HORS SÉRIE - HORS SÉRIE - HORS SÉRIE - HORS SÉRIE - HORS SÉRIE

GNU LINUX MAGAZINE / FRANCE

France Métro : 6,40€ - DOM 6,95€ - BEL : 7,30€ - LUX : 7,30€ - PORT. CONT. : 7,30€ - CH : 13FS - CAN : 12$ - MAR : 65DH

L 15066 - 27 H - F: 6,40 € - RD

Octobre / Novembre 2006

HORS SÉRIE N°27

ÉLECTRONIQUE ET LINUX

MONTAGES DÉTAILLÉS, DÉVELOPPEMENT, EXPLOITATION DE DONNÉES

- +1 poster aide-mémoire de l'électronique
- Rappel des bases
- Acquisition de traces GPS
- Exploitation des données GPS avec Google Maps et Google Earth
- Prise de vue automatisée pour appareil photo numérique
- Utilisation et interfaçage de tubes Nixie
- Introduction à l'embarqué sur carte ACME Fox
- E/S faciles avec les convertisseurs USB/ imprimante
- Mise en œuvre d'un écran LCD couleur 128*128

100 % ÉLECTRONIQUE

ACTUELLEMENT

GNU LINUX MAGAZINE / FRANCE N°87

France Métro : 6,20€ - DOM 6,75€ - TOM 950 XPF - BEL : 6,80€ - LUX : 6,80€ - PORT. CONT. : 6,80€ - CH : 12,70CHF - CAN : 11,60$ - MAR : 70DH

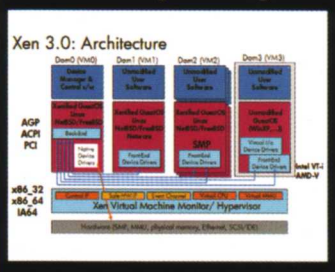
▶OCTOBRE ▶2006 ▶NUMÉRO 87

Le point sur Mono .NET Java et les Brevets

Face à l'engouement pour les plateformes Java et .NET, nous faisons le point sur les dangers que représente l'investissement dans ces technologies. Où en sommes-nous aujourd'hui ? Qui nous protège ? Que se passe-t-il lorsque le Logiciel libre s'acoquine avec des technologies brevetées ?

▶PARAVIRTUALISATION

Xen en pratique



Faites vos premiers pas en paravirtualisation avec Xen et utilisez les blocs de stockages virtuels (VBD)

EN KIOSQUE

edito

Bonjour et bienvenue dans ce nouveau hors-série,

A peu près à la même période, l'année dernière, nous avons consacré un premier hors-série à l'électronique (interface PC et GNU/Linux). Malheureusement, je n'avais réussi, ni à détruire tout mon équipement, ni à enfumer mes collègues et mon voisinage par une odeur pestilentielle de composants carbonisés... C'est donc avec une motivation importante et non sans un certain espoir de réussite que nous avons renouvelé l'expérience*.

Encore une fois, c'est J.-M. Friedt et E. Carry, que je remercie au passage, qui se sont joints à l'aventure (pour des raisons que je soupçonne plus sérieuses que les miennes) en contribuant de manière importante à ce numéro. Ainsi, les articles concernant l'acquisition et l'utilisation de trames GPS, ainsi que celui sur l'automatisation des prises de vue font suite à des réalisations concrètes en milieu scientifique. On notera au passage que l'un et l'autre de ces articles contiennent des informations pouvant être utilisées bien en dehors du cadre électronique/embarqué du hors-série. Vous en apprendrez ainsi beaucoup, par exemple, sur l'utilisation de Google Maps/Earth, ainsi que sur la manière d'utiliser KML.

Sans plus vous faire patienter, je vous laisse découvrir tout cela par vous-même.

