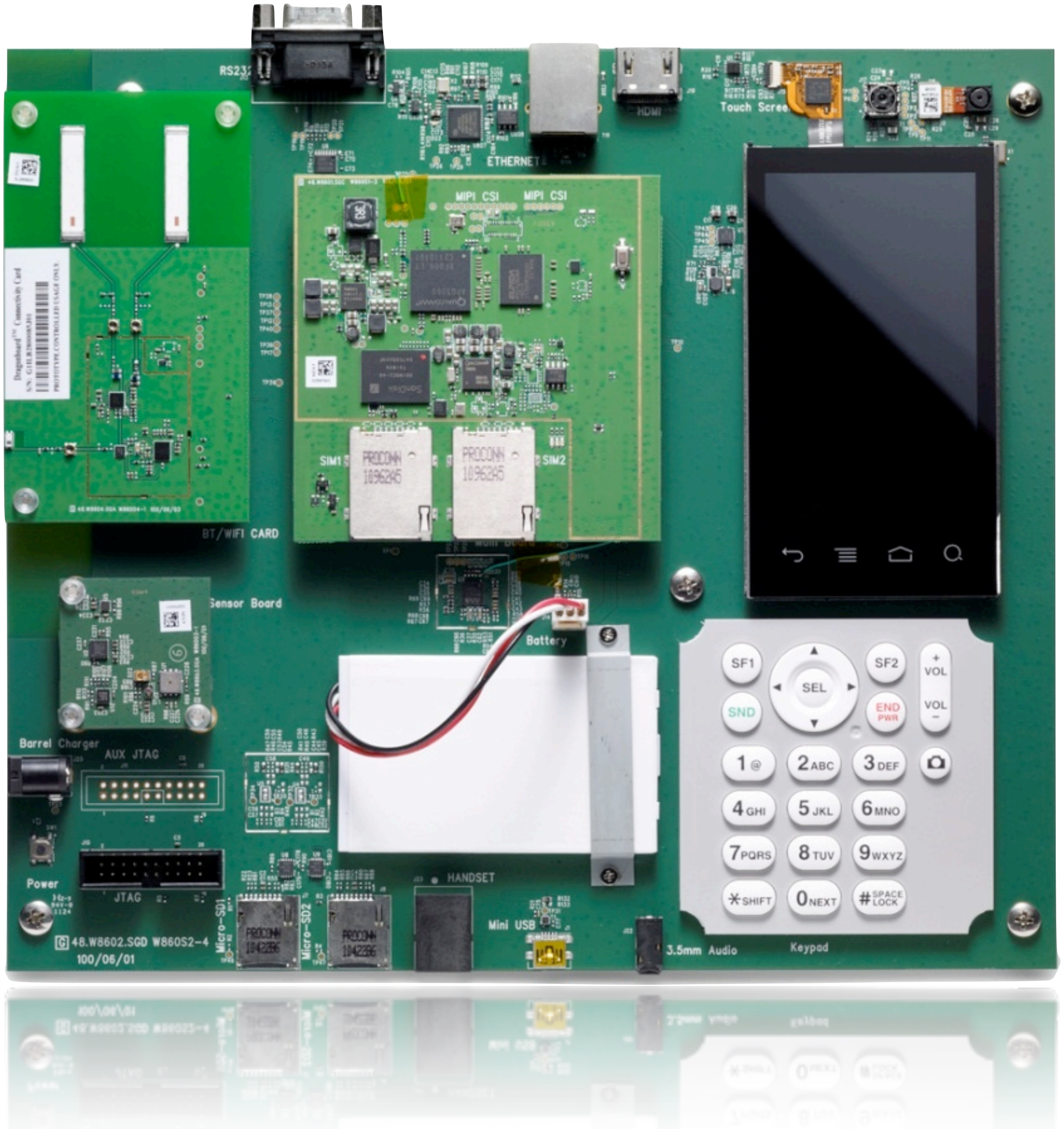
Developing on DragonBoard™

Getting Started with APQ8060 and Pragmatux+Android

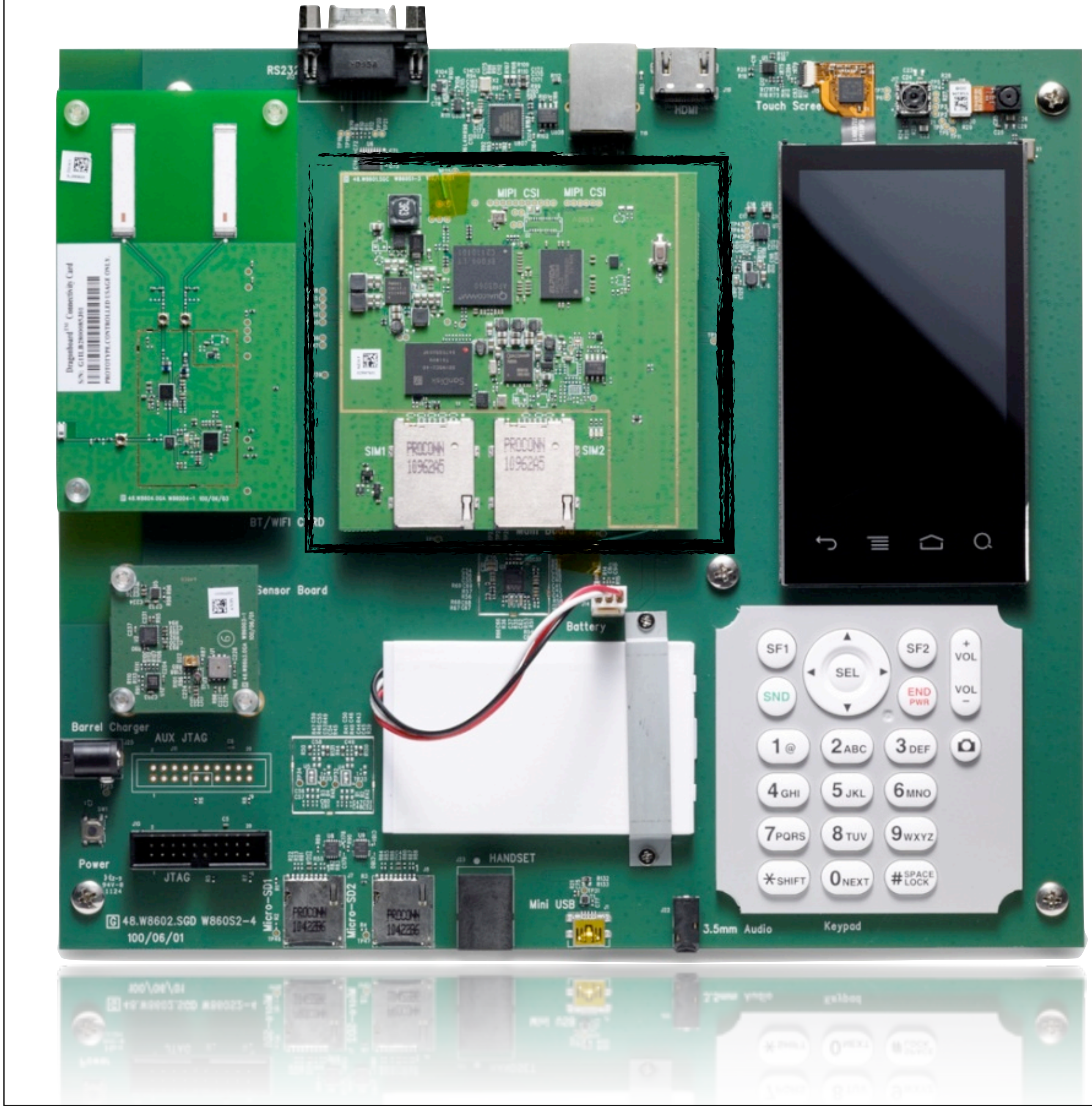
Bill Gatliff
bgat@billgatliff.com

Ryan Kuester
rkuester@insymbols.com

DragonBoard™



