

Parcours de formation SIN

**Déploiement d’une interface de communication web**

Module SIN42 : PLATINE SYSTEME EMBARQUE FOX BOARD G20

Durée : 6h

Objectifs à atteindre : Faire évoluer un système communicant par liaison série RS232 pour qu’il communique via un serveur Web embarqué ou concevoir une centrale de mesure utilisant des capteurs analogiques ou numériques (bus 1 fils) munie d’une interface de communication web et d’une base de données pour le stockage des mesures.

Niveau des connaissances envisageable : Bac STI - BTS

Pré requis :

* Connaissance du système d’exploitation linux en mode terminal (Debian ou Ubuntu)
* Connaissance des protocoles TCP/IP
* Connaissance de l’utilisation d’une liaison série asynchrone RS232
* Connaissance de langages de programmation (C, php,..)
* Connaissance de la manipulation de données de bases de données relationnelles en langage SQL

Systèmes mis en œuvre :

* Carte système embarqué FOX Board G20 de chez ACMESystem   
  (FOX Board G20 combo box 185€ chez le fabricant http://www.acmesystems.it/ ou Pack de développement "FOX BOARD G20" chez lextronic http://www.lextronic.fr/produit.php?id=6427)
* Modem/routeur (livebox, neufbox, …) ou routeur.

Logiciels utilisés :

* VMware + Linux ubuntu
* Sous windows : Putty + WinSCP + notepad++ (freewares)

Outils mobilisés, le cas échéant : Oscilloscope numérique (Tektronix...), multimètre.

Webographie :

<http://www.acmesystems.it/>

<http://www.acmesystems.it/foxg20/doku.php>

<http://www.lextronic.fr/P6389-platine-fox-board-g20.html>

<http://domotique.benchi.fr/robotique/fox-board-g20/premier-pas-avec-la-foxboard-g20/>

<http://domotique.benchi.fr/tag/fox-board/>

<http://www.yoannsculo.fr/tag/g20/>

Bibliographie : GNU/Linux Magazine HS N°51 – DÉCEMBRE 2010/JANVIER 2011

Le livre blanc des systèmes embarqués  
 <http://download.microsoft.com/documents/France/windows/2009/Livre_blanc_Embarque.pdf>

Auteur : Marc Silanus – Académie d’Aix-Marseille – Révision 01/12/2011

Sommaire

[1 La platine FOX Board G20 4](#_Toc288997300)

[2 Connexion 5](#_Toc288997301)

[2.1 Sous linux (ubuntu) 5](#_Toc288997302)

[2.2 Sous Windows 6](#_Toc288997303)

[2.3 Trouvez l’adresse IP de la carte FOX Board G20 6](#_Toc288997304)

[2.3.1 Port débug 6](#_Toc288997305)

[2.3.2 Logiciel ipscan 7](#_Toc288997306)

[2.4 Connexion au site web embarqué 7](#_Toc288997307)

[2.5 Transfert de fichiers 7](#_Toc288997308)

[3 Réglage de la date et de l’heure 8](#_Toc288997309)

[4 Créer une nouvelle carte microSD 9](#_Toc288997310)

[4.1 Préparer la carte microSD 9](#_Toc288997311)

[4.1.1 Identification du périphérique 9](#_Toc288997312)

[4.1.2 Partitionnement et formatage 10](#_Toc288997313)

[4.2 Copie du noyau et du système de fichier 11](#_Toc288997314)

[4.3 Finalisation de la microSD 12](#_Toc288997315)

[4.3.1 Synchroniser les données en mémoire et sur la microSD 12](#_Toc288997316)

[4.3.2 Démontage des partitions 12](#_Toc288997317)

[5 Le serveur web embarqué 13](#_Toc288997318)

[5.1 HTML 13](#_Toc288997319)

[5.2 PHP 14](#_Toc288997320)

[5.3 SQLITE 14](#_Toc288997321)

[5.4 PHP/SQLITE 15](#_Toc288997322)

[6 Programmer en C 16](#_Toc288997323)

[6.1 Premier programme 16](#_Toc288997324)

[6.2 Interfacer SQLITE et C 16](#_Toc288997325)

[7 Programmer en Python 19](#_Toc288997326)

[7.1 Premier programme en Python 19](#_Toc288997327)

[7.2 Interfacer SQLITE et Python 19](#_Toc288997328)

[8 Exécuter une tâche à intervalle régulier : CRON 20](#_Toc288997329)

[9 Utilisation des liaisons séries 22](#_Toc288997330)

[9.1 En ligne de commande (shell) 22](#_Toc288997331)

[9.2 En langage C 22](#_Toc288997332)

[9.3 En Python 25](#_Toc288997333)

[9.4 En PHP 26](#_Toc288997334)

[10 Le CGI 27](#_Toc288997335)

[10.1 Activer le CGI sur le serveur web lighttpd 27](#_Toc288997336)

[10.2 Premier test en C 28](#_Toc288997337)

[10.3 Premier test en Python 29](#_Toc288997338)

[11 Utilisez les ports GPIO 30](#_Toc288997339)

[11.1 Gérer les lignes GPIO à l’aide de l'interface sysfs en ligne de commandes 31](#_Toc288997340)

[11.2 Gérer les lignes GPIO à l’aide de l'interface sysfs en C 32](#_Toc288997341)

[11.3 Gérer les lignes GPIO à l’aide de l'interface sysfs en PHP 33](#_Toc288997342)

[12 Utilisation des convertisseurs analogique/numérique 37](#_Toc288997343)

[12.1 Téléchargement, compilation et installation du driver 37](#_Toc288997344)

[12.2 Lecture d’une ligne analogique en PHP 41](#_Toc288997345)

[13 Utilisation du bus 1 fil 42](#_Toc288997346)

[13.1 Le 1-Wire bus (bus 1 fil) 42](#_Toc288997347)

[13.2 Mesure de température 43](#_Toc288997348)

[13.3 Projet 45](#_Toc288997349)

[13.3.1 Mise en œuvre des capteurs de température 46](#_Toc288997350)

[13.3.2 Enregistrement périodique avec CRON 48](#_Toc288997351)

[13.3.3 Interface web 49](#_Toc288997352)

[14 En savoir plus 51](#_Toc288997353)

# La platine FOX Board G20

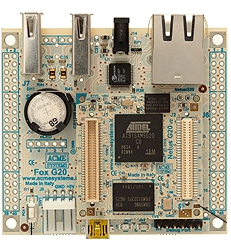
La platine "FOX Board G20" est un système embarqué avec un OS Linux, construite autour d’un processeur ARM9™ AT91SAM9G20 cadencé à 400 MHz d'Atmel™.

Elle est composée d'une platine support avec étage de régulation multiple (2 x 3,3 V / 1,8 V / 1 V) sur laquelle est insérée le module CPU « NETUS G20 » (lequel intègre le processeur AT91SAM9G20).

La platine support fourni :

* un connecteur d'alimentation,
* un connecteur Ethernet (Base 10/100),
* 2 ports USB 2.0 host,
* un port client sur mini USB,
* une pile de sauvegarde pour horloge RTC,
* un bouton-poussoir libre d'utilisation,
* un emplacement pour afficheurs uOLED intelligents de "4D Systems",
* un connecteur mâle au pas de 2,54 mm pour le raccordement du port série "console/Debug"
* une zone de prototypage rapide disposant de ports série, ports d’entrées/sorties parallèle, I2C/SPI, convertisseurs A/N, PWM, …

Ethernet 10/100BaseT



Connexion afficheur uOLED

Ports USB

12 Mb/s

Zone de prototypage

Connecteur J7

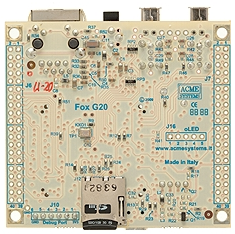
Zone de prototypage

Connecteur J6

Module Netus G20

Port mini USB

Port Debug



Le premier élément de code exécuté lors du démarrage est le RomBOOT. Ce firmware est stocké dans le processeur Atmel. Il cherche un code exécutable sur la mémoire Dataflash. Cette dernière est programmée par défaut avec le chargeur de démarrage appelé AcmeBoot, qui démarre linux à partir de la carte microSD.

Carte microSD. Elle contient deux partitions :

* FAT16 : Noyau (Kernel)
* EXT4 : système de fichiers.

Capacité maximale de 8 Go.

Détail des connexions disponibles : [http://foxg20old.acmesystems.it/doku.php?id=hw:foxg20pinout](http://www.acmesystems.it/foxg20/doku.php?id=hw:foxg20pinout)

# Connexion

Wiki du fabricant  : [http://foxg20old.acmesystems.it/doku.php?id=tutorial:gettingstarted](http://www.acmesystems.it/foxg20/doku.php?id=tutorial:gettingstarted)

|  |  |
| --- | --- |
| Insérez la carte microSD amorçable contenant l’OS linux dans le lecteur. | http://foxg20old.acmesystems.it/lib/exe/fetch.php?media=tutorial:microsd\_socket.jpg |
| Connectez la carte au réseau à l’aide d’un câble LAN.  Connectez l’alimentation à la carte.  Connectez l’alimentation au secteur. | [http://foxg20old.acmesystems.it/lib/exe/fetch.php?media=foxg20:power\_and\_lan.jpg](http://www.acmesystems.it/foxg20/lib/exe/fetch.php?media=foxg20:power_and_lan.jpg) |
| Lorsque la carte est alimentée, le voyant vert noté 3V3 s’éclaire.  Si vous êtes connecté au port débug, vous voyez la séquence de démarrage à l’écran. | http://foxg20old.acmesystems.it/lib/exe/fetch.php?media=tutorial:power\_led.jpg |
| Après environ 5 secondes, le voyant rouge noté PC7 se met à clignoter. S’il reste allumé sans clignoter, c’est que la carte microSD n’est pas présente, n’est pas bootable ou le système qu’elle contient est défectueux. | http://foxg20old.acmesystems.it/lib/exe/fetch.php?media=tutorial:pc7\_led.jpg |

Après environ 20 secondes, le système est opérationnel et il est possible de s’y connecter par le réseau au moyen d’une connexion *ssh*, à condition de connaitre l’adresse IP de la carte.

|  |
| --- |
| **Remarque** : Le mot de passe de *root* par défaut est **netusg20**. |

## Sous linux (ubuntu)

|  |
| --- |
| Ouvrez un terminal, octroyez vous les droits super-utilisateur et ouvrez une connexion *ssh*. |
|  |

## Sous Windows

Wiki du fabricant  : [http://foxg20old.acmesystems.it/doku.php?id=tutorial:ssh\_access](http://www.acmesystems.it/foxg20/doku.php?id=tutorial:ssh_access)

|  |
| --- |
| Vous devez disposer du logiciel Putty (<http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe>) .  Exécutez Putty et établissez une connexion *ssh* avec la carte. |
|  |

## Trouvez l’adresse IP de la carte FOX Board G20

Wiki du fabricant  : [http://foxg20old.acmesystems.it/doku.php?id=tutorial:howtodiscovertheipaddress](http://www.acmesystems.it/foxg20/doku.php?id=tutorial:howtodiscovertheipaddress)

Par défaut, la carte est configurée en adressage automatique. Elle attend donc qu’un serveur DHCP lui attribut ses paramètres TCP/IP.

Pour trouver l’adresse IP qui lui a été attribuée, vous pouvez utiliser le port debug ou un logiciel de scan d’adresses IP (ipscan.exe par exemple).