Denis Bodor

* qui finalement se soldera par quelques poils de barbe grillés et une brûlure à l'avant-bras due à une manipulation hasardeuse de fer à souder. Force est de constater, donc, que l'exercice n'est pas aussi aisé qu'il y paraît et que l'électronique, finalement, n'est pas un domaine inabordable pour un utilisateur Unix qui se respecte.

sommaire

Linux Magazine France
Hors Série

est édité par
Diamond Editions

B.P. 20142 - 67603 Sélestat Cedex

Tél. : 03 88 58 02 08

Fax : 03 88 58 02 09

E-mail :
cial@ed-diamond.com

Service commercial :
abo@ed-diamond.com

Site :
www.ed-diamond.com

Directeur de publication :
Arnaud Metzler

RÉDACTION

Rédacteur en chef :
Denis Bodor

Conception graphique :
Kathrin Troeger

Responsable publicité :
Véronique Wilhelm
Tél. : 03 88 58 02 08

Service abonnement :
Tél. : 03 88 58 02 08

Impression :
VPM DRUCK /
www.vpm-druck.de

Distribution France :
(uniquement pour les dépositaires de presse)

MLP Réassort :
Plate-forme de Saint-Barthélemy-d'Anjou. Tél. : 02 41 27 53 12

Plate-forme de Saint-Quentin-Fallavier. Tél. : 04 74 82 63 04

Service des ventes :
Distri-médias :
Tél. : 05 61 72 76 24

PRINTED IN Germany / Imprimé en allemagne / Dépôt légal : 3e Trimestre 1998 / N° ISSN : 1291-78 34 / Commission Paritaire : 09 03 K78 976 / Périodicité : Bimestrielle / Prix de vente : 6,40 Euros

Denis Bodor :: db@ed-diamond.com :: lefinnois@lefinnois.net

INTRODUCTION ET RAPPEL DES BASES

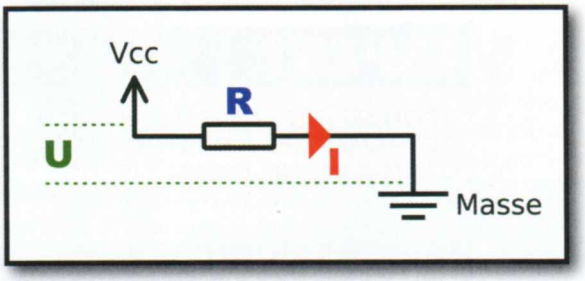
Le précédent hors-série 23 intégrait une dizaine de pages théoriques sur les bases de l'électronique. Je ne vais pas remettre le couvert. Cependant un bref condensé de ce qui a été dit s'impose autant en guise de rappel que de cours accéléré.

LOI D'OHM

Incontournable, indispensable, centrale... elle mérite tous les qualificatifs. La loi d'Ohm est la base en électronique et sa formule est simplement U=RxI. La tension U en volts est égale à la résistance R en ohms multipliée par le courant en ampère.

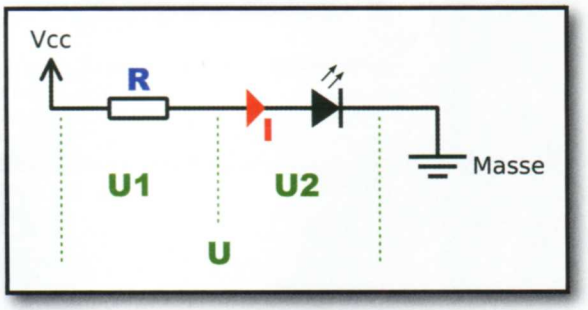
Le courant n'est pas la tension ! Le courant est la quantité d'électricité qui circule dans un circuit (un peu à l'image du débit d'eau dans un tuyau). La tension est une différence de potentiel entre deux points du circuit (à l'image d'une pression existant entre deux contenants hermétiques).

Pour comprendre les relations entre le courant, la tension et la résistance, le plus simple est encore d'utiliser un exemple. Soit le circuit donné en figure 1. Les trois valeurs sont reliées entre elles par la relation U=RxI :



- Si la tension entre Vcc et la masse est de 5 volts et la résistance de 1500 ohms, alors 3,3 mA traversent le circuit, car 5=1500x0.003.
- Si la tension est de 5 volts et que nous mesurons un courant de 10 mA, alors nous pouvons conclure que la valeur de la résistance est de 500 ohms, car 5=500x0.010.
- Si nous mesurons un courant de 15 mA traversant le circuit et que la résistance est de 1000 ohms, alors la tension aux bornes de celle-ci sera de 15 volts.

Voyons un circuit sensiblement plus complexe, mais très courant en péri-informatique (figure 2). Nous avons ajouté une LED. Ce composant se pilote en courant et non en tension. Cela signifie que, pour le faire fonctionner, on doit faire circuler un courant d'une valeur précise à l'intérieur. Le plus souvent, ce courant est de 10 mA ou 20 mA. En faisant cela, une tension apparaît aux bornes du composant. Cette tension est décrite dans la documentation du fabricant de la LED sous l'abréviation Vf.