DragonBoard™



CPU Daughterboard

APQ8060

ARMv7

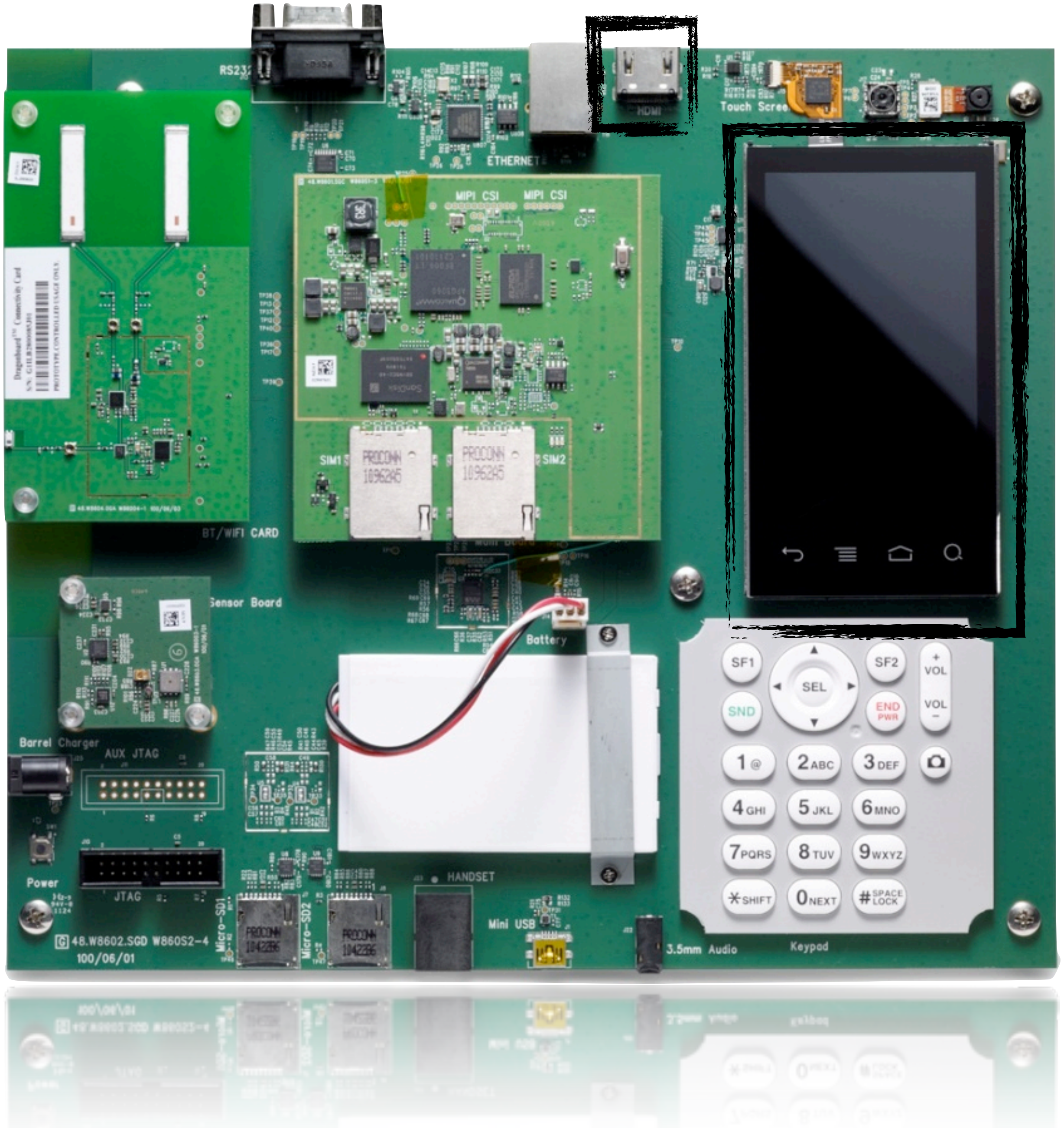
Dual core

1.5 GHz

512 MB RAM

4 GB eMMC

DragonBoard™

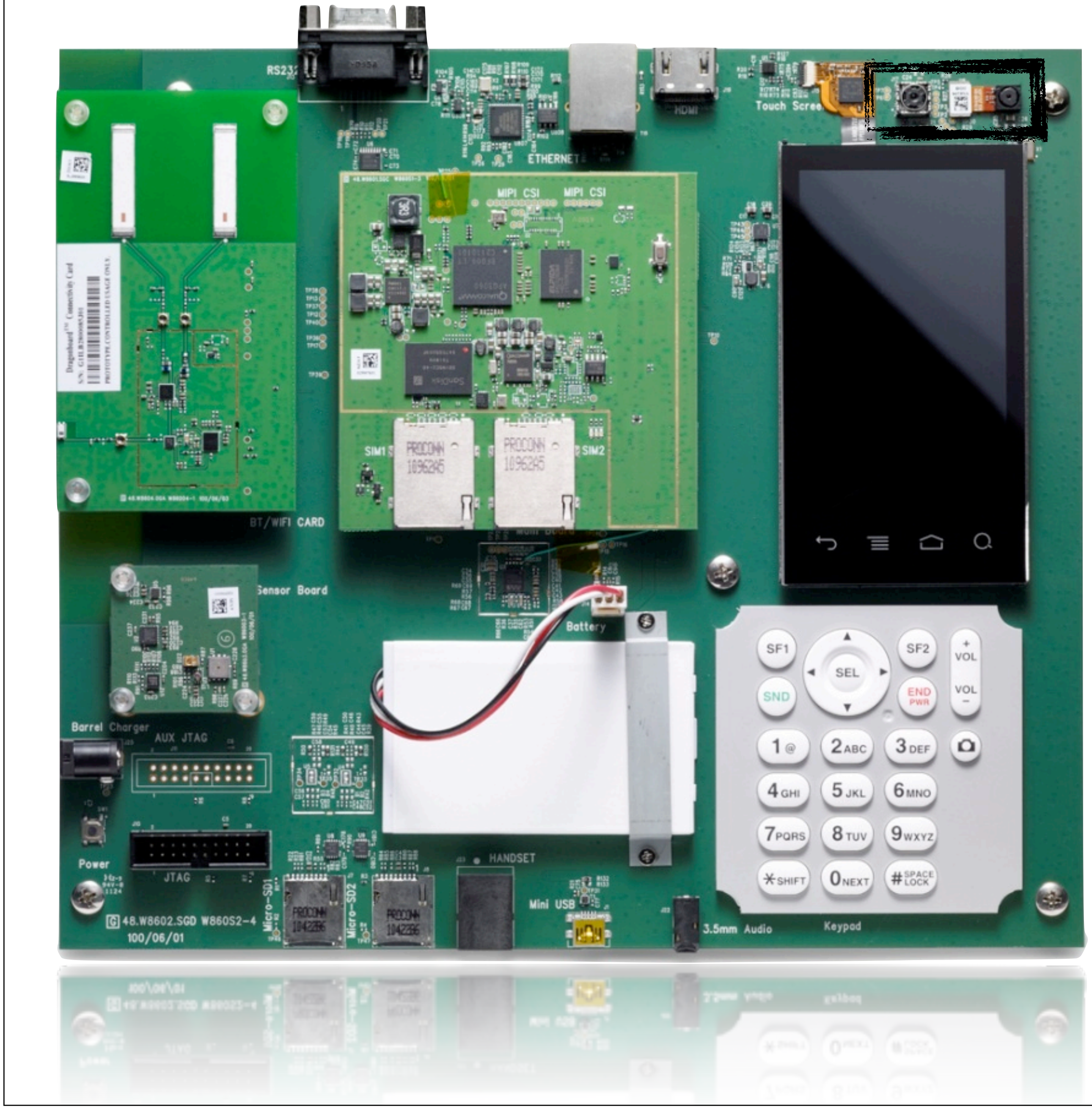


Video

Adreno™ 220 GPU
24-bit 1440x900
RGB
4-lane MIPI DSI