### Port debug

|  |  |
| --- | --- |
| Le port debug (DPI : Debug Port Interface) permet d’établir une connexion avec la carte au moyen d’une connexion USB qui émule une liaison série RS232.  Il est construit autour d’un FTDI FT232 qui ne nécessite pas d’installation de pilotes sous linux. En revanche, sous Windows, il faudra télécharger et installer les pilotes de chez FTDI.   * Sous Windows, on pourra utiliser Hyperterminal. * Sous linux, on pourra utiliser Minicom.   Les caractéristiques de la liaison sont : 115200bds, 8 bits, 1 Stop, NP, pas de contrôle de flux  Voir les tutoriels vidéo. | Port debug  http://www.acmesystems.it/www/DPI/dpionfox.jpg |

### Logiciel ipscan

|  |  |
| --- | --- |
| Pour trouver l’adresse IP fournie par le DHCP, vous pouvez utiliser le logiciel ipscan sous Windows.  Ce logiciel scanne toutes les adresses IP entre les limites que l’utilisateur a précisées et fournit le nom des périphériques identifiés. |  |

## Connexion au site web embarqué

Wiki du fabricant  : [http://foxg20old.acmesystems.it/doku.php?id=tutorial:web\_access](http://www.acmesystems.it/foxg20/doku.php?id=tutorial:web_access)

|  |
| --- |
| Utilisez un navigateur web (IE, Mozilla Firefox, Conqueror, …) et saisissez dans la barre d’adresse l’adresse IP de la FOX Board G20. Vérifiez la prise en charge de l’extension php en cliquant sur le lien « see phpinfo ». |
|  |

## Transfert de fichiers

Wiki du fabricant : [http://foxg20old.acmesystems.it/doku.php?id=tutorial:scp\_access](http://www.acmesystems.it/foxg20/doku.php?id=tutorial:scp_access)

|  |  |
| --- | --- |
| Pour transférer des fichiers, la carte FOX Board G20 dispose nativement du protocole SCP (Secure Copy Protocol).   * Sous Windows, vous pouvez utiliser WinSCP. * Sous linux (ubuntu) :   + commande **scp** dans un terminal :   **scp <file> <username>@<IP address>:<Destination>**   * + Le navigateur de fichier (Nautilus)   **Voir les tutoriels vidéo.** |  |

# Réglage de la date et de l’heure

Au premier démarrage de la FOX Board G20, il est peu probable que la date et l’heure soit correcte.

La commande date permet de consulter ou de régler la date et l’heure sur un système linux.

|  |
| --- |
| **debarm:~#** date  Fri Feb 18 16:47:32 MST 2011 |

Pour modifier la date et l’heure :

|  |
| --- |
| **debarm:~#** date –s mm/dd/yy  **debarm:~#** date –s HH:MM:SS |

Pour fixer la date et l’heure matérielle (RTC) à l’heure internationale de référence :

|  |
| --- |
| **debarm:~#** hwclock --systohc --utc |

* La commande hwclock interroge et positionne l'horloge matérielle RTC.
* La fonction systohc (*system to hardware clock*) indique que l’on souhaite régler l'horloge matérielle à l'heure courante du système.
* L’option utc indique que l'horloge RTC est conservée au format universel UTC.

L'heure UTC (Universal Time Coordinated), en français Temps Universel Coordonné, est l'heure de référence internationale. Elle correspond à l'heure GMT (Greenwich Mean Time).   
Lorsqu'il est 0 heure UTC, il est minuit à Greenwich (Angleterre), sur le méridien de longitude zéro.

Pour fixer la date et l’heure matérielle (RTC) à l’heure locale :

|  |
| --- |
| **debarm:~#** hwclock --systohc --localtime |

|  |
| --- |
|  |

# Créer une nouvelle carte microSD

Wiki du fabricant  : [http://foxg20old.acmesystems.it/doku.php?id=dev:bootable-microsd](http://www.acmesystems.it/foxg20/doku.php?id=dev:bootable-microsd)

La carte microSD contient deux partitions :

* kernel : partition FAT16 qui contient le noyau linux (Version 2.6.35.4 mars 2011).
  + uImage : fichier binaire du noyau téléchargeable ici :  
    <http://foxg20.acmesystems.it/download/microsd_20100922/uImage>
* rootfs : partition ext4 qui contient le système de fichier linux.
  + Ensemble des dossiers et fichiers de l’arborescence du système de fichier de linux.  
    Archive téléchargeable ici :  
    <http://foxg20.acmesystems.it/download/microsd_20100922/rootfs.tar.bz2>

## Préparer la carte microSD

En général, les cartes microSD sont constituées d’une seule partition principale formatée en FAT16. Il nous faudra donc détruire cette partition et en créer deux nouvelles conformément à l’architecture décrite ci-dessus.

### Identification du périphérique

Les supports de stockage de masses sont identifiés dans le dossier */dev* par des noms plus ou moins explicites indiquant la nature de leur connexion.

Ainsi,

* un disque dur IDE s’appelle hda pour le premier, hdb pour le second, etc…
* un disque dur SCSI ou USB ou SATA s’appelle sda pour premier, sdb, etc …

Si vous ne possédez qu’un disque dur SATA ou SCSI, une clé USB est donc identifiée sous le nom sdb.

Les partitions contenues dans un périphérique de stockage de masse est représentée par son numéro d’ordre. Ainsi, la première partition de sda s’appelle sda1, la seconde sda2, …

Une clé USB ne dispose en générale que d’une seule partition. Dans les mêmes conditions que ci-dessus, elle s’appellera donc sdb1.

|  |
| --- |
| **ATTENTION :**  le système de fichier à construire étant de format ext4, spécifique à linux, cette opération ne peut être faite que sous linux.  Les opérations de ce chapitre ont été réalisées sur une machine virtuelle créée avec VMware et avec OS linux Ubuntu. |

Insérez la microSD dans un adaptateur USB / microSD et insérez-le dans un port USB de l’hôte (machine virtuelle active sinon, c’est l’OS de l’hôte qui va la montée et non la VM).

Exécutez la commande :

|  |
| --- |
| marco@marco-virtual-machine:~$ **sudo su**  [sudo] password for marco:  root@marco-virtual-machine:/home/marco# **fdisk -l** |

La commande fdisk –l permet de lister les périphériques de stockage de masse.

Elle affiche aussi leur partitionnement et leurs tailles ainsi que celles des partitions.

|  |
| --- |
| Ma carte miniSD  2Go  1 partition FAT16  Mon disque dur :  SCSI de 20Go  Créer sous VMware |

### Partitionnement et formatage

|  |
| --- |
| **ATTENTION :**  Assurez-vous d’avoir bien identifier votre microSD comme indiqué dans la partie précédente. Ici, ma microSD s’appelle sdb. Le partitionnement et le formatage détruira irrémédiablement les données présentes sur le support indiqué.  Faites attention que sdb ne soit pas un de vos disque dur par exemple. |

Commençons par «démonter» la microSD, c’est-à-dire la déconnecter du système de fichier :

|  |
| --- |
| root@marco-virtual-machine:/home/marco# **umount /dev/sdb1** |

Détruisons le partitionnement existant sur sdb :

|  |
| --- |
| root@marco-virtual-machine:/home/marco# **parted /dev/sdb –s rm 1** |

Créons les deux partitions sur sdb : fat16 de 32Mo et le reste en ext4.

|  |
| --- |
| root@marco-virtual-machine:/ # **parted /dev/sdb**  (parted) **mkpart**  Type de partition ? primary/primaire/extended/étendue? **primaire**  Type de système de fichiers ? [ext2]? **fat16**  Début ? **0**  Fin ? **32M**  Avertissement: L'alignement de la partition ainsi définie n'est pas optimal au niveau performance.  Ignorer/Ignore/Annuler/Cancel? **ignore**  (parted) **mkpart**  Type de partition ? primary/primaire/extended/étendue? **primaire**  Type de système de fichiers ? [ext2]? **ext4**  Début ? **32M**  Fin ? **-1**  (parted) **print**  Modèle: ChipsBnk SD/MMCReader (scsi)  Disque /dev/sdb : 1967MB  Taille des secteurs (logique/physique) : 512o/512o  Table de partitions : msdos  Numéro Début Fin Taille Type Système de fichiers Fanions  1 512B 32,0MB 32,0MB primary fat16 lba  2 32,5MB 1967MB 1935MB primary ext4  (parted) **quit** |

|  |
| --- |
| -1 : jusqu’à la fin des 2Go |

Il ne reste plus qu’a formater les deux partitions. La première en FAT16 nommée kernel :

|  |
| --- |
| root@marco-virtual-machine:/ # **mkdosfs /dev/sdb1 –n kernel** |

La seconde en ext4 nommée rootfs :

|  |
| --- |
| root@marco-virtual-machine:/ # **mke2fs –t ext4 /dev/sdb1 –L rootfs** |

|  |
| --- |
|  |

Retirez l’adaptateur USB/microSD, puis après quelques secondes, rebranchez-le. Les deux partitions nouvellement créées seront automatiquement montées dans le dossier /media.

* /media/kernel
* /media/rootfs

## Copie du noyau et du système de fichier

Téléchargez le noyau linux et l’archive du système de fichier :

* kernel : <http://foxg20.acmesystems.it/download/microsd_20100922/uImage>
* rootfs : <http://foxg20.acmesystems.it/download/microsd_20100922/rootfs.tar.bz2>

Copiez ensuite le fichier uImage dans la partition kernel.

|  |
| --- |
| root@marco-virtual-machine:/ # **cp /home/marco/Téléchargements/uImage /media/kernel/** |

Décompactez l’archive du système de fichiers et copiez le contenu dans la partition rootfs.

|  |
| --- |
| root@marco-virtual-machine:/# **tar xvjpSf /home/marco/Téléchargements/rootfs.tar.bz2 -C /media/kernel/** |

## Finalisation de la microSD

### Synchroniser les données en mémoire et sur la microSD

|  |
| --- |
| root@marco-virtual-machine:/ # **sync** |

### Démontage des partitions

|  |
| --- |
| root@marco-virtual-machine:/ # **umount /media/kernel**  root@marco-virtual-machine:/ # **umount /media/rootfs** |

Retirez l’adaptateur USB/microSD, puis retirez la carte microSD de l’adaptateur.

Insérez la carte microSD dans la FOX Board G20 et démarrez-la.

# Le serveur web embarqué

Le serveur web embarqué de la FOX Board G20 est un serveur Lighttpd (<http://www.lighttpd.net/>).

Ce serveur présente, entre autre, l’avantage d’être bien plus léger qu’un serveur Apache.

Les fichiers du site web embarqué sont situés dans le dossier */var/www*.

Par défaut, ce dossier contient la page d’accueil *index.html*.

## HTML

Créez une page de test HTML sur votre ordinateur et transférez-la dans le dossier du serveur web embarqué. Testez l’affichage de la page dans un navigateur.

|  |
| --- |
| <HTML>  <HEAD>  <meta http-equiv="Content-Type" content="text/html"; charset="utf-8" />  <TITLE>Page Test</TITLE>  </HEAD>  <BODY BGCOLOR="#FF9966">  <H1><CENTER><B>Bonjour &agrave; tous!</B></CENTER></H1>  <H2><CENTER><B>Bienvenue</B></CENTER></H2>  <P>Vous &ecirc;tes ici sur une <B>page de test....</B></P>  <P>Pour vous rendre &agrave; la page d'accueil, <A HREF="index.html"><B><I>cliquez ici!</I></B></A></P>  </BODY>  </HTML> |

|  |
| --- |
| Adresse IP de la FOX Board G20 |

## PHP

PHP pour Lighttpd est installé par défaut sur la FOX Board G20.

