Introduction to OpenEmbedded

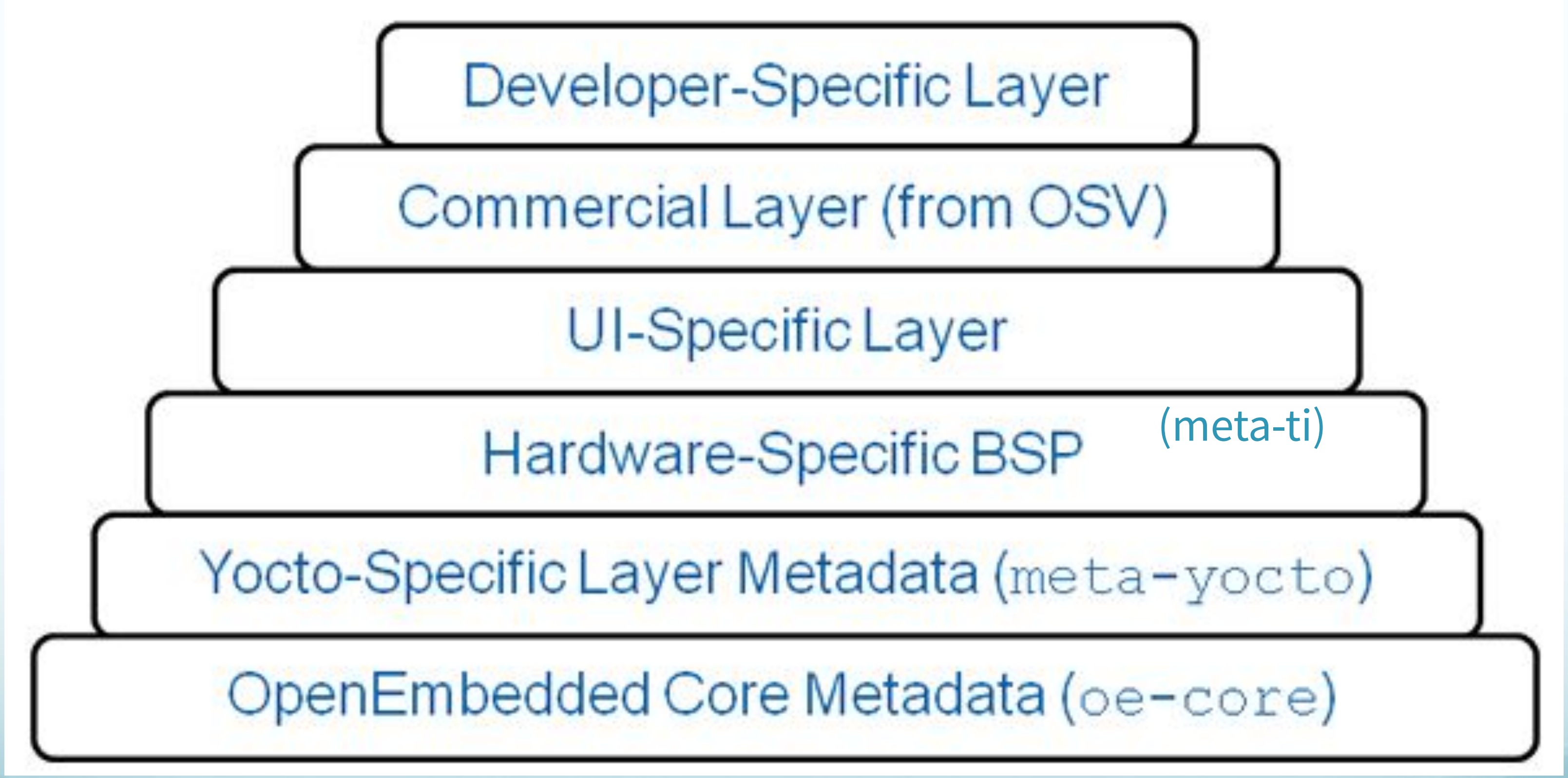
Joel Fernandes
www.LinuxInternals.org
joel@linuxinternals.org

OpenEmbedded

- A build system that can be used to build simple to complex embedded Distributions.
- Fully self-hosting cross-compiler environment!
- Fetches, configures, patches, builds and packages common open source projects. Fully automatic!
- Takes care of versioning and dependencies for you, no hassle!
- Typical build takes 2-3 hours, very fast!