HDMI Out
1080p HD Video

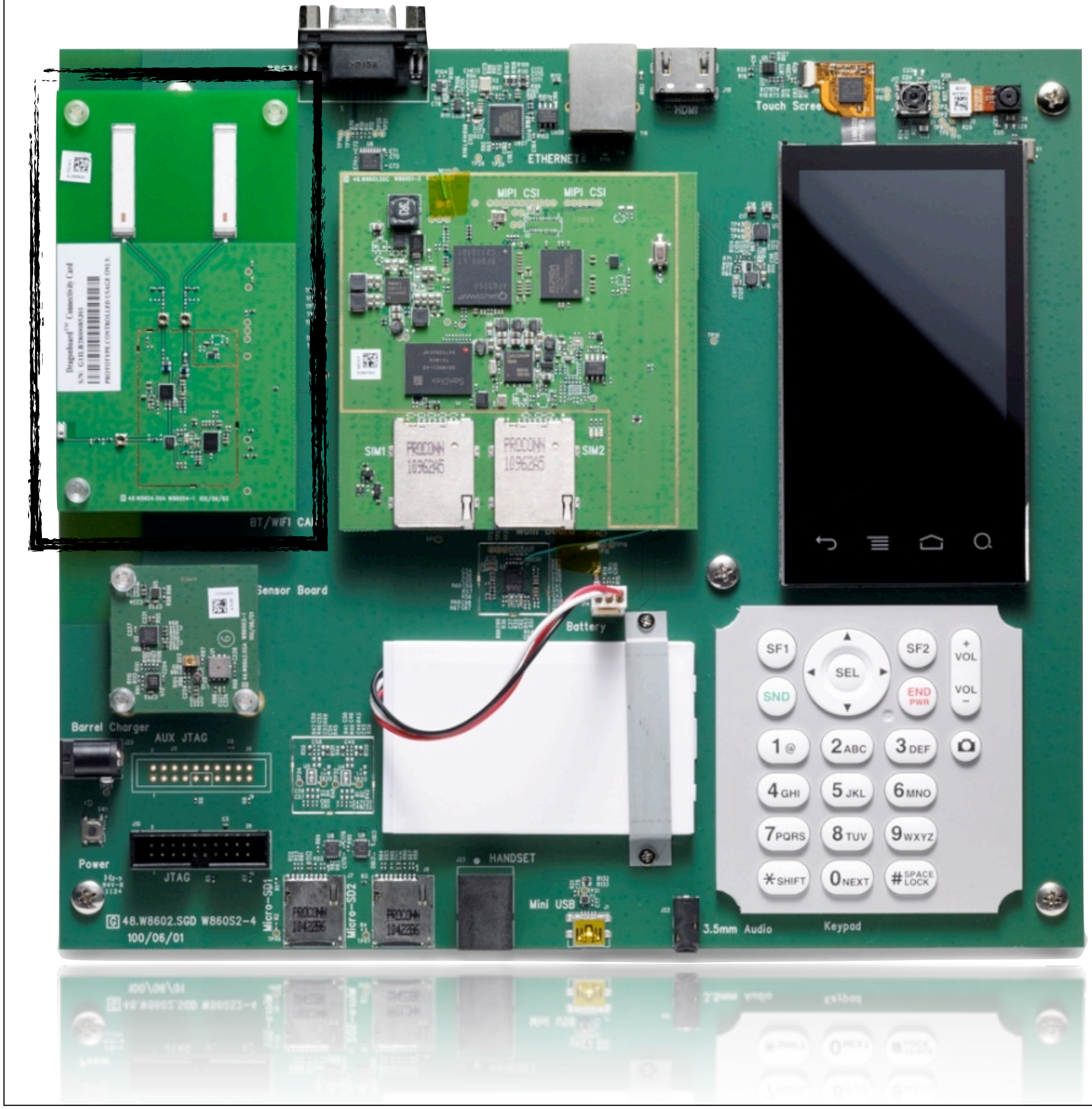
DragonBoard™



Camera

4-lane MIPI CSI
2-lane MIPI CSI

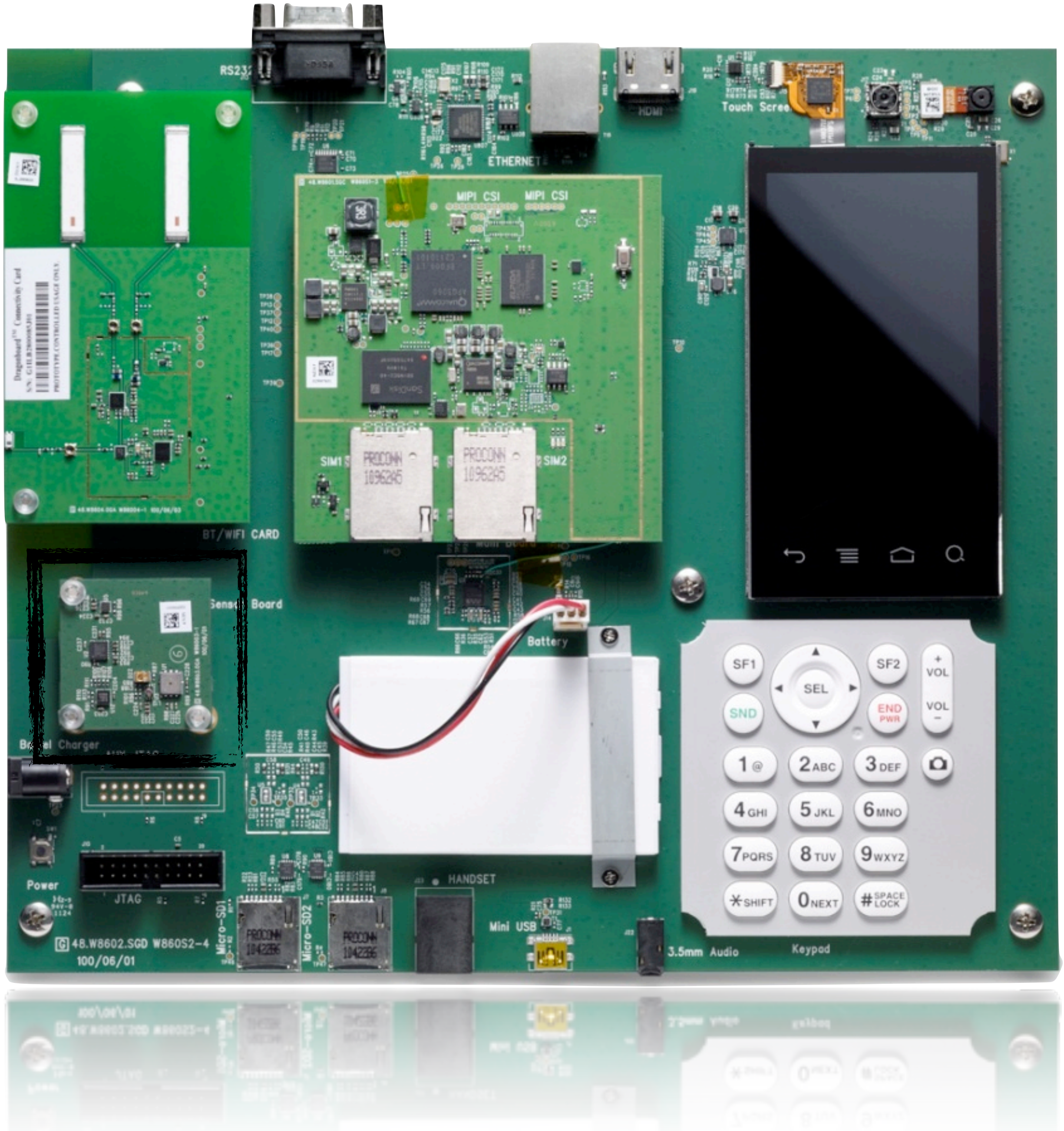
DragonBoard™



Wireless Card

Wi-Fi b/g/n
Bluetooth

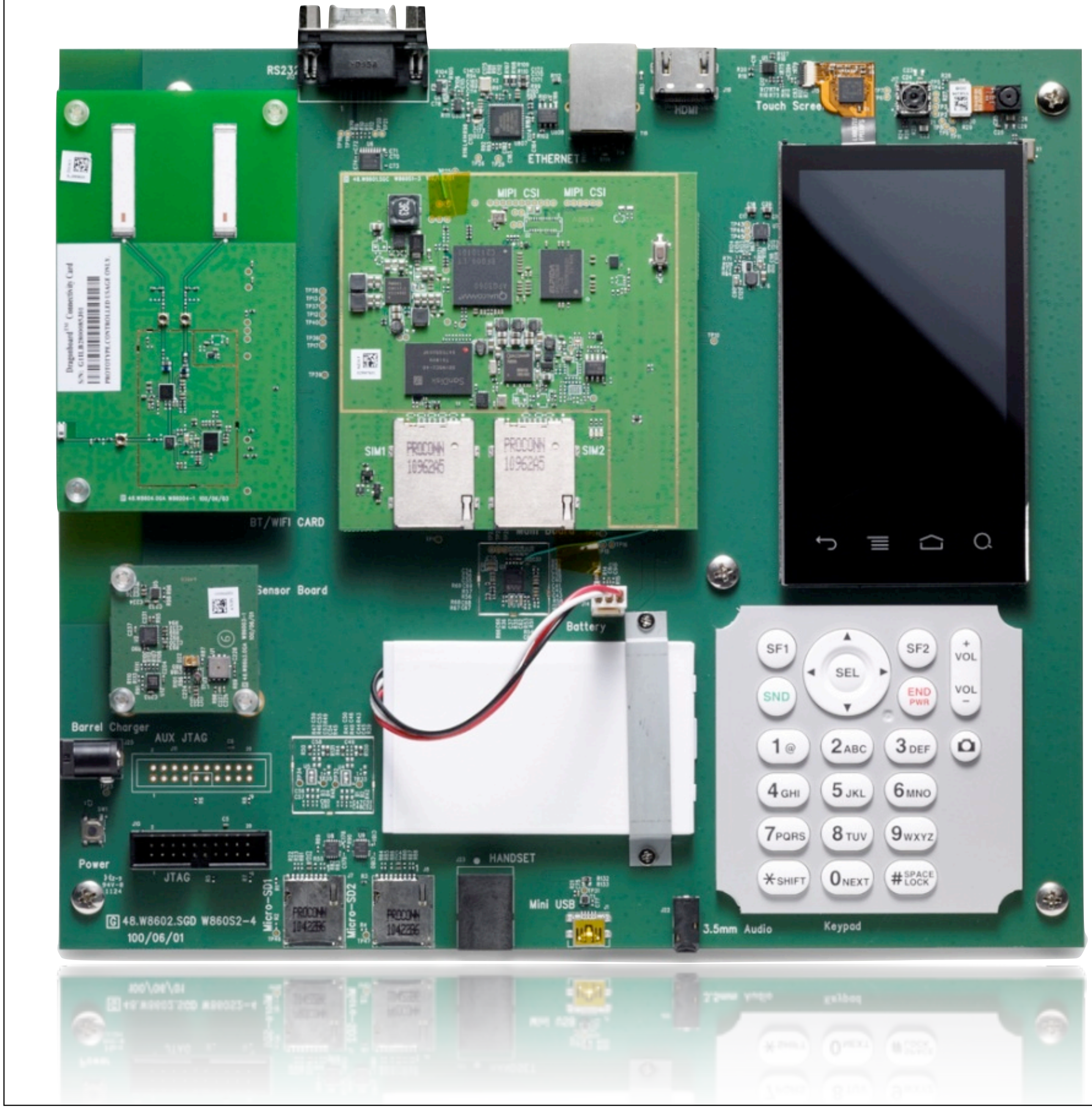