Créez une page de test PHP sur votre ordinateur et transférez-la dans le dossier du serveur embarqué. Testez l’affichage de la page.

|  |
| --- |
| <HTML>  <HEAD>  <meta http-equiv="Content-Type" content="text/html"; charset="utf-8" />  <TITLE>Page Test</TITLE>  </HEAD>  <BODY>  <?php  if(!isset($\_POST["valider"]))  {  echo'<p>Quel est ton prénom :</p>'; echo'<form method="post">'; echo'<p><input type="text" name="prenom" />  <input type="submit" name="valider" value="Valider" /></p>' ; echo'</form>' ;}else{ echo'<p>Bonjour !</p>' ; echo"<p>Tu t’appelles ".$\_POST["prenom"].'</p>' ; echo'<p>Si tu veux changer de prénom, <a href="testphp.php">clique ici  </a> pour revenir à formulaire.php</p>' ;}  ?>  </BODY>  </HTML> |

|  |  |
| --- | --- |
|  |  |

## SQLITE

Le serveur de base de données (SGBD) de la carte FOX Board G20 est SQLITE.

Il n’est pas installé par défaut. Pour l’installer et permettre sa prise ne charge par PHP, suivez les instructions suivantes :

|  |
| --- |
| **debarm:~#** apt-get update  **debarm:~#** apt-get install sqlite3  **debarm:~#** apt-get install php5-sqlite  …  **debarm:~#** /etc/init.d/lighttpd restart |

La dernière commande permet de redémarrer le serveur web afin qu’il puisse gérer les accès au serveur de bases de données SQLITE3.

Malheureusement, SQLITE3 ne dispose pas d’interface graphique de gestion comme PhpMyAdmin pour MySQL. Il faudra donc apprendre à utiliser l’interface en lignes de commandes.Créez une nouvelle base de données *basetest* à l’aide de l’instruction sqlite3, créez une table test et insérez-y des données. Affichez le contenu de la table :

|  |
| --- |
| **debarm:/var/www#** sqlite3 basetest  SQLite version 3.5.9  Enter ".help" for instructions  **sqlite>** create table test ("nom" VARCHAR(20), "prenom" VARCHAR(20), "etablissement" TEXT, "ville" VARCHAR(20));  **sqlite>** insert into test values("silanus","marc","Lycee A. BENOIT","L'ISLE SUR LA SORGUE");  **sqlite>** insert into test values("toto","robert","Lycee A. BENOIT","L'ISLE SUR LA SORGUE");  **sqlite>** select \* from test;  silanus|marc|Lycee A. BENOIT|L'ISLE SUR LA SORGUE  toto|robert|Lycee A. BENOIT|L'ISLE SUR LA SORGUE  **sqlite>** .quit  Création de la base de données  **debarm:/var/www#**  Création de la table |

|  |
| --- |
| Sélection des données  Insertion des données |

## PHP/SQLITE

Comme pour interfacer PHP et MySQL (vue dans le module SIN413), on utilisera l’extension **PDO** (*PHP Data Objects*).

Créez une page de test SQLITE avec PHP sur votre ordinateur et transférez-la dans le dossier du serveur embarqué. Testez l’affichage de la page.

|  |
| --- |
| <?php  // Connexion à la base de données  $dbh = new PDO("sqlite:/var/www/basetest");  // Requète de selection  $sql = "SELECT \* FROM test";  // Affichage des résultats  foreach ($dbh->query($sql) as $row)  {  echo $row[nom]." ".$row[prenom]." - ".$row[etablissement].  " - ".$row[ville]."<br>";  }  $dbh = null;  ?> |

|  |
| --- |
|  |

# Programmer en C

## Premier programme

Nous commencerons par le traditionnel « *helloword* » :

|  |
| --- |
| // hellword.c  #include <stdio.h>  int main(void)  {  printf("Hello World !\n ");  return 0; } |

Pour compiler le programme, on utilise le compilateur **gcc** disponible dans système.

Le fichier *helloword.c* est enregistré ici dans */root*.

|  |
| --- |
| Syntaxe :  **debarm:/root#** gcc <fichier source> -o <fichier de sortie>  Ici :  **debarm:/root#** gcc helloword.c -o helloword |

Exécutez le programme :

|  |
| --- |
| **debarm:/root#** ./helloword |

Remarque : l’exécution se fait partir du dossier courant (./). Sil fichier à exécuter est hors du dossier courant, on saisit le chemin d’accès ( */root/helloword* ).

Compilation

|  |
| --- |
| Exécution |

## Interfacer SQLITE et C

Les opérations sur les bases de données sont effectuées soit directement par l’utilisateur au moyen de la console sqlite3, soit au moyen d’un programme (en C, Python, script Shell, …).

Pour interfacer le gestionnaire de base de données SQLITE et le langage C, il faut inclure dans la libraire libsqlite3-dev qui donne accès aux structures, fonctions et constantes permettant les opérations sur les bases de données :

* sqlite3\_stmt : Structure qui permet de préparer l’accès aux tables
* sqlite3\_open() : Ouvrir une base de données
* sqlite3\_exec() : Exécuter une requête (première méthode)
* sqlite3\_prepare\_v2() : Exécuter une requête (deuxième méthode)
* sqlite3\_column\_count() : Nombre de colonnes de la table
* sqlite3\_column\_name() : Nom des colonnes de la table
* sqlite3\_column\_text() : Valeurs d’un enregistrement
* SQLITE\_ROW : Constante indiquant la présence de donnée dans la ligne
* SQLITE\_DONE : Fin de la table

Consultez la documentation officielle de SQLITE : <http://www.sqlite.org/cintro.html>

Installez la librairie libsqlite3-dev :

|  |
| --- |
| **debarm:~#** apt-get install libsqlite3-dev |

|  |
| --- |
|  |

Saisissez le programme suivant qui ajoute deux lignes à la table *test* créée précédemment. Compilez-le et exécutez-le.

|  |
| --- |
| #include<stdio.h>  #include<sqlite3.h> // apt-get install libsqlite3-dev  #include<stdlib.h>  int main(int argc, char\*\* args)  {  //Créez une variable de type int pour stocker le code de retour pour chaque appel  int retval;  char \*query;  // Préparer l'accès aux tables  sqlite3\_stmt \*stmt;  // Créer un handle pour la connexion à base de données  sqlite3 \*handle;    // Essai de connexion à la base de données  retval = sqlite3\_open("basetest",&handle);//Si la connexion ne marche pas, handle retourne NULL  if(retval)  {  printf("Impossible de se connecter à la base de données\n");  return -1;  }  printf("Connexion réussie\n");    query = "INSERT INTO test VALUES('tyty','robert','Lycee Albert Camus','Paris')";  retval = sqlite3\_exec(handle,query,0,0,0);  query = "INSERT INTO test VALUES('gogo','marcel','Lycee Thiers','Marseille')";  retval = sqlite3\_exec(handle,query,0,0,0);  // Lire le contenu de la table  query = "SELECT \* from test";  retval = sqlite3\_prepare\_v2(handle,queries,-1,&stmt,0);  if(retval)  {  printf("Impossible de lire la table\n");  return -1;  }  // Lire le nombre de lignes récupérées  int cols = sqlite3\_column\_count(stmt);  // Ecriture de l'entête des colonnes  int col;  for(col=0 ; col<cols;col++) printf("%s \t\t | ",sqlite3\_column\_name(stmt,col));  printf("\n");  //Ecriture des données  while(1)  {  // récupérer le status de la ligne (contient de données ou fin de table)  retval = sqlite3\_step(stmt);  if(retval == SQLITE\_ROW)  {  // La ligne contient des données  int col;  for(col=0 ; col<cols;col++)  {  // sqlite3\_column\_text retourne un const void\* => cast en const char\*  const char \*val = (const char\*)sqlite3\_column\_text(stmt,col);  printf("%s \t\t | ",val);  }  printf("\n");  }  else if(retval == SQLITE\_DONE)  {  // Plus de données  printf("Fin de la table\n");  break;  }  else  {  // Erreur  printf("Erreur lors de l'accès aux données\n");  return -1;  }  }  // Fermeture du handle pour libérer la mémoire  sqlite3\_close(handle);  return 0;  } |

Pour le compiler, il faut pr

Préciser au compilateur gcc d’utiliser la librairie libsqlite3-dev :

|  |
| --- |
| **debarm:~#** gcc testsqlite3.c –o testsqlite3 -l sqlite3  **debarm:~#** ./testsqlite3 |

l

|  |
| --- |
| Vérification directe dans la base de données  Lecture de la table par le programme en C |

# Programmer en Python

Python est un langage de programmation à typage dynamique qui à été développé en 1989 par Guido Van Rossum et de nombreux bénévoles.

Python fait partie des langages de script alors que Java, C++ et C sont des langages qui nécessitent une compilation. Les langages de script sont plus rapides au développement que les autres. Les programmes comportent moins de lignes (environ 50 % de moins), par contre leur vitesse d'exécution est plus lente. De plus, la place mémoire prise par ses langages lors de l'exécution d'un programme est plus grande qu'en C/C++.

Python est installé par défaut sur la FOX Board G20.

Consultez la documentation officielle : <http://docs.python.org/>

## Premier programme en Python

Nous commencerons par le traditionnel « *helloword* » :

|  |
| --- |
| # Programme helloword.py  print "Hello Word !" |

Exécutez le programme :

|  |
| --- |
| **debarm:/root#** pyton helloword.py |

|  |
| --- |
| Exécution avec la commande python suivis du nom du fichier avec l’extension .py |

## Interfacer SQLITE et Python

Créez une base de données *basecron* :

|  |
| --- |
| **debarm:~#** sqlite3 basecron  **SQLite version 3.5.9**  **Enter ".help" for instructions**  **sqlite>** create table tablecron ("num" INTEGER PRIMARY KEY AUTOINCREMENT, "date" DATE);  **sqlite>** .quit  **debarm:~#** |

Cette table contient deux champs :

* num : entier, clé primaire et auto-incrémenté
* date : de type date, enregistre la date et l'heure de la création de l'enregistrement

Le script suivant enregistre la date et l'heure de son exécution dans la table *tablecron* de la base de données *basecron* :

|  |
| --- |
| **#!/user/bin/python2.5**  **Shebang** : cette ligne indique le que programme qui suit doit être interprété par l’interpréteur *python2.5* situé dans */user/bin*.  import time  import sqlite3  # Inserer des donnees dans la table  connection = sqlite3.connect("/root/basecron")  cursor = connection.cursor()  cursor.execute("INSERT INTO tablecron ('date') VALUES(datetime('now','localtime')) ")  connection.commit() |

Remarque : Comparez avec le même programme C page 16.

|  |
| --- |
| Date et heure actuelle pour vérification  Nouvelle entrée dans la base de données  Exécution |

# Exécuter une tâche à intervalle régulier : CRON

**Cron** est un service de linux utilisé pour programmer des tâches devant être exécutées à un moment précis. Les actions et leurs périodicités sont indiquées dans le fichier **crontab**.

