Why the Yocto Project for my IoT Project?

Deploy Software Updates for Linux Devices

Drew Moseley
Technical Solutions Architect
Mender.io
Session overview

- Motivation
- Challenges for Embedded, Linux and IoT developers
- Describe IoT workflow
- Overview of Yocto
- Benefits of Linux and Yocto for IoT
Drew Moseley

- 10 years in Embedded Linux/Yocto development.
- Longer than that in general Embedded Software.
- Project Lead and Solutions Architect.

Mender.io

- Over-the-air updater for Embedded Linux
- Open source (Apache License, v2)
- Dual A/B rootfs layout (client)
- Remote deployment management (server)
- Under active development
Embedded Projects increasingly use Linux:
- AspenCore/Linux.com\(^1\): Embedded Linux top 2 in current and planned use.

Huge IoT market opportunity:
- Forbes\(^2\): $267B by 2020

Linux is a big player in IoT
- Nodes & Gateways\(^3\) - 17.18 Billion units by 2023
- Inexpensive prototyping hardware - Raspberry Pi, Beaglebone, etc
- Readily available production hardware - Toradex, Variscite, Boundary Devices
- Wide selection of chipsets - NXP, TI, Microchip, Nvidia

\(^3\) http://www.marketsandmarkets.com/PressReleases/iot-gateway.asp
Challenges for Embedded Linux/IoT Developers

Hardware variety

Storage Media

Software may be maintained in forks

Cross development

Initial device provisioning
1. Buy Hardware¹

¹https://makezine.com/comparison/boards/
1. Buy Hardware¹
2. Connect Hardware

¹https://makezine.com/comparison/boards/
1. **Buy Hardware**
2. **Connect Hardware**
3. **Install OS**
   a. Binary distribution
   b. Distribution build tools
   c. Build system

1. https://makezine.com/comparison/boards/
1. Buy Hardware
2. Connect Hardware
3. Install OS
   a. Binary distribution
   b. Distribution build tools
   c. Build system
4. Develop Application
5. Test Application

1 https://makezine.com/comparison/boards/
1. Buy Hardware
2. Connect Hardware
3. Install OS
   a. Binary distribution
   b. Distribution build tools
   c. Build system
4. Develop Application
5. Test Application
6. Test System

https://makezine.com/comparison/boards/
Getting Started Guide for Embedded/IoT Development

1. Buy Hardware
2. Connect Hardware
3. Install OS
   a. Binary distribution
   b. Distribution build tools
   c. Build system
4. Develop Application
5. Test Application
6. Test System
7. Deploy

1 https://makezine.com/comparison/boards/
Getting Started Guide for Embedded/IoT Development

1. Buy Hardware
2. Connect Hardware
3. Install OS
   a. Binary distribution
   b. Distribution build tools
   c. Build system
4. Develop Application
5. Test Application
6. Test System
7. Deploy

8. PROFIT!!!

1 https://makezine.com/comparison/boards/
Is

- Mechanism to specify and build
  - Define hardware/BSP components
  - Integrate user-space applications; including custom code
- Need reproducibility
- Must support multiple developers
- Allow for parallel processing
- (Cross) Toolchains
- License Management

Is Not

- An IDE
- A Distribution
- A deployment and provisioning tool
- An out-of-the-box solution
Yocto Project - Overview

“It’s not an embedded Linux distribution -- it creates a custom one for you”¹

- Recipes, metadata, dependencies and configuration
- Primary output: package feed
- Secondary output: boot images
- Builds all components from source
- Mechanism, not policy

Products:
- Root filesystem image
- Kernel, Bootloader, Toolchain
- Package Feed

¹See more at https://www.yoctoproject.org
Organized into independent layers:
- Separation of functionality
- Allows different release schedules
- Expandability
  - Recipes developed in python and bash

SDK mechanism
- Separation of system and application devs
- Easily allows multiple developers to contribute

Optimizations:
- Faster build time reusing prebuilt binaries
- Parallel builds

Previous ELC talk estimated ~ 8400 software packages
Yocto Project - Getting Started

$ git clone -b rocko \ git://git.yoctoproject.org/poky.git
$ source poky/oe-init-build-env
$ MACHINE=qemux86 bitbake \ core-image-minimal
$ runqemu qemux86
Why Linux for Embedded (1/2)?

- Ubiquity of:
  - Available Software
  - Expertise
  - Training Materials
- Broad device support
- Support for most common SOCs and reference boards
- Common system architecture on host and target
- Open Source
Why Linux for Embedded (2/2)?

- Industry Support
  - Semiconductor manufacturers
  - Industry groups (Genivi, AGL, etc)
- High performance/low power
- Connectivity options
- Good cross-development support
  - Toolchains, including debug support
  - Serial consoles
  - JTAG
Why Yocto for IoT (1/2)?

- Strong support from major board and semiconductor vendors.
- Well defined workflow.
- Supports large developer teams.
- Easy access to IoT-specific protocols.
- Good connectivity and coexistence with RTOSes used in sensors and actuators.
- CVE updates in upstream stable branches.
Why Yocto for IoT (2/2)?

