Embedded Linux development with Buildroot training

Course duration

5 half days – 20 hours

Language

Materials	English
Oral Lecture	English
	French

Trainer

One of the following engineers

- Thomas Petazzoni

Contact

training@bootlin.com
+33 484 258 097

bootlin
bootlin.com

Online seminar

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, 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

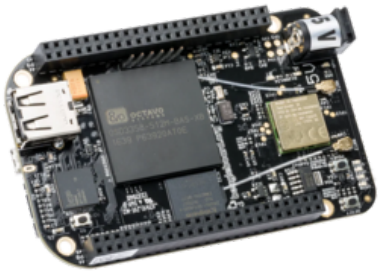
One of these Discovery Kits from STMicroelectronics: **STM32MP157A-DK1**, **STM32MP157D-DK1**, **STM32MP157C-DK2** or **STM32MP157F-DK2**

- 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



Training Schedule

Half day 1

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
Demo	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 *defconfigs* ▪ Defconfig fragments ▪ Other building tips
Lecture	Buildroot source and build trees	▪ Details about the Buildroot source code organization ▪ Details about the Buildroot build tree

Half day 2

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 *multilib* capabilities and integration of toolchains in Buildroot ▪ Generating custom toolchains with *Crosstool-NG*, 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: *skeleton*, installation of packages, overlays, *post-build* and *post-image* scripts. ▪ Customization of the root filesystem contents ▪ System configuration: *console* selection, various `/dev` management methods, the different `init` implementations, etc. ▪ Understand how Buildroot generates filesystem images
Demo	Root filesystem customization	▪ Explore the build output ▪ Customize the root filesystem using a *rootfs overlay* ▪ Customize the kernel with patches and additional configuration options ▪ Add more packages ▪ Use *defconfig* files and *out of tree* build
Lecture	Download infrastructure in Buildroot	▪ Downloading logic ▪ Primary site and backup site, doing offline builds ▪ VCS download, integrity checking ▪ Download-related *make* targets

Half day 3

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 *generic*, *autotools*, *CMake*, *Python* packages and more. ▪ Writing a package `Config.in` 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.
Demo	New packages in Buildroot	▪ Create a new package for *nInvaders* ▪ Understand how to add dependencies ▪ Add patches to *nInvaders* for *Nunchuk* support
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 *hooks* ▪ Overriding commands ▪ Legacy handling ▪ Virtual packages

Half day 4

Demo	Advanced packages	▪ Package an application with a mandatory dependency and an optional dependency ▪ Package a library, hosted on GitHub ▪ Use *hooks* to tweak packages ▪ Add a patch to a package
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	▪ BR2_EXTERNAL to store customizations outside of the Buildroot sources ▪ Package-specific targets ▪ Understanding rebuilds ▪ Tips for building faster
Demo	Advanced aspects	▪ Use build time graphing capabilities ▪ Use dependency graphing capabilities ▪ Use licensing report generation, and add licensing information to your own packages ▪ Use BR2_EXTERNAL

Half day 5

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
Demo	Application development with Buildroot	▪ Build and run your own application ▪ Remote debug your application ▪ Use `<pkg>_OVERRIDE_SRCDIR`
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: *Bugzilla*, *mailing list*, *IRC* ▪ Contributing: understanding the development process, how to submit patches