Introduction à la carte FOX LX832

par Yoann Sculo (Site)

Date de publication : 26 août 2009

Dernière mise à jour :

Vous avez toujours rêvé de construire un robot, de mettre en place un frigo 2.0 relié à Internet, de créer votre propre système de vidéosurveillance ou de télécharger vos torrents quand vous dormez ? Cet article est fait pour vous. Il s'agit du premier article d'une série sur le système embarqué appelé carte fox tournant sous Linux. Je vais introduire les premières notions et expliquer comment cette carte fonctionne. De quoi poser les bases pour nous atteler à des projets concrets.

I - Introduction

Le but de ce tutoriel est de faire une présentation du système embarqué tournant sous Linux appelé carte Fox. L'idée est de donner un aperçu des possibilités offertes par le système dans un premier temps, puis d'expliquer comment installer le SDK, communiquer avec la carte et réaliser son premier programme en C. Ce tutorial s'adresse aux personnes qui seraient intéressées par les systèmes embarqués, bidouilles, robotique etc.

Je n'ai pas la prétention de prêcher la parole ultime. On trouve un certain nombre de documentations sur cette carte, et bien souvent en anglais. Ce tutoriel permet de faire un tour des fonctionnalités de base, rassemblant tout ce que j'ai pu trouver sur Internet mais aussi découvrir à travers mes différents projets.

II - Description

II-A - Spécifications de la carte

Carte fox (acmesystems)

Carte fox

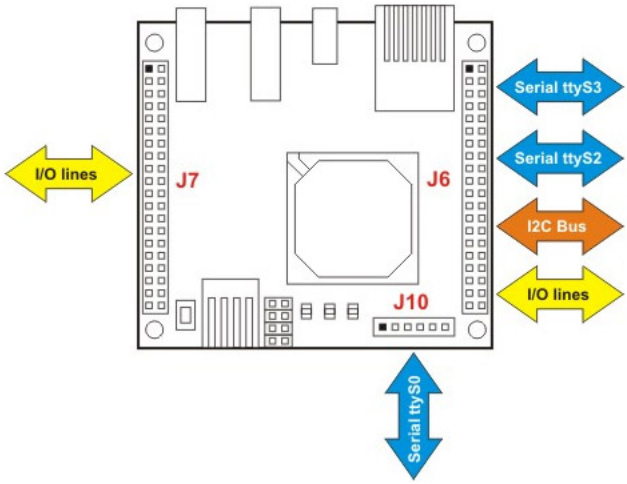


Schéma des entrées sorties (acmesystems)

Schéma des entrées sorties

La carte FOX LX832 est un système embarqué à bas coût utilisant un système d'exploitation Linux et développée par le constructeur italien **Acmesystems**. Elle est dotée d'un processeur AXIS à 100MHz (processeur ETRAX 100LX), de 32 MB de RAM, 8MB de mémoire flash, de connectiques USB, Ethernet, de plusieurs ports séries ainsi que de nombreuses entrées/sorties numériques pouvant être reliées à plusieurs BUS de données. Le système linux est déjà préinstallé sur la carte, ce qui la rend fonctionnelle dès l'achat.

Elle permet de développer en bon nombre de langages : C, C++, PHP, PYTHON, Shell, ... et offre des connexions HTTP, FTP, SSH, TELNET et série. De petite taille et faible consommation électrique (1.4 W), elle est idéale pour mettre en place rapidement des projets de type embarqué.

Ses deux connecteurs 40 broches permettent de connecter la carte Fox à des cartes d'extension (à acheter ou construire séparément) Le constructeur propose des cartes possédant différentes caractéristiques, telles que des connexions Zigbee, GSM/GPRS mais aussi un lecteur de carte mémoire, ou l'ajout d'une horloge système.

