulated fuse
rw,nosuid,nodev,noexec,relatime,user_id=1023,group_id=1023,default_permissions,allow_other 0 0
shell@shamu:/ $ mount

rootfs / rootfs ro,seclabel,relatime 0 0

selinuxfs /sys/fs/selinux selinuxfs rw,relatime 0 0

tmpfs /mnt/asec tmpfs rw,seclabel,relatime,size=1505044k

/dev/block/platform/msm_sdcc.1/by-name/system /system ext4 ro,seclabel,relatime,data=ordered 0 0
Smart Hubs with Android

- Read-Only System partition
- SE-Linux ready
- Application Sandbox
- Encryption
- Verified Boot
- High Performance IPC (Binder)
Smart Hubs with Android

- Performance + HMI
- „App“ like Build-cycle
- Stable + Proven Architecture
- **Pluggable Build System**
- Robust + Secure OTA Update
- Protocols + Hardware integration
Smart Hubs with Android

Hardware
Komponenten

- SBC oder MBC
- Base Board
- (optional) CPU Board

- TFT / LCD Screen
- Touch Sensor

- Screen / Touch Controller Converter
Smart Hubs with Android Hardware

Baseboard - Evaluation

- SOC (CPU / GPU)
- RAM
- Storage
- Interfaces
Smart Hubs with Android Hardware

Baseboard
• SOC (CPU / GPU)
• RAM
• Storage
• Interfaces

SBC

MBC
Smart Hubs with Android Hardware

**Screen + Touch**
- LCD / TFT Screen
- Touch Sensor
- Converter to „connect“ to BaseBoard

[Image of a screen and hardware components]
Smart Hubs with Android

- Performance + HMI
- „App“ like Build-cycle
- Stable + Proven Architecture
- **Pluggable Build System**
- Robust + Secure OTA Update
- Protocols + Hardware integration
Build the Droid

terminology

Product and Board configuration

Modules

Build Target

- **Boardconfiguration**: contain hardware specific configurations
  - Architecture (ARM/x86/mips,…)
  - Peripherals: GPS, Bluetooth,…
  - Boot Parameter,…
  - Inheritance

```
myBoardConfig.mk
fsl-BoardConfigCommon.mk
  fsl-omx.mk
  fsl-gpu.mk
  fsl-codec.mk
```
- **Product configurations**: describes software
  - Device Name, Vendor Name, etc...
  - Pre-Installed Apps, Themes, Sounds
  - Localisation
  - Inheritance

![Diagram showing the product configurations]

- `my_product.mk`
  - `ereader.mk`
  - `imx5x.mk`
    - `generic_no_telephony.mk`
    - `language_small.mk`
      - `core.mk`
Build the Droid

terminology

Product and Board configuration

Modules

Build Target

Everything is a Module
- Apps, ADB, Framework, SDK, Libraries, Fonts, etc. (3713 Modules in 5.0)
- Defined in own Makefile: Android.mk
- Lot of stubs, extensive use of Makefile macros

packages/apps/Calculator/Android.mk:

```
LOCAL_PATH := $(call my-dir)
LOCAL_MODULE_TAGS := optional
LOCAL_STATIC_JAVA_LIBRARIES := libarity android-support-v4 guava
LOCAL_SRC_FILES := $(call all-java-files-under, src)
LOCAL_SDK_VERSION := current
LOCAL_PACKAGE_NAME := Calculator
```
Results:
- **make droid**: complete images for the target device, host tools (boot.img, system.img, recovery.img…)
- **make dist**: `device_target_files`
  (a zip archive, containing all information to create every build system output)

Releasetools:
a set of tools to create images, sign and verify images and updates
- `.sign_target_files_apks`
  sign a build with test or debug certificates. All builds are signed!
- `.img_from_target_files`
  create flashable images (called from ‘make droid’)
- `.ota_from_target_files`
  create full or **incremental** over-the-air updates
For a Nexus 7 (2013)

1. **configure the build**: loads Product and Board configuration in environment
   
   ```
   $> source build/envsetup.sh && lunch aosp_flo-userdebug
   ```

2. **compile**: builds “core” and each module that is included in this configuration
   
