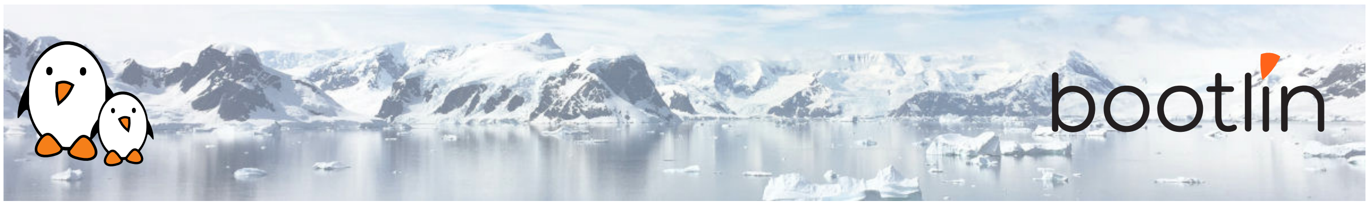
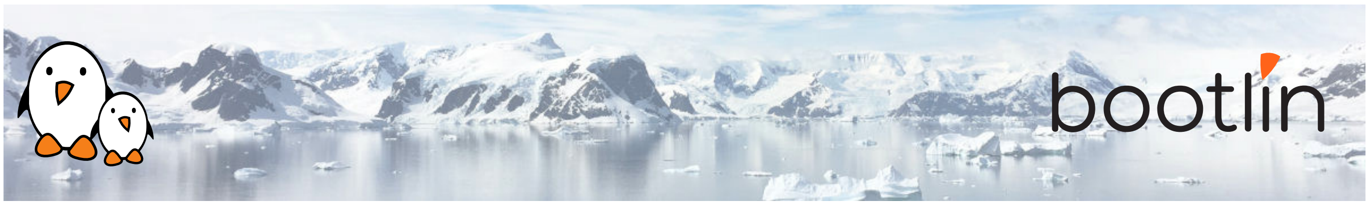


Embedded Linux system development training

On-line seminar

Title	**Embedded Linux system development training**
Overview	Bootloaders Kernel (cross) compiling and booting Block and flash filesystems C library and cross-compiling toolchains Lightweight building blocks for embedded systems Embedded system development tools Embedded application development and debugging Implementing real-time requirements in embedded Linux systems Optional practical labs proposed on an virtual ARM board emulated by QEMU (to be done between each session), followed by corresponding practical demos on the ARM based SAMA5D3 Xplained board from Microchip
Materials	Check that the course contents correspond to your needs: `https://bootlin.com/doc/training/embedded-linux`.
Duration	**Seven** half days - 28 hours (4 hours per half day). 75% of lectures, 25% of practical demos, plus optional practical labs on QEMU between sessions.
Trainer	One of the engineers listed on: `https://bootlin.com/training/trainers/`
Language	Oral lectures: English Materials: English.
Audience	People developing devices using the Linux kernel People supporting embedded Linux system developers.
Prerequisites	**Familiarity with UNIX or GNU/Linux commands** People lacking experience on this topic should get trained by themselves, for example with our freely available on-line slides: `https://bootlin.com/blog/command-line/`.

Required 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 • For people interested in our optional practical labs, an installation of VirtualBox and about 30 GB of free disk space.
Materials	Electronic copies of presentations, lab and demo instructions and data.



Emulated hardware

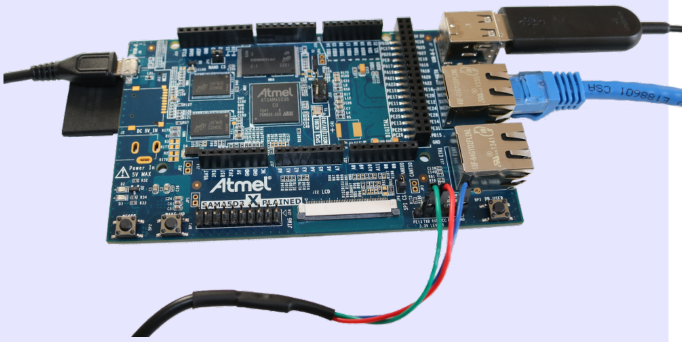
Optional labs (between sessions) proposed on the QEMU emulated ARM Vexpress A9 board.