- **IOT-Specific protocols well supported:**
  - MQTT (python-paho-mqtt)
  - AMQP (rabbitmq)
  - CoAP (libcoap/python3-aiocoap)
  - ZeroMQ (zeromq)
  - GPS (gpsd)
  - Bluetooth (bluez5)

- **Networking**
  - systemd-networking
  - connman
  - NetworkManager

- **Developer frameworks:**
  - Python (meta-python layer)
  - NodeJS (meta-nodejs layer)
## Yocto IoT Layers

<table>
<thead>
<tr>
<th>Layer name</th>
<th>Description</th>
<th>Type</th>
<th>Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td>meta-refkit</td>
<td>IoT Reference OS Kit for Intel(r) Architecture distro layer</td>
<td>Distribution</td>
<td><a href="https://github.com/intel/intel-iot-refkit">https://github.com/intel/intel-iot-refkit</a></td>
</tr>
<tr>
<td>meta-flatpak</td>
<td>Flatpak support layer</td>
<td>Miscellaneous</td>
<td><a href="https://github.com/intel/intel-iot-refkit">https://github.com/intel/intel-iot-refkit</a></td>
</tr>
<tr>
<td>meta-refkit-computervision</td>
<td>IoT Reference OS Kit for Intel(r) Architecture profile for computer vision use cases</td>
<td>Miscellaneous</td>
<td><a href="https://github.com/intel/intel-iot-refkit">https://github.com/intel/intel-iot-refkit</a></td>
</tr>
<tr>
<td>meta-refkit-core</td>
<td>Core layer for the IoT Reference OS Kit for Intel(r) Architecture</td>
<td>Miscellaneous</td>
<td><a href="https://github.com/intel/intel-iot-refkit">https://github.com/intel/intel-iot-refkit</a></td>
</tr>
<tr>
<td>meta-refkit-extra</td>
<td>IoT Reference OS Kit for Intel(r) Architecture demo layer</td>
<td>Miscellaneous</td>
<td><a href="https://github.com/intel/intel-iot-refkit">https://github.com/intel/intel-iot-refkit</a></td>
</tr>
<tr>
<td>meta-refkit-industrial</td>
<td>IoT Reference OS Kit for Intel(r) Architecture profile for industrial use cases</td>
<td>Miscellaneous</td>
<td><a href="https://github.com/intel/intel-iot-refkit">https://github.com/intel/intel-iot-refkit</a></td>
</tr>
<tr>
<td>meta-intel-iot-middleware</td>
<td>Shared middleware recipes for Intel IoT platforms</td>
<td>Software</td>
<td>git://git.yoctoproject.org/meta-intel-iot-middleware</td>
</tr>
<tr>
<td>meta-iot-cloud</td>
<td>OpenEmbedded layer to add support for multiple cloud service provider solutions.</td>
<td>Software</td>
<td><a href="https://github.com/intel-iot-devkit/meta-iot-cloud.git">https://github.com/intel-iot-devkit/meta-iot-cloud.git</a></td>
</tr>
<tr>
<td>meta-iot-web</td>
<td>IoT web components</td>
<td>Software</td>
<td><a href="https://github.com/intel/iot-web-layers">https://github.com/intel/iot-web-layers</a></td>
</tr>
<tr>
<td>meta-oic</td>
<td>Support for building the Open Interconnect Consortium IoT Ready framework</td>
<td>Software</td>
<td>git://git.yoctoproject.org/meta-oic</td>
</tr>
<tr>
<td>meta-security-framework</td>
<td>adds higher-level security middleware and tools</td>
<td>Software</td>
<td><a href="https://github.com/01org/meta-intel-iot-security.git">https://github.com/01org/meta-intel-iot-security.git</a></td>
</tr>
<tr>
<td>meta-security-smack</td>
<td>adds the Smack LSM to OpenEmbedded distros</td>
<td>Software</td>
<td><a href="https://github.com/01org/meta-intel-iot-security.git">https://github.com/01org/meta-intel-iot-security.git</a></td>
</tr>
</tbody>
</table>

Source: https://layers.openembedded.org/layerindex/branch/master/layers/
Yocto Project for IoT; in summary

Pros:
- Widely supported by board and semiconductor vendors
- Active developer community
- Wide functionality and board support enabled by layer mechanism
- Customizable and expandable
- Minimal native tooling required
- Predictability of software contents

Cons:
- Steep learning curve
- Unfamiliar environment to non-embedded developers
- Resource-intensive
  - Long initial build times
  - Disk space
Q&A - Thank you!

Resources:

- [https://ubm.io/2Iazdfn](https://ubm.io/2Iazdfn) - Deeper dive into the Yocto project
- [https://hub.mender.io/t/raspberry-pi-3-model-b-b/57](https://hub.mender.io/t/raspberry-pi-3-model-b-b/57) - Building Yocto for Raspberry Pi with Mender.

@drewmoseley
drew.moseley@mender.io