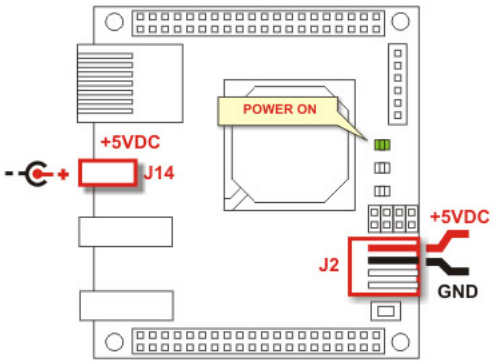
Il est important de noter qu'à partir du 31 Juillet 2010, le processeur de la carte changera au profit d'un processeur ARM9. La carte " FOX Board LX832 " changera de nom pour " FOX Board G20 ". La production de cartes fox LX continuera jusqu'à ce qu'ils ne trouvent plus de processeurs ETRAX 100LX. Le constructeur estime que ses stocks seront épuisés bien après la date du 31 Juillet 2010.

Le prix **dans le commerce** pour cette carte tourne autour de 170 euros.

Note : Acmesystems a annoncé la sortie prochaine (Septembre 2009) de sa nouvelle carte G20 quelque temps après la rédaction de ce tutoriel ... dévoilant toutes les caractéristiques de son nouveau produit. Le constructeur invite ses nouveaux clients à regarder l'éventualité de la nouvelle carte Fox G20. Cette dernière embarque la distribution linux OpenWRT et son architecture se rapproche très fortement de la carte fox LX832 actuelle. Ce tutoriel n'est pas pour autant inutile car on ne trouve pas encore de modèle G20 dans le commerce auprès les revendeurs français, et de part le fait que les cartes seront très proches. Je publierai un nouveau tutoriel complémentaire sur la G20 à l'occasion de la sortie de la nouvelle carte.

II-B - Quelques informations

La carte a besoin d'une alimentation régulée de +5V DC, que cela soit via son connecteur J14 ou le J2. La tension ne doit en aucun cas fluctuer, sinon la carte va tout simplement griller. La carte seule consomme 280mA, valeur qui augmentera plus l'on rajoutera de périphériques. Pour allumer la carte, il suffit de brancher l'alimentation. Le voyant vert indique la mise sous tension.



Alimentation

II-C - Utilisations et possibilités

Internet regorge d'exemples de systèmes embarqués. Il s'avère que la carte Fox est assez souvent utilisée dans des projets amateurs et même professionnels, en raison de son prix attractif et de ses caractéristiques techniques très intéressantes.
Différents projets réalisés avec la carte fox sont listés sur le **site du constructeur**.

On y trouve différentes utilisations possibles :

- Carte mère pour robots mobiles
- Appareil de téléchargement autonome de fichiers torrents.
- Carte mère pour ballon dirigeable
- Streaming vidéo
- Système de contrôle pour des trains (modélisme)
- Système de domotique
- Lecteur de puce RFID
- Serveur GSM / envoi de SMS

Les utilisations sont infinies, et ne dépendent que de l'imagination du développeur !

III - Installation du SDK

L'environnement de développement (SDK) fonctionne à la fois sous Linux et Windows (via l'utilisation d'une machine virtuelle sous linux, le SDK n'étant pas disponible pour les plateformes Windows)
Dans tous les cas, le SDK a besoin pour fonctionner des éléments suivants installés sur votre système :

compilateur GCC C, cross-compilateur CRIS, GNU make, GNU wget, Subversion, awk (ou gawk), bc, byacc (ou yacc), lex ou flex, perl, sed, tar, zlib, md5sum, pmake, curses ou ncurses, bison, which
Nous prendrons les exemples détaillés de Mandriva et Ubuntu, mais le principe est le même pour chaque distribution linux. Pour d'autres distributions, se référer au site d'**acmesystems**

III-A - Sous Linux

Mandriva
A partir d'une distribution Mandriva Linux One 2009 Spring fraichement installée, nous commençons par installer les paquets suivants à l'aide du gestionnaire de logiciels :
subversion, gcc, make, libncurses-devel, zlib1-devel, flex, byacc, bc,

Normalement le système installé possède par défaut tous les autres prérequis nécessaires à l'installation. En cas de package manquant, se référer à la liste en début de partie.
Puis nous téléchargeons le compilateur croisé ainsi que pmake pour l'installation du SDK avec les commandes suivantes :