   ```
   $> make droid -j8
   
   ....zzzZZZzzzZZZzzz...
   ```

3. **Flash / distribute the result**: runs releasetools, packs, signs & flashes the build
   
   ```
   $> make dist || fastboot flashall
   ```
Smart Hubs with Android

- Performance + HMI
- „App“ like Build-cycle
- Stable + Proven Architecture
- Pluggable Build System
- Robust + Secure OTA Update
- Protocols + Hardware Integration
Smart Hubs with Android

OTA Updates
Results:

- **make droid**: complete images for the target device, host tools (boot.img, system.img, recovery.img…)
- **make dist**: `device_target_files` (a zip archive, containing all information to create every build system output)

Releasetools:

a set of tools to create images, sign and verify images and updates

- `./sign_target_files_apks`  
  sign a build with test or debug certificates. All builds are signed!
- `./img_from_target_files`  
  create flashable images (called from ‘make droid’)
- `./ota_from_target_files`  
  create full or **incremental** over-the-air updates
Smart Hubs with Android

OTA Updates

Android

OTA Update

RecoverySystem.VerifyUpdate()

OK

read args from file

verify signature

execute edify update script

execute updater binary

unpack archive

cleanup

save log & reboot

Recovery

fail

Smart Hubs with Android

OTA Updates
Smart Hubs with Android

- Performance + HMI
- „App“ like Build-cycle
- Stable + Proven Architecture
- Pluggable Build System
- Robust + Secure OTA Update
- Protocols + Hardware Integration
Smart Hubs with Android
Hardware Integration
Smart Hubs with Android

Hardware Integration

Application

Application logic

SystemService

Java API / SDK / Documentation
Access to service over RPC

Manager Interface

Binder Interface for the service (AIDL)

Manager Service

Logic, Security Checks, Native JNI calls

libhardware

Native “Interface” (header file)

driver.so

Device Dependent Driver

/dev/foo

Kernel / Module
Smart Hubs with Android

Hardware Integration

- Grove 16x2 display with I2C extension
  - LCD-Module
  - RGB-Module
- MarS Board with Android 4.0.4 (BSP)
- The LCD is connected through I2C
  - LCD 0x3e
  - RGB 0x62

```
#include <driver.so>

// Open the device file
open(/dev/foo, NULL, O_RDWR);
```
Smart Hubs with Android
Hardware Integration

- The Android Kernel is based on a Linux Kernel with some android specific adaptations
- The Linux Kernel provides an I2C-Core-Driver
  - It is responsible for the communication between the devices connected to the bus
- For the LCD-Module and the RGB-Module an I2C-Client-Driver is needed
Smart Hubs with Android

Hardware Integration

driver.so

/dev/foo
Smart Hubs with Android

Hardware Integration

Application

SystemService

ManagerInterface

Manager Service

libhardware

driver.so

/dev/foo
Smart Hubs with Android
Hardware Integration

- Access to drivers through a HAL library (user space)
- HAL library uses the system call interface to get access to the device files (/dev/lcd1313M1 u. /dev/rgb1313M1)
- A proprietary license can be used (Driver has to use GPL)

libhardware
The HAL library consists of two components:

- Implementation under:
  - /device/fsl/marsboard_6q/lcd1313M1

- Header file under:
  - /hardware/libhardware/include/hardware

The header file represents the interface of the HAL library.
Some functions of the HAL library

- `displayInit()`
- `setText(...)`
- `setRGB(...)`
- `backlightOn() / backlightOff()`
- `...`
Smart Hubs with Android

Hardware Integration

Application

SystemService

Manager Interface

Manager Service

libhardware

driver.so

/dev/foo
The Manager Service consists of two components:

- the native methods in `com_android_server_LCDService.cpp`
- `LCDService.java`

- `com_android_server_LCDService.cpp` includes the header file of the HAL library and loads the library
- `LCDService.java` calls the native methods over the JNI
Smart Hubs with Android
Hardware Integration

