

# Embedded Systems



**Mail: [Stephane.Lavirotte@unice.fr](mailto:Stephane.Lavirotte@unice.fr)**  
**Web: <http://stephane.lavirotte.com/>**  
**Université de Nice - Sophia Antipolis**

# A Smart Object

- ✓ A Smart Object: What is it ?
- ✓ The Nabaztag example
  - 23 cm high
  - 418 g
  - connected to Internet with WiFi 802.11b/g
  - communicates with its user by sending voice messages, light or stirring ears
  - disseminates information such as weather, stock, air quality, traffic from the Paris ring road, incoming e-mail, etc..
  - RFID reads and activates service on reading RFID tags
- ✓ And if you looked inside...







## NSLU 2

A concentrate of technology

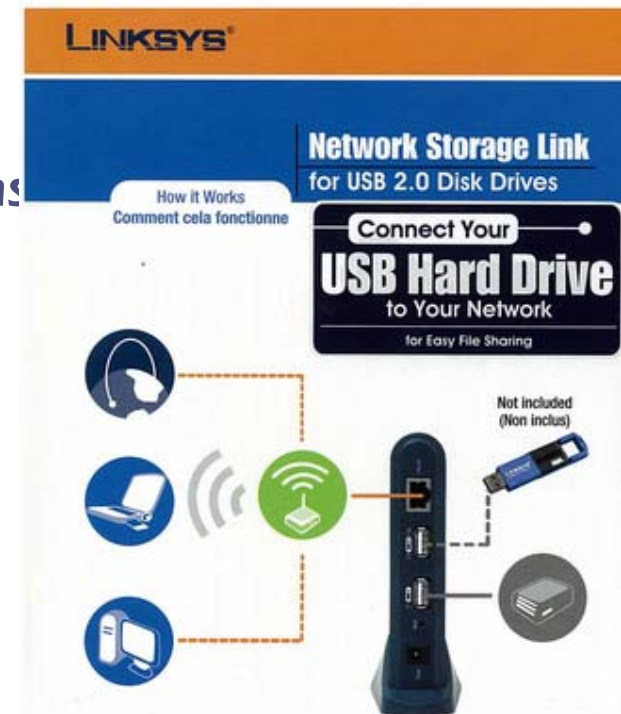
# Network Storage Link for Usb 2

## ✓ NSLU 2

- A storage unit
- Made by LinkSys since 2004
- No longer on sale since 2010
- Around 80€

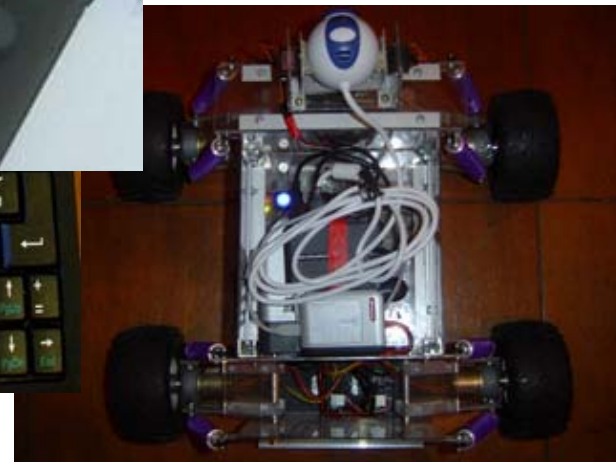
## ✓ Functionalities

- To give a network accessibility to Flash
- Adapted Linux release
- Uses the SMB protocol
- EXT3, NTFS et FAT32 disks
- Web Interface for configuration
  - User and Group Permissions
  - Network options





# Various Uses



# Adding New Services

- ✓ **Alternatives configurations**
  - Web server
  - Mail server
  - Share of media over the local network
    - DAAP (Digital Audio Access Protocol) iTunes
    - Audio/Video UPnP server
  - Client Bittorrent
  - Router (add a new USB network interface)
  - Vocal and chat communication
    - FreeSwitch server
    - Asterix server
  - ... and more with your own ideas
    - Smart Objects

# Hardware Configuration

- ✓ **Compatible ARM processor**
  - XScale-IXP42x Family rev 2 (v5b)
  - 266 MHz (133MHz until 2006)
- ✓ **Memory**
  - 32MB of RAM
  - 8MB of Flash memory
- ✓ **Network**
  - 10/100Mb (Realtek RTL8201CP)
- ✓ **An scalable platform**
  - Adding a new serial port
  - Adding a JTAG port (automate testing of electronic boards)
  - Adding more memory (64MB: FatSLUGs)
  - Automatic startup (ignition)
  - ...