OpenEmbedded

Concept: **Layers** in an OpenEmbedded-built distribution.
Each layer has collection of **recipes** for building software programs.



Developer-Specific Layer

Commercial Layer (from OSV)

UI-Specific Layer

Hardware-Specific BSP (meta-ti)

Yocto-Specific Layer Metadata (`meta-yocto`)

OpenEmbedded Core Metadata (`oe-core`)

OpenEmbedded

Concepts:

- **Recipes** describe how to build a particular program like busybox
- Each **Layer** has Recipes, recipes in upper-layers override lower ones
- A **package** is the output of a recipe, can have dependencies
- A package is **installed** onto a file system
- A **distribution** is a collection of **packages** installed together
- bitbake controls and builds everything in OpenEmbedded

OpenEmbedded

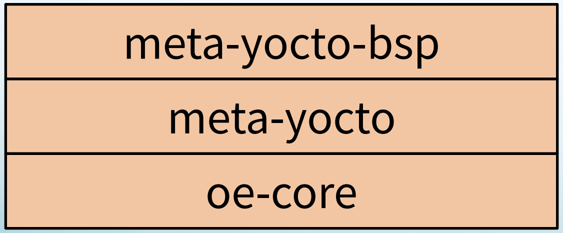
Poky Distribution

- **Part of the Yocto project:**

  From yoctoproject.org:

  Why use the Yocto Project? It's a complete embedded Linux development environment with tools, metadata, and documentation - everything you need.

Provides Layers which you can build on top of and build a **custom** distribution (next slide) :