Pour ouvrir et modifier **crontab**:

|  |
| --- |
| **debarm:~#** crontab -e |

La syntaxe est la suivante :

|  |
| --- |
| **# m h dom mon dow command** |

Chaque ligne du fichier correspond à une commande que l'on veut voir exécutée régulièrement.

|  |  |  |
| --- | --- | --- |
| <http://www.siteduzero.com/tutoriel-3-73916-executer-un-programme-a-une-heure-differee.html> | Exemples : | |
| **Crontab** | **Signification** |
| 47 \* \* \* \* commande | Toutes les heures à 47 minutes exactement.  Donc à 00h47, 01h47, etc. |
| 0 0 \* \* 1 commande | Tous les lundis soir à minuit. |
| 0 4 1 \* \* commande | Tous les premiers du mois à 4h du matin. |
| 0 4 \* 12 \* commande | Tous les jours du mois de décembre à 4h du matin. |
| 0 \* 4 12 \* commande | Toutes les heures les 4 décembre. |
| \* \* \* \* \* commande | Toutes les minutes ! |

Remarque : la commande sera exécutée dans le dossier personnel de l’utilisateur, ici root.

|  |
| --- |
| # Ecrire Salut toutes les minutes dans /root/testcron.log  \*\*\*\*\* echo "Salut" >> testcron.log |

Nous allons utiliser **cron** pour enregistrer des données à intervalle régulier dans une table. La base de données doit donc être préalablement créée.

Nous utiliserons *basecron* définie dans la partie précédente.

**Remarque** : Le champ **date** enregistrera la date et l’heure système sous le format suivant :

**aaaa-mm-jj hh:mm:ss**.

Il est donc nécessaire de régler la date et l’heure du système conformément à la partie 3 « Réglage de la date et de l’heure ».

Le programme en C suivant ajoute les lignes à la table :

|  |
| --- |
| #include<stdio.h>  #include<sqlite3.h> //apt-get install libsqlite3-dev  #include<stdlib.h>  int main(int argc, char\*\* args)  {  // Créez une variable de type int pour stocker le code de retour pour chaque appel  int retval;  char \*query;  // Préparer l'accès aux tables  sqlite3\_stmt \*stmt;  // Créer un handle pour la connexion à base de données  sqlite3 \*handle;  // Essai de connexion à la base de données  retval = sqlite3\_open("/root/basecron",&handle);  // Si la connexion ne marche pas, le handle retourne NULL  if(retval)  {  printf("Impossible de se connecter à la base de données\n");  return -1;  }  printf("Connexion réussie\n");  query = "INSERT INTO tablecron ('date') VALUES(datetime('now','localtime'))";  retval = sqlite3\_exec(handle,query,0,0,0);  // Fermeture du handle pour libérer la mémoire  sqlite3\_close(handle);  return 0;  } |

Compilez et exécutez le programme. Vérifiez le contenu de la table. Une ligne a dû être ajoutée.

|  |
| --- |
| **debarm:~#** gcc testcron.c –o testcron –l sqlite3  **debarm:~#** ./testcron  **debarm:~#** Connexion réussie  **debarm:~#** sqlite3 basecron  SQLite version 3.5.9  Enter ".help" for instructions  **sqlite>** select \* from tablecron;  1|2010-11-27 22:59:22  **sqlite>** .quit  **debarm:~#** |

Rendez la tâche répétitive :

|  |
| --- |
| **debarm:~#** crontab –e  # Ecrire dans une table toutes les minutes  # Un commentaire commence par #  \* \* \* \* \* /root/testcron |

|  |
| --- |
|  |

**Remarque** : le fichier *crontab* s’ouvre dans l’éditeur de texte nano installé par défaut dans la FOX Board G20 et qui est courant sous linux.

* Pour enregistrer les modifications : Ctrl + O
* Pour quitter : Ctrl + X

Après quelques minutes, consultez la table :

|  |
| --- |
| **debarm:~#** sqlite3 basecron  SQLite version 3.5.9  Enter ".help" for instructions  **sqlite>** select \* from tablecron;  1|2010-11-27 22:59:22  2|2010-11-27 23:00:36  3|2010-11-27 23:01:38  4|2010-11-27 23:20:37  **sqlite>** .quit  **debarm:~#** |

L’exécution de la tâche définie dans le fichier crontab se fait de manière silencieuse, on dit couramment « en tâche de fond ». Ceci illustre bien le caractère multi-tâche de l’OS linux.

# Utilisation des liaisons séries

Wiki du fabricant  : <http://foxg20.acmesystems.it/doku.php?id=tutorial:serial_port>

La FOX Board G20 dispose de plusieurs ports séries :

|  |
| --- |
| **ATTENTION :**  Les ports série fonctionnent en 3,3V.  Ne tolère pas le 5V.  [http://foxg20old.acmesystems.it/doku.php?id=tutorial:serial\_port](http://www.acmesystems.it/foxg20/doku.php?id=tutorial:serial_port) |

Le fabricant propose un module de connexion aux ports séries (Daisy RS232 module et Daisy adapter). Consultez le site du fabricant à l’adresse suivante :

<http://foxg20.acmesystems.it/doku.php?id=daisy:daisy9_rs232>

## En ligne de commande (shell)

**Sous Linux, chaque port série est représenté par un fichier de périphérique situé dans le dossier /dev.**

Connectez les lignes RX et TX du port COM4 (ttyS5) et COM3 (ttyS4) entre elles et testez l’envoi et la réception de caractères à partir du terminal (ouvrez deux terminaux simultanément) :

|  |  |
| --- | --- |
| **Premier terminal :**  **debarm:~#** echo "test d’ecriture sur le ttyS5" > /dev/ttyS5 | **Deuxième terminal :**  **debarm:~#** cat /dev/ttyS4  test d’ecriture sur le ttyS5  CTRL+C  **debarm:~#** |

|  |
| --- |
| Ctrl + C pour stopper la lecture |

## En langage C

Dans un programme en C, on ouvre ces fichiers comme n'importe quel autre fichier grâce à la fonction **open()** :

|  |
| --- |
| int fd;  if ((fd = open("/dev/ttyS1",O\_RDWR))<0)  {  fprintf (stderr,"Erreur d’ouverture %s\n", strerror(errno));  exit(-1);  } |

Une fois que le port est ouvert, on peut y lire et y écrire des caractères au moyen des fonctions **read()** et **write()**.

Mais auparavant, il faut configurer les paramètres de la liaison. Ils répondent à la norme **POSIX** et sont regroupés dans une structure nommée **termios** définie dans le fichier *termios.h* qu'il faut inclure. Ce fichier est situé dans */usr/include* et */usr/include/bits*.

Cette structure comporte les champs suivants

|  |
| --- |
| struct termios {  tcflag\_t c\_iflag; /\* modes d'entrée \*/  tcflag\_t c\_oflag; /\* modes de sortie \*/  tcflag\_t c\_cflag; /\* modes de contrôle \*/  tcflag\_t c\_lflag; /\* modes locaux \*/  cc\_t c\_cc[NCCS]; /\* caractères de controle \*/  }; |

Consultez la documentation de termios pour plus d’informations (en français, sous licence GPL, traduite par Christophe Blaess ([ccb@club-internet.fr](mailto:ccb@club-internet.fr))) :

<http://www.linux-kheops.com/doc/man/manfr/man-html-0.9/man3/termios.3.html>

Ecrire des données : *ecrireSerialPort.c*

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <unistd.h>  #include <fcntl.h>  #include <errno.h>  #include <sys/types.h>  #include <sys/stat.h>  #include <stdlib.h>  #include <stdarg.h>  #include <termios.h>  int main(void)  {  int fd;  struct termios termios\_p;  /\* Ouverture de la liaison serie \*/  if ( (fd=open("/dev/ttyS4",O\_WRONLY))<0)  {  fprintf (stderr,"Erreur d’ouverture %s\n", strerror(errno));  exit(-1);  }    /\* Lecture des parametres courants \*/  tcgetattr(fd,&termios\_p);  /\* On ignore les BREAK et les erreur de parité\*/  termios\_p.c\_iflag = IGNBRK | IGNPAR;  /\* Pas de mode de sortie particulier \*/  termios\_p.c\_oflag = 0;    /\* Liaison a 9600 bps avec 8 bits de donnees \*/  termios\_p.c\_cflag = B9600 | CS8;  /\* Mode non-canonique sans echo \*/  termios\_p.c\_lflag &= ~(ECHO);  termios\_p.c\_lflag &= ~(ICANON);  /\* Caracteres immediatement disponibles \*/  termios\_p.c\_cc[VMIN] = 1;  termios\_p.c\_cc[VTIME] = 0;  /\* Sauvegarde des nouveaux parametres \*/  tcsetattr(fd,TCSANOW,&termios\_p);  int i=0;  while(i<10)  {  char a\_transmettre[100];  sprintf(a\_transmettre,"%d - test de transmission",i);  write(fd,a\_transmettre,strlen(a\_transmettre));  i++;  sleep(1);  printf(a\_transmettre);  printf("\nnombre de caractères transmis : %d\n\n",strlen(a\_transmettre));  }  write(fd,"fin.",4);  /\* Fermeture \*/  close(fd);  /\* Bye... \*/  exit(0);  } |

Lire des données : *lireSerialPort.c*

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <unistd.h>  #include <fcntl.h>  #include <errno.h>  #include <sys/types.h>  #include <sys/stat.h>  #include <stdlib.h>  #include <stdarg.h>  #include <termios.h>  int main(void)  {  int fd;  struct termios termios\_p;  /\* Ouverture de la liaison serie \*/  if ( (fd=open("/dev/ttyS5",O\_RDONLY))<0)  {  fprintf (stderr,"Erreur d’ouverture %s\n", strerror(errno));  exit(-1);  }    /\* Lecture des parametres courants \*/  tcgetattr(fd,&termios\_p);  /\* On ignore les BREAK \*/  termios\_p.c\_iflag = IGNBRK | IGNPAR;  /\* Pas de mode de sortie particulier \*/  termios\_p.c\_oflag = 0;    /\* Liaison a 9600 bps avec 8 bits de donnees et une parite paire \*/  termios\_p.c\_cflag = CREAD | B9600 | CS8; //CREAD  /\* Mode non-canonique sans echo \*/  termios\_p.c\_lflag &= ~(ECHO);  termios\_p.c\_lflag &= ~(ICANON);  /\* Caracteres immediatement disponibles \*/  termios\_p.c\_cc[VMIN] = 1;  termios\_p.c\_cc[VTIME] = 0;  /\* purge donnees non lues ou non envoyees \*/  tcflush(fd, TCIFLUSH);  /\* Sauvegarde des nouveaux parametres \*/  tcsetattr(fd,TCSANOW,&termios\_p);  int fin=0;  while(!fin)  {  //lecture de 100 caractères max  char c[100]="";  read(fd,&c,100);  if(c[strlen(c)-1]=='.') fin=1;  printf("%s",c);  printf("\nnombre de caractères reçus : %d\n\n",strlen(c));  /\* La transmission se termine par un . \*/    }  /\* Fermeture \*/  close(fd);  /\* Bye... \*/  exit(0);  } |

Compilez et exécutez ces deux programmes dans des terminaux différents après avoir connecter entre eux les ports ttyS4 et ttyS5 comme au paragraphe précédant.

Vérifiez que les données sont bien transmises d’un port à l’autre.

# En Python

Pour transmettre des caractères sur le port ttyS5:

|  |
| --- |
| **#!/usr/bin/python2.5**  #serialPortPrint.py  import serial  ser2 = serial.Serial(port='/dev/ttyS5', baudrate=9600, timeout=1, parity=serial.PARITY\_NONE, stopbits=serial.STOPBITS\_ONE, bytesize=serial.EIGHTBITS) s = ser2.read(1) # lire un caractere  chaine = ""while (s != "q"):  s = ser2.readline() # lire une ligne (fini par un retour chariot) if (s != ""):  print s  ser2.close() |

Pour recevoir des caractères sur le port ttyS4 :

|  |
| --- |
| **#!/usr/bin/python2.5**  #serialPortInput.py  import serial    ser1 = serial.Serial(port='/dev/ttyS4', baudrate=9600, timeout=1, parity=serial.PARITY\_NONE, stopbits=serial.STOPBITS\_ONE, bytesize=serial.EIGHTBITS)  x = "rien"  while (x != "q"):  #print x  x = raw\_input("caractere a transmettre(q pour arreter) : ")  ser1.write(x)  ser1.close() |

Connectez les ports ttyS4 et ttyS5 entre eux et testez les programmes dans deux terminaux simultanément. Encore une fois, constatez la compacité des programmes en Python par rapport à ceux en C.