Shell
```
wget http://foxlx.acmesystems.it/download/cris-dist-1.63-1.i386.rpm
wget http://foxlx.acmesystems.it/download/pmake-1.45-16.i386.rpm
```

Ensuite, en root (su -) nous tapons

Shell
```
rpm -U pmake-1.45-16.i386.rpm
rpm -U cris-dist-1.63-1.i386.rpm
```

Ubuntu
A partir d'une distribution Ubuntu 8.10 vierge, nous commençons par installer les paquets suivant en tapant les commandes apt-get suivantes:

Shell
```
sudo apt-get install make
sudo apt-get install gcc
sudo apt-get install libc6-dev
sudo apt-get install libncurses5-dev
sudo apt-get install pmake
sudo apt-get install zlib1g-dev
sudo apt-get install flex
sudo apt-get install bison
sudo apt-get install subversion
```

Ensuite, sous Ubuntu, le lien symbolique /bin/sh pointe vers /bin/dash au lieu de /bin/bash. Or cela est nécessaire pour notre installation. Ainsi il suffit de faire :

Shell
```
sudo ln -sf /bin/bash /bin/sh
```

pour changer le lien symbolique.
Il faut ensuite télécharger le compilateur croisé :

Shell
```
wget http://foxlx.acmesystems.it/download/cris-dist_1.63-1_i386.deb
```

et installer le package en tapant:

```Shell
dpkg -i cris-dist_1.63-1_i386.deb
```

Procédure commune
Maintenant que tous les packages préliminaires ont été récupérés, nous allons maintenant créer un dossier pour le SDK, télécharger son script d'installation et l'exécuter en tapant :

```Shell
mkdir ~/foxboard
wget http://www.acmesystems.it/download/install_svn_sdk.sh
chmod +x install_svn_sdk.sh
./install_svn_sdk.sh
```

L'installation démarre et télécharge le SDK (devboard-R2_01-distfiles.tar.gz) et commence à l'installer. Au bout de quelques temps apparaît le message suivant :

```Shell
### Selected product: "fox" ###
etrax100boot must be run by root.
To make this easier (but less secure) you can make etrax100boot setuid root.
Do you want to make etrax100boot setuid root now [yn]? (default n):
```

Répondre y pour permettre à etrax100boot d'être lancé en root (programme chargé de flasher la carte fox)
Une fois l'installation terminée, un dossier devboard-R2_01 est alors créé et contient notre SDK.
Il sera alors possible de configurer ce SDK pour changer le noyau de la carte fox en activant/désactivant différentes options et en chargeant différents drivers. Cette manipulation fera l'objet d'un autre tutoriel.
En attendant nous allons d'abord verifier qu'il n'existe pas de mise à jour du SDK en tapant :

```Shell
cd devboard-R2_01
./sdk_update
```

III-B - Sous Windows

Comme indiqué précédemment, il n'est pas possible d'utiliser directement le SDK sous Windows. Il faut passer par une machine virtuelle. A titre d'exemple, j'ai utilisé **VirtualBox** pour ce tutoriel. L'installation d'un système linux avec cette application est relativement simple et intuitive. Il suffit de télécharger une distribution au format .iso et de faire "nouveau" et suivre les indications. Puis en faisant "lancer" nous allons ensuite chercher le .iso fraîchement installé. Une fois le système en place, il n'y a plus qu'à se référer au paragraphe précédent sur la partie linux.

Il est important à noter que pour bénéficier d'une connexion réseau opérationnelle il faudra activer la connexion de pont, sinon la carte fox ne sera pas joignable.

Le constructeur fournit également une image de machine virtuelle préinstallée basée sur debian Sarge 3.1 avec le SDK déjà installé. Le principe est alors le même, il faut par contre installer l'application VMware et double cliquer simplement sur le .vmx contenu dans le fichier zip suivant : http://foxlx.acmesystems.it/download/foxsdk.zip Vous pourrez trouver plus d'informations sur la configuration du SDK sous VMware **sur le site du constructeur**

IV - Connexions à la carte

La connexion à la Fox peut se faire de plusieurs manières

IV-A - Ethernet

Bénéficiant d'un port Ethernet il est possible de la brancher à n'importe quel réseau. La DEL jaune clignotante indique l'activité réseau de la carte.