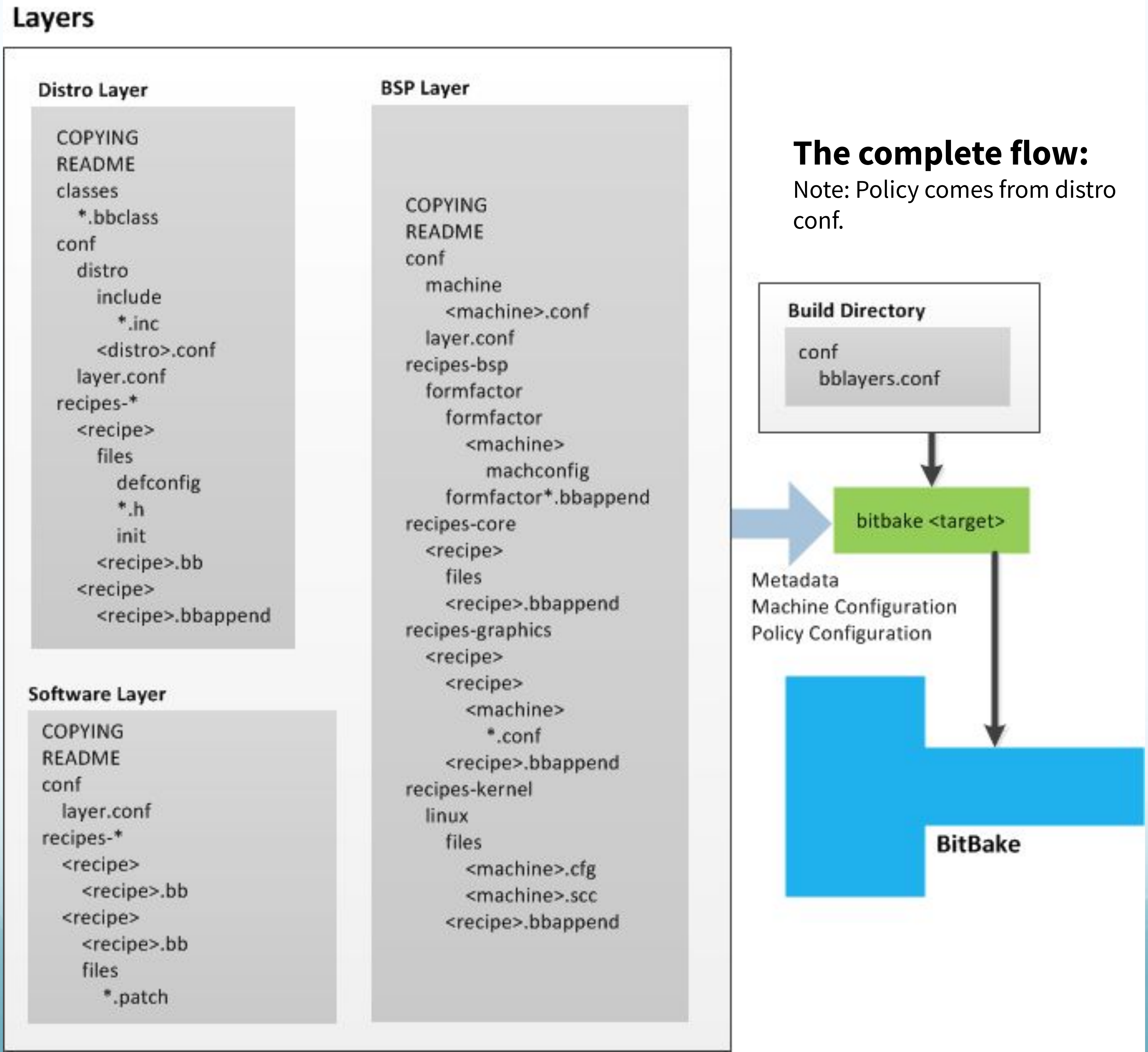
meta-yocto-bsp

meta-yocto

oe-core

The complete flow:

Note: Policy comes from distro conf.



Layers

Distro Layer

```
COPYING
README
classes
  *.bbclass
conf
  distro
    include
      *.inc
    <distro>.conf
  layer.conf
recipes-*
  <recipe>
    files
      defconfig
      *.h
      init
    <recipe>.bb
  <recipe>
    <recipe>.bbappend
```

Software Layer

```
COPYING
README
conf
  layer.conf
recipes-*
  <recipe>
    <recipe>.bb
  <recipe>
    <recipe>.bb
    files
      *.patch
```

BSP Layer

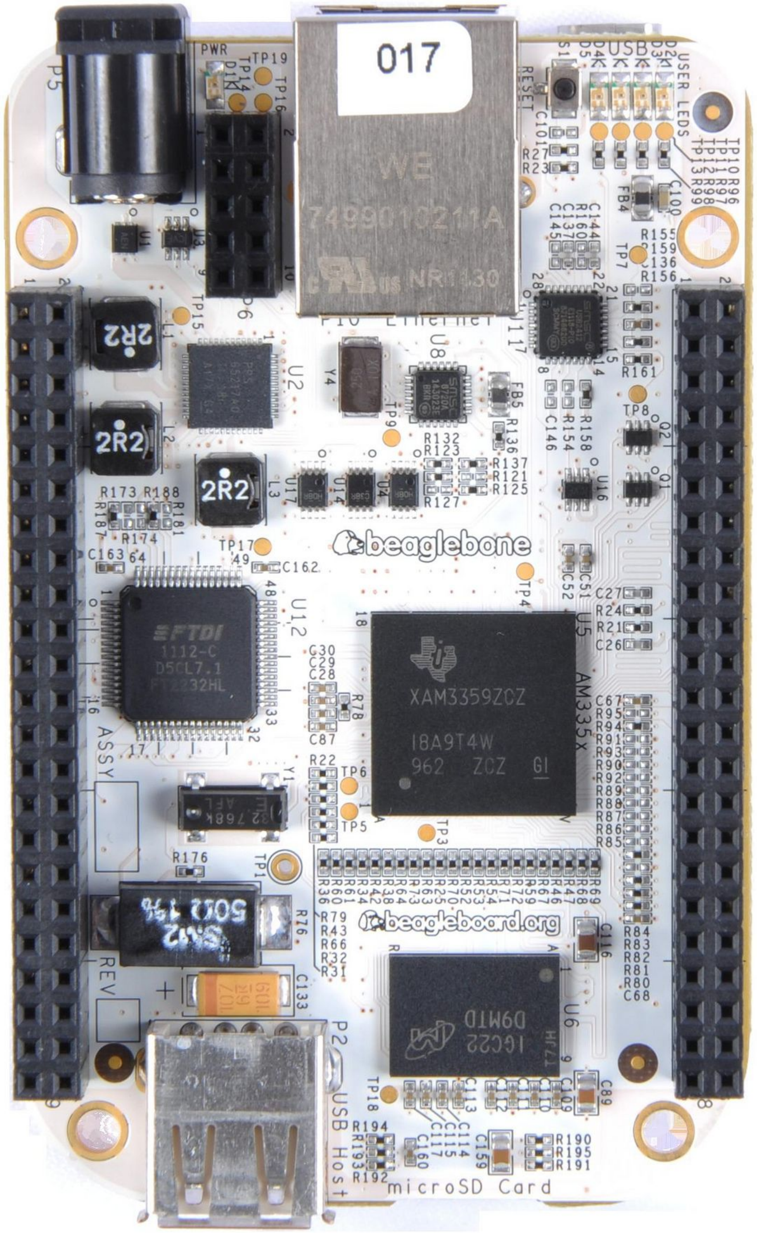
```
COPYING
README
conf
  machine
    <machine>.conf
  layer.conf
recipes-bsp
  formfactor
    formfactor
      <machine>
        machconfig
    formfactor*.bbappend
recipes-core
  <recipe>
    files
    <recipe>.bbappend
recipes-graphics
  <recipe>
    <recipe>
      <machine>
        *.conf
    <recipe>.bbappend
recipes-kernel
  linux
    files
      <machine>.cfg
      <machine>.scc
    <recipe>.bbappend
```