Image credits (Wikipedia): https://frama.link/mW71eosa

Real hardware in practical demos

Using the Microchip SAMA5D3 Xplained board (Cortex A5 from Microchip) in all practical demos performed by the trainer, used as solutions for the above labs:

- USB powered
- 256 MB DDR2 RAM
- 256 MB NAND flash
- 2 Ethernet ports (Gigabit + 100 Mbit)
- 2 USB 2.0 host ports
- 1 USB device port
- 1 MMC/SD slot
- 3.3 V serial port (like Beaglebone Black)
- Arduino R3-compatible header
- Misc: JTAG, buttons, LEDs

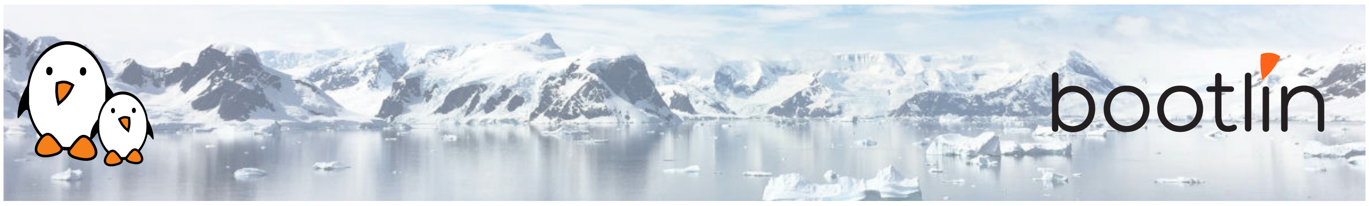


Half day 1

Lecture - Introduction to embedded Linux

- Advantages of Linux versus traditional embedded operating systems. Reasons for choosing Linux.
- Global picture: understanding the general architecture of an embedded Linux system. Overview of the major components in a typical system.

The rest of the course will study each of these components in detail.



Lecture - Embedded Linux development environment

- Operating system and tools to use on the development workstation for embedded Linux development.
- Desktop Linux usage tips.

Lecture - Cross-compiling toolchain and C library

- What's inside a cross-compiling toolchain
- Choosing the target C library
- What's inside the C library
- Ready to use cross-compiling toolchains
- Building a cross-compiling toolchain with automated tools.

During spare time between sessions

Optional lab - Cross compiling toolchain

- Configuring Crosstool-NG
- Executing it to build a custom uClibc toolchain.

Half day 2

Solution demo - Cross compiling toolchain

Corresponding to the above optional lab

Lecture - Bootloaders

- Available bootloaders
- Bootloader features
- Installing a bootloader
- Detailed study of U-Boot

Lecture - Linux kernel

- Role and general architecture of the Linux kernel
- Features available in the Linux kernel, with a focus on features useful for embedded systems
- Kernel user interface
- Getting the sources
- Understanding Linux kernel versions.
- Using the patch command

Lecture – Configuring and compiling a Linux kernel

- Kernel configuration.
- Using ready-made configuration files for specific architectures and boards.
- Kernel compilation.
- Generated files.
- Using kernel modules