Par défaut, une carte neuve est configurée de la sorte :

Adresse IP: **192.168.0.90**
Netmask: **255.255.255.0**
Passerelle: **192.168.0.1**

Si votre réseau est configuré de la sorte, la Fox s'y intègre sans problème et celle-ci est accessible directement. L'idéal dans un premier temps est de connecter la carte à notre PC à l'aide d'un câble croisé.
Déjà nous pouvons vérifier la connexion de la carte avec un :

```
Shell
ping 192.168.0.90
```

Si la connexion ne se fait pas, essayons en root la commande suivante:

```
Shell
ifconfig interface_ethernet 192.168.0.10 netmask 255.255.255.0
```

où interface_ethernet est l'interface réseau de notre PC (eth1 dans mon cas). Nous nous allouons alors l'adresse IP 192.168.0.10 pour être dans le même réseau que la FOX.

IV-A-1 - TELNET

On peut se connecter à la carte fox en Telnet via la commande

```
Shell
telnet 192.168.0.90
```

avec
login : root
Password : pass
On peut alors naviguer sur la carte fox simplement. Taper CTRL + D pour quitter la session

IV-A-2 - SSH

Telnet n'est pas sécurisé et il est préférable d'utiliser le protocole sécurisé SSH pour s'y connecter. Il suffit alors de taper :

```
Shell
ssh root@192.168.0.90
```

Et de taper le mot de passe. De la même façon, pour quitter la connexion, taper CTRL + D

IV-A-3 - Web

La carte fox embarque un serveur web accessible tout simplement sur http://192.168.0.90/
La page qui apparaît alors est située dans le répertoire /usr/html/index.html. Cependant ce dossier se situe dans la mémoire RAM de la carte (voir plus loin d'architecture de la carte) et toute modification dans ce dossier sera perdue au prochain démarrage. Pour rajouter ses propres pages web il faudra déposer les fichiers dans /usr/html/local accessibles sur http://192.168.1.90/local/
J'aborderai certainement plus en profondeur le serveur web de la fox et les scripts CGI dans un autre tutoriel.

IV-A-4 - Envoi de fichiers

Parce qu'il faut bien envoyer nos programmes développés sur notre PC directement sur la fox, il y a possibilité d'utiliser ftp et scp.
Pour le ftp, rien de plus classique, connectez-vous avec un client FTP classique avec les infos suivantes :
Hôte : **192.168.0.90**
Identifiant : **root**
Mot de passe : **pass**
Port : **21**
Vous pouvez alors déposer vos fichiers où bon vous semble.

Mais il est possible également d'utiliser scp, qui permet de faire du transfert de fichiers via ssh. Ainsi, tapée depuis votre PC, la commande suivante permet d'envoyer un ou plusieurs fichiers manuellement sur la carte fox. Le mot de passe sera par contre demandé.

Shell
```
scp <fichier> root@192.168.0.90:<dossier_absolu>
```

IV-A-5 - Connexion Réseau et Internet

Et comme il peut être utile de connecter la carte à autre chose qu'à un seul PC, nous allons voir comment nous connecter à un réseau local, voir Internet.
Cela commence tout d'abord par un changement de l'adresse IP. Ce dernier peut être fait via :

Shell
```
ifconfig eth0 x.x.x.x
```

où x.x.x.x est la nouvelle adresse. Mais la configuration ne sera pas reportée lors du prochain redémarrage. Par conséquent, il faut inscrire cela en dur dans la mémoire ROM de la carte en allant ouvrir avec vi ou easyedit le fichier etc/conf.d/net.eth0. Nous modifions alors le bas du fichier:

# If you are using DHCP the following variables will not be used.
IP="192.168.1.90"
NETMASK="255.255.255.0"
BROADCAST="192.168.1.255"
GATEWAY="192.168.1.1"

