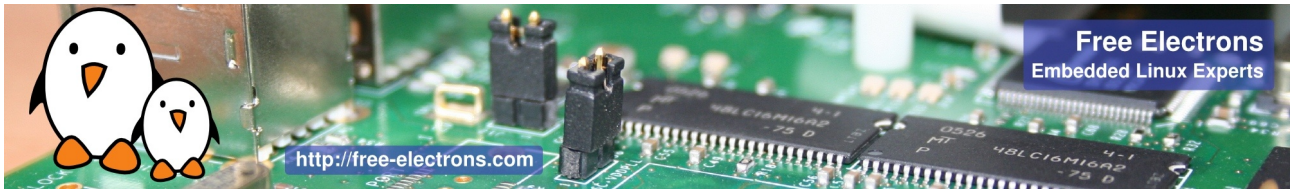
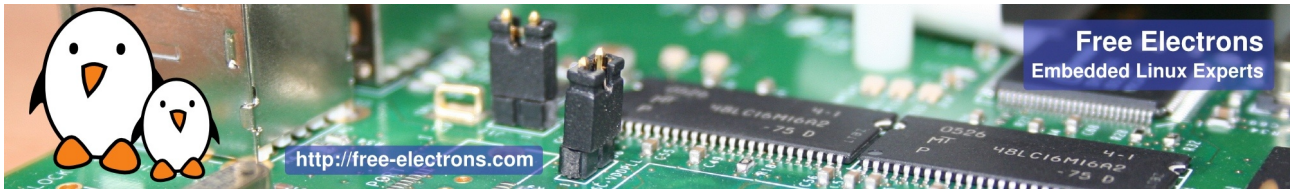



# Embedded Linux system development training

### 5 day session

## Overview

| | |
|---|---|
| **Title** | **Embedded Linux system development training** |
| **Overview** | Bootloaders. Kernel (cross) compiling and booting.<br>Block and flash filesystems. C library and cross-compiling toolchains. Lightweight applications for embedded systems. Embedded system development tools. Target debugging. Hotplugging.<br>Implementing real-time requirements in embedded Linux systems.<br>Techniques to optimize system size, RAM, power, performance and cost.<br>Practical labs with the Beagle ARM board. |
| **Duration** | 5 days. 40% of presentations and 60% of practical labs. |
| **Trainer** | Gregory Clement, Thomas Petazzoni or Michael Opdenacker. |
| **Language** | Oral lectures: English<br>Materials: English |
| **Audience** | People developing devices using the Linux kernel<br>People supporting embedded Linux system developers. |
| **Prerequisites** | **Knowledge and practice of Unix or GNU/Linux commands**<br>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/)<br>Possibility to order an extra training day on this topic. |
| **Needed equipment** | Video projector<br>1 PC computer with a bootable cdrom drive on each desk (1 or 2 people)<br>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.<br>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. |
| **Materials** | Print and electronic copy of presentations and labs.<br>Electronic copy of lab files. |