During spare time between sessions

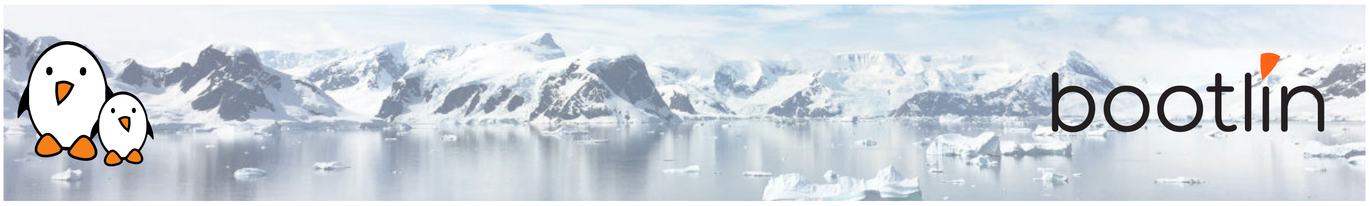
Optional lab - Bootloader and U-boot

Using the QEMU emulated ARM Vexpress A9 board

- Configure U-Boot for your virtual board
- Build U-Boot with your new toolchain
- Start U-Boot directly from QEMU and test it.
- Store the U-Boot environment in an emulated SD card
- Setup networking between your PC and the QEMU emulated machine
- Setup tftp for transferring files between the host and U-Boot in QEMU

Optional lab - Kernel sources

- Downloading kernel sources
- Applying kernel patches



Optional lab - Kernel cross-compiling and booting

Using the QEMU emulated ARM Vexpress A9 board

- Configuring the Linux kernel and cross-compiling it for the ARM board.
- Boot Linux from U-Boot through tftp
- Automate the boot sequence in U-Boot

Half day 3

Solution demo - Bootloader and U-boot

Using the Microchip SAMA5D3 Xplained board

- Set up serial communication with the board.
- Configure, compile and install the first-stage bootloader and U-Boot on the Xplained board.
- Become familiar with U-Boot environment and commands.
- Set up TFTP communication with the board. Use TFTP U-Boot commands.

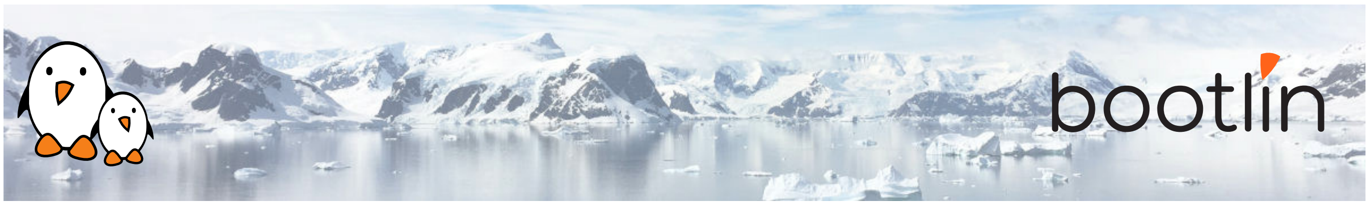
Solution demo - Kernel sources

- Downloading kernel sources
- Apply kernel patches

Solution demo - Kernel cross-compiling and booting

Using the Microchip Xplained board

- Configuring the Linux kernel and cross-compiling it for the ARM board.
- Downloading your kernel on the board through U-boot's tftp client.
- Booting your kernel from RAM.
- Copying the kernel to flash and booting it from this location.
- Storing boot parameters in flash and automating kernel booting from flash.



Lecture – Root filesystem in Linux

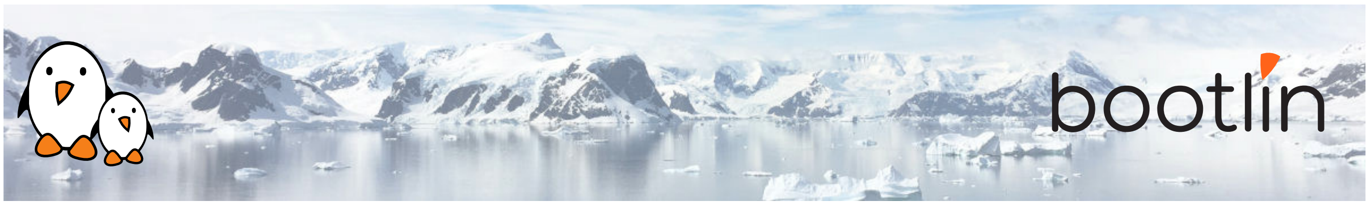
- Filesystems in Linux.
- Role and organization of the root filesystem.
- Location of the root filesystem: on storage, in memory, from the network.
- Device files, virtual filesystems.
- Contents of a typical root filesystem.

