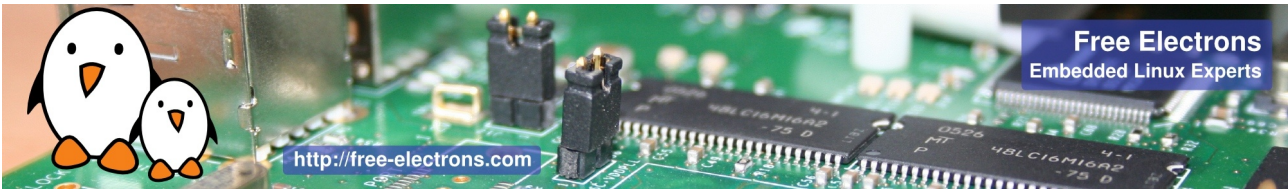


Embedded Linux system development training

5 day session

Overview

Title	Embedded Linux system development training
Overview	<p>Bootloaders. Kernel (cross) compiling and booting. Block and flash filesystems. C library and cross-compiling toolchains. Lightweight applications for embedded systems. Embedded system development tools. Target debugging. Hotplugging. Implementing real-time requirements in embedded Linux systems. Techniques to optimize system size, RAM, power, performance and cost. Practical labs with the Beagle ARM board.</p>
Duration	5 days. 40% of presentations and 60% of practical labs.
Trainer	Gregory Clement, Thomas Petazzoni or Michael Opdenacker.
Language	Oral lectures: English Materials: English
Audience	People developing devices using the Linux kernel People supporting embedded Linux system developers.
Prerequisites	<p>Knowledge and practice of Unix or GNU/Linux commands People lacking experience on this topic should get trained by themselves with our freely available on-line slides (http://free-electrons.com/training/intro_unix_linux/) Possibility to order an extra training day on this topic.</p>
Needed equipment	<p>Video projector 1 PC computer with a bootable cdrom drive on each desk (1 or 2 people) PC computers with at least 1 GB of RAM and a free partition of at least 10 GB for installing Linux (will be done by training participants). Connection to the Internet (direct or through the company proxy). PC computers with valuable data must be backed up before being used in our sessions. Some people have already made mistakes during our sessions and damaged work data. If Linux is already installed, we need a 32 bit (i386) version of Ubuntu Desktop 10.04 (Xubuntu and Kubuntu variants are fine). We don't support other distributions, because we can't test all possible package versions.</p>
Materials	Print and electronic copy of presentations and labs. Electronic copy of lab files.

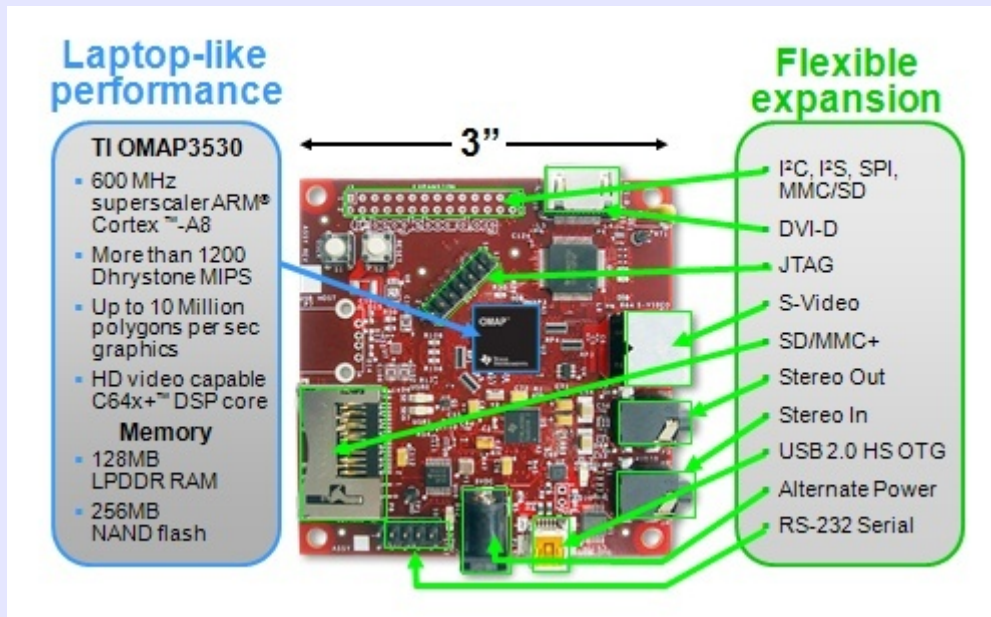


Training - Embedded Linux system development

See our training materials on <http://free-electrons.com/doc/training/beagle>
 This way, you can check by yourself that they correspond to your needs.