## En PHP

Il peut s’avérer intéressant de pouvoir disposer d’une interface web capable d’intervenir sur un système physique (commande de la rotation ou du zoom d’une caméra IP motorisée, modifications des paramètres réseau d’un système communicant, transmission d’une chaine de caractères à afficher sur un afficheur industriel, …). Plusieurs solutions techniques sont alors envisageables :

* les CGI (*Common Gateway Interface*), qui sont des programmes écrit en C/C++, Python, Perl, … et exécutés sur le serveur. Ils peuvent renvoyer du contenu HTML au client et agir directement sur le système hôte hébergeant le serveur (configuration, liaisons séries, parallèles, …).
* Les langages web dynamiques (php, asp, aspx) qui sont des langages de script interprétés via un serveur http. Ils peuvent aussi fonctionner localement comme n’importe quel langage interprété et ainsi agir sur le système hôte.

**Attention** : L’utilisateur par défaut du serveur web (lighttpd) est www-data. Jusqu’à présent, lorsque nous exécutions un programme écrit en C ou en Python, l’utilisateur était root. Un coup d’œil sur le contenu du dossier */dev* nous indique que le propriétaire des ports séries est root et le groupe est dialout.

|  |
| --- |
| Port debug  C : périphérique de lecture ou écriture de caractères  r : accès en lecture  w : accès en écriture |

Par conséquent, un script PHP exécuter à partir d’un navigateur web ne sera pas autorisé à accéder aux ports séries puisqu’il sera exécuté par www-data. Il faut donc soit modifier les droits d’accès au ports pour permettre à tous de lire ou d’écrire, ce qui peut représenter un problème de sécurité, soit intégrer l’utilisateur www-data dans le groupe dialout afin qu’il puisse bénéficier des droits qui lui sont accordés, puis redémarrer le serveur : /etc/init.d/lighttpd restart

|  |
| --- |
| Modification de www-data pour l’ajouter au groupe dialout  Les groupes aux quel appartient www-data |

Pour transmettre des caractères sur le port ttyS4 :

|  |
| --- |
| <HTML>  <HEAD>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <TITLE>Page Test</TITLE>  </HEAD>  <?php  if(!isset($\_POST["valider"]))  {  echo'<h1>Chaine à transmettre sur ttyS4 :</h1>';  echo'<form method="post">';  echo'<br><input type="text" name="chaine" />  <input type="submit" name="valider" value="Valider" />';  echo'</form>';  }  else  {  echo'<h1>Transmission de données sur un port série</h1>';  echo'<br>Transmission sur ttyS4 de '.$\_POST["chaine"];  $port = fopen("/dev/ttyS4","w");  fwrite ($port,$\_POST["chaine"]);  fclose($port);  echo'<br><a href="serie.php">clique ici</a> pour transmettre une nouvelle chaine';  }  ?>  </BODY> |

|  |  |
| --- | --- |
|  |  |

# Le CGI

Un script CGI (*Common Gateway Interface*) est un programme exécuté sur le serveur web (« côté serveur ») et capable de générer du code HTML qui sera transmis au navigateur.

Ce programme peut être écrit en divers langages compilés ou interprétés comme par exemple le C/C++, le Perl, le Python, …

Nous traiterons ici uniquement le C et le Python au travers de quelques exemples simples.

## Activer le CGI sur le serveur web lighttpd

La configuration du serveur web lighttpd se trouve dans le dossier */etc/lighttp*.

On y trouve :

* **lighttpd.conf** : fichier de configuration à partir duquel les directives sont chargées.
* **conf-available** : dossier qui contient les fichiers de configuration des différents modules.
* **conf-enabled** : dossier qui contient des liens vers les modules disponibles pour les activer.

Pour activer les modules, on utilise la commande **lighty-enable-mod** :

|  |
| --- |
| **debarm:~#** lighty-enable-mod cgi |

Cette commande a pour action de créer un lien nommé *10-cgi.conf* dans */etc/lighttpd/conf-enabled* qui pointe sur le fichier du même nom dans */etc/lighttpd/conf-available*.

Editez ce fichier et modifiez la directive **HTTP["$url"]** comme suit :

|  |
| --- |
| $HTTP["url"] =~ "^/cgi-bin/" {  cgi.assign = ( ".py" => "/usr/bin/python",  ".php" => "/usr/bin/php-cgi",  "" => "" ) |

Cette directive indique au serveur le chemin des interpréteurs des langages utilisés.

Enregistrez les modifications et redémarrez le serveur avec la commande suivante :

|  |
| --- |
| **debarm:~#** /etc/init.d/lighttpd restart |

Il est à noter que comme tous les services de linux qui doivent démarrer au démarrage du système, l’exécutable *lighttpd* se trouve dans le dossier */etc/init.d*

Les scripts CGI doivent être localisés dans un dossier bien précis sur le serveur : *cgi-bin*. Créez ce dossier à la racine du site, c’est-à-dire dans */var/www*

|  |
| --- |
| **debarm:~#** mkdir /var/www/cgi-bin |

## Premier test en C

Saisissez le programme suivant et enregistrez le sous le nom *test.c* dans le dossier *cgi-bin*.

|  |
| --- |
| #include<stdio.h>  main()  {  printf("Content-type: text/html\n\n");  printf("<html>\n");  printf("<head><title>Mon premier script CGI</title>\n");  printf("<meta http-equiv='content-type' content='text/html; charset=utf-8'></head>\n");  printf("<body bgcolor=\"#FFFF00\">\n");  printf("<br><br><br><br>\n");  printf("<center>\n");  printf("<h1>Salut à tous<br>voici mon premier script CGI</h1>\n");  printf("</center>\n");  printf("</body>\n");  printf("</html>\n");  } |

Compilez le programme et donnez comme extension au fichier de sortie *.cgi*

|  |
| --- |
| **debarm:/var/www/cgi-bin#** gcc test.c –o test.cgi |

Testez le programme dans un navigateur :

|  |
| --- |
|  |

## Premier test en Python

Saisissez le programme suivant et enregistrez le sous le nom *test.py* dans le dossier *cgi-bin*.

|  |
| --- |
| **#!/usr/bin/python2.5**  print 'Content-type: text/html'  print '''  <html>  <head><title>Mon premier script CGI</title>  <meta http-equiv='content-type' content='text/html; charset=utf-8'>  </head>  <body bgcolor="#FFFF00">  <br><br><br><br>  <center>  <h1>Bonjour<br>voici mon premier script CGI ecrit en Python</h1>  </center>  </body>  </html>  ''' |

Rendez le exécutable :

|  |
| --- |
| **debarm:/var/www/cgi-bin#** chmod +x test.py |

Testez le programme dans un navigateur :

|  |
| --- |
|  |

# Utilisez les ports GPIO

Wiki du fabricant : [http://foxg20old.acmesystems.it/doku.php?id=hw:foxg20pinout](http://www.acmesystems.it/foxg20/doku.php?id=hw:foxg20pinout)

Le CPU AT91SM9G20 utilisé sur la FOX Board G20 dispose de 3 ports appelés A, B et C avec 32 bits chacun. Tous les bits ne sont pas disponibles sur les connecteurs de la carte et certains sont multiplexés avec d’autres types de liaisons comme des ports série, I2C, SPI, convertisseur A/N, USB, ….

L’utilisateur dispose de 28 lignes GPIO sur les deux connecteurs J6 et J7 de la carte :

|  |
| --- |
| **ATTENTION :**  Les ports GPIO fonctionnent en 3,3V.  Ne tolère pas le 5V.  J7  J6 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Pin #** | **ID** | **I/O line** | **Alt** | **Description** | **Pin #** | **ID** | **Line** | **Alt** | **Description** |
| J7.1 |  | GND |  | Signal ground | J6.1 |  | 3.3V |  | 3.3 volt DC power line |
| J7.2 |  | GND |  | Signal ground | J6.2 |  | 3,3V |  | 3.3 volt DC power line |
| J7.3 | 82 | PB18 |  | General purpose I/O | J6.3 | 92 | PB28 | RTS1 | Request to send ttyS2 |
| J7.4 | 83 | PB19 |  | General purpose I/O | J6.4 | 71 | PB7 | RXD1 | Receive data ttyS2 |
| J7.5 | 80 | PB16 |  | General purpose I/O. | J6.5 | 70 | PB6 | TXD1 | Transmit line ttyS2 |
| J7.6 | 81 | PB17 |  | General purpose I/O | J6.6 | 93 | PB29 | CTS1 | Clear to send ttyS2 |
| J7.7 | 66 | PB2 |  | General purpose I/O | J6.7 | 90 | PB26 | RTS0 | Request to send ttyS1 |
| J7.8 | 67 | PB3 |  | General purpose I/O | J6.8 | 69 | PB5 | RXD0 | Receive data ttyS1 |
| J7.9 | 64 | PB0 |  | General purpose I/O | J6.9 | 68 | PB4 | TXD0 | Transmit data ttyS1 |
| J7.10 | 65 | PB1 |  | General purpose I/O | J6.10 | 91 | PB27 | CTS0 | Clear to send ttyS1 |
| J7.11 | 110 | PC14 |  | General purpose I/O 1.8V | J6.11 |  |  |  | N.C. (See the schematic) |
| J7.12 | 111 | PC15 |  | General purpose I/O 1.8V | J6.12 |  | 5V |  | 5 volt DC power live |
| J7.13 | 108 | PC12 |  | General purpose I/O 1.8V | J6.13 | 75 | PB11 | RXD3 | Receive data ttyS4 |
| J7.14 | 109 | PC13 |  | General purpose I/O 1.8V | J6.14 | 74 | PB10 | TXD3 | Transmit data ttyS4 |
| J7.15 | 105 | PC9 |  | General purpose I/O 1.8V | J6.15 | 77 | PB13 | RXD5 | Receive data ttyS6 |
| J7.16 | 106 | PC10 | CTS3 | Clear to send ttyS4 1.8V | J6.16 | 76 | PB12 | TXD5 | Transmit data ttyS6 |
| J7.17 | 103 | PC7 |  | Red led line 1.8V | J6.17 | 85 | PB21 |  | General purpose I/O |
| J7.18 | 104 | PC8 | RTS3 | Request to send ttyS4 1.8V | J6.18 | 84 | PB20 |  | General purpose I/O |
| J7.19 | 101 | PC5 |  | General purpose I/O 1.8V | J6.19 | 95 | PB31 |  | General purpose I/O |
| J7.20 | 102 | PC6 |  | Reserved line. Used to read if the client USB port is wired to a PC | J6.20 | 94 | PB30 |  | General purpose I/O |
| J7.21 | 73 | PB9 | RXD2 | Receive data ttyS3 | J6.21 | 63 | PA31 | TXD4 | Transmit data ttyS5 |
| J7.22 | 72 | PB8 | TXD2 | Transmit data ttyS3 | J6.22 | 62 | PA30 | RXD4 | Receive data ttyS5 |
| J7.23 |  | BATT |  | RTC Battery input | J6.23 |  |  |  | N.C. (See the schematic) |
| J7.24 |  | PGD |  | Power good NetusPS1 | J6.24 | 38 | PA6 |  | General purpose I/O |
| J7.25 |  | POK |  | 3.3V NetusPS1 aux output stable | J6.25 | 39 | PA7 |  | General purpose I/O |
| J7.26 |  | SHDNPS# |  | PS1 shutdown. Active low | J6.26 | 41 | PA9 |  | General purpose I/O |
| J7.27 |  | NRST |  | Reset output | J6.27 | 99 | PC3 | AD3 | Analog input 3 |
| J7.28 |  | SHDN# |  | Turn off the CPU when low | J6.28 | 98 | PC2 | AD2 | Analog input 2 |
| J7.29 |  | 5V |  | 5 volt power line | J6.29 | 97 | PC1 | AD1 | Analog input 1 |
| J7.30 |  | WAKEUP |  | Wake up input | J6.30 | 96 | PC0 | AD0 | Analog input 0 |
| J7.31 | 87 | PB23 | DCD0 | Data carrier detect ttyS1 | J6.31 | 56 | PA24 | SCL | I2C Clock |
| J7.32 | 86 | PB22 | DSR0 | Data set ready ttyS1 | J6.32 | 55 | PA23 | SDA | I2C Data |
| J7.33 | 89 | PB25 | RI0 | Ring indicator ttyS1 | J6.33 |  | AVDD |  | Clean 3.3V out for A/D circuitry |
| J7.34 | 88 | PB24 | DTR0 | Data terminal ready ttyS1 | J6.34 |  | VREF |  | A/D voltage reference input |
| J7.35 | 60 | PA28 |  | General purpose I/O | J6.35 |  | AGND |  | Analog ground |
| J7.36 | 59 | PA27 |  | General purpose I/O | J6.36 | 42 | PA10 |  | General purpose I/O |
| J7.37 | 58 | PA26 |  | General purpose I/O | J6.37 | 54 | PA22 |  | General purpose I/O |
| J7.38 | 57 | PA25 |  | General purpose I/O | J6.38 | 43 | PA11 |  | General purpose I/O |
| J7.39 |  | 3.3V |  | 3.3 volt power line | J6.39 |  | GND |  | Signal ground |
| J7.40 |  | 3.3V |  | 3.3 volt power line | J6.40 |  | GND |  | Signal ground |

