How Much Work Does it Take and What Is it Like to Integrate an Android SW Stack on a Gadget?

Mark Gross June 2012

Prolog: the lecture I'm not giving

- My first few attempts at writing this talk started with an enumeration of all the different things you are likely to have to deal with when attempting to make a production phone or gadget.
  - I had a lot of content (see backup slides) but nothing very interesting, outside of a business plan.
- The messages I wanted to get across was
  - Doing a new phone with a 300$ BOM will cost you O(50,000,000$) engineering / factory costs to build 100,000 units and ~12months
  - Doing a prototype “gadget” can be done lots cheaper!
  - There are a few options for doing a prototype to consider
  - There is a way to avoid too much low level work.

The story I'm going to tell today

- Some friends of mine have a killer idea
- Its a gadget that uses Android and does something they think is useful.
- I scare them away from going to go for mass production until after a full featured prototype is working.
- I help them evaluate a few critical choices early on that are not so easy to make.
- They go the prototype route reusing production devices and a IOIO to integrate Android with their widget
- They find they still have a lot of work to do customizing android that is reusable if the get funding to take the idea to the masses.
- They go off and start making! In 6 months they start applying for for VC money. In 12 they get 10M in VC funding, quit day jobs to grow a 100M/year 50 full time employee biz in the following 3 years and live happily ever after.

The BIG idea

Wait for it....

Android based vend-sewing machine for maker-bars and Hacker spaces!

Code name: Snowflake

Snowflake

- Smart sewing machine with cloud based social media back-end for shared “stitch” patterns and credit card interface for pay as you go use and purchase of license “stitches” for use on that machine. in a “maker-bar” or Hacker space.
  - They think using android as an OS to tie things together and will work GREAT!
    - Android touch screen based UI
    - Android app integrated to some cloud social media stitch sharing thing
    - TBD HW interface to machine for programming stitches
    - TBD metering of the machine use done by android
    - TBD credit card integration
    - TBD NFC integration
    - TBD UI for creating custom stitches
    - TBD back end for storing your stitches and selectively sharing other stitches.
    - TBD back end e-commerce for buying and selling your stitches for cash and creating a virtual economy based on “stitches”!
      - So great!

We have our programmable Sewing machine picked out, now what?

- It depends:
  - Any real time interactions needed? (no)
  - Any high bandwidth interactions needed? (no)
  - What are the protocols involved?
    - Spi, i2c, serial, 1-wire, CAN, other?
  - UI needs?
  - Network needs?
  - Security needs?
  - Local storage needs?

We want our device to not look like a phone running an app.

- They want to customize the Android OS
  - Change boot splash screens
  - Change Launcher
  - Add special system applications and services
  - Have only their applications run on it / block installing of other APK's
  - Integrated with special cloud services over wifi
    - (or maybe 3g)

Ok, you have the following options

- HW
  - Use a production (COTS) yet hackable device and a IOIO/ADK
  - Use a development board and direct wire the sewing HW to the board.
- SW
  - AOSP
  - Cyanogen
  - Arowboat (only if you use beagle board)
- Versions:
  - Gingerbread?, ICS? Master? J-desert?

We know you don't like dev boards but tell us about them anyway!

- Sigh
  - Development boards are open air devices, most times with no integrated peripherals (other than what's on the SOC)
    - Come with BSP's and reference implementations (maybe not using a version of Android you are interested in)
    - Limited support for individuals
    - Seldom stress tested in a way that matters to you.
    - Exposes you to all the complexities of HW and SW integration
    - Useful when developing for mass production based on specific dev board electrical designs or common SOC's
  - Most common commodity boards at this time are:
    - Panda (OMAP4), Beagle (OMAP3)
    - X86 (netbook class things)
  - Lots of SOC vendors are offering development boards!

x86

- Supported in AOSP android explicitly since ICS.
  - Will take effort to host on real hardware.
- X86 full
  - Makes a great SDK target out of the box. (fastest SDK emulator possible!)
  - Needs kernel work to use on a native device (video/graphics/touch screen...)
- Vbox
  - Has a great root FS you could drop on a netbook and it will pretty much will just work.
  - Graphics is not optimized out of the box.

Panda board using AOSP

- http://pandaboard.org/
- Has explicit support as a build target in AOSP
- I don't know the panda very well but, if you have your heart set on an omap base android I recommend this one.
  - Only because its supported in AOSP master branch.

