About Me

• Spent 25 years developing DoD and communications equipment
• Spent 4 years working with a research group developing a retinal implant
• Spent the last 5 years playing with robots
• Rocket Scientist
• Brain Surgeon
• Roboticist
• Linux Enthusiast
  • Running Linux since 1995
Gratuitous Star Wars Reference
Agenda

Then
• DoD Robots
• x86 Processors (mostly)
• Open Embedded
  • With iRobot layer on top

And (almost) Now
• Consumer Robots
• ARM processors
• Yocto & buildroot
  • With iRobot customizations
Packbot

• 2005
• x86 Pentium III
• Bluecat Linux
• Came up pretty easily
• Core problems:
  • RT characteristics
  • Terrible WiFi support
    — Hard to get AdHoc networking to work
One Linux to Rule them

• 2007 – 2010

• Multiple Hardware Platforms:
  • x86, ARM, PowerPC
  • Kernel was on a separate branch, not as well supported
  • Needed latest kernel
    – But older Wifi drivers needed older kernels

• Needed an embedded Linux Distribution

• "Common OS" project
  • IRobot Layer on top of Open Embedded
  • BSPs for various products
  • Common build system
  • Started with "OE Classic" migrated to Yocto/OE
Ava

• Remote Presence Robot
• Off-the-shelf x86 COM Express module
  • running Ubuntu 12.04 (and then 14.04)
• ROS-like Robotics layer
• LIDAR used for mapping
  • Connected to CPU via Ethernet
• UART connection to mobility module
Issues & Problems

• Compiler support was terrible
  • No scripts, no buildroot, had to buy cross compilers
• Backporting drivers is hard
• Various custom Linuxes for radios
• Boot time was minutes
  • Not good for military applications
• No good power management sleep/wake support
Then What?

• Defense Group was 100% Linux
• Home/Consumer Group was 100% NOT Linux
  • Minimal FLASH, RAM, processing power and cost!
Fast Forward to 2015

• Roomba 980 released
  • First product with vision based mapping
  • LPC3250 Processor from NXP
  • ARM9 SoC
  • 2 MByte FLASH
  • 16 Mbyte SDRAM
  • WiFi Connected via separate module

• New product developments considering SoCs such as the SAMA5 processor from Atmel
  • Cortex A5 SoC
  • 16 MByte FLASH (more on that in a bit)
  • 128 MByte SDRAM
  • WiFi Connected directly
New board `/bin/sh` in 8 days

- Received a board from the Electrical Designer on a Thursday afternoon in March
- Celebrated a "We got the prompt" party the following Friday
- Customized Atmel's `at91bootstrap` bootloader to support our FLASH and SDRAM memory configuration
- Minor tweaks to U-Boot
- No modifications to Linux source tree
  - Except, of course for our custom device tree
  - And a few bugfixes/enhancements submitted upstream
- Second board came up in 2 days
  - But that one only had a FLASH change
  - WiFi took longer
Application Development Model

• Develop, debug and test the application on the Desktop
  • Using standard driver models (v4l2, USB, audio, network stack, etc...)
  • Doesn't work so well for I2C, SPI, or GPIO devices

• Optional: Recompile natively on the target
  • Works if you have a native distribution such as Ubuntu running on the target

• Cross compile for the target
  • buildroot and Yocto help a lot here!
  • Debug with gdbserver
  • Can use USB networking, or even PPP/SLIP!

• Fight to keep your boot console!
  • Don't let the hardware design take that from you
    -- Perhaps adb can help here
Praise for (and a plea) to Chip Manufacturers

• SoC manufactures now maintain Linux kernels for their devices
  • And Yocto distributions as well
• Please work to get your kernel mainlined
• Please work to isolate your Yocto changes to a single meta package that can be dropped into the standard Yocto distribution
Going Forward

• More Cores
• More FLASH
• More SDRAM
• More Off-the-Shelf Software
  • Amazon Echo
  • Google Things
  • Stacks are provided, assume more resources
• GPL vs NDA
• Security, Security, Security
• STEM
Gratuitous iRobot Video
Thank you

We’re hiring! (Software, Cloud, Mobile, IoT…)
http://www.irobot.com/careers

iRobot Ventures: early stage investing program