Manager Service
Smart Hubs with Android

Hardware Integration

Application

SystemService

Manager Interface

Manager Service

libhardware

driver.so

/dev/foo
Smart Hubs with Android

Hardware Integration

- A ILCDService.aidl file for the LCDService is needed
- This file contains the declaration of the functions callable from Binder
- The .aidl file is used to generate a Binder-Interface for the Java service (ILCDService.java)
Smart Hubs with Android

Hardware Integration

Application

SystemService

ManagerInterface

Manager Service

libhardware

driver.so

/dev/foo
public class LCDManager
{
    public static LCDManager getLCDManager()
    {
        IBinder b = ServiceManager.getService("lcd1313M1");
        ILCDService service = ILCDService.Stub.asInterface(b);
        return new LCDManager(service);
    }
    public int setText(String mString) {
        try {
            return mService.setText(mString);
        } catch (RemoteException e) {
            return -1;
        }
    }
}

SystemService e
Smart Hubs with Android
Hardware Integration

Application
  ↓
SystemService
  ↓
Manager Interface
  ↓
Manager Service
  ↓
libhardware
  ↓
driver.so
  ↓
/dev/foo
Smart Hubs with Android
Application Development
How to get an App-Developer started?

“Well, thats easy! Just ..."

- Set up a linux box / VM
- Install build chain
  - `repo init` && `repo sync`
- `lunch full-eng`
- `make` ...
## SDK Manager Integration

The image shows the Android SDK Manager window with SDK path set to `/Users/dhelleberg/dev/android-sdk-mac_x86`. The window is displaying installed packages:

- **MIPS System Image**
- **Google APIs**
- **Sources for Android SDK**

Under the package for **Android 4.0.3 (API 15)**:

- **SDK Platform**
- **Samples for SDK**
- **ARM EABI v7a System Image**
- **Intel x86 Atom System Image**
- **MIPS System Image**
- **Google APIs**

The **LCD SDK-Addon** is also installed under the **Android 4.0 (API 14)** package. The window includes options for showing updates/new and installed items, and buttons for installing, deleting, and deselecting packages.

The message at the bottom of the window reads, "Done loading packages."
SDK Addon

Contents

- Java Libraries / Stubs
- Java Docs
- Metadata
- System Image
- Kernel
- Hardware Definitions (skins)
### Contents

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>addon.xml</td>
<td>886 bytes</td>
<td>Today 21:27</td>
</tr>
<tr>
<td>lcd_sdk_addon-eng.inovex-linux-x86.zip</td>
<td>96.9 MB</td>
<td>Today 20:54</td>
</tr>
</tbody>
</table>
This lets you manage a list of user-contributed external add-on sites URLs.

Add-on sites can provide new add-ons and extra packages. They cannot provide standard Android platforms, system images or docs. Adding a URL here will not allow you to clone an official Android repository.

file:///Users/dhelleberg/work/embedded/addon/addon.xml
## SDK Addon

### SDK Manager Integration II

![Android SDK Manager](image)

**SDK Path:** `/Users/dhelleberg/dev/android-sdk-mac_x86`

### Packages

<table>
<thead>
<tr>
<th>Name</th>
<th>API</th>
<th>Rev.</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIPS System Image</td>
<td>16</td>
<td>2</td>
<td>Not installed</td>
</tr>
<tr>
<td>Google APIs</td>
<td>16</td>
<td>2</td>
<td>Installed</td>
</tr>
<tr>
<td>Sources for Android SDK</td>
<td>16</td>
<td>2</td>
<td>Not installed</td>
</tr>
<tr>
<td>Android 4.0.3 (API 15)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDK Platform</td>
<td>15</td>
<td>5</td>
<td>Installed</td>
</tr>
<tr>
<td>Samples for SDK</td>
<td>15</td>
<td>2</td>
<td>Not installed</td>
</tr>
<tr>
<td>ARM EABI v7a System Image</td>
<td>15</td>
<td>2</td>
<td>Not installed</td>
</tr>
<tr>
<td>Intel x86 Atom System Image</td>
<td>15</td>
<td>1</td>
<td>Installed</td>
</tr>
<tr>
<td>MIPS System Image</td>
<td>15</td>
<td>1</td>
<td>Not installed</td>
</tr>
<tr>
<td>Google APIs</td>
<td>15</td>
<td>2</td>
<td>Installed</td>
</tr>
<tr>
<td><strong>LCD SDK-Addon</strong></td>
<td>15</td>
<td>2</td>
<td>Installed</td>
</tr>
<tr>
<td>Sources for Android SDK</td>
<td>15</td>
<td>2</td>
<td>Not installed</td>
</tr>
<tr>
<td><strong>Android 4.0 (API 14)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDK Platform</td>
<td>14</td>
<td>4</td>
<td>Installed</td>
</tr>
<tr>
<td>Google APIs</td>
<td>14</td>
<td>2</td>
<td>Installed</td>
</tr>
<tr>
<td>Sources for Android SDK</td>
<td>14</td>
<td>1</td>
<td>Not installed</td>
</tr>
</tbody>
</table>

