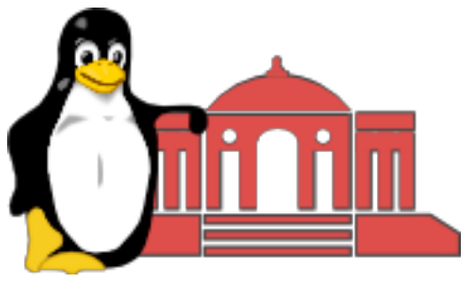
Workshop GNU/Linux embedded

Ottavio Campana <ottavio@campana.vi.it>



Sommario

- Introduzione allo sviluppo embedded
- Introduzione alla scheda
- Introduzione ad OpenEmbedded
- Test con la scheda

Sviluppo embedded

La differenza principale rispetto a sviluppare software per pc consiste nella minore disponibilità di risorse:

- hardware - minore CPU, minore RAM, memorie FLASH al posto dei dischi fissi;
- software - non tutti i programmi per pc sono disponibili per lo sviluppo embedded.

OpenEmbedded

OpenEmbedded è un framework per la generazione di sistemi operativi GNU/Linux embedded.

A differenza di altre soluzioni, è completamente opensource.

Non è legato a nessun archtettura.

OpenEmbedded

Il primo passo per sviluppare un sistema con OpenEmbedded è scaricare il download del framework.

- Per fare questo è necessario:
- scaricare bitbake;
- scaricare il repository di OpenEmbedded.

OpenEmbedded

Non è necessario installare bitbake nel sistema, basta scaricarlo dal suo repository in una cartella locale, perché è scritto in python

dsp@pc:~ $ mkdir stuff

dsp@pc:~ $ cd stuff

dsp@pc:stuff $ svn co svn://svn.berlios.de/bitbake/branches/bitbake-1.8/ bitbake

OpenEmbedded

OpenEmbedded è scaricabile con git:

dsp@pc:stuff $ git clone git://
git.openembedded.org/openembedded

Directory layout

```
stuff
 |---> bitbake
 |---> org.openembedded.dev
          |---> classes
          |---> conf
          |---> contrib
          |---> docs
          |---> file
          |---> recipes
          |---> site
```

Configurazione di OE

```
dsp@pc:stuff $ cd org.openembedded.dev
dsp@pc:org.openembedded.dev $ cd conf
dsp@pc:conf $ mv local.conf.sample local.conf
```

I minimi parametri da impostare sono:

```
BBFILES := "/home/dsp/stuff/org.openembedded.dev/
packages/*/*.bb"

MACHINE = "mx27ads"

DISTRO = "angstrom-2008.1"
```

Variabili d'ambiente

Per terminare la configurazione di Openembedded ed iniziare la compilazione è necessario impostare delle variabili d'ambiente:

```
export PATH=$PATH:${HOME}/stuff/bitbake/bin/

export BBPATH=${HOME}/stuff/build:${HOME}/stuff/
org.openembedded.dev

export BBFILES=${HOME}/stuff/org.openembedded.dev/
packages/*/*.bb
```

Compilazione

Per ricompilare un sistema minimale completo:

```
dsp@pc:org.openembedded.dev $ bitbake minimal-image
```

Questo comando compilerà tutto il framework, includendo tutti i tool necessari per scaricare il software, patcharlo, i compilare, la libreria C, i programmi necessari ad avere un sistema minimale e tutte le

Il risultato della compilazione

Al termine del processo di compilazione si saranno ottenuti:

- il kernel
- il rootfs

L'immagine del kernel dovrà essere esportata mediante TFTP mentre il rootfs via NFS .

Prima del boot

Ci sono due *common issues* che possono far fallire il boot del sistema generato:

- non effettuare un chown -R 0.0 del rootfs copiato nella directory del server NFS
- configuare erroneamente eth0 nel rootfs nel file /etc/network/interfaces. Se si vuole usare il client dhcp del kernel, è necessario commentare la riga con scritto *auto eth0* .

Setup per lo sviluppo

La configurazione più usata durante la fase iniziale dello sviluppo di sistemi embedded è di solito l'uso di una scheda *target* che si appoggia ad un pc *host* che esporta tutto il software necessario.

Questo permette di fare le modifiche al software più rapidamente.

Setup per lo sviluppo

host

Console

RS 232

TFTP, DHCP, NFS

IP

target

Setup per lo sviluppo

Sul computer *host* tre servizi sono necessari:

- DHCP ;
- NFS ;
- TFTP .