DragonBoard™



Sensor Card

Pressure
Temperature
Compass
Accelerometer
Gyroscope

DragonBoard™

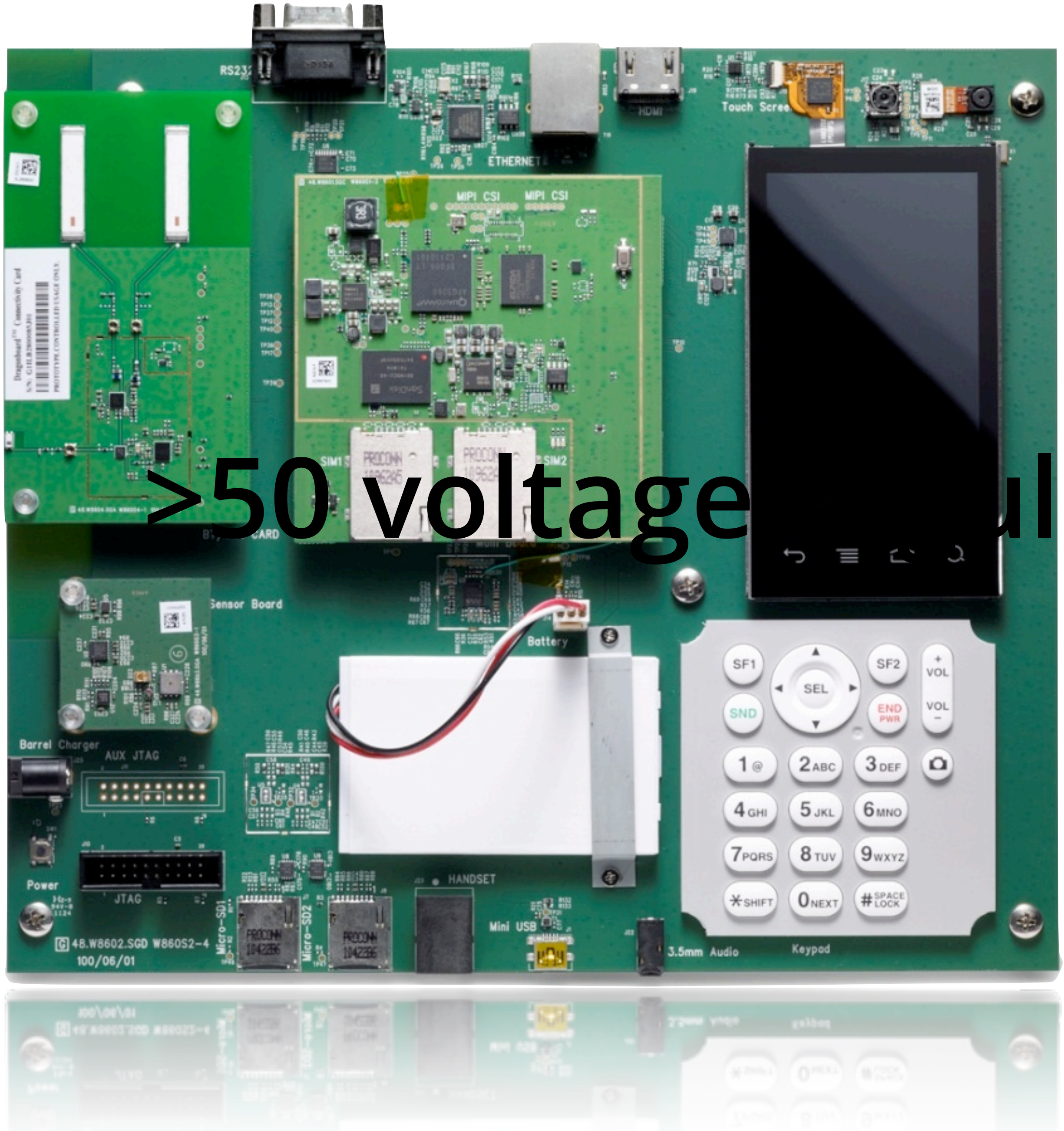


Also

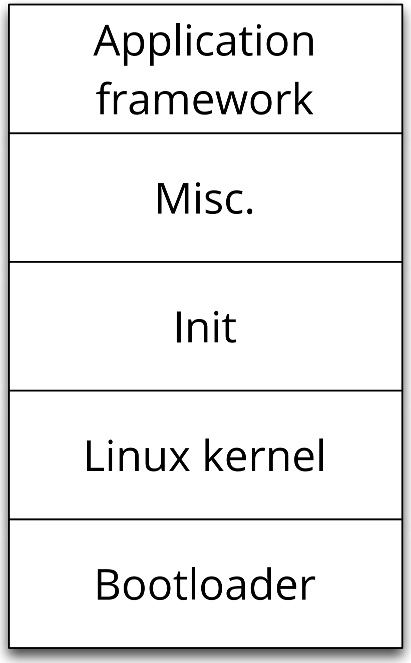
Ethernet
5 SDIO
173 GPIO
I2C
SPI
UART