Nous partons du principe que la LED choisie a besoin de 20 mA et qu'avec ce courant apparaît une tension à ses bornes de 2 volts. Notre tension d'alimentation, Vcc ou U sur le schéma est de 5 volts. Nous savons également qu'aux bornes de la LED doivent apparaître 2 volts, c'est U2. On calcule facilement U1=U-U2, soit 3=5-2. Nous avons tout ce qu'il faut pour calculer R.

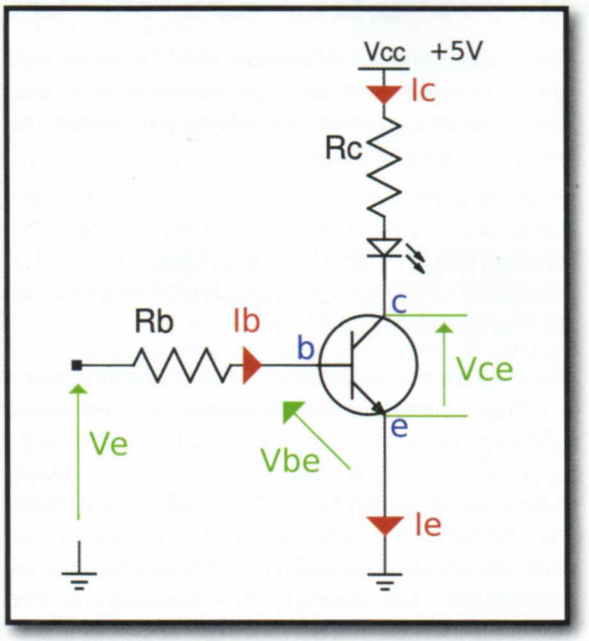
U=RxI et notre U, dans ce cas, c'est U1, la tension aux bornes de la résistance. Le courant traverse tout le circuit et nous devons le limiter à 20 mA. Nous avons ainsi 3=Rx0.020, soit 3/0.020=R, soit 150. En plaçant une résistance de 150 ohms, nous fournissons à la LED le courant qu'elle attend et elle s'allume.

TRANSISTORS

Le transistor, inventé en 1947, représente la pierre angulaire de l'informatique. Ses inventeurs ont reçu le prix Nobel en 1956, tant l'importance de la découverte était grande. Ce composant est un semi-conducteur tout comme la diode, mais se compose de trois bornes ou électrodes : l'émetteur, la base et le collecteur. Il existe plusieurs types de transistors et plusieurs modes de fonctionnement. Ceux qui nous intéressent en informatique sont les transistors bipolaires en régime ou mode de commutation. Un transistor bipolaire est un amplificateur de courant. On injecte du courant entre la base et l'émetteur pour créer un courant entre l'émetteur et le collecteur qui est multiplié par le gain du transistor. Il existe deux types de transistors bipolaires, les PNP et les NPN.

Ce qui nous intéresse, nous, informaticiens, ce n'est pas d'amplifier proportionnellement un courant, mais d'utiliser un transistor comme porte logique. Prenons le port parallèle. Celui-ci serait incapable d'alimenter un groupe de LED monté en parallèle nécessitant quelques 100 mA.

Nous voulons que la sortie du port puisse piloter un interrupteur contrôlant l'alimentation. Notre transistor en commutation est soit bloqué (interrupteur OFF), soit saturé (interrupteur ON).



Le montage complet en situation est donné en figure 3. Le transistor est un BC547. Il s'agit d'un modèle courant NPN. Sa documentation nous apprend plusieurs choses :

-Vce_max = 45V est la tension maximale que le transistor peut supporter entre le collecteur et l'émetteur. Lorsque le transistor est saturé, le problème ne se pose pas. Lorsqu'il est bloqué, il ne faut pas que la différence de potentiel soit supérieure à cette valeur.

- ß = 200 est le gain, le coefficient multiplicateur noté Hfe dans la documentation.
- Vce_sat = 0.2V est la tension présente entre le collecteur et l'émetteur lorsque le transistor est saturé. Il faut en prendre compte dans le calcul de la résistance Rc.
- Vbe_sat = 0.7V est la tension présente entre la base et l'émetteur lorsque le transistor est saturé.