Per la console serve invece un emulatore di terminale come minicom o kermit.

Setup per lo sviluppo

Installare il server DHCP

```
sudo apt-get install dhcp3-server
```

Il file di configurazione del servizio è

/etc/dhcp3/dhcpd.conf

Setup per lo sviluppo

Configurare il server DHCP

```
vim /etc/dhcp3/dhcpd.conf
```

Dichiarare una subnet:

```
subnet 192.168.0.0 netmask 255.255.255.0 {
        range 192.168.0.100 192.168.0.150;
}
```

Setup per lo sviluppo

Installare il server NFS

```
sudo apt-get install nfs-kernel-server
```

Il file di configurazione del servizio è

/etc/exports

Setup per lo sviluppo

Configurare il server NFS

```
vim /etc/exports
```

Dichiarare una share:

```
/srv/nfs/rootfs 192.168.0.0/24
(rw,insecure,no_root_squash,sync,subtree_check)
```

Setup per lo sviluppo

Installare il server TFTP

```
sudo apt-get install tftpd
```

Questo servizio non ha file di configurazione, ma i lsuo funzionamento viene specificato da *inetd* .

Setup per lo sviluppo

Configurare il server TFTP

```
vim /etc/inetd.conf
```

Dichiarare un servizio:

```
tftp dgram udp wait nobody /usr/sbin/tcpd /usr/
bin/in.tftpd /srv/tftp
```

Setup per lo sviluppo

Gli emulatori di terminale usati più frequentemente sono *minicom* e *kermit* . Di quest'ultimo esistono più versioni, quella presente in Debian è *ckermit* .

```
apt-get install minicom ckermit
```

Minicom ha una interfaccia basata su curses, kermit è un interprete dei comandi.

Setup per lo sviluppo

```
#!/usr/bin/kermit +
#
# Author: Ottavio Campana <ottavio@campana.vi.it>
# Runs kermit without having to type always the same commands.
#

set modem type none
set line \%1
set carrier-watch off
set speed 115200
set parity none
set stop-bits 1
robust
connect
```

Accendere la scheda

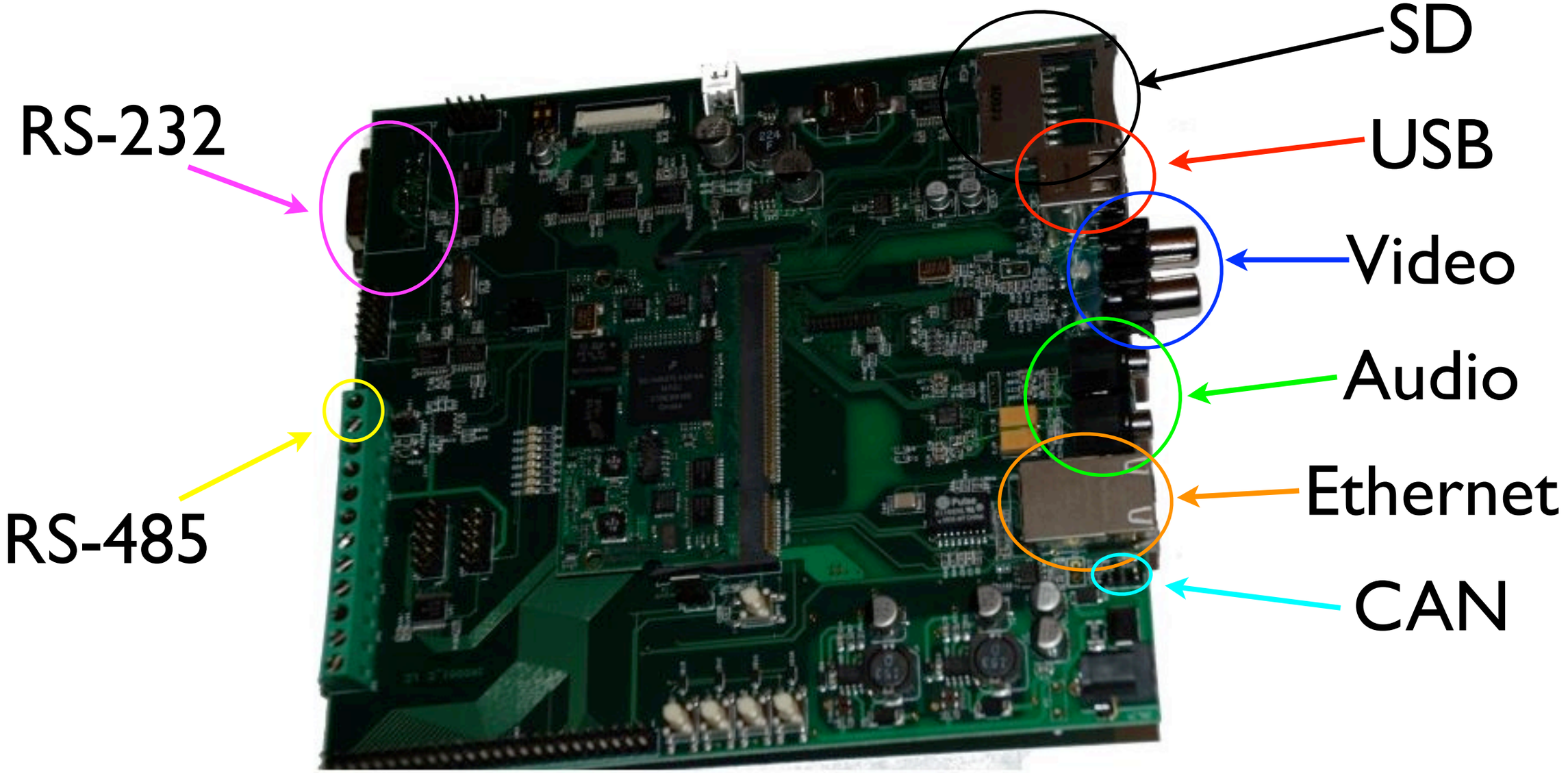
La scheda ha il sistema operativo GNU/Linux pre-installato.

