KOAN

embedded software engineering

Yocto Project and OpenEmbedded training

2-day session

Title	**Yocto Project and OpenEmbedded development training**
Overview	Understanding the Yocto Project Using it to build a root filesystem and run it on your target Writing and extending recipes Creating layers Integrating your board in a BSP Creating custom images Application development
Duration	**Two** days - 16 hours (8 hours per day). 40% of lectures, 60% of practical labs.
Trainer	Marco Cavallini m.cavallini (AT) koansoftware.com
Language	Oral lectures: English, Italian. Materials: English.
Audience	Companies and engineers interested in using the Yocto Project to build their embedded Linux system.
Prerequisites	**Knowledge of embedded Linux** as covered in our embedded Linux training (`http://koansoftware.com/en/content/linux-embedded-course`) **Knowledge and practice of Unix or GNU/Linux commands** People lacking experience on this topic should not attend this course.

Required equipment	**For public sessions** Everything is supplied by KOAN in public sessions except the PC. Participants must have **their own PC laptop computer** with: • USB3 port support (for Disk provided) We will use a 32GB USB3 disk to work with Lubuntu 12.04 (32 bit) • USB to power the target board • USB to connect the serial adapter (provided) • Ethernet connector (for communication with the target) • Wifi • **PC computers with valuable data must be backed up** before being used in our sessions. Some people have already made mistakes during our sessions and damaged work data. **For on-site sessions** please add the following • Video projector • **Connection to the Internet** (direct or through the company proxy).
Materials	Print and electronic copies of presentations and labs. Electronic copy of lab files.

Hardware

The hardware platform used for the practical labs of this training session is the **BeagleBone Black board**, which features:

- An ARM AM335x processor from Texas Instruments (Cortex-A8 based), 3D acceleration, etc.
- 512 MB of RAM
- 2 GB of on-board eMMC storage (4 GB in Rev C)
- USB host and device
- HDMI output
- 2 x 46 pins headers, to access UARTs, SPI buses, I2C buses and more.

Note:
Content and order of this agenda may slightly vary between sessions and will be determined by the participants and the specific needs of the class.

Day 1

Lecture - Introduction to embedded Linux build systems

- Overview of an embedded Linux system architecture
- Methods to build a root filesystem image
- Usefulness of build systems

Lecture - Overview of the Yocto Project and the Poky reference system

- Organization of the project source tree
- Building a root filesystem image using the Yocto Project
- Organization of the build output
- Flashing and installing the system image
- Configuring the build system
- Customizing the package selection
- Writing a minimal recipe
- Adding dependencies
- Development workflow with *bitbake*

Lab - First Yocto Project build

- Downloading the Poky reference build system
- Building a system image
- Building a cross-compilation toolchain
- Flashing and booting the image on the BeagleBone
- Configuring the BeagleBone to boot over NFS
- Learn how to use the `PREFERRED_PROVIDER` mechanism
- Writing a recipe for *nInvaders*
- Adding *nInvaders* to the final image

Day 2

Lecture - Writing recipes, layers and a BSP

- Writing a minimal recipe
- Adding dependencies
- Development workflow with *bitbake*
- Extending and overriding recipes
- Adding steps to the build process
- Learn about classes
- Debugging dependencies
- What layers are and where to find them
- Creating a layer
- Extending an existing BSP
- Adding a new machine
- Bootloaders
- Linux and the linux-yocto recipe
- Adding a custom image type
- Writing an image recipe
- Adding users/groups
- Adding custom configuration
- Writing and using package groups recipes

Lab - Adding a recipe and learning how to configure packages

- Learning how to configure packages
- Extending a recipe to add configuration files
- Using `ROOTFS_POSTPROCESS_COMMAND` to modify the final rootfs
- Studying package dependencies
- Learn how to write a layer and add the layer to the build
- Move *nInvaders* to the new layer
- Adding *nInvaders* to the custom image
- Writing a custom image recipe