USB OTG
I2S
PCM

DragonBoard™

>50 voltage regulators



DragonBoard™



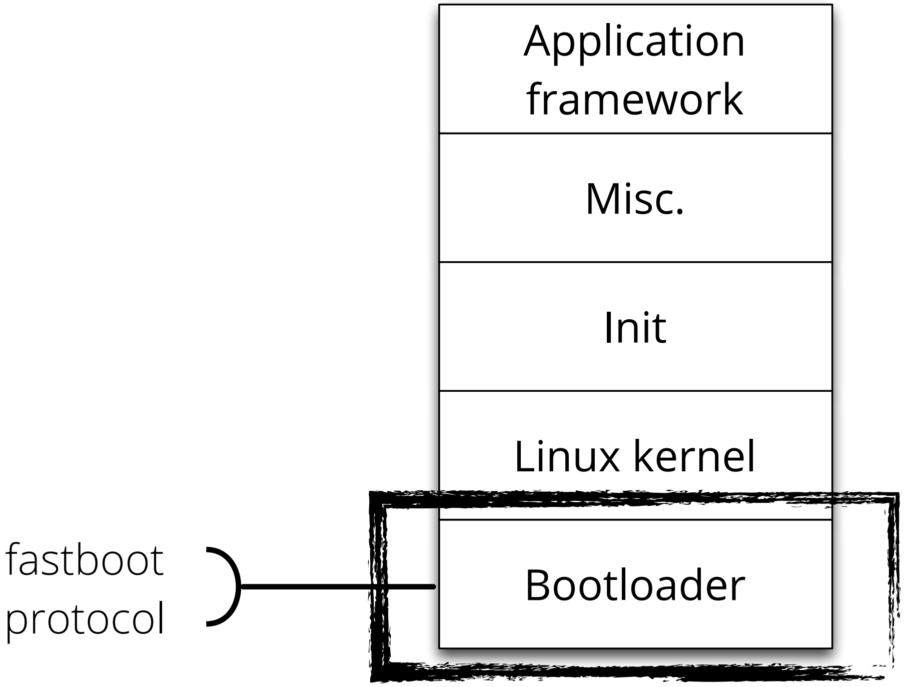
Application framework

Misc.

Init

Linux kernel

Bootloader



Application framework

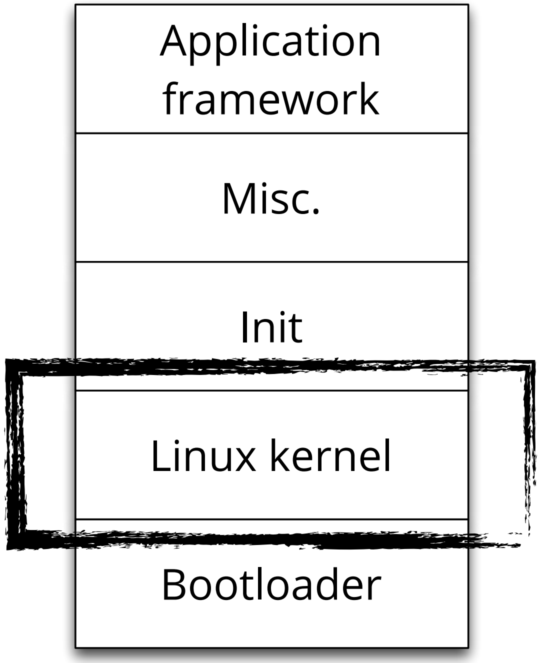
Misc.