**Show:** Updates/New Installed

Select **New or Updates**

- Obsolete

**Deselect All**

**Install packages...**

**Delete packages...**

**Done loading packages.**
```groovy
android {

    compileSdkVersion "inovex:LCD SDK-Addon:15"

    buildToolsVersion '21.1.2'

    sourceSets {
        main {
```
SDK Addon

Android Studio
### Edit Android Virtual Device (AVD)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVD Name</td>
<td>LCD_device</td>
</tr>
<tr>
<td>Device</td>
<td>10inchLCD (1024 x 786: ldpi)</td>
</tr>
<tr>
<td>Target</td>
<td>LCD SDK-Addon (inovex) - API Level 15</td>
</tr>
<tr>
<td>CPU/ABI</td>
<td>ARM (armeabi-v7a)</td>
</tr>
<tr>
<td>Keyboard</td>
<td>Hardware keyboard present</td>
</tr>
<tr>
<td>Skin</td>
<td>No skin</td>
</tr>
</tbody>
</table>
D/LCDService( 203): LCD changed:
D/LCDService( 203): /------------------\
D/LCDService( 203): | inovex GmbH         |
D/LCDService( 203): | www.inovex.de/em   |
D/LCDService( 203): \------------------/
D/LCDService( 203): Background: #FF5500
Is there anything new?

We do have

- Windows Embedded
- QNX
- Embedded Linux(s)

....
Smart Hubs with Android

• Performance + HMI
• „App“ like Build-cycle
• Stable + Proven Architecture
• Pluggable Build System
• Robust + Secure OTA Update
• Protocols + Hardware integration
So tell me about the dark side...

- No Play Services (unless you go the CTS Route)
- (Almost) no documentation
- No Roadmap
- Obvious (?) things missing
- Realtime ?
Smart Hubs with Android

```
mschaff@pluto:~ » adb shell dexopt -h
Usage:
Short version: Don't use this.
Slightly longer version: This system-internal tool is used to produce optimized dex files. See the source code for details.
mschaff@pluto:~ »
```
/*
* Parse arguments. We want:
* 0. (name of dexopt command -- ignored)
* 1. "--zip"
* 2. zip fd (input, read-only)
* 3. cache fd (output, read-write, locked with flock)
* 4. filename of zipfile being optimized (used for debug messages and
*    for comparing against BOOTCLASSPATH; does not need to be
*    accessible or even exist)
* 5. dexopt flags
*
* The BOOTCLASSPATH environment variable is assumed to hold the correct
* boot class path. If the filename provided appears in the boot class
* path, the path will be truncated just before that entry (so that, if
* you were to dexopt "core.jar", your bootclasspath would be empty).
*
* This does not try to normalize the boot class path name, so the
* filename test won't catch you if you get creative.
*
static int fromZip(int argc, char* const argv[])
{
    int result = -1;
    "OptMain.cpp"
So tell me about the dark side...

- No Play Services (unless you go the CTS Route)
- (Almost) no documentation
- No Roadmap
- Obvious (?) things missing
- Realtime ?
Embedded Android + IoT

https://entwickler.de/IoT-Mag
Vielen Dank für Ihre Aufmerksamkeit!

Kontakt
Dominik Helleberg
http://dominik-helleberg.de/+  
Mail: dominik.helleberg@inovex.de