Yocto Project and OpenEmbedded training

3-day session

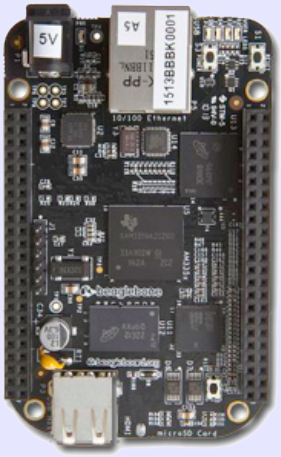
Title	**Yocto Project and OpenEmbedded development training**
Overview	Understanding the Yocto Project Using it to build a root filesystem and run it on your target Writing and extending recipes Creating layers Integrating your board in a BSP Creating custom images Application development with the Yocto Project SDK
Duration	**Three** days - 24 hours (8 hours per day). 40% of lectures, 60% of practical labs.
Trainer	One of the engineers listed on `https://bootlin.com/training/trainers/`
Language	Oral lectures: English, French. 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 (`https://bootlin.com/training/embedded-linux/`) **Knowledge and practice of Unix or GNU/Linux commands** People lacking experience on this topic should get trained by themselves, for example with our freely available on-line slides: `https://bootlin.com/blog/command-line/`

Required equipment	**For on-site sessions only.** Everything is supplied by Bootlin in public sessions. • Video projector • PC computers with at least 8 GB of RAM, a CPU at least equivalent to an Intel Core i5 and Ubuntu Linux installed in a **free partition of at least 40 GB. Using Linux in a virtual machine is not supported**, because of issues connecting to real hardware. • We need Ubuntu Desktop 16.04 (Xubuntu and other variants are fine). We don't support other distributions, because we can't test all possible package versions. • **High Speed Connection to the Internet** (direct or through the company proxy). • **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.
Materials	Print and electronic copies of presentations and labs. Electronic copy of lab files.

Hardware, first option

BeagleBone Black board

- 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.



Hardware, second option

STMicroelectronics STM32MP157A-DK1 Discovery board

- STM32MP157A (dual Cortex-A7) CPU from STMicroelectronics
- USB powered
- 512 MB DDR3L RAM
- Gigabit Ethernet port
- 4 USB 2.0 host ports
- 1 USB-C OTG port
- 1 Micro SD slot
- On-board ST-LINK/V2-1 debugger
- Arduino Uno v3-compatible headers
- Audio codec
- Misc: buttons, LEDs

Day 1 - Morning

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

Lab - First Yocto Project build

- Downloading the Poky reference build system
- Building a system image

Day 1 - Afternoon

Lecture - Using Yocto Project - basics

- Organization of the build output
- Flashing and installing the system image

Lab - Flashing and booting

- Flashing and booting the image on the board

Lecture - Using Yocto Project - advanced usage

- Configuring the build system
- Customizing the package selection

Lab - Using NFS and configuring the build

- Configuring the board to boot over NFS
- Learn how to use the `PREFERRED_PROVIDER` mechanism

Day 2 - Morning

Lecture - Writing recipes - basics

- Writing a minimal recipe
- Adding dependencies
- Development workflow with *bitbake*

Lab - Adding an application to the build

- Writing a recipe for *nInvaders*
- Adding *nInvaders* to the final image

Lecture - Writing recipes - advanced features

- Extending and overriding recipes
- Adding steps to the build process
- Learn about classes
- Analysis of examples
- Logging
- Debugging dependencies

Day 2 - Afternoon

Lab - Learning how to configure packages

- Extending a recipe to add configuration files
- Using `ROOTFS_POSTPROCESS_COMMAND` to modify the final rootfs
- Studying package dependencies

Lecture - Layers

- What layers are
- Where to find layers
- Creating a layer

Lab - Writing a layer

- Learn how to write a layer
- Add the layer to the build
- Move *nInvaders* to the new layer

Day 3 - Morning

Lecture - Writing a BSP	Lab - Implementing the kernel changes
• Extending an existing BSP • Adding a new machine • Bootloaders • Linux and the linux-yocto recipe • Adding a custom image type	• Extend the kernel recipe to add the nunchuk driver • Configure the kernel to compile the nunchuk driver • Play *nInvaders*

Day 3 - Afternoon

Lecture - Creating a custom image	Lab - Creating a custom image
• Writing an image recipe • Adding users/groups • Adding custom configuration • Writing and using package groups recipes	• Writing a custom image recipe • Adding *nInvaders* to the custom image
Lecture - Creating and using an SDK	**Lab - Experimenting with the SDK**
• Understanding the purpose of an SDK for the application developer • Building an SDK for the custom image	• Building an SDK • Using the Yocto Project SDK