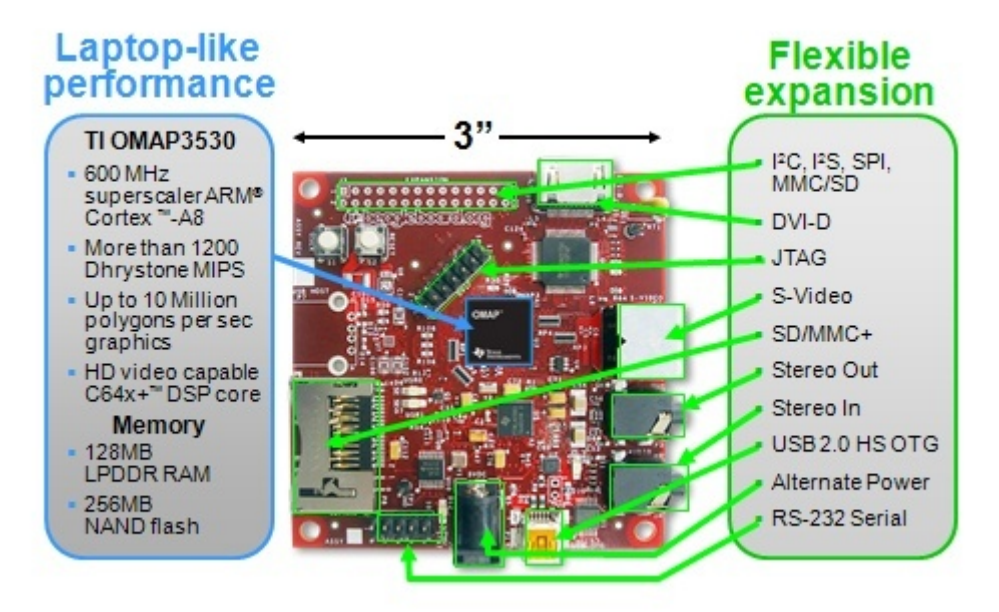



# Training - Embedded Linux system development

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

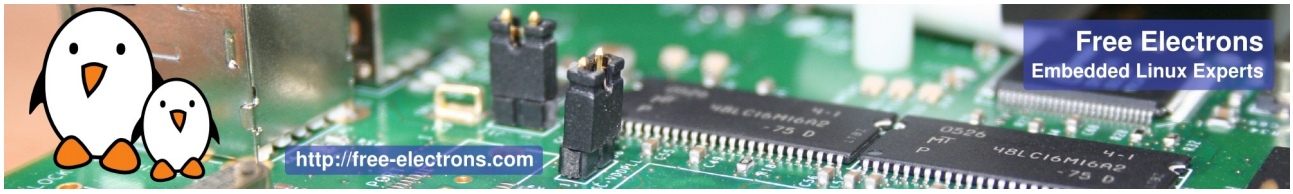
Using Beagle Boards based on TI OMAP 3530 in most practical labs




## Day 1 - Morning

| **Lab - Lab environment setup** |
|---|
| Using the Ubuntu GNU/Linux distribution (http://ubuntu.com)<br>Installing Ubuntu in a free partition in the hard disk. |

| **Lecture – Introduction to embedded Linux** | **Lecture - System administration basics** |
|---|---|
| Advantages of Linux versus traditional embedded operating systems. Reasons for choosing Linux.<br>Global picture: understanding the general architecture of an embedded Linux system. Overview of the major components in a typical system.<br>*The rest of the course will study each of these components in detail.* | *You will be the administrator of your embedded system*<br>Setting up networking<br>Manipulating disk (block) partitions<br>Filesystems: creating and mounting<br>Misc: file ownership, shutting down... |




| Lecture - Cross-compiling toolchain and C library |
|---|
| What's inside a cross-compiling toolchain<br>Choosing the target C library<br>What's inside the C library<br>Ready to use cross-compiling toolchains<br>Building a cross-compiling toolchain with automated tools. |

### Day 1 – Afternoon

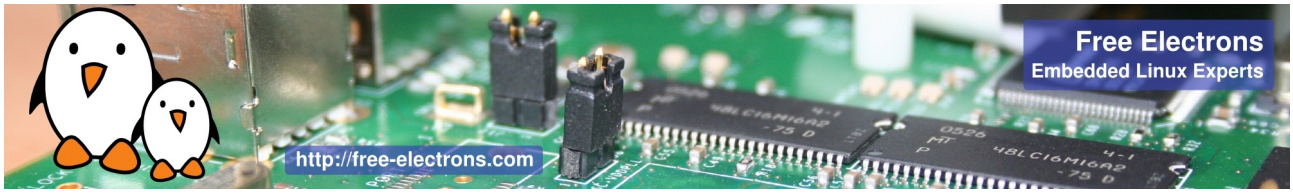
| Lab – Cross compiling toolchain | Lecture – Bootloaders |
|---|---|
| Configuring Crosstool-NG<br>Executing it to build a custom uClibc toolchain. | Available bootloaders<br>Bootloader features<br>Installing a bootloader<br>Detailed study of U-Boot |

| Lab - U-boot | Lecture – Linux kernel |
|---|---|
| Interact with U-boot through a serial console. | Role and general architecture of the Linux kernel<br>Features available in the Linux kernel, with a focus on features useful for embedded systems<br>Kernel user interface<br>Getting the sources<br>Understanding Linux kernel versions.<br>Using the `patch` command |