Cette configuration est faite pour s'inscrire dans un réseau local de Neufbox.
Maintenant que notre carte fox est reconnue sur le réseau, il faut lui donner accès à Internet. Par défaut, aucun serveur DNS n'est spécifié sur la carte, il faut donc modifier sa valeur en tapant :

Shell
```
nameserver 192.168.1.1
```

De la même façon, la configuration sera perdue au prochain démarrage. Il faut donc éditer le fichier /etc/resolv.conf.def en lui précisant bien l'adresse du serveur DNS, càd en y inscrivant nameserver 192.168.1.1

Maintenant il peut être intéressant de donner à nom notre carte fox auprès du serveur DNS. En associant 192.168.1.90 à fox par exemple. Il sera alors possible de se connecter via un ssh root@fox, ce qui est tout de même plus pratique à taper.

IV-B - Série

La carte fox possède plusieurs ports série, dont un port console (port J10 sur la carte). En connectant ce port à un PC grâce à un câble RS232 classique, nous avons accès à la console linux. Cette technique peut s'avérer extrêmement utile dans le cas d'une erreur de manip qui bloque la connexion Ethernet. Exemple, si vous vous trompez dans l'adresse IP à allouer à la carte fox et que la connexion n'est plus possible en Ethernet après reboot, il y a possibilité d'aller corriger le problème grâce au port série. A noter qu'il est possible de réinitialiser la carte dans sa configuration d'usine en allumant et éteignant avec le bouton SW1 enfoncé pendant environ 10 secondes. C'est assez radical comme solution, mais ça a le mérite d'exister.

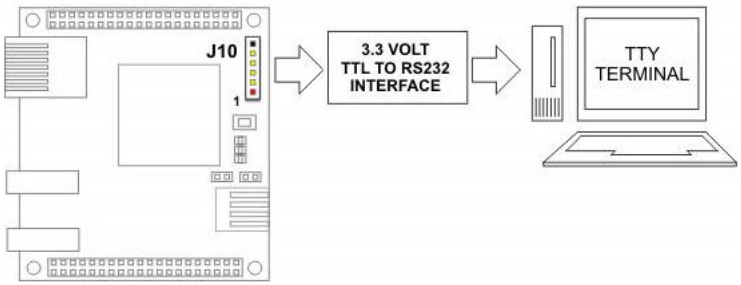
Il faut par contre posséder :

- un port série de libre sur le PC (port série sur carte PCI par exemple)
- un câble RS232
- un adaptateur série pour la carte Fox
- L'hyperterminal sous windows ou minicom sous linux

L'adaptateur série est nécessaire car le protocole RS232 fonctionne sur PC à 12V et la carte fox délivre du 3.3V. Il y a possibilité de faire soit même l'adaptateur ou d'en acheter un en même temps que la Fox.

La connexion se configure alors de la sorte:
Débit : **115200 bauds**
Bits de données : **8**
Parité : **Aucune**
Bits d'arrêt : **1**
Contrôle de flux : **aucun**
En choisissant votre port série selon votre configuration.

L'accès au terminal est alors direct. A noter que de part sa nature de " console " le port J10 (/dev/ttyS0) envoie des trames au démarrage de la carte. Donc faire attention tout de même à ne pas utiliser ce port comme port série classique, sinon votre périphérique va recevoir un gros flux de données qui ne lui sont pas destinées (périphérique fou !) Il y a toutefois possibilité de désactiver le mode console du port série (à vos risques et périls)



Utilisation du port série console

IV-C - Autres

Bénéficiant de 2 ports USB, la carte fox peut être connectée à différents périphériques USB tels que des dongles Wifi ou bluetooth. Elle peut également être reliée à une webcam ou tout autre périphérique usb, du moment que les drivers sont présents sur la Fox ou bien que vous les avez à votre disposition. Cependant ces points ne seront abordés ici, et trouveront leur place peut être dans d'autres tutoriels.