Per seguire il boot della scheda è sufficiente collegare il cavo null modem al pc e alla scheda.

Per comunicare via rete è necessario connettere il cavo di rete.

Il cavo null modem

Connettori scheda



Connessione seriale

Ambiente di sviluppo

La scheda viene a corredo con una versione precompilata di OpenEmbedded.

```
cp angstrom*toolchain.tar.bz2 /
tar xfvj angstrom*toolchain.tar.bz2
```

Questo scompatterà l'ambiente nella directory /usr/local/angstrom/arm

Ambiente di sviluppo

Affinché l'ambiente di sviluppo funzioni correttamente devono essere definite alcune variabili d'ambiente

```
source /usr/local/angstrom/arm/environment-setup
```

Questo imposterà tutte le variabili necessarie, abilitando anche la gestione dei pacchetti ipk.

Il prompt di U-Boot

All'avvio della scheda il primo programma ad essere eseguito è U-Boot, un bootloader usato in molti sistemi embedded.

A differenza di LILO o GRUB, U-Boot ha una shel molto evoluta che permette molte operazioni di debug sull'hardware.

Le immagini dei kernel per U-Boot devono essere create con *make ulmage* .

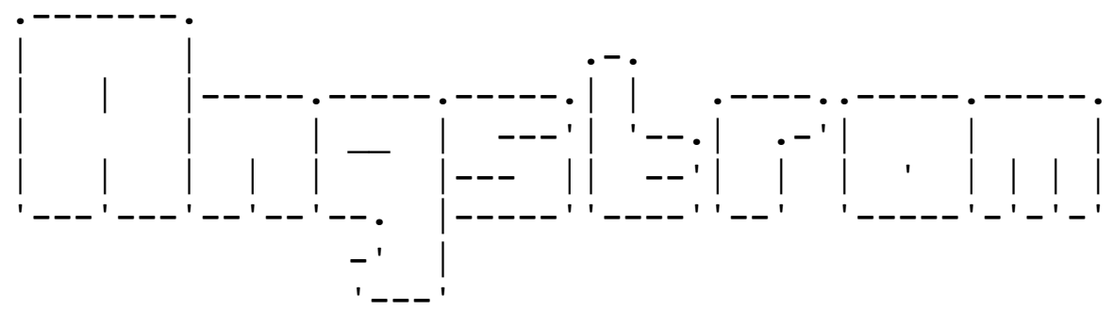
Il prompt di U-Boot

```
U-Boot 2.0.0-rc7-svn3092-dirty2 (Sep 22 2009 - 17:31:50)

Board: M31 SODIMM module
NAND device: Manufacturer ID: 0x20, Chip ID: 0x36 (ST Micro NAND 64MiB 1,8V 8-bit)
Scanning device for bad blocks
Using environment in NAND Flash
chip id: [2,882,1,01d]
mpll:     398999390 Hz
spll:     239999725 Hz
arm:      398999390 Hz
perclk1:   66499898 Hz
perclk2:   66499898 Hz
perclk3:   66499898 Hz
perclk4:   66499898 Hz
clkin26:   26000000 Hz
ahb:      132999796 Hz
ipg:       66499898 Hz
Malloc space: 0xa3b00000 -> 0xa3f00000 (size  4 MB)
Stack space : 0xa3af8000 -> 0xa3b00000 (size 32 kB)
running /env/bin/init...

Hit any key to stop autoboot:  3

type update_kernel nand|nor [<imagename>] to update kernel into flash
type update_root nand|nor [<imagename>] to update rootfs into flash

uboot:/
```