Lecture - BusyBox

- Detailed overview. Detailed features.
- Configuration, compiling and deploying.

Lecture - Block filesystems

- Filesystems for block devices.
- Usefulness of journaled filesystems.
- Read-only block filesystems.
- RAM filesystems.
- How to create each of these filesystems.
- Suggestions for embedded systems.



During spare time between sessions

Optional lab – Tiny root filesystem built from scratch with BusyBox

Using the QEMU emulated ARM Vexpress A9 board

- Now build a basic root filesystem from scratch for your ARM system
- Setting up a kernel to boot your system on a host directory exported by NFS
- Passing kernel command line parameters to boot on NFS
- Creating the full root filesystem from scratch. Populating it with BusyBox based utilities.
- Creating device files and booting the virtual system.
- System startup using BusyBox /sbin/init
- Using the BusyBox http server.
- Controlling the target from a web browser on the PC host.
- Setting up shared libraries on the target and compiling a sample executable.

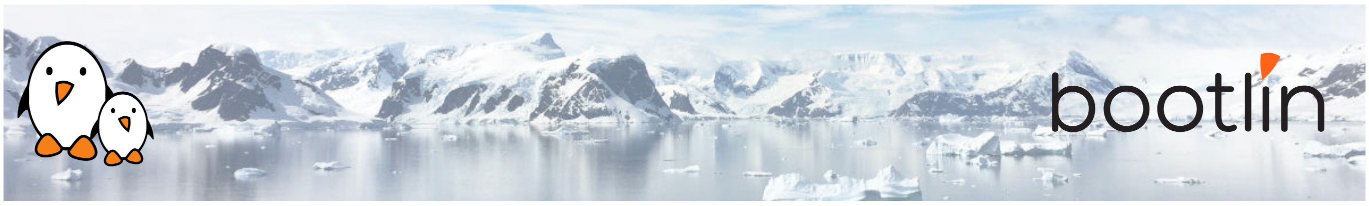
Half day 4

Solution demo – Tiny root filesystem built from scratch with BusyBox

Implementing the same functionality as on QEMU but with the Microchip Xplained board

Lecture - Flash filesystems

- The Memory Technology Devices (MTD) filesystem.
- Filesystems for MTD storage: JFFS2, Yaffs2, UBIFS.
- Kernel configuration options
- MTD storage partitions.
- Focus on today’s best solution, UBI and UBIFS: preparing, flashing and using UBI images.



During spare time between sessions

Optional lab - Block filesystems

Using the QEMU emulated ARM Vexpress A9 board

- Booting your system with a mix of filesystems on MMC/SD storage: SquashFS for applications, ext4 for configuration and user data, and tmpfs for temporary system files.

Half day 5

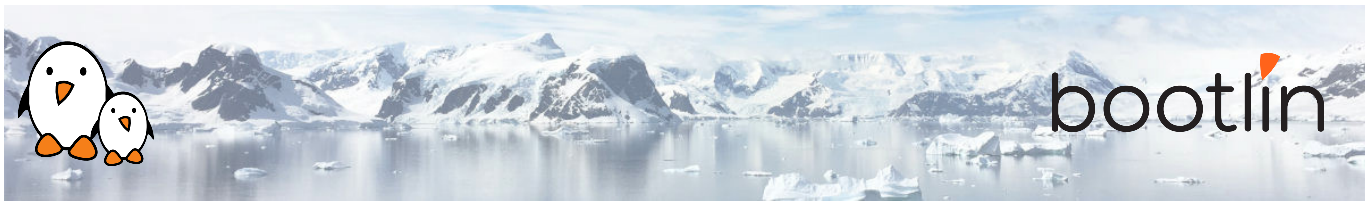
Solution demo - Block filesystems

Implementing the same functionality as on QEMU but with the Microchip Xplained board

Demo – Flash filesystems

Using the SAMAD3 Xplained ARM board

- Defining partitions in U-Boot for your internal flash storage instead of using raw offsets.
- Sharing these definitions with Linux.
- Creating a UBI image on your workstation, flashing it from U-Boot and booting your system on one of the UBI volumes with UBIFS.