Build Directory

```
conf
  bblayers.conf
```

OpenEmbedded Demo: Beaglebone

- Introducing beaglebone
- Arago Linux distribution (built using OE) – we will be building another distribution called “Poky”
- 256MB DDR
- Cortex-A8 x1 720MHz
- USB, Ethernet, I2C, SPI etc.
- $89 (newer one costs $45)



OpenEmbedded

This is how I setup my sources for building the poky distribution for Beaglebone

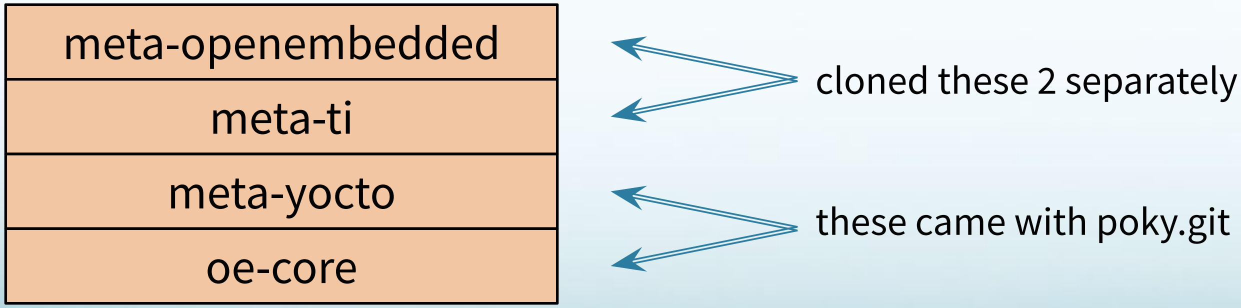
```
git clone -b jethro git://git.yoctoproject.org/poky.git poky-jethro

cd poky-jethro

git clone git://git.openembedded.org/meta-openembedded

git clone git://git.yoctoproject.org/meta-ti
```

We will be building a custom poky with the follow layers:



meta-openembedded

meta-ti

meta-yocto

oe-core

cloned these 2 separately

these came with poky.git

OpenEmbedded

Initialize the build and setup the environment

```
joel@joelbox:~/repo/poky-jethro$ source oe-init-build-env build

You had no conf/local.conf file. This configuration file has therefore been
created for you with some default values. You may wish to edit it to use a
different MACHINE (target hardware) or enable parallel build options to take
advantage of multiple cores for example. See the file for more information as
common configuration options are commented.

You had no conf/bblayers.conf file. The configuration file has been created for
you with some default values. To add additional metadata layers into your
configuration please add entries to this file.
```