Init

Linux kernel

Bootloader

fastboot protocol



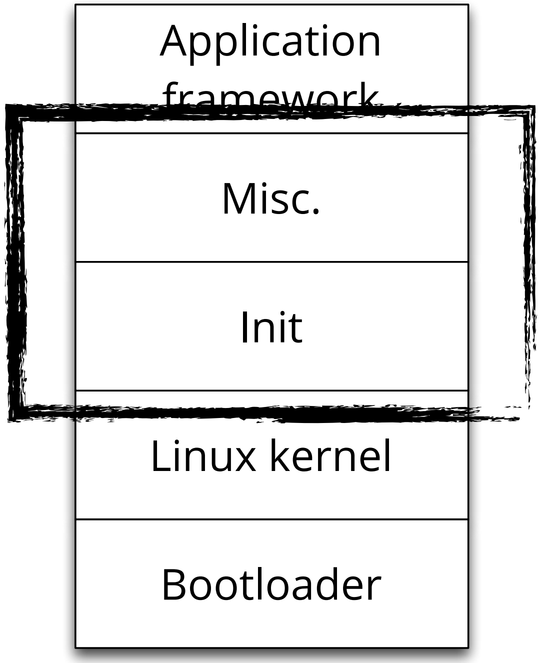
Application framework

Misc.

Init

Linux kernel

Bootloader



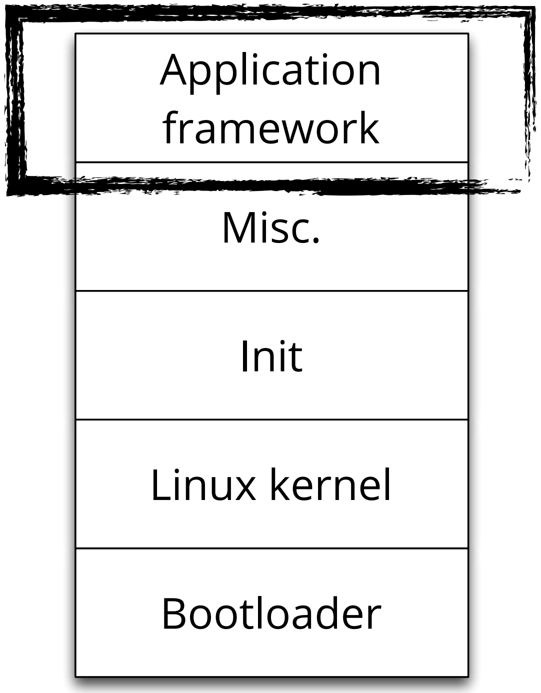
Application framework

Misc.

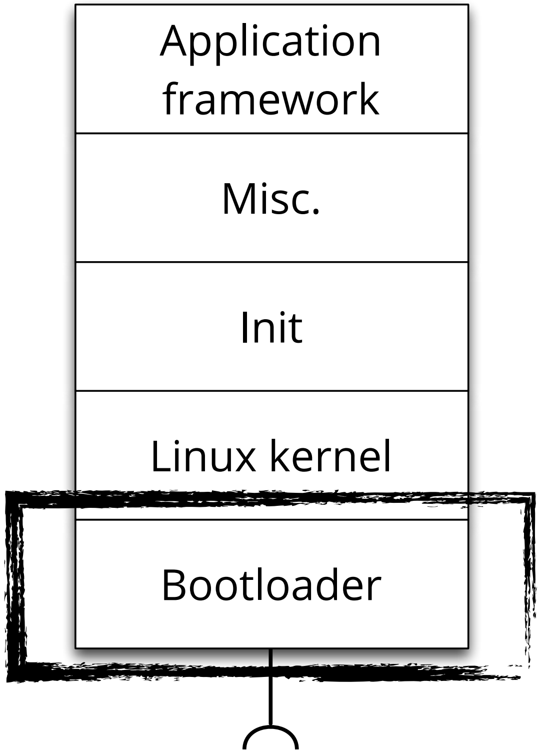
Init

Linux kernel

Bootloader



Application framework
Misc.
Init
Linux kernel
Bootloader



Application framework

Misc.

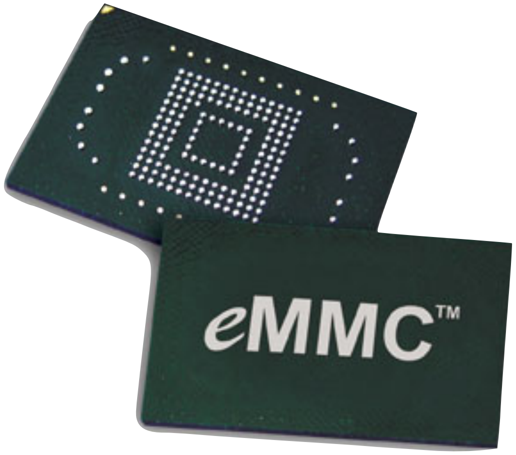
Init

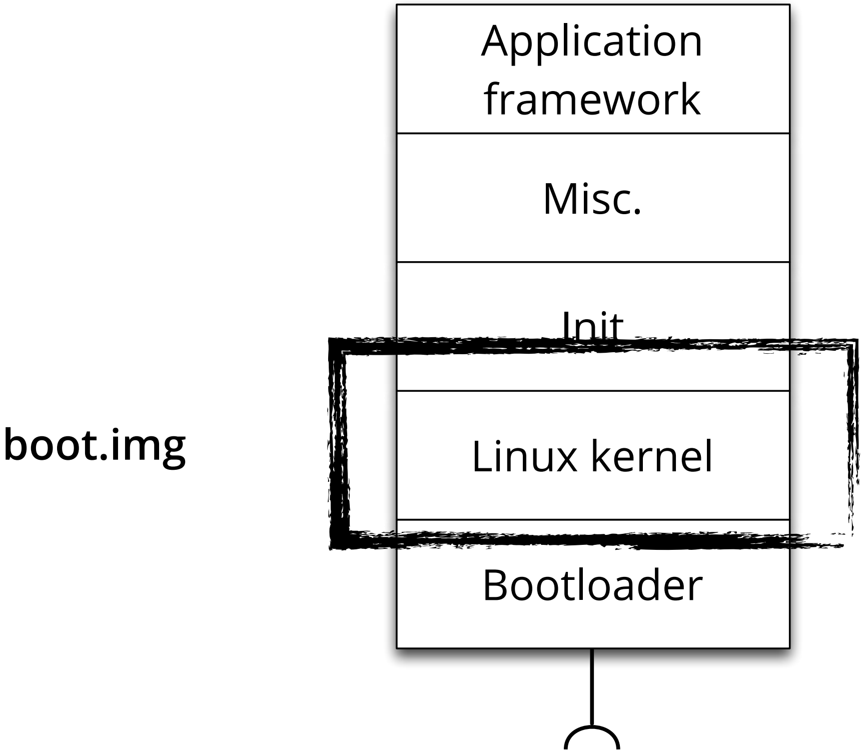
Linux kernel

Bootloader

Normally don’t touch this

# fastboot flash $IMAGE





Application framework

Misc.

