Making a Splash: Digital Signage
Powered by the MinnowBoard MAX
and the Yocto Project

Nitin A Kamble & John Hawley
Open Source Technology Center
Intel Corporation
Agenda

- Digital Signage
- MinnowBoard MAX
- Yocto Project
Digital Signage

Background and Motivation

- Digital signs help us satisfy our need for information in our modern, data-driven lives.
- Digital signs are moving into retail businesses, and can offer **interactivity** to create improved engagement with customers.
- Digital signs have evolved beyond basic grids of information and now often include advanced user interface design.
Advantages:

- Users can quickly adapt to web-based interface styles
- Clean separation of back-end data architecture and UI design
- Large pool of talented web application engineers
- Web toolkits including JavaScript animation libraries are mature and have been optimized for performance
“Heads-up” Demo Concept
Some factors to consider when considering browser-based embedded platforms for digital signage:

- Strong CPU and graphics performance to run DHTML animations
- Boot disk options
- Level of custom integration required
- Tradeoff between hardware capabilities and engineering cost / time to market
Meet MinnowBoard MAX

OHAI!
Embedded Board Selection

- Strong CPU and graphics performance
- Offers x86 PC standards, multi-core options, and Gigabit Ethernet
- Media flexibility: can boot from microSD card, SATA disk, mSATA SSD, USB 3.0
- Save engineering/developer time with ability to run demanding applications (e.g., OpenCV using high level languages)
- Open Hardware design can be adapted or integrated into custom solutions
MinnowBoard MAX Specs

- $99: Bay Trail E3815 1.46 GHz (single-core, 64-bit, VT-x), 1 GB RAM
- $129: Bay Trail E3825 1.33 GHz (dual-core, 64-bit, VT-x), 2 GB RAM
- DDR3 RAM
- Intel HD Graphics (w/ open source Linux drivers)
- Micro-HDMI w/ HDMI audio
- 10/100/1000 Ethernet
- SATA2 3.0 Gbps, microSD via SDIO
- USB Host Ports: 1x USB 3.0, 1x USB 2.0
- 8 GPIO (2 support PWM), I2C, SPI, I2S Audio, 2 UARTS
- High-speed expansion connector w/ PCIe 1x, SATA2, USB 2.0 x1, JTAG
Yocto Project Introduction

- Flexible framework for developing Linux-based embedded OSes
- Centered around the OpenEmbedded build system – uses BitBake and recipe metadata
- Offers a powerful layer-based architecture to easily configure software stacks and swap BSPs
- Allows you to define your own Linux distribution and make policy decisions on how your embedded OS will function
- Supports all major architectures – ARM, x86(-64), MIPS, PPC
OpenEmbedded = BitBake + metadata

- **OpenEmbedded** – build system used by the Yocto Project
- **BitBake** – a task executor and scheduler
- **Metadata** – task definitions
  - **Configuration (*.conf)** – global definitions of variables
  - **Classes (*.bbclass)** – encapsulation and inheritance of build logic, packaging, etc.
  - **Recipes (*.bb)** – the logical units of software/images to build
Build System Workflow

Openembedded Architecture Workflow

Upstream Source
Metadata/Inputs
Build system

Output Packages
Process steps (tasks)
Output Image Data

Source Mirror(s)

Upstream Project Releases
Local Projects
SCMs (optional)

User Configuration
Metadata (.bb + patches)
Machine (BSP) Configuration
Policy Configuration

Source Fetching
Patch Application
Configuration / Compile / Autoreconf as needed

Output Analysis for package splitting plus package relationships
.rpm Generation
.deb Generation
.ipk Generation

QA Tests

Package Feeds

Image Generation
SDK Generation
Images
Application Development SDK
SUMMARY = "Display or change ethernet card settings"
DESCRIPTION = "A small utility for examining and tuning the settings of your ethernet-based network interfaces."
HOMEPAGE = "http://sourceforge.net/projects/gkernel/
LICENSE = "GPLv2+"
SRC_URI = "${SOURCEFORGE_MIRROR}/gkernel/ethtool-
            ${PV}.tar.gz"

inherit autotools
Building recipes involves executing the following functions, which can be overridden when needed for customizations:

- `do_fetch`
- `do_unpack`
- `do_patch`
- `do_configure`
- `do_compile`
- `do_install`
- `do_package`
OpenEmbedded Layers

- Developer-Specific Layer
- Commercial Layer (from OSV)
- UI-Specific Layer
- Hardware-Specific BSP
- Yocto-Specific Layer Metadata (meta-yocto)
- OpenEmbedded Core Metadata (oe-core)
The meta-web-kiosk Layer

Includes an image recipe `core-image-web-kiosk.bb` which:

- Boots up to Xorg
- Starts the Midori web browser in fullscreen mode
- Goes to a configured URL

To discover more layers and recipes you can use with the Yocto Project, take a look at the OpenEmbedded metadata index:

`http://layers.openembedded.org`
The Heads-Up Demo Custom Layer

Includes heads-up-demo-image.bb which contains:

- RFID support
- WebCam support
- open cv support
- x264 & webm codecs support
Designing for Security

- Digital signs that display remotely-hosted data should be hardened against potential attacks
- Use cryptographic network protocols – HTTPS, TLS, SSH – certificate validation is your friend
- Custom handshake protocols can raise the bar but amount to “security through obscurity”
- Ensure there is sensible degradation behavior if the security tests fail or the content server is not available
Try to Avoid This...
Summary

- Digital Signage Solution
  - Web based, Interactive, Media rich

- MinnowBoard MAX
  - Meets the heavy computing demand
  - Improves time to market by using readily available s/w
  - Open Hardware Design

- Yocto Project
  - Create your own Embedded Linux solution
  - Optimal reuse because of the Layered architecture
Resources

- Yocto Project Home: http://yoctoproject.org
- Web-kiosk Layer for OpenEmbedded: http://layers.openembedded.org/layerindex/branch/master/layer/meta-web-kiosk/
- MinnowBoard MAX Announcement: http://www.minnowboard.org/meet-minnowboard-max/
- Meta-heads-up demo layer: http://git.yoctoproject.org/cgit/cgit.cgi/poky-contrib
Thank you!

Questions?