V - Architecture de la carte

L'arborescence de la carte est relativement classique pour un système linux. Cependant de part sa nature embarquée, il est important de noter la différenciation RAM/ROM et l'utilisation qu'il faut en faire.

En effet, tout le système linux est chargé en mémoire au démarrage et tout est perdu à chaque redémarrage, sauf tout ce qui se trouve dans /mnt/flash (càd /dev/flash2). Ce dossier est monté sur de la mémoire flash (4.6Mo) et contient tous les fichiers de configuration du système, en l'occurrence tout ce qui doit rester en mémoire après un reboot. Cette mémoire EPROM n'est pas infinie comme on pourrait le croire. A chaque modification de fichier, toute la mémoire est reprogrammée. Et cette reprogrammation a un cycle de vie autour de 100 000 utilisations. En d'autres termes, évitez d'y sauvegarder des logs, compteurs ou autres informations régulièrement rafraichies. Il est plutôt préférable de stocker le tout en mémoire RAM, /tmp par exemple et de travailler sur des sauvegarde espacées via service cron qui n'écriront sur la mémoire flash que toutes les X heures/jours.

Il peut être également intéressant de noter que tous les fichiers présents dans le dossier /etc/init.d/boottime seront exécutés au démarrage.

VI - Programmer en C

La carte Fox offre plusieurs possibilités en termes de langages de programmation, nous allons commencer par un exemple basique en C pour illustrer le fonctionnement de la compilation.

Tout d'abord, commençons par écrire un programme basique (bonjour.c) que nous allons compiler et envoyer sur la carte .

bonjour.c
```c
#include <stdio.h>
int main()  {
  printf(" Bonjour !\n " ) ;
  return 0 ;
}
```

Il existe deux méthodes pour compiler un programme en C.

VI-A - Web compilateur

La première consiste à se rendre à **cette adresse**. Acmesystems met à disposition un compilateur web gcc qui vous propose de compiler un fichier source avec le compilateur adapté à la Fox. Il suffit alors d'aller chercher notre fichier bonjour.c sur notre disque et de cliquer sur compiler. Nous recevons alors un fichier .out que nous allons envoyer sur la carte Fox grâce à la commande suivante :

Shell
```
scp bonjour.out root@192.168.0.90:/mnt/flash/
```

Il faut alors indiquer à Linux que notre fichier .out est un exécutable. On se connecte alors à la Fox et se rend dans le répertoire /mnt/flash pour exécuter le code suivant :

Shell
```
chmod +x bonjour.out
```

Il ne reste alors plus qu'à l'exécuter en tapant :

Shell
```
./bonjour.out
```

L'intérêt de cette méthode est de permettre le développement de programmes très rapidement sur une machine n'ayant pas le SDK d'installé. L'inconvénient est tout de même de ne pas pouvoir linker notre code à d'éventuelles bibliothèques.

VI-B - Cross-Compilateur

La seconde méthode est un peu plus complexe. Elle nécessite en effet l'installation du SDK, ce que nous venons de faire. Pour pouvoir commencer à développer, nous allons devoir activer l'environnement, c'est-à-dire charger les variables d'environnement qui vont pointer vers notre compilateur croisé. Il faut déjà se rendre dans notre dossier devboard-R2_01 créé lors de l'installation du SDK, puis taper :

Shell
```
. init_env
```

Nous nous rendons alors dans apps puis créons un nouveau dossier

Shell
```
cd apps
mkdir helloworld
```

Nous pouvons ainsi y déposer notre bonjour.c
Nous allons ensuite changer le compilateur habituel, au profit de celui proposé par notre SDK en tapant :

Shell
```
make cris-axis-linux-gnu
```

Le programme qui sera généré par la suite ne pourra pas être exécuté par notre système Linux. Pour revenir à notre compilateur gcc standard et tester en local, il suffit de taper :

Shell
```
make host
```

Nous allons alors avoir besoin d'un Makefile pour pouvoir compiler notre programme. Ne serait-ce que pour charger toutes les informations relatives à la cross-compilation présentes dans le fichier devboard-R2_01/tools/build/Rules.axis