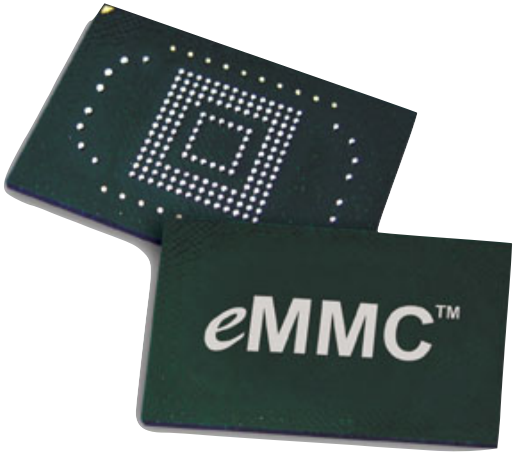
Init

boot.img

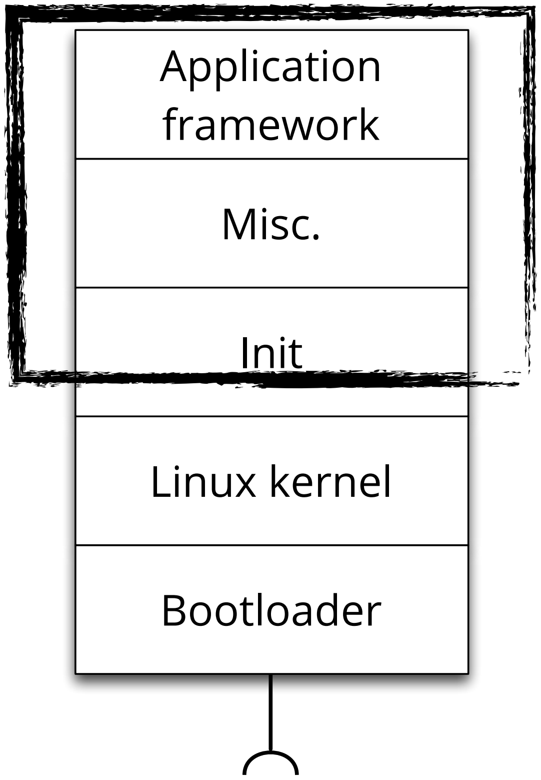
Linux kernel

Bootloader

# fastboot flash $IMAGE



system.img



Application framework

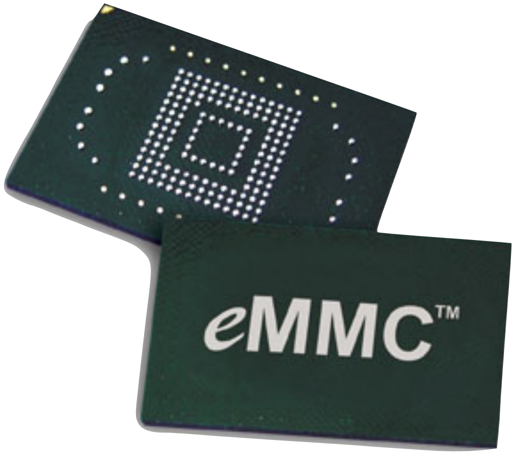
Misc.

Init

Linux kernel

Bootloader

```
# fastboot flash $IMAGE
```



DragonBoard™

myDragonBoard.org

BSQUARE

FORGING SMART CONNECTIONS

The challenges of DragonBoard

Powerful Hardware

Complex Software

Broad Applicability

DragonBoard™

The challenges of DragonBoard

Embedded systems are more than just

Android applications!

DragonBoard™

Limitations of the AOSP Android Framework

No real-time scheduling

Limited thread prioritization

`java.*`, `android.*`, or `your-own.*`

Android runs on Linux, but…

Android is not an operating system

No production-grade platform management

Limited software availability

Android is purpose-built

DragonBoard™

Android is purpose-built

If it brings what you need, great!

(Otherwise, not so much)

The right way to use Android

Let Android do what it does well

Don’t let Android do anything else!

DragonBoard™

The right way to use Android

Leave the rest to Linux, where it belongs

Connect the two at the right places

We need a split-personality workflow!

DragonBoard™

Pragmatux

DragonBoard™

Pragmatux

Adopted by Qualcomm for all APQ platforms

Road-tested embedded Linux operating system

Based on familiar FOSS tools and concepts

Open and extensible

DragonBoard™

Pragmatux

Workstation environment

Target device operating system

Deployment management system

Developer ecosystem

DragonBoard™

Pragmatux

Not “beta”

