Yocto Project and OpenEmbedded development training

Onsite training

Course duration

3 days – 24 hours

Language

Materials	English
Oral Lecture	English
	French
	Portuguese
	Italian

Trainer

One of the following engineers

- Alexandre Belloni
- Antonin Godard
- Jérémie Dautheribes
- João Marcos Costa
- Luca Ceresoli

Contact

training@bootlin.com

+33 484 258 097

bootlin.com

Audience

Companies and engineers interested in using the Yocto Project to build their embedded Linux system.

Training objectives

- Be able to understand the role and principle of an embedded Linux build system, and compare Yocto Project/OpenEmbedded to other tools offering similar functionality.
- Be able to configure and build basic embedded Linux system with Yocto, and install the result on an embedded platform.
- Be able to write and extend recipes, for your own packages or customizations.
- Be able to use existing layers of recipes, and create your own new layers.
- Be able to integrate support for your own embedded board into a BSP layer.
- Be able to create custom images.
- Be able to use the Yocto Project SDK to develop applications.
- Be able to use devtool to generate and modify recipes.

Prerequisites

- **Knowledge and practice of UNIX or GNU/Linux commands**: participants must be familiar with the Linux command line. Participants lacking experience on this topic should get trained by themselves, for example with our freely available on-line slides.
- **Minimal experience in embedded Linux development**: participants should have a minimal understanding of the architecture of embedded Linux systems: role of the Linux kernel vs. user-space, development of Linux user-space applications in C. Following Bootlin's Embedded Linux course allows to fulfill this pre-requisite.
- **Minimal English language level: B1**, according to the *Common European Framework of References for Languages*, for our sessions in English. See the CEFR grid for self-evaluation.

Pedagogics

- Lectures delivered by the trainer: 40% of the duration
- Practical labs done by participants: 60% of the duration
- Electronic copies of presentations, lab instructions and data files. They are freely available here.

Certificate

Only the participants who have attended all training sessions, and who have scored over 50% of correct answers at the final evaluation will receive a training certificate from Bootlin.

Disabilities

Participants with disabilities who have special needs are invited to contact us at *training@bootlin.com* to discuss adaptations to the training course.

Required equipement

For on-site session delivered at our customer location, our customer must provide:

- Video projector
- One PC computer on each desk (for one or two persons) with at least 16 GB of RAM, and Ubuntu Linux 24.04 installed in a free partition of at least 30 GB
- Distributions other than Ubuntu Linux 24.04 are not supported, and using Linux in a virtual machine is not supported.
- Unfiltered and fast connection to Internet: at least 50 Mbit/s of download bandwidth, and no filtering of web sites or protocols.
- PC computers with valuable data must be backed up before being used in our sessions.

For on-site sessions organized at Bootlin premises, Bootlin provides all the necessary equipment.

Hardware platform for practical labs

STM32MP1 Discovery Kit

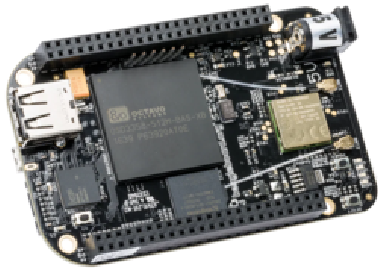
One of these Discovery Kits from STMicroelectronics: **STM32MP157A-DK1**, **STM32MP157D-DK1**, **STM32MP157C-DK2** or **STM32MP157F-DK2**

- STM32MP157, dual Cortex-A7 processor 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 compatible headers
- Audio codec, buttons, LEDs
- LCD touchscreen (DK2 kits only)

BeagleBone Black

BeagleBone Black or **BeagleBone Black Wireless** board

- An ARM AM335x (single Cortex-A8) processor from Texas Instruments
- USB powered
- 512 MB of RAM
- 2 or 4 GB of on-board eMMC storage
- USB host and device
- HDMI output
- 2 x 46 pins headers, to access UARTs, SPI buses, I2C buses and more.
- Ethernet or WiFi



BeaglePlay

BeaglePlay board

- Texas Instruments AM625x (4xARM Cortex-A53 CPU)
- SoC with 3D acceleration, integrated MCU and many other peripherals.
- 2 GB of RAM
- 16 GB of on-board eMMC storage
- USB host and USB device, microSD, HDMI
- 2.4 and 5 GHz WiFi, Bluetooth and also Ethernet
- 1 MicroBus Header (SPI, I2C, UART, ...), OLDI and CSI connector.