# Newest Models

## ✓ Other newest models

### – NAS200

- Possibility to directly connect SATA hard drives
- About 150-160€



### – WRT600N and WRT300N/350N

- Wifi access point
- Share network connection
- Switch
- Storage link
- About 150€





# Cross Compiling

Produce code for another processor

# Compiling for Another Architecture

## ✓ Cross Compiling

- Faster on the workstation than on the target
- Easier to have a development environment on the workstation on the target
  - Space problem on the target
  - Resources on the target (screen size, keyboard, storage, ...)
- Cross compiling toolchain: compiler with a prefix depending on the name of the architecture
  - `arm-linux-gnueabi-gcc`

# Kernel Cross Compiling

- ✓ **The architecture of the CPU and compiler tool are defined in the Makefile top level**
  - Defined by the `ARCH` and `CROSS_COMPILE` variables
- ✓ **The Makefile defines:**
  - `CC = $(CROSS_COMPILE)gcc`
- ✓ **The simplest way is to redefine these variables:**
  - Example for the ARM architecture
    - `ARCH = arm`
    - `CROSS_COMPILE = arm-linux-gnueabi-`
- ✓ **Solutions of modifications to achieve this:**
  - In the toplevel Makefile
  - On the command line:
    - Be careful to remember the call parameters
  - Redefine the environment variables
- ✓ **Adding the cross compiler to your PATH**

# Choice for a Cross Compiling Toolchain

- ✓ Finding a cross-compiling toolchain is not an easy task
  - Many components to compile (the compiler itself)
  - Choices to do
    - Compiler version, kernel version, C library version, Operating systems tools, ...
  - Many details about which you should be familiar:
    - From configuration to compilation (kernel, tools, ...)
    - gcc versions, the differences and special patches for your architecture
  - Be sure that the toolchain corresponds to your needs
    - CPU, little or big endian, version of libraries, of tools, ....
  - 26 pages HowTo to set up a toolchain:
    - <http://www.aleph1.co.uk/oldsite/armlinux/docs/toolchain/toolchHOWTO.pdf>

# Cross-Compiling Toolchain for Various Architectures

## ✓ ARM

- Code Sourcery (supports GNU/Linux, EABI and uClinux):
  - [http://www.codesourcery.com/gnu\\_toolchains/arm/](http://www.codesourcery.com/gnu_toolchains/arm/)
- Also available for Windows workstations

## ✓ MIPS

- <http://www.linuxmips.org/wiki/Toolchains>

## ✓ Coldfire

- Code Sourcery (supports ELF, GNU/Linux and uClinux):
  - [http://www.codesourcery.com/gnu\\_toolchains/coldfire](http://www.codesourcery.com/gnu_toolchains/coldfire)

## ✓ PowerPC

- Code Sourcery (supports GNU/Linux and EABI)
  - [http://www.codesourcery.com/gnu\\_toolchains/power](http://www.codesourcery.com/gnu_toolchains/power)

# Tools to Build a Cross-Compiling Toolchain

## ✓ Buildroot

- <http://buildroot.uclibc.org/>
- Makefile dedicated to the build of a cross-compiling toolchain based on uClibc
- Also allows the build of a complete filesystem

## ✓ Crosstool

- <http://www.kegel.com/crosstool/>
- Script dedicated to the build of a cross-compiling toolchain based on the glibc
- Do not support uClibc for the moment