OpenEmbedded

Main configuration files

- build/conf/local.conf: Contains local user’s settings

```
MACHINE ?= "beaglebone”
```

- build/conf/bblayers.conf

```
# meta-openembedded and meta-ti added by Joel for vim
# Remember to check out a recent version into poky

BBLAYERS ?= " \
  /home/joel/repo/poky-jethro/meta \
  /home/joel/repo/poky-jethro/meta-yocto \
  /home/joel/repo/poky-jethro/meta-ti \
  /home/joel/repo/poky-jethro/meta-openembedded/meta-oe \
  "
BBLAYERS_NON_REMOVABLE ?= " \
  /home/joel/repo/poky-jethro/meta \
  /home/joel/repo/poky-jethro/meta-yocto
```

OpenEmbedded

Now let's build

```
joel@joelbox:~/repo/poky-jethro$ cd build/
joel@joelbox:~/repo/poky-jethro/build$ bitbake beaglebone-image
```

After a few hours..

Final Output:

```
joel@joelbox:~/repo/poky-jethro/build/tmp/deploy/images/beaglebone/

-rw-r--r-- Nov 12 22:20
beaglebone-image-beaglebone-20151112163703.rootfs.tar.gz
-rwxr-xr-x Nov 12 22:22 MLO
-rwxr-xr-x Nov 12 22:22 u-boot.img
```

OpenEmbedded

Programming an SD Card with the images just built:

Step 1: Use mkcard script by Graeme Gregory to create the SD card.

Step 2: Copy the binaries from build/tmp/deploy/ to the card.

```
cd build/tmp/deploy/
cp MLO /media/joel/boot/
cp u-boot.img /media/joel/boot/
tar -xvf beaglebone-image-beaglebone-20151112163703.rootfs.tar.gz \
  -C /media/joel/Angstrom/
```

OpenEmbedded

Time to see a demo of the result!

Typical commands to run:

Find out kernel version:

```
root@beaglebone:~# dmesg|grep -i linux
```

Run vim:

```
root@beaglebone:~# vi /tmp/file
```

Find CPU details:

```
root@beaglebone:~# cat /proc/cpuinfo
```

Find the distro name:

```
root@beaglebone:~# cat /etc/issue
```

Mount the boot partition:

```
root@beaglebone:~# mkdir bootpart
root@beaglebone:~# mount /dev/mmcblk0p1 bootpart/
```

OpenEmbedded

Demo: Turns out i2c-tools is missing in my image.

I found that the “OE-core” layer had a recipe for it under
meta/recipes-devtools/i2c-tools/ called “i2c-tools_3.1.2.bb”

Lets add it to our beaglebone-image recipe:
meta/recipes-extended/images/beaglebone-image.bb

Next rebuild beaglebone-image

```
cd build
bitbake beaglebone-image
```

Results are in:

```
cd tmp/deploy/images/beaglebone
tar —tf beaglebone-image-beaglebone.tar.gz
```

Lets copy it from the build machine:

```
scp
joel@192.168.0.101:/home/joel/repo/poky-jethro/build/tmp/deploy/images/beaglebone/beaglebone-image-beaglebone.tar.gz
```

OpenEmbedded

Testing OpenEmbedded images with Qemu

- Oe-core layer has a qemu ARM machine at:

meta/conf/qemuarm.conf

- Change MACHINE variable in your local.conf file
- Run again: bitbake beaglebone-image
- Now everything will be built for qemu ARM machine

OpenEmbedded

Build qemu emulator for your Host machine:

- To run the qemu image just built, build the qemu-native package, using: “bitbake qemu-native” . This builds the qemu-native package which will run on your build machine.
- qemu-native package install “qemu-arm” and “qemu-system-arm” native binaries in build/sysroots/x86-64/usr/bin/qemu-*

OpenEmbedded

- Qemu demo:

In your OpenEmbedded root directory, run:

```
./scripts/runqemu qemuarm -nographic
```