



Embedded Linux development with Buildroot training

Course duration _____

 3 days – 24 hours

Language _____

Materials English

Oral Lecture English
French

Trainer _____

One of the following engineers

- Thomas Petazzoni

Contact _____

 training@bootlin.com

 +33 484 258 097

Audience

Companies already using or interested in using Buildroot to build their embedded Linux systems.

Training objectives

- Be able to understand the role and principle of an embedded Linux build system, and compare Buildroot to other tools offering similar functionality.
- Be able to create a simple embedded Linux system with Buildroot: create a configuration, run the build, install the result on an embedded platform.
- Be able to adjust the Buildroot configuration to build an embedded Linux system tailored to specific needs: choice of the cross-compilation toolchain, management of the Linux kernel configuration, customization of the root filesystem contents, etc.
- Be able to create new packages in Buildroot to integrate additional applications and libraries into the embedded Linux system.
- Be able to use the tools offered by Buildroot to manage and analyze the build: security vulnerability tracking, license compliance, etc.
- Be able to develop and debug Linux user-space applications in the context of Buildroot.
- Be able to interact with the Buildroot open-source community, and to understand the internals of Buildroot.

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.



Onsite
training

Required equipment

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 × 46 pins headers, to access UARTs, SPI buses, I2C buses and more.
- Ethernet or WiFi



Day 1 - Morning

Lecture	Embedded Linux and build system introduction	▪ The general architecture of an embedded Linux system ▪ Build systems vs. binary distributions ▪ Role of a build system ▪ Comparison of existing build systems
Lecture	Introduction to Buildroot	▪ Key facts about the project ▪ Getting Buildroot ▪ Basic configuration of Buildroot ▪ Doing a first build
Lab	Basic Buildroot usage	▪ Getting and setting up Buildroot ▪ Configuring and building a basic system with Buildroot for an embedded platform ▪ Flash and test the generated system on the embedded platform
Lecture	Managing the build and configuration	▪ Out of tree build ▪ Using and creating <i>defconfigs</i> ▪ Defconfig fragments ▪ Other building tips

Day 1 - Afternoon

Lecture	Buildroot source and build trees	▪ Details about the Buildroot source code organization ▪ Details about the Buildroot build tree
Lecture	Toolchains in Buildroot	▪ The different choices for using toolchains in Buildroot ▪ Overview of the toolchain options ▪ Using existing binary toolchains, such as Bootlin toolchains, understanding <i>multilib</i> capabilities and integration of toolchains in Buildroot ▪ Generating custom toolchains with <i>Crosstool-NG</i>, and re-use them as external toolchains
Lecture	Managing the Linux kernel configuration	▪ Loading, changing and saving the kernel configuration
Lecture	Root filesystem construction in Buildroot	▪ Understand how Buildroot builds the root filesystem: <i>skeleton</i>, installation of packages, overlays, <i>post-build</i> and <i>post-image</i> scripts. ▪ Customization of the root filesystem contents ▪ System configuration: <i>console</i> selection, various <i>/dev</i> management methods, the different <i>init</i> implementations, etc. ▪ Understand how Buildroot generates filesystem images
Lab	Root filesystem customization	▪ Explore the build output ▪ Customize the root filesystem using a <i>rootfs overlay</i> ▪ Customize the kernel with patches and additional configuration options ▪ Add more packages ▪ Use <i>defconfig</i> files and <i>out of tree</i> build

Day 2 - Morning

Lecture	Download infrastructure in Buildroot	▪ Downloading logic ▪ Primary site and backup site, doing offline builds ▪ VCS download, integrity checking ▪ Download-related <i>make</i> targets
Lecture	GNU Make 101	▪ Basics of make rules ▪ Defining and referencing variables ▪ Conditions, functions ▪ Writing recipes
Lecture	Integrating new packages in Buildroot	▪ How to integrate new packages in the Buildroot configuration system ▪ Understand the different package infrastructures: for <i>generic</i>, <i>auto-tools</i>, <i>CMake</i>, <i>Python</i> packages and more. ▪ Writing a package <code>Config.in</code> file: how to express dependencies on other packages, on toolchain options, etc. ▪ Details on writing a package recipe: describing the package source code location, download method, configuration, build and installation steps, handling dependencies, etc.
Lab	New packages in Buildroot	▪ Create a new package for <i>nInvaders</i> ▪ Understand how to add dependencies ▪ Add patches to <i>nInvaders</i> for <i>Nunchuk</i> support

Day 2 - Afternoon

Lecture	Advanced package aspects	▪ Licensing report ▪ Patching support: patch ordering and format, global patch directory, etc. ▪ User, permission, device tables ▪ Init scripts and systemd unit files ▪ Config scripts ▪ Understanding <i>hooks</i> ▪ Overriding commands ▪ Legacy handling ▪ Virtual packages
Lab	Advanced packages	▪ Package an application with a mandatory dependency and an optional dependency ▪ Package a library, hosted on GitHub ▪ Use <i>hooks</i> to tweak packages ▪ Add a patch to a package

Day 3 - Morning

Lecture	Analyzing the build: licensing, dependencies, build time	▪ Usage of the legal information infrastructure ▪ Graphing dependencies of packages ▪ Collecting and graphing build time information
Lecture	Advanced topics	▪ <code>BR2_EXTERNAL</code> to store customizations outside of the Buildroot sources ▪ Package-specific targets ▪ Understanding rebuilds ▪ Tips for building faster
Lab	Advanced aspects	▪ Use build time graphing capabilities ▪ Use dependency graphing capabilities ▪ Use licensing report generation, and add licensing information to your own packages ▪ Use <code>BR2_EXTERNAL</code>

Day 3 - Afternoon

Lecture	Application development with Buildroot	▪ Using Buildroot during application development ▪ Usage of the Buildroot environment to build applications outside of Buildroot ▪ Generate an SDK for other developers ▪ Remote debugging with Buildroot
Lab	Application development with Buildroot	▪ Build and run your own application ▪ Remote debug your application ▪ Use <code><pkg>_OVERRIDE_SRCDIR</code>
Lecture	Understanding Buildroot internals	▪ Detailed description of the Buildroot build process: toolchain, packages, root filesystem construction, stamp files, etc. ▪ Understanding virtual packages.
Lecture	Getting support and contributing	▪ Getting support: <i>Bugzilla</i>, <i>mailing list</i>, <i>IRC</i> ▪ Contributing: understanding the development process, how to submit patches