## Gérer les lignes GPIO à l’aide de l'interface sysfs en ligne de commandes

Wiki du fabricant  : [http://foxg20old.acmesystems.it/doku.php?id=tutorial:gpio](http://www.acmesystems.it/foxg20/doku.php?id=tutorial:gpio)

Sysfs est un système de fichiers virtuel introduit par le noyau Linux 2.6.

Sysfs permet d'exporter depuis le noyau vers l'utilisateur des informations sur les périphériques du système et leurs pilotes.

Ainsi, la gestion des I/O se fait par le biais de simples fichiers situés dans */sys/class/gpio*.

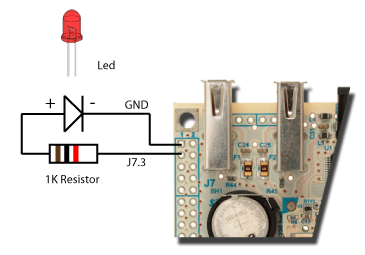
Le fichier export permet d’exporter une ligne GPIO identifiée par son Id :

|  |
| --- |
| **debarm:~#** echo 84 > /sys/class/gpio/export |

|  |
| --- |
| Ecrire **84** dans ***export*** a créé le dossier des fichiers de gestion de la broche **J6.18** d’Id **84**.  On accède à ses fichiers par le lien *gpio84* qui pointe vers */sys/devices/virtual/gpio/gpio84* |

Le dossier gpio84 contient, entre autre, le fichier de direction et le fichier de valeur de la broche GPIO :

|  |
| --- |
| Ecrire out dans direction :  Configuration de la broche en sortie.  Ecrire 1 dans value :  Mettre J6.18 à l’état haut. |

**Exemple** : Commander une led sur J7.3 (82)

Disposez une LED comme indiquez sur le schéma suivant :

|  |
| --- |
| **debarm:/#** echo 82 > /sys/class/gpio/export  **debarm:/#** echo out > /sys/class/gpio/gpio82/direction  **debarm:/#** echo 1 > /sys/class/gpio/gpio82/value  **debarm:/#** echo 0 > /sys/class/gpio/gpio82/value |

## Gérer les lignes GPIO à l’aide de l'interface sysfs en C

Comme on l’a vu précédemment, l’utilisation des ports GPIO est liée à l’accès à plusieurs fichiers. Les fonctions fopen(), fwrite(), fread() et fclose() sont donc nécessaires pour mener à bien cette tâche. Yoann Sculo, ingénieur en informatique et système embarqué, a développé une petite bibliothèque de fonction permettant de simplifier la programmation des GPIO.

<http://www.yoannsculo.fr/utiliser-gpio-en-c-sur-la-foxboard-g20/>

|  |
| --- |
| #include <string.h>  #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  struct S\_GPIO\_LINE {  char id\_number[4];  int direction;  int value;  };  typedef struct S\_GPIO\_LINE S\_GPIO\_LINE;  int load\_gpio\_line( S\_GPIO\_LINE \*ps\_line, char id\_number[4], int i\_direction ) {  FILE \*p\_gpio\_line;  /\* Exporting GPIO line \*/  if ((p\_gpio\_line = fopen("/sys/class/gpio/export", "ab")) == NULL)  {  printf("Cannot open export file.\n");  exit(1);  }  rewind(p\_gpio\_line);  strcpy(ps\_line->id\_number, id\_number);  fwrite(&ps\_line->id\_number, sizeof(char), 2, p\_gpio\_line);  fclose(p\_gpio\_line);  set\_gpio\_direction(ps\_line, i\_direction);  return 0;  }  int set\_gpio\_direction( S\_GPIO\_LINE \*ps\_line, int i\_direction ) {  FILE \*p\_gpio\_direction;  char gpio\_direction[35];  char c\_direction[4];  sprintf(gpio\_direction, "/sys/class/gpio/gpio%s/direction", ps\_line->id\_number);  /\* Setting line direction \*/  if ((p\_gpio\_direction = fopen(gpio\_direction, "rb+")) == NULL)  {  printf("Cannot open direction file.\n");  exit(1);  }  rewind(p\_gpio\_direction);  if( i\_direction ) {  strcpy(c\_direction,"in");  ps\_line->direction = 1;  }  else{  strcpy(c\_direction,"out");  ps\_line->direction = 0;  }  fwrite(&c\_direction, sizeof(char), 3, p\_gpio\_direction);  fclose(p\_gpio\_direction);  return 0;  }  int set\_gpio\_line(S\_GPIO\_LINE \*ps\_line, int value ) {  FILE \*p\_gpio\_value;  char gpio\_value[35];  char c\_value[2];  if( ps\_line->direction == 0 ) {  sprintf(gpio\_value, "/sys/class/gpio/gpio%s/value", ps\_line->id\_number);  /\* Setting value \*/  if ((p\_gpio\_value = fopen(gpio\_value, "rb+")) == NULL)  {  printf("Cannot open value file.\n");  exit(1);  }  rewind(p\_gpio\_value);  sprintf(c\_value, "%d", value);  ps\_line->value = value;  fwrite(&c\_value, sizeof(char), 1, p\_gpio\_value);  fclose(p\_gpio\_value);  }  else{  printf("Wrong access.\n");  exit(1);  }  return 0;  }  int main() { // Ce programme fait clignoter une led sur J7.3 d’Id 82  FILE \*fp;  S\_GPIO\_LINE s\_led82;  load\_gpio\_line(&s\_led7, "82", 0);  while(1)  {  set\_gpio\_line(&s\_led82, 1);  usleep(100000);  set\_gpio\_line(&s\_led82, 0);  usleep(100000);  }  return 0;  } |

## Gérer les lignes GPIO à l’aide de l'interface sysfs en PHP

L’intérêt d’un serveur web embarqué est de pouvoir piloter ou interroger un système à distance. Pour cela, l’interface web est sans doute la solution la plus pratique pour l’utilisateur (pas de logiciel supplémentaire à installer).

Tout comme en C, l’accès au système de fichier virtuel va se faire grâce aux fonctions PHP de gestions des fichiers.

Cependant, nous devons nous conserver à l’esprit que l’utilisateur par défaut du serveur web est www-data et qu’il n’a aucun droit par défaut sur les fichiers *export*, *direction* et *value*.

Il faudra donc prendre soin de créer le système de fichiers virtuels en mode super utilisateur (root) et modifier leur permissions afin que www-data (ou n’importe qui) puisse y avoir un accès complet.

|  |
| --- |
| Modification des permissions : accès complet à tout le monde  Permissions inadaptées à l’utilisation par le serveur web (*www-data*) pour les fichiers direction et value  Apparition du dossier *gpio82*  Création du système de fichier virtuel *gpio82* |

Le système de fichier virtuel ainsi créé existe tant qu’il n’a pas été détruit par *root* ou tant que la carte reste sous tension. En effet, lors du redémarrage de celle-ci, tous les gpio créés disparaissent. Une solution à ce problème consiste à créer un script shell de définition des GPIO utilisés qui sera exécuté lors du démarrage de la carte :

|  |
| --- |
| **#!/bin/sh**  echo "Activer les ports GPIO" echo -n "J7.3 : " if test -e /sys/class/gpio/gpio82  then echo "Déja actif"  else echo 82 > /sys/class/gpio/export  chmod 777 /sys/class/gpio/gpio82/direction  chmod 777 /sys/class/gpio/gpio82/value  echo "OK" fi  Si le dossier gpio existe (test –e) Ecrire **Déja actif**  Sinon le créer et modifier les permissions pour donner l’accès complet à tout le monde.  La première ligne indique que chemin d’accès à l’interpréteur shell (**shebang**) |

Afin de pouvoir exécuter ce script au démarrage de la carte, il faut le rendre exécutable puis le déplacer dans le dossier */etc/init.d* qui regroupe tous les programmes qui peuvent être exécutés au démarrage et créer un lien vers le dossier */etc/rc2.d* qui indique au système de démarrage les programmes à exécuter en fonction du niveau 2 d'exécution de Linux.

Les niveaux d’exécution (runlevel) sont au nombre de 7. Ils définissent l’ordre dans lequel les programmes démarrent :

* 0 : sert pour l'arrêt du système (commande init 0)
* 1 : programmes du mode mono utilisateur
* 2 : programmes du mode multi utilisateurs et réseau sans NFS (partage de fichier)
* 3 : programmes du mode multi utilisateurs et réseau
* 4 : ne sert pas
* 5 : programmes de démarrage de l'environnement graphique (proposer un environnement)
* 6 : sert pour rebooter le système

Le niveau par défaut dans une distribution Debian est le niveau 2.