DragonBoard™

Pragmatux

A “Linux distribution”, but also much more

debian

emdebian

Pragmatux

Utilizes tools, concepts from the Debian Project

Isn’t “Debian”, but has a similar look and feel

Is suitable for any embedded Linux platform

DragonBoard™

Pragmatux

Builds on a decade of embedded Linux

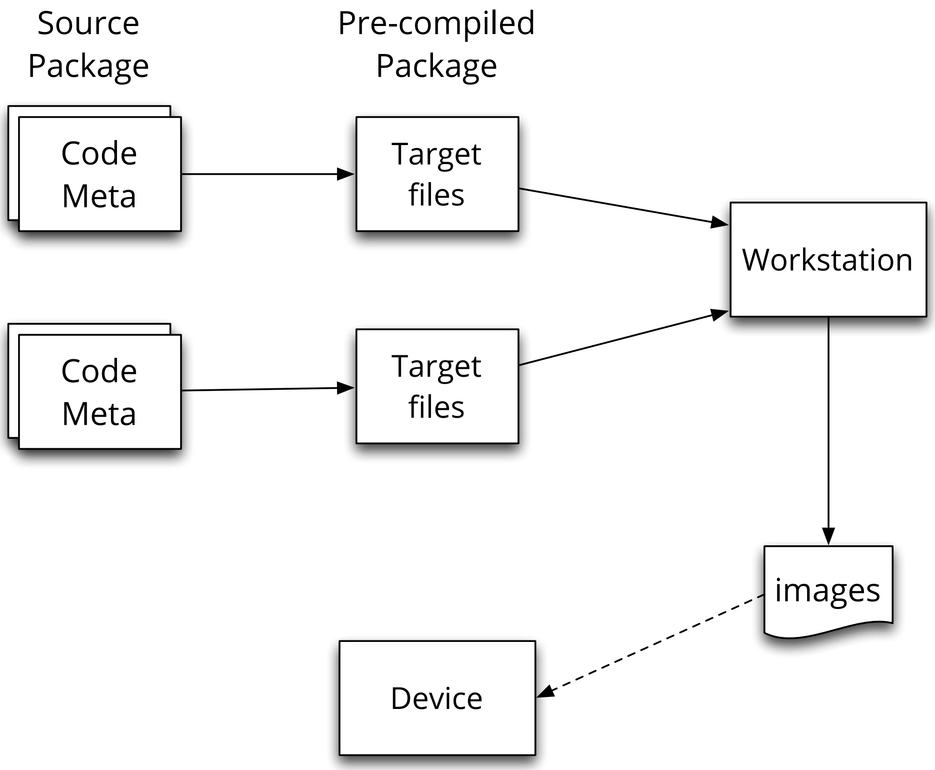
and Debian lessons-learned

Pragmatux

Best-practices, real-world embedded Linux

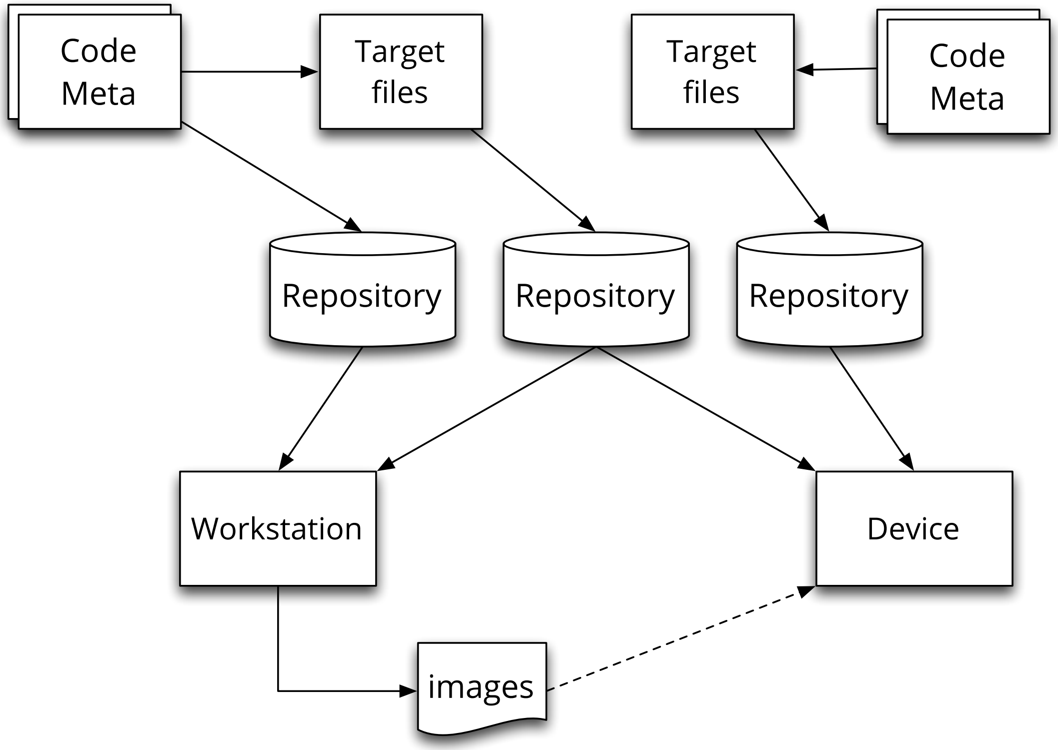
DragonBoard™

Fantasy Workflow



DragonBoard™

Pragmatux Workflow



DragonBoard™

Case study: kernel module update

DragonBoard™

Problem description:

Vendor issues a bugfix to binary-only module

Not all deployed devices have that kernel

Some devices with that kernel are no-touch

Other constraints limit applicability

DragonBoard™

We want:

Hands-free update mechanism

High-assurance of correctness

No risk of customer data loss

DragonBoard™

But also…

Some of the target devices aren’t networked!

DragonBoard™

This is exactly the scenario
Pragmatux was developed to solve

DragonBoard™

Ye Olde Skule way:

Wipe-and-reinstall the filesystem

DragonBoard™

Ye Olde Skule way:

(but only on the appropriate machines!)

(don’t forget to save, restore customer data!)

(don’t kick out the power cord!)

(don’t forget to schedule downtime!)

DragonBoard™

The Pragmatux way:

Update the kernel module “package” file

Push the new package to the repository

Target devices “phone home” for updates

Targets install package at next opportunity

DragonBoard™

What about “wrong” machines?

Package meta-data indicates compatibility

Repository structure limits distribution

(other machines can’t see the package)

… and disconnected machines?

Repository tools create crypto-signed manifest

Manifest file delivered by USB, carrier pigeon

Targets install package at next opportunity

No! It’s too good to be true!

How about a demonstration? :-)

DragonBoard™

Demonstrations

Set up a developer workstation

Build a target filesystem image

Install that image on DragonBoard

Boot!

DragonBoard™

Demonstrations

Update, distribute a package

(and a special surprise guest bug!)

DragonBoard™

Demonstrations

(follow along!)

DragonBoard™

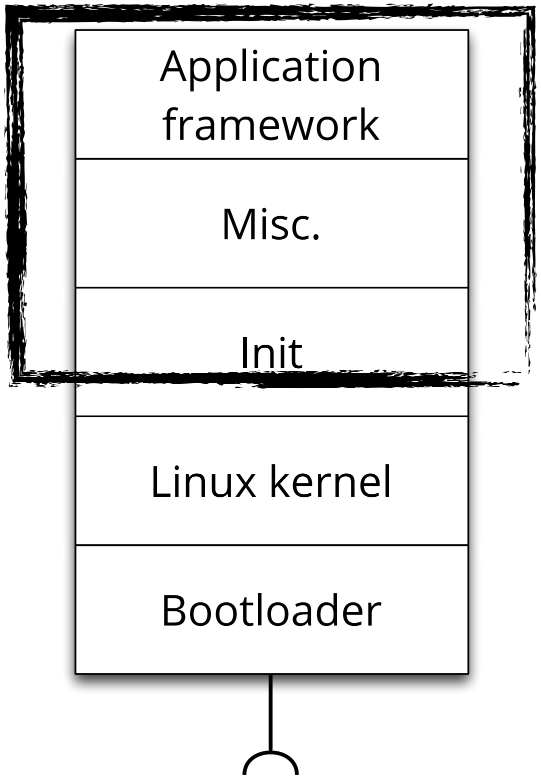
What about Android APKs?

They work for Android applications only!

(a lot of embedded Android isn’t Android)

DragonBoard™

system.img



Application framework

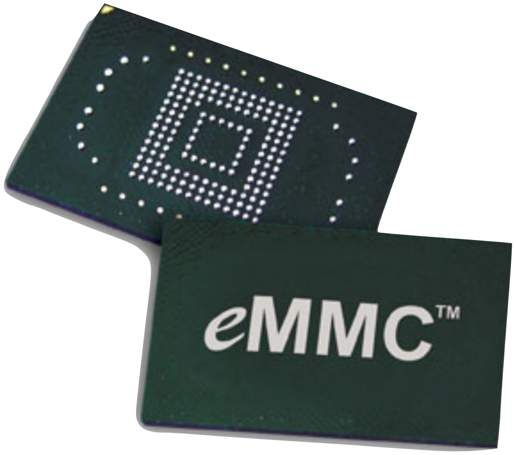
Misc.

Init

Linux kernel

Bootloader

# fastboot flash $IMAGE



Pragmatux isn’t rocket science

We are just using great tools well

… and we want you to do likewise!

DragonBoard™

Pragmatux brings the platform

… so you can focus on the problem

DragonBoard™