Commençons par recalculer Rc en suivant la loi d'Ohm en tenant compte de Vce. Nous avons donc :

```
Vcc - Vf - Vce_sat = ?
5 - 2 - 0.2 = 2.8

U = Rc.I
2.8 = Rc * 0.01
2.8/0.01 = 280

Rc = 280 Ohms
```

Choisissons Rc à 220 comme résistance standard et déduisons l'intensité du courant :

```
U = R.I
2.8 = 220 * I
2.8/220 = 0.0127A = I
```

12.7mA, c'est un peu plus que la normale pour une petite LED, mais cela reste acceptable. MAIS, ce que nous venons de calculer, c'est Ic. Or, avec un transistor, le courant traversant la LED n'est pas Ic, mais Ie. Ie est égal à Ic+Ib. Ib est le courant minimum à fournir pour que le transistor sature. Il se calcule en fonction du gain du transistor (Hfe) : Ib min = Ic/ß, soit ici 0.0127/200 = 0.00006. Comme le préconisent les électroniciens, multiplions cette intensité par 1.5 pour ne laisser aucun doute quant à la saturation du transistor, ce qui nous donne 0.00009 que nous appelons « Ib_sat ». Le courant traversant la LED sera donc 0.0127 + 0.00009, soit 0.01279 ou 13 mA arrondis largement. C'est toujours acceptable pour la LED. Nous en avons fini avec Rc.

Nous connaissons déjà l'intensité du courant de saturation, Ib_sat. Il devient donc simple de calculer la résistance Rb à utiliser en appliquant la loi d'Ohm. Le transistor saturé présente 0.7 volt entre la base et l'émetteur (Vbe_sat). La tension aux bornes de Rb sera donc 5 - 0.7 soit 4.3 :

```
U = R.I
4.3 = R * 0.00009
4.3/0.00009 = 47777
Rb = 47777 Ohms
```

On utilisera une résistance de 47 K ohms nous permettant d'avoir une intensité de 0.091 mA, mais nous pourrions également utiliser une 33 K ohms qui nous donnerait 0.13 mA. La résistance peut être ainsi réduite car, plus elle est petite, plus le courant sera important et dépassera l'intensité minimum permettant de saturer le transistor. Il faut simplement prendre garde à ne pas dépasser le courant maximum acceptable spécifié dans la documentation. Avec un BC547, c'est 200 mA... nous avons de la marge.

Il ne faut pas oublier non plus que Ie = Ic + Ib. Ici, ce n'est pas une différence de 0.03 mA qui changera quoi que ce soit pour la LED, mais ce n'est pas toujours le cas pour tous les composants.

Vous comprenez sans doute tout l'intérêt du transistor avec cet exemple. Avec un courant de 0.13 mA, nous pilotons un dispositif traversé par un courant cent fois plus important.

Les transistors PNP reposent sur le même principe, mais fonctionnent à l'inverse. Grossièrement, avec le même type de circuit et les calculs révisés, c'est lorsque la sortie est à la masse que le transistor est saturé et lorsqu'elle est au niveau haut qu'il est bloqué.

CONCLUSION

Loi d'Ohm et calcul de transistor, avec ces deux éléments pleinement acquis, vous possédez déjà un bagage important pour débuter. Ces deux bagages et un peu de courage et de temps, c'est tout ce qu'il vous faut.

Denis Bodor :: db@ed-diamond.com :: lefinnois@lefinnois.net

ENTRÉES/SORTIES SIMPLES SUR USB

Ce n'est pas un scoop, les ports d'E/S faciles d'accès sont en voie de disparition. Le port série n'est presque plus présent sur les nouvelles configurations et le port parallèle ne devrait pas tarder à suivre le même destin. L'USB s'est donc définitivement installé comme port à tout faire par défaut.

Il y a bien longtemps (à l'échelle de l'évolution des technologies), les ordinateurs de type PC disposaient d'un bus très facile d'accès permettant toutes sortes de bidouilles : le bus ISA. L'arrivée en masse du PCI a non seulement éliminé l'ISA, mais a également marqué la fin de l'accès aisé direct au matériel. Le développement de cartes prototypes PCI « faites maison » est une activité coûteuse et pénible. Restait alors le port série permettant de dialoguer avec tout montage externe, mais celui-ci est maintenant en voie d'extinction. Certes, il est toujours possible d'utiliser une carte PCI série, mais il faut se rendre à l'évidence : l'USB remplace globalement toutes les autres interfaces d'E/S petit à petit.