Makefile
```
AXIS_USABLE_LIBS = UCLIBC GLIBC
include $(AXIS_TOP_DIR)/tools/build/Rules.axis

PROGS = bonjour

all: $(PROGS)
$(PROGS): $(PROGS).o
 $(CC) $(LDFLAGS) $^ $(LDLIBS) -o $@
 scp bonjour root@192.168.1.90:/mnt/flash/

clean:
 rm -f $(PROGS) *.o core
```

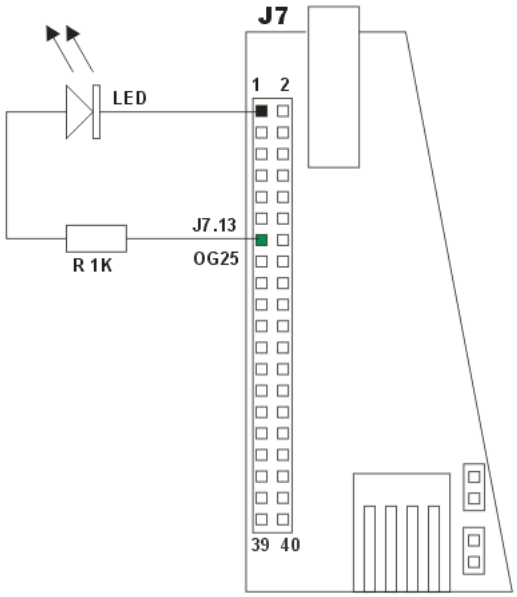
En tapant make, le fichier est alors compilé de la même façon qu'avec le web compilateur. Il peut même être intéressant de glisser l'envoi du fichier dans le makefile via scp. Cela fait gagner un peu de temps si on passe son temps par la suite à compiler et tester son programme sur la carte.
Le programme est alors prêt à être exécuté.

VII - Exemples d'utilisation

Maintenant que nous avons réalisé notre helloworld, nous pouvons nous atteler à des choses plus intéressantes. Je compte aborder un exemple simple de circuit réalisable avec la carte fox. Cela permet de donner un premier aperçu des possibilités sans rentrer dans des circuits très complexes. On pourra trouver sur le site d'**acmesystems** un listing de toutes les entrées sorties présentes sur la carte.

La carte fox délivre une tension logique de 3.3V en sortie jusqu'à 12mA. Ce n'est pas énorme, mais peut être suffisant pour travailler sur de l'électronique numérique (logique).

Ce premier circuit (proposé dans les docs du constructeur) est vraiment très simple mais permet de voir que les entrées/sorties de la FOX sont manipulables directement depuis le shell. Nous allons brancher une simple LED associée à une résistance sur la pine J7.13 (sortie OG25). L'idée est de ne pas dépasser les 12mA. Acmesystems nous conseille de n'utiliser que des LED basse consommation dans l'exemple suivant.



Exemple de connexion de led

VII-A - Commandes setbits et readbits

La commande **setbits** va nous permettre de manipuler les valeurs de sortie de la carte

```
Shell
setbits -p port -b bit -s état
```

où **port** est la valeur de port à utiliser (a, b ou g), **bit** est le numéro de bit à changer et **état** sa valeur binaire (0 ou 1)
On peut alors, allumer notre LED

```
Shell
setbits -p g -b 25 -s 1
```

et l'éteindre

```
Shell
setbits -p g -b 25 -s 0
```

On peut même créer un script shell pour la faire clignoter

Shell
```
#/bin/sh
while [ 1 ]
do
 setbits -p g -b 25 -s 1
 sleep 1
 setbits -p g -b 25 -s 0
 sleep 1
done
```

La commande **readbits** nous permet, quant à elle, de lire la valeur des sorties de la fox.

Shell
```
input=`readbits -p a -b 1`
```

Nous permettra alors de lire le bit 1 du port a. Il deviendra ainsi très simple de tester la valeur d'une entrée pour faire par exemple fluctuer d'autres bits de sortie. On pourra par exemple placer un interrupteur en entrée, qui une fois pressé déclenchera une action prédéfinie par la carte sur les sorties. Les possibilités sont infinies.

