yocto PROJECT

Yocto Project and OpenEmbedded development training

Course duration

4 half days – 16 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
bootlin.com

Online seminar

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, over video-conference. Participants can ask questions at any time.
- Practical demonstrations done by the trainer, based on practical labs, over video-conference. Participants can ask questions at any time. Optionally, participants who have access to the hardware accessories can reproduce the practical labs by themselves.
- Instant messaging for questions between sessions (replies under 24h, outside of week-ends and bank holidays).
- 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

Mandatory equipment:

- Computer with the operating system of your choice, with the Google Chrome or Chromium browser for videoconferencing.
- Webcam and microphone (preferably from an audio headset).
- High speed access to the Internet.

Optionnally, if the participants want to be able to reproduce the practical labs by themselves, they must separately purchase the hardware platform and accessories, and must have a PC computer with a native installation of Ubuntu Linux 24.04.

Hardware platform for practical labs

STM32MP1 Discovery Kit

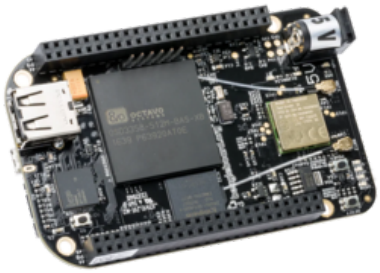
One of these Discovery Kits from STMicroelectronics: **STM32MP157A-DK1**, **STM32MP157D-DK1**, **STM32MP157C-DK2** or **STM32MP157F-DK2**

- 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

Half 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	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
Demo	First Yocto Project build	▪ Downloading the Poky reference build system ▪ Configuring the build system ▪ Building a system image
Demo	Flashing and booting	▪ Flashing and booting the image on the board

Half day 2

Lecture	Using Yocto Project - advanced usage	▪ Variable assignment, operators and overrides ▪ Package variants and package selection ▪ bitbake command line options
Demo	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
Lecture	Writing recipes - basics	▪ Recipes: overview ▪ Recipe file organization ▪ Applying patches ▪ Recipe examples
Demo	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

Half day 3

Lecture	Layers	▪ What layers are ▪ Where to find layers ▪ Creating a layer

Demo	Writing a layer	▪ Learn how to write a layer ▪ Add the layer to the build ▪ Move *ninvaders* to the new layer
Demo	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
Demo	Create a custom machine configuration	▪ Create a new machine configuration ▪ Build an image for the new machine
Lecture	Distro layers	▪ Distro configuration ▪ Distro layers

Half day 4

Lecture	Images	▪ Writing an image recipe ▪ Image types ▪ Writing and using package groups recipes
Demo	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
Demo	Develop your application in the Poky SDK	▪ Building an SDK ▪ Using the Yocto Project SDK
Lecture	Devtool	▪ About devtool ▪ Devtool use cases
Demo	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

Possible extra time

Extra time (up to 4 hours) may be proposed if the agenda didn't fit in 4 half days, according to the time spent answering questions from participants.