Le port parallèle, précédemment traité dans le hors-série 23, est encore présent sur les machines PC actuelles, mais ce n'est qu'une question de temps. De plus, les architectures non-PC comme les Mac ou d'autres plateformes n'ont jamais disposé d'interfaces compatibles IEEE 1284 (port parallèle).

Il faut donc se tourner vers le seul port disponible et présent sur toutes les plateformes matérielles : l'USB. Malheureusement, bien que plus accessible que le bus PCI, l'USB n'est pas simple à mettre en œuvre. La solution pour obtenir un jeu d'E/S via USB est alors d'utiliser les modules dédiés comme ceux utilisant les puces FTDI disposant de pilotes pour Linux. Ces composants ou modules coûtent entre 20 et 40 euros et peuvent être utilisés directement via le support du noyau ou en utilisant la libUSB. Il existe cependant une solution à la fois plus économique et plus simple à mettre en œuvre. Pour peu que l'on contourne un certain problème.

ADAPTATEUR USB/PARALLÈLE : LE PROBLÈME

La gestion des ports parallèles sous Linux se compose en plusieurs éléments ou couches. Nous avons tout d'abord le pilote chargé de dialoguer directement avec le matériel. Sur PC, il s'agit du support `parport_pc`. Reposant sur cette couche, se place ensuite le support `parport` permettant l'interfaçage avec une API unique pour l'accès au port. Une troisième couche, `ppdev`, permet un accès au port depuis l'espace utilisateur en utilisant directement une entrée `/dev/parport[0-9]`.

En jetant un œil aux sources du noyau, on constate qu'il existe plusieurs supports de bas niveau pour les ports parallèles (`parport_cs`, `parport_sunbpp` ou encore `parport_amiga`), mais pas de `parport_usb`. Côté support USB, nous trouvons `usblp` et un mystérieux `uss720`.

`usblp` n'offre pas de support `parport` contrairement à `uss720` qui, pourtant, est bien un pilote pour convertisseur USB/parallèle de fabrication Lucent ou Belkin. Pourquoi ? Le pilote `uss720` concerne un matériel bien précis utilisant la puce du même nom. Il a été développé et est maintenu par Thomas Sailer. C'est le seul pilote vous permettant d'obtenir un `/dev/parport[0-9]` et ainsi de bénéficier du support `ppdev`. Lors du chargement sous la forme d'un module, un avertissement vous signale que, pour une utilisation avec une imprimante, il est fortement recommandé d'utiliser `usblp` plutôt que ce module.

Les adaptateurs USB/parallèle sont en réalité mal nommés. On devrait, en effet, plutôt parler d'adaptateur ou de ponts USB vers imprimante. Les circuits logiques spécialisés utilisés pour ces adaptateurs sont, en effet, destinés à fonctionner avec des imprimantes et non à offrir un accès aux lignes du port parallèle. Ils intègrent ainsi un certain nombre d'opérations spécifiques à la communication avec une imprimante.

`usblp` n'est pas un pilote pour un ou plusieurs périphériques USB précis, mais pour une classe de périphériques. Le système de classes permet de regrouper les constructeurs de périphériques USB autour de spécifications communes. Une interface unique permet ainsi de faciliter la mise en œuvre du matériel. C'est pour cette raison que n'importe quel adaptateur USB/imprimante se déclarant comme faisant partie de la classe `USB Printer` fonctionnera sous Linux. Il en va de même pour d'autres périphériques comme les claviers et les souris répondant aux spécifications de la classe HID ou encore les disques USB supportés par `usb-storage`.

En toute logique, si vous voulez jouer avec les signaux du port parallèle d'un adaptateur USB, soit vous avez la chance d'avoir l'un des périphériques intégrant une puce uss720, soit vous devez développer votre propre support noyau pour le périphérique en question. Il n'existe aucune volonté parmi les développeurs noyau d'écrire des dizaines de pilotes pour utiliser une architecture `parport`. C'est bien compréhensible lorsqu'on sait que la quasi-totalité

des utilisateurs font un usage « normal » de ce type de périphériques.