Lecture – Leveraging existing open-source components in your system

- Reasons for leveraging existing components.
- Find existing free and open source software components.
- Choosing the components.
- The different free software licenses and their requirements.
- Overview of well-known typical components used in embedded systems: graphical libraries and systems (framebuffer, Gtk, Qt, etc.), system utilities, network libraries and utilities, multimedia libraries, etc.
- System building: integration of the components.

Lecture – Cross-compiling applications and libraries

- Configuring, cross-compiling and installing applications and libraries.
- Details about the build system used in most open-source components.
- Overview of the common issues found when using these components.

During spare time between sessions

Optional lab – Cross-compiling applications and libraries

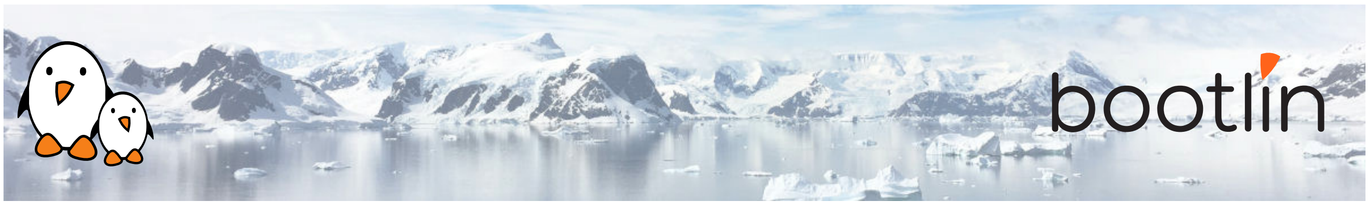
Using the QEMU emulated ARM Vexpress A9 board

- Building a system with audio libraries and a sound player application.
- Manual compilation and installation of several free software packages.
- Learning about common techniques and issues.

Half day 6

Solution demo – Cross-compiling applications and libraries

Implementing the same functionality as on QEMU but with the Microchip Xplained board



Lecture - Embedded system building tools

- Review of existing system building tools.
- Buildroot example.

Lecture - Application development and debugging

- Programming languages and libraries available.
- Overview of the C library features for application development.
- Build system for your application, how to use existing libraries in your application.
- Source browsers and Integrated Development Environments (IDEs).
- Debuggers. Debugging remote applications with gdb and gdbserver. Post-mortem debugging with core files.
- Code checkers, memory checkers, profilers.

During spare time between sessions

Optional lab - System build with Buildroot

Using the QEMU emulated ARM Vexpress A9 board

- Using Buildroot to rebuild the same system as in the previous lab.
- Seeing how easier it gets.

Optional lab – Application development and debugging

On the Microchip Xplained board

- Develop and compile an application relying on the ncurses library
- Using strace, ltrace and gdbserver to debug a crappy application on the remote system.

Half day 7

Solution demo - System build with Buildroot

Implementing the same functionality as on QEMU but with the Microchip Xplained board

Solution demo - Application development and debugging

Implementing the same functionality as on QEMU but with the Microchip Xplained board

Lecture - Linux and real-time

Very useful for many kinds of devices, industrial or multimedia systems.

- Understanding the sources of latency in standard Linux.
- Soft real-time solutions for Linux: improvements included in the mainline Linux version.
- Understanding and using the latest RT preempt patches for mainline Linux.
- Real-time kernel debugging. Measuring and analyzing latency.
- Xenomai, a hard real-time solution for Linux: features, concepts, implementation and examples.

Demo - Linux latency tests

- Tests performed on the Xplained ARM board.
- Latency tests on standard Linux, with preemption options.
- Latency tests using the `PREEMPT_RT` kernel patchset.
- Setting up Xenomai.
- Latency tests with Xenomai.

Possible extra time

Extra time (up to 4 hours) may be proposed if the agenda didn't fit in 7 half days, according to the time spend answering questions from participants.