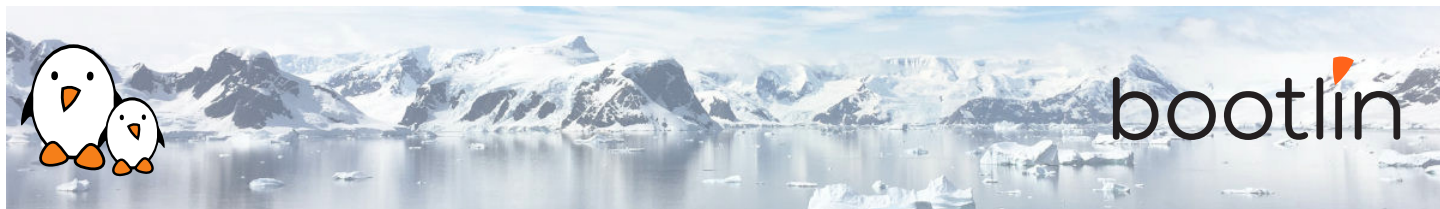


Yocto Project and OpenEmbedded development training

On-site training, 3 days

Latest update: May 17, 2024

Title	Yocto Project and OpenEmbedded development training
Training objectives	<ul style="list-style-type: none">• 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.
Duration	Three days - 24 hours (8 hours per day)
Pedagogics	<ul style="list-style-type: none">• 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 at https://bootlin.com/doc/training/yocto.
Trainer	One of the engineers listed on: https://bootlin.com/training/trainers/
Language	Oral lectures: English, French, Italian. Materials: English.
Audience	Companies and engineers interested in using the Yocto Project to build their embedded Linux system.



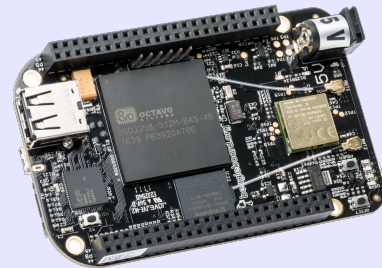
Prerequisites	<ul style="list-style-type: none"> • Knowledge and practice of UNIX or GNU/Linux commands: participants must be familiar with the Linux command line. Participants lacking experience on this topic should get trained by themselves, for example with our freely available on-line slides at bootlin.com/blog/command-line/. • 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 <i>Embedded Linux</i> course at bootlin.com/training/embedded-linux/ allows to fulfill this pre-requisite. • Minimal English language level: B1, according to the <i>Common European Framework of References for Languages</i>, for our sessions in English. See bootlin.com/pub/training/cefr-grid.pdf for self-evaluation.
Required equipment	<ul style="list-style-type: none"> • Video projector • One PC computer on each desk (for one or two persons) with at least an Intel Core i5 processor, 8 GB of RAM, and Ubuntu Linux 22.04 installed in a free partition of at least 100 GB • Distributions other than Ubuntu Linux 22.04 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.
Certificate	<p>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.</p>
Disabilities	<p>Participants with disabilities who have special needs are invited to contact us at training@bootlin.com to discuss adaptations to the training course.</p>



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.

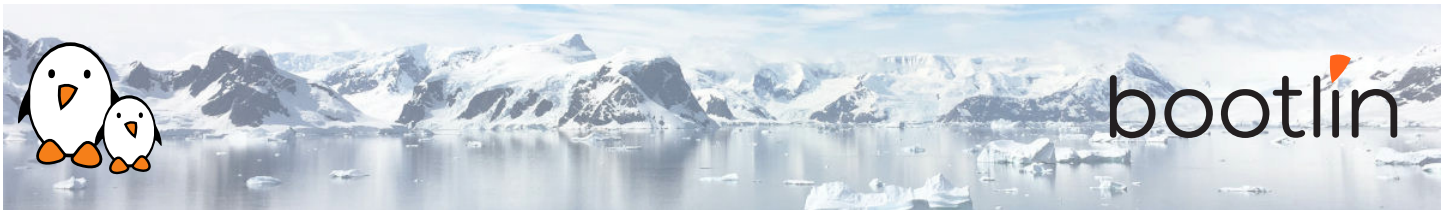


Hardware, second option

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

- STM32MP157 (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
- LCD touchscreen (DK2 kits only)





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

Lab 1 - First Yocto Project build

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

Day 1 - Afternoon

Lecture - Using Yocto Project - basics

- Organization of the build output

Lab 1 - 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 2 - 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 3 - 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 4 - 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 5 - 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

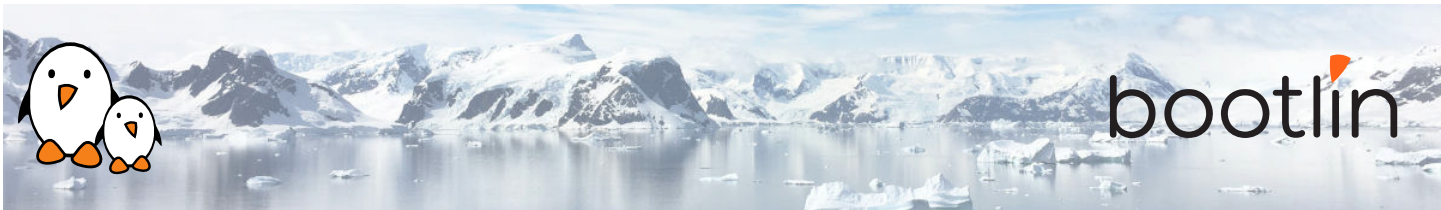
- Introduction to BSP layers
- Adding a new machine
- Bootloader configuration
- Linux: the kernel bbclass and the linux-yocto recipe

Lab 6 - Create a custom machine configuration

- Create a new machine configuration
- Build an image for the new machine

Lecture - Distro layers

- Distro configuration
- Distro layers



Lecture - Images

- Writing an image recipe
- Image types
- Writing and using package groups recipes

Lab 7 - 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

Day 3 - Afternoon

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 8 - Develop your application in the Poky SDK

- Building an SDK
- Using the Yocto Project SDK

Lecture - Devtool

- About devtool
- Devtool use cases

Lab 9 - 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