Hardware

Using Beagle Boards based on TI OMAP 3530 in most practical labs



Day 1 - Morning

Lab - Lab environment setup

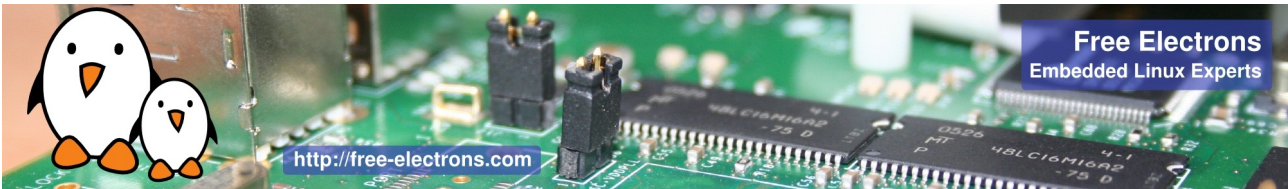
Using the Ubuntu GNU/Linux distribution (<http://ubuntu.com>)
 Installing Ubuntu in a free partition in the hard disk.

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 - System administration basics

You will be the administrator of your embedded system
 Setting up networking
 Manipulating disk (block) partitions
 Filesystems: creating and mounting
 Misc: file ownership, shutting down...



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.

Day 1 - Afternoon

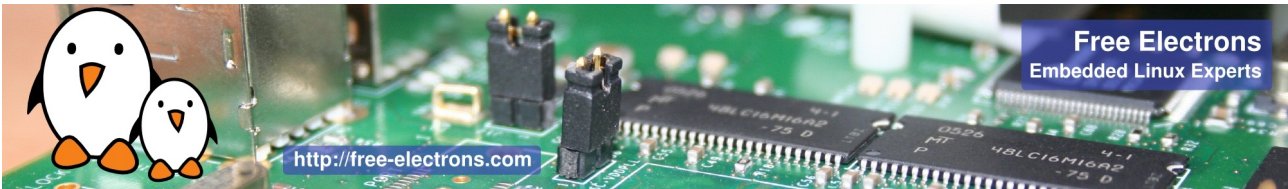
Lab - Cross compiling toolchain	Lecture - Bootloaders
Configuring Crosstool-NG Executing it to build a custom uClibc toolchain.	Available bootloaders Bootloader features Installing a bootloader Detailed study of U-Boot

Lab - U-boot	Lecture - Linux kernel
Interact with U-boot through a serial console.	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

Day 2 - Morning

Lecture - Configuring and compiling a Linux kernel
Kernel configuration. Useful settings for embedded systems. Native compiling. Generated files. Using kernel modules

Lab - Configuring, compiling and booting the kernel	Lecture - Booting
Getting kernel sources. Applying kernel patches. Configuring, compiling and booting the kernel on a virtual PC.	Linux system booting overview. Linux kernel booting. Booting parameters. NFS boot example. Booting on an initramfs System startup.



Day 2 - Afternoon

Lecture - Kernel cross-compiling

Kernel cross-compiling setup.
Using ready-made configuration files for specific architectures and boards.
Cross-compiling Linux.

Lab - Kernel cross-compiling and booting

Using the ARM board

Setting up a cross-compiling environment for the ARM instruction set.
Configuring the Linux kernel and cross-compiling it for the ARM board.
Setting up a tftp server on your GNU/Linux workstation.
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 - BusyBox

Detailed overview. Detailed features.
Configuration, compiling and deploying.
Saving space by implementing your own applets.

Day 3 - Morning

Lab - Basic root filesystem created from scratch

Now build a basic root filesystem from scratch for your ARM system

Setting up a kernel to boot your system on a workstation 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 developing a sample application.

Day 3 - Afternoon

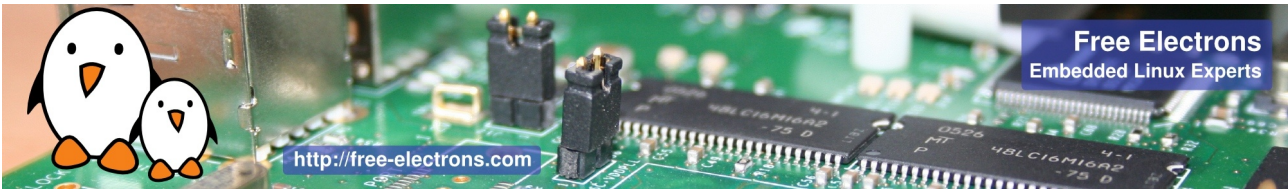
Lecture - Block filesystems

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.

Lab - Block filesystems

Using the ARM board

Creating partitions on your block storage
Booting a system with a mix of filesystems: SquashFS for applications, ext3 for configuration and user data, and tmpfs for temporary system files.