PARPORT!=LP

Qu'est-ce qui différencie donc un `/dev/parport[0-9]` d'un `/dev/lp[0-9]` ? La réponse tient en quelques mots : le niveau d'abstraction vis-à-vis du matériel. `lp` signifie « *Line Printer* ». L'entrée dans `/dev` est un point d'accès vers une imprimante. En accédant à une entrée `parport`, vous contrôlez l'état des lignes en entrée et en sortie. Avec `lp`, vous envoyez simplement des données à une imprimante, le noyau prend en charge le protocole de communication. C'est ce qui vous permet de faire `cat fichier.ps > /dev/lp0` par exemple. Si cela ne fonctionne pas (plus de papier, imprimante *offline*, etc.), le support noyau se charge de gérer les erreurs et de retourner, au besoin, un message à l'utilisateur.

Le protocole de communication vers une imprimante n'est pas quelque chose de strictement défini. Il en va de même avec les lignes du port lui-même. La succession de standards ayant pour origine l'IBM PC puis l'AT a fait qu'une partie des spécifications ont évolué et se sont déclinées plus ou moins précisément en fonction des fabricants. Il n'en reste pas moins que, dans les grandes lignes, un véritable standard a vu le jour sous le nom d'IEEE 1284.

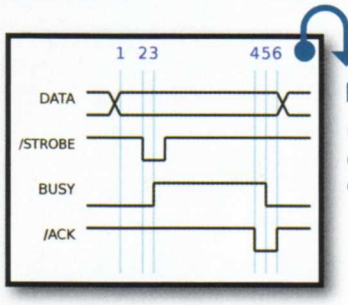


Fig. 1

Chronogramme d'une communication entre un PC et une imprimante

La figure 1 montre un dialogue simplifié entre un ordinateur et une imprimante ainsi que l'utilisation des différentes lignes de l'interface :

1. Préparation des données et configuration des lignes D0 à D7.
2. L'ordinateur passe /STROBE à l'état bas pendant 1.5usec (+/- 500nsec), signalant la disponibilité des données à traiter.
3. Le périphérique alerté par le changement d'état de /STROBE se signale occupé en passant BUSY à l'état haut. Dès lors, l'ordinateur sait qu'il ne doit plus envoyer de données.
4. Après traitement des données, le périphérique signale la fin du traitement en passant ACK à l'état bas. Il accuse réception des informations transmises.
5. Le périphérique se déclare libre pour le traitement des prochaines données et repasse BUSY à l'état bas.
6. Les données présentées par l'ordinateur sont maintenant invalides (obsolètes) et le processus peut reprendre pour les données suivantes.

On comprend clairement tout le travail fait à votre place en envoyant des données sur `/dev/lp[0-9]`, et tout aussi clairement que `lp` n'offre pas autant de libertés que le support `ppdev`.

ADAPTATEUR USB/IMPRIMANTE : LA SOLUTION

`usblp` ne permet pas d'utiliser `ppdev`, mais court-circuite le processus, n'offrant ainsi que le support `lp` direct. Il n'existe pas de technique logicielle simple pour contourner le problème. Il faut donc se pencher sur une solution matérielle. Puisque `usblp` veut parler à une imprimante, nous n'avons qu'à lui proposer une imprimante.

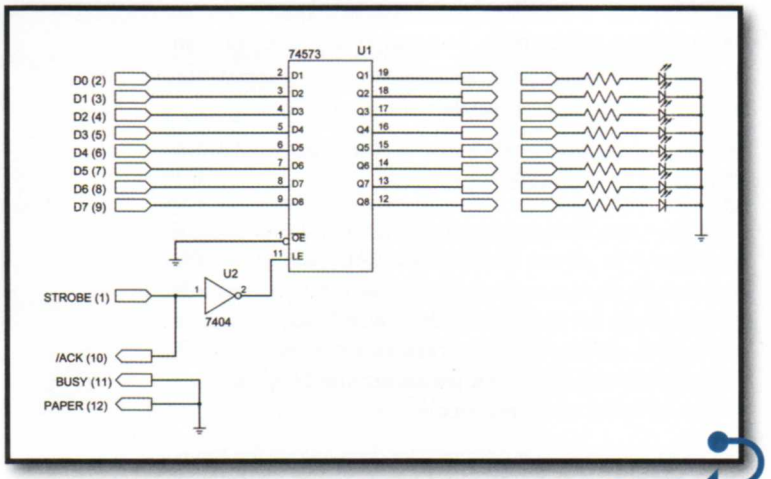
Nous allons toutefois simplifier la communication entre l'interface et la pseudo-imprimante :