### Day 2 - Morning

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

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

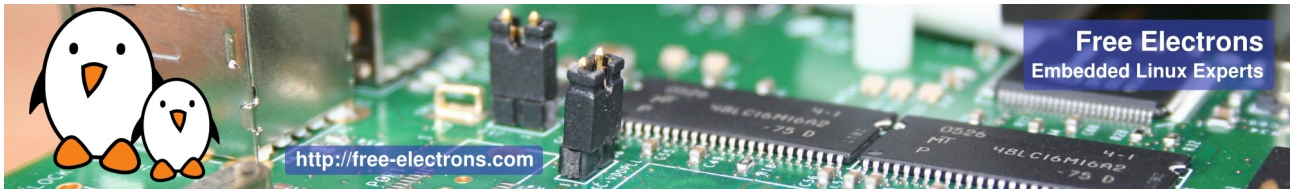
## Day 3 - Afternoon

### 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 |
|---|---|
| The Memory Technology Devices (MTD) filesystem.<br>Filesystems for MTD storage: JFFS2, Yaffs2, ubifs.<br>Kernel configuration options<br>MTD storage partitions.<br>Mounting MTD filesystem images. | *Using the ARM board*<br>Creating partitions in your internal flash storage.<br>Formating the main partition with SquashFS.<br>Using jffs2 for system data. |

### Day 4 - Morning

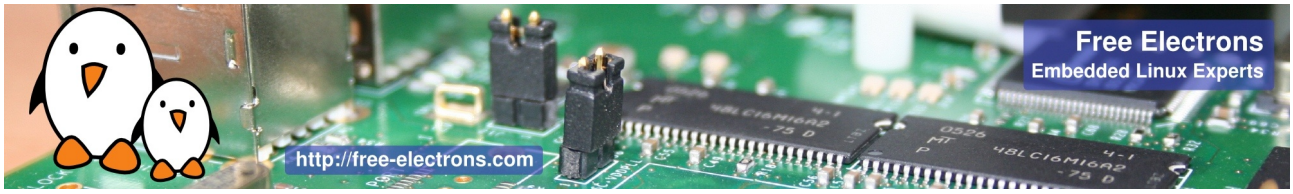
| Lecture – Leveraging existing open-source components in your system |
|---|
| Reasons for leveraging existing components.<br>Find existing free and open source software components.<br>Choosing the components.<br>The different free software licenses and their requirements.<br>Overview of well-known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

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

### Day 4 - Afternoon

| Lab – Cross-compiling applications and libraries |
|---|
| ... continued from the morning |

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




## 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.

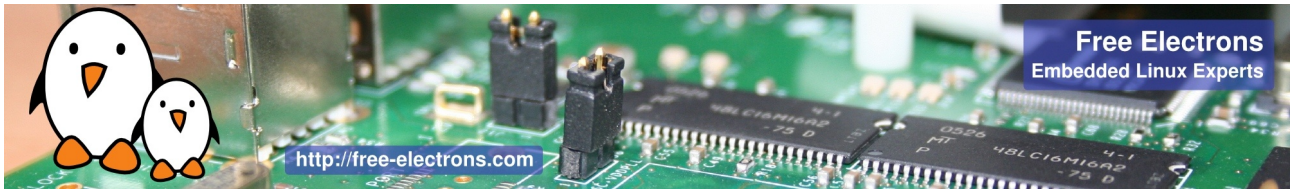
## Day 5 - Afternoon

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

| Lecture - System optimizations |
|---|
| Ideas, techniques and resources for reducing cost, size, boot time, run time and power consumption. |