Training Schedule

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	Yocto Project and Poky reference system overview	▪ Introduction to the Yocto / OpenEmbedded build system and its lexicon ▪ Overview of the Poky reference system
Lecture	Using Yocto Project - basics	▪ Setting up the build directory and environment ▪ Configuring the build system ▪ Building a root filesystem image ▪ Organization of the build output
Lab	First Yocto Project build	▪ Downloading the Poky reference build system ▪ Configuring the build system ▪ Building a system image

Day 1 - Afternoon

Lab	Flashing and booting	▪ Flashing and booting the image on the board
Lecture	Using Yocto Project - advanced usage	▪ Variable assignment, operators and overrides ▪ Package variants and package selection ▪ bitbake command line options
Lab	Using NFS and configuring the build	▪ Configuring the board to boot over NFS ▪ Add a package to the root filesystem ▪ Learn how to use the `PREFERRED_PROVIDER` mechanism ▪ Get familiar with the bitbake command line options

Day 2 - Morning

Lecture	Writing recipes - basics	▪ Recipes: overview ▪ Recipe file organization ▪ Applying patches ▪ Recipe examples
Lab	Adding an application to the build	▪ Writing a recipe for *ninvaders* ▪ Troubleshooting the recipe ▪ Troubleshooting cross-compilation issues ▪ Adding *ninvaders* to the final image
Lecture	Writing recipes - advanced features	▪ Extending and overriding recipes ▪ Virtual packages ▪ Learn about classes ▪ BitBake file inclusions ▪ Debugging recipes ▪ Configuring BitBake network usage

Day 2 - Afternoon

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

Lab	Extend a recipe	▪ Extend the kernel recipe to add patches ▪ Configure the kernel to compile the nunchuk driver ▪ Edit the ninvaders recipe to add patches ▪ Play *ninvaders*
Lecture	Writing a BSP	▪ Introduction to BSP layers ▪ Adding a new machine ▪ Bootloader configuration ▪ Linux: the kernel bbclass and the linux-yocto recipe
Lab	Create a custom machine configuration	▪ Create a new machine configuration ▪ Build an image for the new machine
Lecture	Distro layers	▪ Distro configuration ▪ Distro layers

Day 3 - Afternoon

Lecture	Images	▪ Writing an image recipe ▪ Image types ▪ Writing and using package groups recipes
Lab	Create a custom image	▪ Add a basic image recipe ▪ Select the image capabilities and packages ▪ Add a custom package group ▪ Add an image variant for debugging
Lecture	Writing recipes - going further	▪ The per-recipe sysroot ▪ Using Python code in metadata ▪ Variable flags ▪ Packages features and PACKAGECONFIG ▪ Conditional features ▪ Package splitting ▪ Dependencies in detail
Lecture	Licensing	▪ Managing open source licenses
Lecture	The Yocto Project SDK	▪ Goals of the SDK ▪ Building and customizing an SDK ▪ Using the Yocto Project SDK
Lab	Develop your application in the Poky SDK	▪ Building an SDK ▪ Using the Yocto Project SDK
Lecture	Devtool	▪ About devtool ▪ Devtool use cases
Lab	Using devtool	▪ Generate a new recipe ▪ Modify a recipe to add a new patch ▪ Upgrade a recipe to a newer version

Lecture	Automating layer management	▪ Automating layer management
Lecture	Runtime Package Management	▪ Introduction to runtime package management ▪ Build configuration ▪ Package server configuration ▪ Target configuration