Il prompt di U-Boot

```
uboot:/ printenv
locals:
kernel=nand
root=nand
uimage=uImage
jffs2=rootfs.jffs2
autoboot_timeout=3
nfsroot=/opt/mx27rootfs
bootargs=console=ttymxc0,115200
nand_parts=256k(uboot),128k(ubootenv),3072k(kernel),-(root)
rootpart_nand=/dev/mtdblock3
globals:
PATH=/env/bin
uboot:/
```

Il prompt di GNU/ Linux



```
The Angstrom Distribution mx27 ttymxc0

Angstrom 2009.X-test-20090522 mx27 ttymxc0

mx27 login:
```

Dispositivi scheda

- Gestione dei LED
- Gestione dei bottoni
- Gestione dell'audio
- Gestione del video

Gestione led

Per utilizzare il led sulla scheda è necessario caricare il modulo mxc_gpio

modprobe mxc_gpio

Questo modulo creerà nella directory /dev molti dipositivi /dev/gpio*

Gestione led

Di tutti i dispositivi GPIO, quelli connessi ai led sono:

/dev/gpio069	LED0
/dev/gpio072	LED1
/dev/gpio071	LED2
/dev/gpio074	LED3
/dev/gpio073	LED4
/dev/gpio076	LED5
/dev/gpio075	LED6
/dev/gpio078	LED7

Gestione led

Configurare il GPIO come output

root@mx27:~# echo O > /dev/gpio069

Spegnere il LED

root@mx27:~# echo I > /dev/gpio069

Accendere il LED

root@mx27:~# echo 0 > /dev/gpio069

Gestione bottoni

Il processore iMX.27 è dotato di uno *scan matrix*, che permette di interpretare i bottoni come se fossero parte di una tastiera.

Per utilizzarlo è necessario caricare il driver

modprobe mxc_keyb

Gestione bottoni

Il modulo mxc_keyb crea il file dispositivo

/dev/input/event0

I driver dei dispositivi di input acquisicono tutti gli eventi che vengono generati dai dispositivi HID, quali per esempio tastiere, mouse, joystick, volanti, tavolette grafiche...

Gestione bottoni

Questa classe di driver viene usata leggendo delle strutture input_event che segnalano quello che accade

```
struct input_event {
    struct timeval time;
    unsigned short type;
    unsigned short code;
    unsigned int value;
};
```

Il codice di esempio è nel file evtest.c

Gestione audio

Per gestire l'audio è necessario caricare il modulo:

modprobe snd-soc-mx27-wm8750

Per prima cosa è necessaria eseguire alsamixer e abilitare i canali *Left Mixer* e *Right Mixer* . A questo punto l'audio funziona esattamente come le schede pc.

Gestione audio

I device file relativi ad also si trovano in /dev/snd/ e le informazioni sui driver possono essere trovate in /proc/asound/

Gestione audio

Il sistema preinstallato prevede solamente i comandi amixer e alsactl, ma non ha altri programmio installati

```
root@mx27:~# opkg install alsa-utils-aplay_1.0.18-r1.2_armv5te.ipk
Installing alsa-utils-aplay (1.0.18-r1.2) to root...
Configuring alsa-utils-aplay
root@mx27:~#
```

Allo stesso modo possiamo installare madplay e le sue dipendenze.

Gestione audio

A questo punto, se vogliamo per esempio ascoltare un mp3, possiamo usare il comando:

madplay file.mp3 -o wave:- | aplay

Gestione video

- La gestione del video avviene con le API di Video4Linux