VII-B - Les entrées/sorties en C

Travailler avec le shell peut être intéressant car rapide à utiliser. Toutefois si l'on souhaite passer à des programmes plus complexes nous pouvons travailler directement en C. Ce dernier gère sans problèmes les entrées/sorties. L'exemple suivant est la traduction en C du précédent script shell qui fait clignoter la LED :

blink.c
```
#include "stdio.h"
#include "stdlib.h"
#include "unistd.h"
#include "sys/ioctl.h"
#include "fcntl.h"
#include "asm/etraxgpio.h"

int main(void) {
  int fd;
  int i;
  int iomask;

  if ((fd = open("/dev/gpiog", O_RDWR))<0) {
    printf("Open error on /dev/gpiog\n");
    exit(0);
  }

  iomask=1<<25;

  while (1) {
    printf("Led allumée\n");
    ioctl(fd,_IO(ETRAXGPIO_IOCTYPE,IO_SETBITS),iomask);
    sleep(1);

    printf("Led éteinte\n");
    ioctl(fd,_IO(ETRAXGPIO_IOCTYPE,IO_CLRBITS),iomask);
    sleep(1);
  }
  close(fd);
  exit(0);
}
```

Nous commençons par ouvrir le port choisi, en l'occurrence /dev/gpiog pour le port G dans notre cas. De la même façon, nous avons /dev/gpioa et /dev/gpiob pour les ports A et B. Attention à ne pas oublier de fermer les ports ouverts à la fin du programme.

Nous mettons ensuite en place le masque qui permet de définir quels bits nous allons modifier, et l'appliquons grâce à la fonction ioctl. IO_SETBITS permet comme son nom d'indique de mettre à 1 les bits définis par le masque et IO_CLRBITS les passe à 0.

Le principe de lecture de bit en entrée est légèrement différent. On utilise la syntaxe suivante :

```
input=ioctl(fd, _IO(ETRAXGPIO_IOCTYPE, IO_READBITS));
```

ioctl ne renvoie pas directement la valeur du bit, mais un masque de bits correspondant à tout le port. En appliquant un ET logique entre input et notre masque de bits (définissant le bit qui nous intéresse) il nous est possible de connaître la valeur du bit. Voici le code, dans le cas d'une entrée numérique sur le bit n°16 du port G:

blink.c
```
#include "stdio.h"
#include "stdlib.h"
#include "unistd.h"
#include "sys/ioctl.h"
#include "fcntl.h"
#include "asm/etraxgpio.h"

int main(void) {
  int fd;
  int input;
  int iomask;

  if ((fd = open("/dev/gpiog", O_RDWR))<0) {
    printf("Open error on /dev/gpiog\n");
    exit(0);
  }

  iomask=1<<16;

  while (1) {
    input=ioctl(fd, _IO(ETRAXGPIO_IOCTYPE, IO_READBITS));
    if ((input&iomask)==0) {
      printf("Valeur d'entrée : 1\n");
    } else {
      printf("Valeur d'entrée : 0\n");
 }
    sleep(1);
  }
}
```

VIII - Conclusion

Nous avons donc pu faire un tour rapide de la carte fox LX832 et de ses possibilités. Nous ne sommes pas rentrés en détails sur certains points, qui feront certainement l'objet d'autres tutoriels dans un futur proche. Si vous êtes intéressés par des fonctionnalités avancées de la Fox, on trouve des informations détaillées sur son utilisation sur le **site du constructeur**.

Ainsi, on peut voir toutes les perspectives qu'une telle carte nous ouvre ! Elles sont illimitées, et comme je l'avais dit un peu plus haut, ne dépendent que de l'imagination du développeur.

IX - Remerciements

Je tiens à remercie **Alp** pour m'avoir poussé à écrire mon premier tutoriel pour ses suggestions et sa relecture.
Merci également à **RougeCitron** pour la relecture.
Merci enfin à la société Acmesystems qui m'a autorisé à utiliser ses images.