1. Pour rendre la pseudo-imprimante disponible, nous ramenons le signal PAPEREND/PAPEROU/OUTPA, broche 12 du connecteur Centronic (36 broches) à la masse. Nous avons toujours du papier. Nous faisons de même avec la broche 11 correspondant au signal BUSY. Le périphérique n'est donc jamais occupé, ce qui est une violation évidente du protocole.
2. L'adaptateur nous présente les données sur les lignes D0 à D7 (broches 2 à 9).
3. Une fois ces lignes de données stables, l'adaptateur passe brièvement le signal /STROBE (broche 1) à l'état bas (masse). Il nous signale la disponibilité des données.
4. Nous utiliserons directement le signal /STROBE (qui est inversé) pour accuser réception des données en le connectant à l'entrée /ACK (qui est aussi inversée).
5. Le signal /STROBE nous permettra également de stocker les données transmises dans un *latch*, un composant relativement simple, sorte d'interrupteur logique stockant les informations qu'il reçoit.
6. Une fois le signal ACK reçu, l'adaptateur USB/imprimante repasse toutes les lignes de données à l'état bas ou, éventuellement, présente les données suivantes.

Le composant principal du montage émulant une imprimante est donc un latch. Le modèle choisi ici est un 74HCT573. Il s'agit d'un latch à trois états et son fonctionnement est très simple. On présente les 8 bits de données en entrée sur les broches D0 à D7 et on envoie une impulsion sur la broche LE (*Lacth Enable*) pour les stocker. La broche /OE (*Output Enable*) permet d'envoyer ou non les 8 bits stockés sur les sorties Q0 à Q7. Dans le cadre de notre utilisation, /OE est relié à la masse afin d'activer en permanence les sorties.

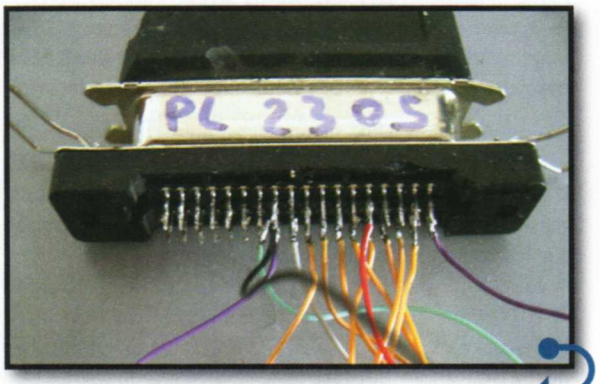
Remarquez que le signal LE est actif au niveau haut, mais que le signal /STROBE de l'adaptateur est actif au niveau bas. Il nous est donc nécessaire d'inverser le signal. Il existe beaucoup de solutions pour ce faire. J'ai choisi ici la plus simple en utilisant un 74LS04, un sextuple inverseur. On reliera donc l'une des entrées de l'inverseur sur /STROBE et la sortie sur le LE du 74HCT573. Les cinq autres inverseurs contenus dans le 74LS04 ne sont pas utilisés.

Notez que n'importe quel latch TTL fera l'affaire, ainsi que n'importe quel composant logique pour l'inversion du signal /STROBE. Des portes NAND (NON ET), par exemple, conviendraient parfaitement. Il suffit de relier les deux entrées de la porte ET ensemble sur /STROBE et la sortie sera naturellement inversée à l'arrivé du signal.



L'émulateur d'imprimante **Fig. 2**

La figure 2 montre le schéma de notre montage. Les résistances utilisées pour piloter les LED sont des 470 ohms, mais selon les LED en question (bleues par exemple), leur valeur peut varier. N'oubliez pas LA formule de base : U=R*I. Coté réalisation, la partie délicate concerne principalement le connecteur Centronic 36 broches. La figure 3 présente un connecteur femelle récupéré sur une vielle imprimante hors d'usage, mais il est parfaitement possible d'en commander chez un détaillant de composants électroniques pour moins de 3 euros.