Le script précédent est nommé *DefGpio.sh* (sh pour *shell*) est est situé dans le dossier */root*.

|  |
| --- |
| **debarm:~#** chmod +x DefGpio.sh  **debarm:~#** mv DefGpio.sh /etc/init.d  **debarm:~#** cd /etc/rc2.d  **debarm:~#** ln –s /etc/init.d/DefGpio.sh S99DefGpio.sh  Ou (variante)  **debarm:~#** chmod +x DefGpio.sh  **debarm:~#** mv DefGpio.sh /etc/init.d  **debarm:~#** cd /etc/init.d  **debarm:/etc/init.d#** update-rc.d DefGpio.sh default 99  Cette dernière commande créée elle-même le lien symbolique dans *rc2.d* avec le préfix *S99*. |

|  |
| --- |
| S : start (K : kill)  99 : ordre de démarrage |

Procédons à un redémarrage de la carte : coupez son alimentation et rebranchez-la.

|  |
| --- |
|  |

Vérifions que le dossier *gpio82* a bien été créé et que les fichiers direction et value ont bien les bonnes permissions :

|  |
| --- |
|  |

Il ne reste plus qu’à écrire le script PHP qui va allumer ou éteindre la LED :

|  |
| --- |
| <HTML>  <HEAD>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <TITLE>LED sur J7.3</TITLE>  </HEAD>  <BODY>  <?php  if (isset($\_GET["ok"]))  {  // Configuration de la direction  $fp = fopen("/sys/class/gpio/gpio82/direction","r+b");  if (!$fp) die("Erreur ouverture direction");  else  {  fwrite($fp,"out"); //J7.3en sortie  fclose($fp);  }  if($\_GET["J7-3"]=="ON")  {  $fp = fopen("/sys/class/gpio/gpio82/value","ab");  if ($fp)  {  fwrite($fp,"1"); //Allumer la LED  fclose($fp);  echo "<br><br><center><h1>Led allumée</h1></center>";  }  }  else  {  $fp = fopen("/sys/class/gpio/gpio82/value","ab");  if ($fp)  {  fwrite($fp,"0"); //Eteindre la LED  fclose($fp);  echo "<br><br><center><h1>Led éteinte</h1></center>";  }  }  }  echo '<form method="GET">';  echo '<center><h1>Contrôler la LED sur J7.3 </h1>';  echo '<input type="radio" name="J7-3" value="ON" ';  if(isset($\_GET["J7-3"]))  {  if($\_GET["J7-3"]=="ON") echo "checked";  }  echo '> ON';  echo '<input type="radio" name="J7-3" value="OFF" ';  if(isset($\_GET["J7-3"]))  {  if($\_GET["J7-3"]=="OFF") echo "checked";  }  echo '> OFF&nbsp;&nbsp;<input type="submit" value="Valider" name="ok">';  echo '</center></form>';  ?>  </body>  </html> |

|  |
| --- |
|  |

# Utilisation des convertisseurs analogique/numérique

Wiki du fabricant : [http://foxg20old.acmesystems.it/doku.php?id=contributes:antoniogalea:at91\_adc](http://www.acmesystems.it/foxg20/doku.php?id=contributes:antoniogalea:at91_adc)

La FOX Board G20 dispose de 4 lignes de conversion analogique/numérique d’une résolution de 10 bits directement issues de l’architecture du processeur AT91-SAM9G20 qui équipe la carte.

Malheureusement, le noyau linux adapté par le fabricant et installé sur la microSD d’origine n’intègre pas le pilote des convertisseurs A/N. Il va donc falloir les installer préalablement à leur utilisation.

Ceci fait, les lignes A/N, apparaîtront comme un simple ensemble de fichiers dans Sysfs. Comme pour les lignes GPIO, il suffira de venir lire le fichier correspondant à une ligne A/N qui contiendra une valeur numérique comprise entre 0 et 1023 pour une tension comprise entre 0 V et 3,3 V.

## Téléchargement, compilation et installation du driver

Cette opération doit être effectuée sous linux obligatoirement. En effet, le driver adapté pour le processeur AT91-SAM9G20 par Antonio Galea (contributeur au projet FoxG20) doit être compilé pour ce processeur.

Vous pouvez télécharger le résultat ici : <http://silanus.fr/sin/linux/at91_adc.ko>

Sinon téléchargez l’archive du driver : [http://foxg20old.acmesystems.it/lib/exe/fetch.php?media=contributes:antoniogalea:at91\_adc\_driver-20100518\_0821.tgz](http://www.acmesystems.it/foxg20/lib/exe/fetch.php?media=contributes:antoniogalea:at91_adc_driver-20100518_0821.tgz)

Copiez-la et décompressez-la dans votre dossier personnel :

|  |
| --- |
|  |

Pour le compiler, il nous faut les sources de la dernière version stable du noyau linux : <http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.35.4.tar.bz2>

|  |
| --- |
| Décompression  Téléchargement.  Si vous utilisez un proxy, indiquez son adresse :  export http\_proxy="http://login:motdepasse@ip\_proxy:port\_proxy" |

Ce noyau linux est écrit pour une architecture PC. Il faut donc l’adapter à une architecture ARM en utilisant le patch du fabricant.

Pour cela, il nous faut installer le programme patch, télécharger le patch du fabricant :

|  |
| --- |
|  |

|  |
| --- |
|  |

Téléchargez les fichiers nécessaires à la compilation :

|  |
| --- |
|  |

Compilez le noyau de linux :

|  |
| --- |
|  |

Compilez le driver des convertisseurs A/N :

|  |
| --- |
| Chemin d’accès au noyau (kernel) |

Le dossier at91\_adc\_driver contient maintenant le fichier at91\_adc.ko qu’il faut transférer dans la FOX Board G20, dans le dossier root par exemple :

|  |
| --- |
|  |

Vous pouvez aussi effectuer ce transfert en utilisant la commande scp :

scp at91\_adc.ko root@192.168.1.14

Remplacez l’adresse ip ci-dessus par celle de votre carte FOX Board G20.

Procédez à l’installation du driver dans le noyau du linux embarqué de la FOX Board G20 :

|  |
| --- |
|  |

Un nouvel ensemble de fichier apparait alors dans */sys/bus/platform/devices*. Il correspond au driver que l’on vient d’installer et on trouve à l’intérieur les quatre lignes du convertisseur sous forme de quatre fichiers *chan0* à *chan3*, accessibles à tous en lecture seule.

|  |
| --- |
|  |

Physiquement, les lignes A/N sont situées sur le connecteur J6 :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Connecteur** | **Adresse kernel** | **Port** | **Nom** | **Fonction** |
| J6.27 | 99 | PC3 | chan3 | Analog input 3 |
| J6.28 | 98 | PC2 | chan2 | Analog input 2 |
| J6.29 | 97 | PC1 | chan1 | Analog input 1 |
| J6.30 | 96 | PC0 | chan0 | Analog input 0 |

Connectez un générateur de tension entre J6.30 (chan0) et J6.35 (AGND). Connectez les masses analogique et numérique entre elles : J6.35 (AGND) et J6.40 (GND), ainsi que J6.33 (VREF) et J6.34( AVDD= 3.3V).

**ATTENTION : Ne pas dépasser les 3,3 V en entrée du CAN.**

|  |
| --- |
| Tension réglable de 0 V à 3,3 V |

Pour lire la valeur présente sur la ligne chan0, il suffit de lire le fichier :

|  |
| --- |
| **debarm:/sys/bus/platform/devices/at91\_adc#** cat chan0  2  **debarm:/sys/bus/platform/devices/at91\_adc#** cat chan0  25  **debarm:/sys/bus/platform/devices/at91\_adc#** cat chan0  506  **debarm:/sys/bus/platform/devices/at91\_adc#** cat chan0  1021 |

## Lecture d’une ligne analogique en PHP

Le script *an0.php* ci-dessous permet d’afficher la valeur lue sur la ligne chan0 et la valeur de la tension correspondante (Van0 = N x 3,3 / 1024).

|  |
| --- |
| <HTML>  <HEAD>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <TITLE>Page Test</TITLE>  </HEAD>  <BODY><center>  <?php  $erreur=0;  **$filename="/sys/bus/platform/devices/at91\_adc/chan0";**  **$fp = fopen($filename,"r");**  if (!$fp)  {  echo "Erreur ouverture chan0";  $erreur=1;  }  else  {  echo "<br><h1>Tension sur J6.30 : </h1>";  **$an0=fread($fp,filesize($filename));**  **fclose($fp);**  **$tension=$an0\*3.3/1024;**  **printf("<br><h2>%.4f V</h2>",$tension);**  **echo "<br>N = $an0</center> ";**    }  ?>  </BODY> |

|  |
| --- |
|  |

# Utilisation du bus 1 fil

## Le 1-Wire bus (bus 1 fil)

Le bus 1-Wire, conçu par DALLAS Semiconductor, permet de faire dialoguer entre eux des circuits sur un seul fil de données.

Les circuits sont alimentés entre 3V et 5V et l’alimentation peut être véhiculée sur le fil de données (mode « parasite »). Malgré son nom (1-Wire), il faut deux fils pour connecter les circuits : le fil de données et la masse.

|  |
| --- |
| DS18B20 Datasheet – Dallas Semiconductor © |

Ce bus supporte les topologies série, parallèle ou étoile. Il fonctionne selon le principe maître / esclave.

|  |  |  |
| --- | --- | --- |
| Linear topologie :  Les esclaves sont distants de moins de 3m du bus | Stubbed topologie :  Les esclaves sont distants de plus de 3m du bus | Star topologie |

Maxim © Note d’application 148 : <http://www.maxim-ic.com/app-notes/index.mvp/id/148>

Le nombre maximum d’esclaves à connecter sur le bus est quasiment illimité et la longueur totale du bus peut atteindre plusieurs dizaines de mètres (maximum définit à 750 m).

Chaque circuit possède une adresse physique unique de 64 bits, gravée dans la puce à la fabrication. Le premier octet de cette adresse détermine le type de famille auquel appartient le circuit. Les 6 octets suivants, constituent le code propre du circuit. Le dernier octet est le CRC. C'est un octet de contrôle calculé à partir des 56 bits précédents.

Ce type de bus est généralement utilisé en domotique (capteurs et interrupteurs commandés) et dans les circuits de gestion des batteries intelligentes.

## Mesure de température

Wiki du fabricant  : [http://foxg20old.acmesystems.it/doku.php?id=tutorial:1wire](http://www.acmesystems.it/foxg20/doku.php?id=tutorial:1wire)

Pour illustrer l’utilisation du bus 1 fil, le fabricant propose une mesure de température à l’aide d’un capteur DS18B20 de chez Dallas Semiconductor.

Le schéma de câblage est le suivant :

|  |
| --- |
| [http://foxg20old.acmesystems.it/lib/exe/fetch.php?media=tutorial:multi\_ds18b20\_wiring.jpg](http://www.acmesystems.it/foxg20/lib/exe/fetch.php?media=tutorial:multi_ds18b20_wiring.jpg) |

La FOX Board G20 intègre par défaut un driver de gestion de son bus 1 fil. Ce driver scan toute les 10 secondes si un nouveau périphérique a été connecté au bus. Pour chaque nouveau périphérique détecté, un nouveau dossier est créé dans */sys/bus/w1/devices/*

|  |
| --- |
|  |

La présence de deux liens symboliques dont le nom commence par 28 indique la présence de deux capteurs de températures DS18B20 connecté sur le bus.

Pour interroger un capteur, il faut lire son fichier *w1\_slave* :

|  |
| --- |
| La température mesurée est de 19,125°C |

La température mesurée est transmise dans les deux premiers octets selon le format suivant :

|  |
| --- |
|  |

Ici, nous avons MBS = 0x01 et LSB = 0x32 ce qui donne en effet une température de 19.125°C

Le script *temperature.php* ci-dessous permet de lire la température fournie par un capteur :

|  |
| --- |
| <HTML>  <HEAD>  <meta http-equiv="Content-Type" content="text/html"; charset="utf-8"/>  <TITLE>Formation STI2D SIN4</TITLE>  </HEAD>  <BODY>  <h1><center>Température</center></h1>  <br><br>  <?php  **$filename = '/sys/bus/w1/devices/28-000002afc6af/w1\_slave';**  **$handle = fopen($filename, 'r');**  echo "<h2><center>Transmission du capteur 28-000002afc6af : </center></h2><br>";  $i=0;  **while (! feof ($handle)) // Tant qu’on n’est pas à la fin du fichier**  **{**  **$line[$i] = fgets($handle); // Lire la ligne courante**  **echo $line[$i]."<br>"; // Afficher la ligne puis retour à la ligne**  **$i++;**  **}**  **$temp = substr($line[1],-6)/1000;**  echo "<h1><center>Temp : ".$temp."°C</center></h1><br>";  fclose($handle);  ?>  </BODY>  </HTML> |

|  |
| --- |
|  |

Identifiez vos capteurs et modifiez le script précédant pour qu’il affiche leurs températures ainsi que la date et l’heure.