Beagle / developer approach

- You'll have to pick a display and input devices
- You'll have ok community support
- You will become expert at a few of very low level details of the platform you may not have expected.
  - i.e. you'll end up becoming a u-boot, kernel and Android OS ROM hacker to get it all working.
  - PM enabling does not just work.
  - Optimized audio / video does not work out of the box either

Beagle Bone Android's

- http://processors.wiki.ti.com/index.php/BeagleBone-Android-DevKit_Gui
  - Arowboat based
  - **http://code.google.com/p/rowboat/**
- http://www.opersys.com/blog/beaglebone-android-start
- http://fosiao.com/node/21
- Arowboat is a Ti/OMAP specific android.
  - Its root file system violates android security
  - It will not be suitable for a device needing android branding. But, will get you pretty far.
- Enabling accelerated video will take hunting for the right recipe and closed source binaries needed.

But wait! you still need interfaces!

- Need to wire up the peripherals and sewing machine to dev board
- Need drivers to talk to sewing machine hardware
- Need drivers to talk to card reader
- Need to get a RIL and radio drivers for a modem (3G or CDMA)
- Need drivers to talk to display
- Need drivers to talk to touch screen
- Need drivers to talk to NFC
- Need drivers to talk to digital relay's
- Need power enabling and tuning
- …
- Then you need to test and test and test they all work across all use cases important to your business.

ok OK! you scared us away from dev boards. Now what?

- Use a production android device that is hackable and use a IOIO to interface it with the machine!
- IOIO is a pic24 MCU that integrates with Android over USB.
  - It will be the interface between the COTS android device and the sewing machine, card reader, power strip
- Cyanogen :http://www.cyanogenmod.com/
- AOSP :http://source.android.com/source/index.html

How do we pick a hackable device?

- Option 1 a device with an existing Cyanogen mod.
  - http://www.cyanogenmod.com/devices
  - Finding the right kernel tree can be hard.
- Option 2 buy a google branded Nexus “developer” device and root it.
  - http://source.android.com/source/building-devices.html
  - Use AOSP reference code for your work.

What is this ADK / IOIO stuff?

- These are external micro controllers that interface with the android device over the USB (or Bluetooth) connection
  - http://www.seeedstudio.com/depot/seeeduino-adk-main-board-
  - http://www.sparkfun.com/products/10748
  - http://www.microchipdirect.com/ProductSearch.aspx?Keywords
- Basically the IOIO can set up a network socket communication between the micro controller and the Android OS / android applications.
- ADK uses a custom USB protocol and control pipe handshake for setting up communication channels.
  - IOIO implements either sockets or ADK protocol

IOIO and ADK

- There good starting point data and links for ADKs at:
  - http://developer.android.com/guide/topics/usb/adk.html
- IOIO is open HW (eagle files in git repo on github)
  - https://github.com/ytai/ioio
  - https://github.com/ytai/ioio/wiki
- IOIO supports BT.

Working with IOIO

- Get Android SDK
- Get eclipse
- Learn how to build and run simple Android applications.
- Optional (only if you need to)
  - Get a Pickit3 programmer.
  - Get mplapX

IOIO Gotchas

- IF you are compelled to build the IOIO FW you many find the build options/targets confusing
  - The only ones to look at tend to be: IOIO0023, SPRK0016, PIC24FJ256DA206, PIC24FJ256DA206_ADB
    - Others you will see are mostly legacy targets
  - Set build targets and build the libraries first.
- http://thegnar.org/sync/?p=225
- Eclipse bites.

We want to Customize our Android ROM!

- Ok, you want to change things
  - Key guard and System applications
  - Home screen / Launcher
  - Startup application / intents
  - OTA enabling
- be sure you can build the kernel, OS, flash and boot before you start Hacking
- Search: “android internals” and “embedded android” for content
  - Marakana has an excellent class and on line content.
    - http://marakana.com/s/tags/android/?page=1
  - http://www.opersys.com/training/embedded-android
    - also has an excellent book and also training content
- https://events.linuxfoundation.org/images/stories/pdf/lf_abs12_boie.pdf

But where is the code?

- AOSP
  - repo init -u https://android.googlesource.com/mirror/manifest --mirror; repo sync
  - cd to new dir and do something like: “repo init -u $HOME/work/aosp-mirror/platform/manifest.git”
  - http://source.android.com/source/downloading.html
  - Kernel/*.git
- Cyanogen (Gtab example):
  - repo init -u git://github.com/CyanogenMod/android.git -b gingerbread
  - git clone https://github.com/pershoot/gtab-2632.git

Where to learn?