Lecture - Flash filesystems	Lab - Flash filesystems
<p>The Memory Technology Devices (MTD) filesystem.</p> <p>Filesystems for MTD storage: JFFS2, Yaffs2, ubifs.</p> <p>Kernel configuration options</p> <p>MTD storage partitions.</p> <p>Mounting MTD filesystem images.</p>	<p><i>Using the ARM board</i></p> <p>Creating partitions in your internal flash storage.</p> <p>Formatting the main partition with SquashFS.</p> <p>Using jffs2 for system data.</p>

Day 4 - Morning

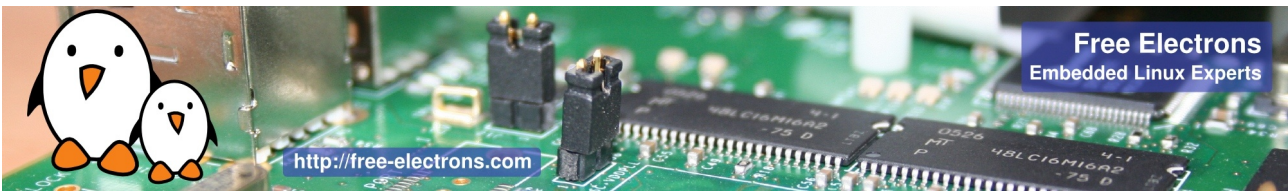
Lecture - Leveraging existing open-source components in your system
<p>Reasons for leveraging existing components.</p> <p>Find existing free and open source software components.</p> <p>Choosing the components.</p> <p>The different free software licenses and their requirements.</p> <p>Overview of well-known typical components used in embedded systems : graphical libraries and systems (framebuffer, DirectFB, Gtk, Qt, etc.), system utilities, network libraries and utilities, multimedia libraries, etc.</p> <p>Example of a typical consumer electronic product leveraging many open-source components.</p> <p>System building : integration of the components.</p>

Lecture - Cross-compiling applications and libraries	Lab - Cross-compiling applications and libraries
<p>Configuring, cross-compiling and installing applications and libraries.</p> <p>Details about the build system used in most open-source components.</p> <p>Overview of the common issues found when using these components.</p>	<p><i>Building a system with a graphical system based on DirectFB</i></p> <p>Manual compilation and installation of several free software packages.</p> <p>Learning about common techniques and issues.</p>

Day 4 - Afternoon

Lab - Cross-compiling applications and libraries
<p>... continued from the morning</p>

Lecture - Embedded system building tools	Lab - System build with Buildroot
<p>Review of existing system building tools.</p> <p>Buildroot example.</p>	<p>Using Buildroot to rebuild the same system as in the previous lab.</p> <p>Seeing how easier it gets.</p>



Day 5 - Morning

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.
Developing on Windows.

Lab - Application development and debugging

Develop and compile an application relying on the DirectFB library
Using strace, ltrace and gdbserver to debug a crappy application on the remote system.
Do post-mortem analysis of a crashed application.

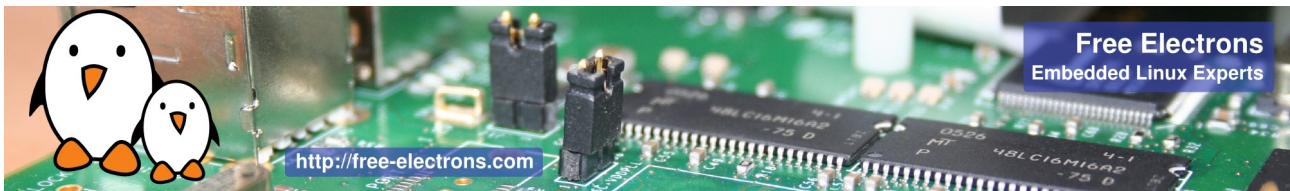
Day 5 - Afternoon

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 brought by Linux 2.6.
Understanding and using the latest RT preempt patches for mainstream Linux.
Real-time kernel debugging. Measuring and analyzing latency.
Xenomai, a hard real-time solution for Linux: features, concepts, implementation and examples.

Lab - Linux latency tests

Tests performed on our ARM board
Using the Linux real-time API.
Measuring and comparing scheduling latency in standard Linux and in Linux with the latest RT patches.



Additional topics

These topics can be covered at the end of the training if enough time is left.

Lecture - Hotplugging	Lab - Hotplugging
<p>Udev: handling hardware events from user-space: creating and removing device files, identifying drivers, notifying programs and users.</p> <p>Using BusyBox mdev, a simpler implementation.</p>	<p>Using BusyBox mdev to populate the /dev directory with all available devices.</p> <p>Adding rules for mdev.</p> <p>Making mdev automatically mount the partitions of an MMC card when it is inserted.</p>

Lecture - System optimizations
<p>Ideas, techniques and resources for reducing cost, size, boot time, run time and power consumption.</p>