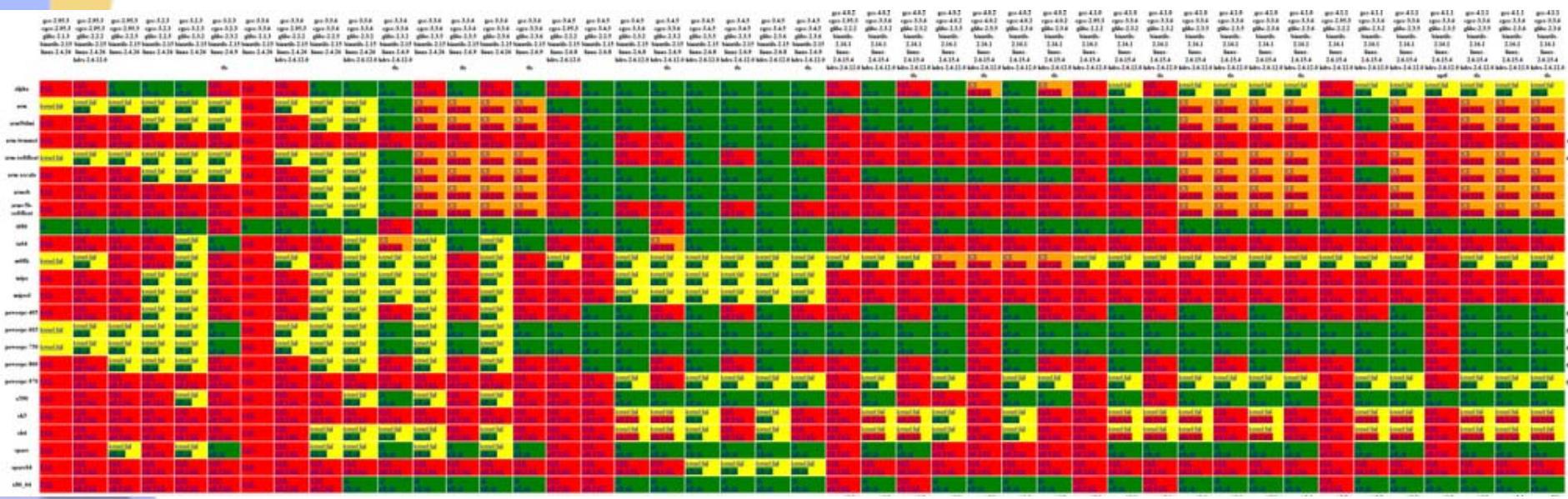
# Buildroot

## ✓ Buildroot

- <http://buildroot.uclibc.org/>
- Supports many architectures
- Automatically downloads the sources and applies necessary patches to the sources
- Can compile most of the applications you need
  - BusyBox, bzip2, Cairo, dbus, Dillo, DirectFB, Dropbear, lighttpd, Python, Qtopia4, sqlite, thttpd, tinyX, xorg...
- Very easy to implement
  - make menuconfig
  - make

# Crosstool: Combinaisons gcc / glibc / binutils / noyau

- ✓ Summary table
  - gcc / glibc / binutils / kernel versions
  - Depending on the architecture
- ✓ Can help you to make a choice



The table displays compatibility for various architectures including alpha, arm, armv7, armv8, armv9, aarch64, avr32, avr32bf, avr32bfr, avr32bfrd, avr32bfrf, avr32bfrg, avr32bfrh, avr32bfrj, avr32bfrk, avr32bfrl, avr32bfrm, avr32bfrn, avr32bfrp, avr32bfrq, avr32bfrs, avr32bfrt, avr32bfru, avr32bfrv, avr32bfrw, avr32bfrx, avr32bfrz, avr32bfr1, avr32bfr2, avr32bfr3, avr32bfr4, avr32bfr5, avr32bfr6, avr32bfr7, avr32bfr8, avr32bfr9, avr32bfr10, avr32bfr11, avr32bfr12, avr32bfr13, avr32bfr14, avr32bfr15, avr32bfr16, avr32bfr17, avr32bfr18, avr32bfr19, avr32bfr20, avr32bfr21, avr32bfr22, avr32bfr23, avr32bfr24, avr32bfr25, avr32bfr26, avr32bfr27, avr32bfr28, avr32bfr29, avr32bfr30, avr32bfr31, avr32bfr32, avr32bfr33, avr32bfr34, avr32bfr35, avr32bfr36, avr32bfr37, avr32bfr38, avr32bfr39, avr32bfr40, avr32bfr41, avr32bfr42, avr32bfr43, avr32bfr44, avr32bfr45, avr32bfr46, avr32bfr47, avr32bfr48, avr32bfr49, avr32bfr50, avr32bfr51, avr32bfr52, avr32bfr53, avr32bfr54, avr32bfr55, avr32bfr56, avr32bfr57, avr32bfr58, avr32bfr59, avr32bfr60, avr32bfr61, avr32bfr62, avr32bfr63, avr32bfr64, avr32bfr65, avr32bfr66, avr32bfr67, avr32bfr68, avr32bfr69, avr32bfr70, avr32bfr71, avr32bfr72, avr32bfr73, avr32bfr74, avr32bfr75, avr32bfr76, avr32bfr77, avr32bfr78, avr32bfr79, avr32bfr80, avr32bfr81, avr32bfr82, avr32bfr83, avr32bfr84, avr32bfr85, avr32bfr86, avr32bfr87, avr32bfr88, avr32bfr89, avr32bfr90, avr32bfr91, avr32bfr92, avr32bfr93, avr32bfr94, avr32bfr95, avr32bfr96, avr32bfr97, avr32bfr98, avr32bfr99, avr32bfr100.

✓ <http://kegel.com/crosstool/crosstool-0.43/buildlogs/>

# ScratchBox

## ✓ ScratchBox

- <http://scratchbox.org/>
- A toolbox project for a system production
- Used by Nokia to develop their products (770, N800, N810)
- Make the cross-compiling esay for a Linux embedded system
- Supported platforms:
  - arm, x86
- Experimental support:
  - ppc, mips, cris
- Support for uClibc and glibc
- Uses the qemu emulator to be able to run the ARM binaries



# OpenEmbedded

## ✓ OpenEmbedded

- <http://wiki.openembedded.net/>
- Supports many architectures
- Easy to customize
- Works on many Linux distributions
- Able to compile many applications (more than 1000 packages)
  - Including GTK+, Xwindows, Mono, Java, ...
- Allow to use glibc or uClibc
- Useful Documentation
  - <http://www.uv-ac.de/openembedded/>



# Cross-Compiling Toolchain: Summary

- ✓ **Build its own toolchain**
  - Hard and long to master
- ✓ **Ready to use toolchain**
  - Available for many platforms
- ✓ **Tool to build a cross-compiling toolchain**
  - [Buildroot](#) et [Crosstool](#)
  - Simplifies creation for specific needs
- ✓ **System build**
  - [ScratchBox](#), [OpenEmbedded](#)
  - Support for the creation and population of a complete system
- ✓ **Resources: [http://elinux.org/Tool\\_Chains](http://elinux.org/Tool_Chains)**