- A number of android training and consulting companies are putting up their content.
- http://www.opersys.com/training/embedded-android
  - Check the Course-ware tab
- http://marakana.com/training/android/
  - http://marakana.com/s/tags/android/
  - http://marakana.com/bookshelf.html
  - Search you tube for “marakana android”
- http://www.vogella.com/android.html
- Github has some interesting nuggets too!
  - https://github.com/marakana

Where can we started on writing our own custom android applications?

- http://developer.android.com
- Lots of training available for free on the net.
  - http://www.vogella.com/articles/Android/article.html
  - http://developer.android.com/develop/index.html
  - http://androidopen.com/android2011/public/schedule/pr
  - http://video.linux.com/categories/2012-android-builders-
  - https://events.linuxfoundation.org/events/android-builde
  -

And team Snowflake is off an running!

- Over time they did find they needed to hack the IOIO pic code a little
- They added a few android services to the framework of their android
- They stripped out non-Snowflake aps and customized their images quite a bit.
- Wrote a customized OTA installer based on clockwork mod.
  - They rewrote it again later to align more with upstream AOSP
- They wrote a lot of cloud based back end code using AppEngen
- They used BOTH cyanogen mode Gingerbread on a G-Tablet AND a nexus tablet running AOSP master branch.
  - To have a current stack ready to go when getting to funded stage.

The end.

BACK UP

Stuff if I have time to burn.

Mass produced devices are a big deal.

- This is where a lot of capital and risk is.
  - Not hard to end up with 10's of M$'s in commodities tied up at factory for a large build.
  - A build of 10000 units with a 300$ BOM translates into someone coughing up 3M$ up front with significant risk.
  - If the design, software or some other critical ingredient is bad at build time you could have 3M$ worth of doorstops.
    - Or 100's of K$ of rework to do before you can put them on the market.

engineering costs

- Factory automation
  - Factory testing
  - Factory provisioning
- Factory refurbishment
- Regulatory and Certifications
  - For a phone with new radio this is a long poll (3-5 months)
- Carrier coordination / partnerships
- OTA back end and device side enabling
- Supply chain changes
- Field trials and beta programs

SW Engineering Costs

- Base OS
- Kernel
- Drivers
- FW / boot loader
- DRM security enabling
- Power Management and performance tuning
- Hardening and testing
- Target or hero feature and applications
- Back end application and enabling
- MTBF, stress and use case testing

HW engineering costs

- Industrial design
- Mechanical engineering
  - Tooling
- Thermal design
- Design for testing
- Design for manufacturing
- Layout and component selection
  - Be sure components have existing drivers or that the part is SOOooo cheep that its worth the SW costs to do a custom driver
- ESD, thermal, RF, reliability testing
- Stress, environmental, regulatory testing
- Work closely with factory and kernel folks

Specialized costs

- Radios and modems
- Antenna design and testing
- Camera and ISP
- WiFi/BT/GPS
- Graphics
- Video and Media
- Audio
- Power management and Performance
- OTA
- Security

Other Engineering costs

- Technical and support documentation
- Training customer support
- Root cause investigations for shipped product.
- Packaging, Printed material
- Tool creation, care and feeding
- DRM biz + eng engagements with content or service providers that have something to loose.
- SCM and build automation
- Licensed IP
- IP compliance checking and GPL fulfillment
- Bug scrubbing and issue tracking / quality metrics and trending
- Seasonal or recurring efforts (kernel and OS updates)
- Working with trademark holders (google) and other Biz partners or customers.

Overheads

- Building's to work in.
- Salaries and benefits
- Program managers
- Managers, VP's or principles.
  - PHB's got to eat
- Admin, HR …
- Legal
- Marketing
- Travel

So what's it take to do a new Phone?

- About 100 to 200 different heads off and on with a peak burn of 90% for 2 to 5 months.
  - 12 months for first, then 6-9 for derivatives
- 5 to 100 Million $ for commodities used in production depending on how big your first build is.
- 20-40 Million in direct engineering
  - Likely more I don't have experience with.
- Phones are pretty much the worst case device to make.

How about a production “gadget”?

- This is a special / limited use case device who's use is part of a larger system.
  - 50 heads 6-12 months
- Still with a lot of $ tied up in commodities.