## Projet

Le laboratoire bas bruit de Rustrel (84) est situé dans d’anciennes installations souterraines militaires. La zone la plus profonde ayant été initialement conçue pour être entièrement durcie et sécurisée dans le cadre de la dissuasion nucléaire, bénéficie d’un environnement très bas bruit (sismique, anthropique, électromagnétique). Elle est parfaitement adaptée à la qualification de systèmes et composants nanoélectroniques et l’étalonnage de dispositifs métrologiques avancés.

L’ancien poste de commande de tir nucléaire du plateau d’Albion sous 500m de roche est une chambre blindée unique au monde par son volume (1250m3).

La température et l’humidité dans les galeries sont des paramètres très important et nécessite une surveillance permanente.

|  |
| --- |
|  |

Pour surveiller ces paramètres dans les galeries non-utiliser à ce jour, le LSBB a choisi de s’adresser au lycée Alphonse Benoit de l’Isle sur la Sorgue et a confié ce projet à un groupe d’élève de terminale S SI dans le cadre du Projet Personnel Encadré (PPE).

Le projet consiste donc à fournir une solution technique viable pour la surveillance d’une seule galerie d’une longueur 150 m avec une forte inclinaison. Les capteurs de température seront espacés de 10m chacun afin d’en établir le gradient dans la galerie.

Les mesures doivent pouvoir être accessibles sur le réseau informatique du LSBB et via internet. Leurs enregistrements horodatés (toutes les 10 mn) dans une base de données permettront d’effectuer des statistiques et d’en conserver un historique.

La FOX Board G20 dispose d’un bus 1 fil permettant un raccordement aisé des capteurs de température et de ports analogiques pour les capteurs d’humidité.

Elle dispose en outre d’un serveur web et d’un serveur de base de données.

Le projet des élèves a consisté à mettre en œuvre un prototype composé de la FOX Board G20 et de deux capteurs de températures DS18S20, à stocker à intervalles réguliers (toutes les minutes) les mesures et à produire une interface web de consultation.

### Mise en œuvre des capteurs de température

Les capteurs de températures sont connectés sur le bus 1 fil conformément à la description de la page 37.

Un programme en écrit en C permet de relever les températures de chaque capteur et de les enregistrer dans une base de données dont la structure est la suivante :

* Base de données : *LSBB*
* Table : *temperature*
  + id : Entier, auto incrémenté, clé primaire
  + date : date
  + cap1 : réel
  + capt2 : réel

|  |
| --- |
| #include <string.h>  #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include<sqlite3.h> // apt-get install libsqlite3-dev  #include <time.h>  char \*str\_sub (const char \*s, unsigned int start, unsigned int end);  FILE \*fp;  char ligne[35];  char \*capt1;  char \*capt2;  int main()  {  printf("**Transmission du capteur 28-000002afc6af :**\n");    **fp = fopen("/sys/bus/w1/devices/28-000002afc6af/w1\_slave","r");**  if (fp==NULL)  {  printf("Erreur ouverture direction");  exit(0);  Lecture du capteur  }  else  {  **while (fgets(ligne, sizeof(ligne), fp) != NULL)**  **{**  Affichage de la température du capt1  **printf(ligne);**  **}**    **capt1=str\_sub(ligne,sizeof(ligne)-6,sizeof(ligne));**  **printf("Temp=%.3f°C",atof(capt1)/1000);**  printf("\n");  }  printf("**Transmission du capteur 28-000002afdad9 :**\n");    **fp = fopen("/sys/bus/w1/devices/28-000002afdad9/w1\_slave","r");**  if (fp==NULL)  {  printf("Erreur ouverture direction");  exit(0);  }  else  {  **while (fgets(ligne, sizeof(ligne), fp) != NULL)**  **{**  **printf(ligne);**  **}**    **capt2=str\_sub(ligne,sizeof(ligne)-6,sizeof(ligne));**  **printf("Temp=%.3f°C",atof(capt2)/1000);**  printf("\n");  Ouverture de la base située dans cgi-bin  }  int retval;  char query[1000];  sqlite3\_stmt \*stmt;  sqlite3 \*handle;  **retval = sqlite3\_open("/var/www/cgi-bin/LSBB",&handle);**  if(retval)  {  printf("Impossible de se connecter à la base de données\n");  return -1;  }  printf("Connexion réussie\n");  //Construction de la requete d’insertion  Construction de la requête d’insertion  **strcpy(query,"INSERT INTO temperature (date, capt1, capt2)");**  **strcat(query," VALUES(datetime(\"now\",\"localtime\"),");**  **strcat(query,capt1);**  **strcat(query,"/1000.0,");**  **strcat(query,capt2);**  **strcat(query,"/1000.0);\n");**  **printf(query);**  **retval = sqlite3\_exec(handle,query,0,0,0);**  sqlite3\_close(handle);  }  **// Fonction str\_sub pour récupérérer une sous-chaine entre les caractères**  **// start et end**  **// ex : s = "tartempion" start = 6 et end = sizeof(s)**  **// retourne "pion"**  **char \*str\_sub (const char \*s, unsigned int start, unsigned int end)**  {  char \*new\_s = NULL;  if (s != NULL && start < end)  {  new\_s = malloc (sizeof (\*new\_s) \* (end - start + 2));  if (new\_s != NULL)  {  int i;  for (i = start; i <= end; i++)  {  new\_s[i-start] = s[i];  }  new\_s[i-start] = '\0';  }  else  {  fprintf (stderr, "Memoire insuffisante\n");  exit (EXIT\_FAILURE);  }  }  return new\_s;  } |

Ce programme est enregistré et compilé dans le dossier *cgi-bin* du serveur web afin de pouvoir l’exécuter depuis un navigateur web.

Compilation et exécution :

|  |
| --- |
|  |

|  |
| --- |
|  |

Vérifions la présence de nos enregistrements dans la table température :

|  |
| --- |
|  |

### Enregistrement périodique avec CRON

L’automatisation de l’exécution du programme précédent sera obtenue grâce à CRON.

Son utilisation est détaillée page 19.

Le projet impose une prise de mesure toutes les 10 mn mais pour tester le fonctionnement de la tâche chronologique, nous la réaliserons toutes les minutes :

|  |
| --- |
| **debarm:/var/www/cgi-bin#** crontab -e |

|  |
| --- |
|  |

* Pour enregistrer les modifications : Ctrl + O
* Pour sortir de l’éditeur : Ctrl + X

|  |
| --- |
|  |

Après quelques minutes, consultons la table *temperature* :

|  |
| --- |
| Enregistrement toutes les minutes à partir de 17h23 |

### Interface web

L’interface réalisée par les élèves est disponible dans le dossier *LSBB*.

Elle est composée de pages web écrites en PHP et HTML. La partie graphique est définie dans une feuille de style (*PageCss.css*).

Les pages disponibles permettent de :

* Prendre connaissance du sujet du projet.
* Rechercher des mesures par mois.
* Effectuer des mesures instantanées.
* Visualiser les mesures d’une journée graphiquement.
* Accéder au site du lycée, du laboratoire LSBB et de Acme Systems.

|  |
| --- |
|  |

Les graphs de l’évolution de la température sont obtenus grâce à l’utilisation de ***Fusion Charts Free***. Cet outil gratuit permet générer une animation flash représentant des graphiques.

***Fusion Charts Free*** est disponible à l’adresse suivante : <http://www.fusioncharts.com/free>

Après avoir téléchargé et décompressé le dossier, il est copié dans le dossier du site.

Le principe de la génération de l’animation repose sur l’utilisation d’un fichier XML qui contient les coordonnées des points de la courbe.

|  |
| --- |
| try  Création d’un fichier XML temporaire  {  **$handle2 = fopen("/tmp/data2.xml", "w");**  Ecriture dans le fichier XML  /\*\*\* connect to SQLite database\*/  $dbh = new PDO("sqlite:/var/www/cgi-bin/LSBB");  /\*\*\* The SQL SELECT statement \*/  $sql = "SELECT \* FROM temperature WHERE date LIKE '$\_GET[date]%'";  **fprintf($handle2,"<graph xAxisName='Temps (hh:mm)' yAxisName='Temp2 (deg C)' showNames='1' decimalPrecision='1' formatNumberScale='0'>\n");**  foreach ($dbh->query($sql) as $row)  {  **fprintf($handle2,"<set name='" . substr($row['date'],-8,5) . "' value='" . $row['capt2'] . "' />\n");**  }  **fprintf($handle2,"</graph>\n");**  Fin du fichier XML  Ecriture dans le fichier XML des coordonnées des points  $dbh = null;  fclose($handle2);  }  catch(PDOException $e)  {  echo $e->getMessage();  }  echo "</td><td>";  // FusionChart component  // Eléments modifiables width, height, quality, dataURL (chemin d’accès au fichier temporaire).  ?>  **<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase=http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0" width="640" height="360" id="Column3D" >**  **<param name="movie" value="FusionCharts/Line.swf" />**  **<param name="FlashVars" value="&dataURL=tmp/data.xml&chartWidth=800&chartHeight=360">**  **<param name="quality" value="high" />**  **<embed src="FusionCharts/Line.swf" flashVars="&dataURL=tmp/data.xml&chartWidth=800&chartHeight=360" quality="high" width="800" height="360" name="Column3D" type="application/x-shockwave-flash" pluginspage="http://www.macromedia.com/go/getflashplayer" />**  **</object>** |

|  |
| --- |
|  |

|  |
| --- |
|  |

# En savoir plus

Sur son wiki, le fabricant fournit de nombreuses informations utiles et des exemples de projets de démonstration.

Il y est par exemple détaillé l’utilisation des périphériques sur :

* Bus USB
* Bus I2C
* Bus 1 fil
* Bus SPI
* Convertisseur Analogique/Numérique
* Lignes PWM
* Modem GPRS et GPS

De nombreuses cartes d’extensions (DAISY modules) permettent de mettre en œuvre rapidement des périphériques de bases sur la FOX Board G20 et réduire ainsi le temps de prototypage.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Daisy ID | Description | Availability | | DAISY-1 | Daisy adapter for the FOX Board G20 board | Available | | DAISY-2 | Stepper motor controller | Coming soon | | DAISY-4 | One relay | Available | | DAISY-5 | 8 push-button module for educational purpose | Available | | DAISY-7 | Digital MEMS accelerometer and giroscope | Coming soon | | DAISY-9 | RS232 interface | Available | | DAISY-10 | Optoisolated RS485/422 interface | Available | | DAISY-11 | 8 led module for educational purpose | Available | | DAISY-12 | Prototiping board | Available soon | | DAISY-13 | Teltonika GPRS modem | Coming soon | | DAISY-14 | Ethernet interface | Coming soon | | DAISY-15 | Adapter for 4D systems serial display | Coming soon | | DAISY-16 | Audio adapter | Coming soon | | DAISY-17 | Breadboard adapter | Coming soon | |

On y trouvera aussi :

* Les schémas de la carte FOX Board G20 et de son cœur, la Netus G20.
* La documentation du processeur AT91SAM9G20.