Brochage du connecteur Centronic **Fig. 3**

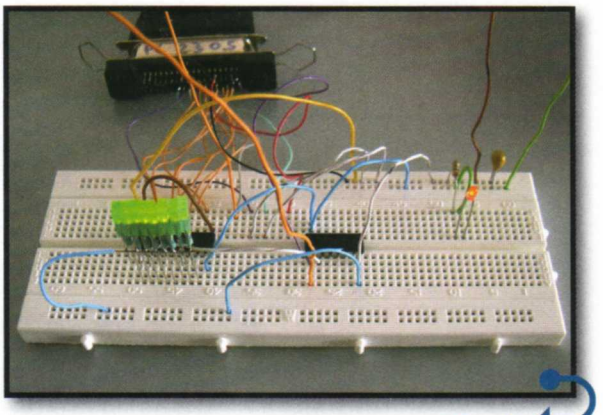
Pour ce qui concerne les brochages à proprement parler, nous ne rencontrons pas de difficulté. Les signaux qui nous intéressent sont tous côte à côte en partant de la broche 1 : /STROBE, D0 à D7, /ACK, BUSY et PAPEROUT. Il faut simplement de petits doigts et de la patience pour les soudures.

LOGICIEL

Une fois le montage réalisé sur platine (figure 4) avec une alimentation directement tirée sur bus USB, nous pouvons expérimenter. Le premier test sera des plus simples. Après chargement si nécessaire du module `usblp`, nous envoyons des données aléatoires sur `/dev/usb/lp0` :

```
$ cat /dev/urandom > /dev/usb/lp0
```

Vous devriez alors voir s'affoler les LED connectées au latch au rythme des données reçues. Ceci montre non seulement que le montage fonctionne malgré les libertés prises avec le protocole, mais également la simplicité d'accès en sortie. Il suffit, en effet, d'envoyer un ou plusieurs caractères dans `/dev/usb/lp0` pour contrôler les LED. Cette simplicité offre un large choix quant aux langages qu'il est alors possible d'utiliser.



L'émulateur sur platine à essai **Fig. 4**

Même en C, les opérations se limitent à une simple écriture dans le pseudo-fichier :

```
int fd;
char tab[] = { 1, 2, 4, 8, 16, 32, 64, 128 };

fd = open("/dev/usb/lp0",O_WRONLY, 0);
if (pp < 0) {
  fprintf(stderr,"Open Error : %s (%d)\n",
      strerror(errno),errno);
  exit(EXIT_FAILURE);
} else {
  printf("Open\n");
}

if(write( fd, &tab, 8 ) != 8){
  fprintf(stderr,"Write Error : %s (%d)\n",
      strerror(errno),errno);
  exit(EXIT_FAILURE);
```

```
}

close(fd);
```

Ce code écrit tout simplement les 8 caractères stockés dans `tab[]` dans `/dev/usb/lp0`. Bien entendu, le traitement par le montage est si rapide que seule est visible la dernière valeur sous la forme de l'allumage de la LED sur D7. Pour voir quelque chose, vous devrez écrire caractère par caractère en incrémentant le pointeur sur `tab[]` (avec un `usleep` si besoin).

Notons qu'un certain nombre d'`ioctl` sont normalement disponibles pour manipuler un `/dev/lp[0-9]`. Malheureusement, la plupart ne sont pas utilisables avec un adaptateur USB/imprimante, car le périphérique n'offre pas assez de finesse en termes de contrôle des lignes. Seule une ou deux lignes en entrée sont facilement utilisables, dont PAPEROUT. Pour lire l'état de la ligne, il suffit d'utiliser `LPGETSTATUS` :

```
char ibuf;

if(ioctl( fd, LPGETSTATUS, &ibuf ) < 0){
  fprintf(stderr,"LPGETSTATUS Error : %s (%d)\n",
      strerror(errno),errno);
  exit(EXIT_FAILURE);
} else {
  printf(":Status %X\n", ibuf);
  if(ibuf & LP_POUTPA) {
    printf(":LP_POUTPA\n");
  }
}
```

Il faudra inclure le fichier `linux/lp.h` (oui, utiliser les en-têtes du noyau c'est mal, mais on est plus à ça près). L'utilisation du signal PAPEROUT n'est pas bloquant. L'imprimante peut, en effet, signaler l'absence de papier tout en continuant à traiter les données et remplir un *buffer* interne. Lorsque le buffer est plein, l'imprimante utilise le signale BUSY. Nous disposons ainsi de 8 sorties et deux entrées librement utilisables pour tout type d'applications (SLCT est la seconde).

AMÉLIORATIONS

Un certain nombre de choses peuvent être améliorées par rapport au simple montage présenté ici. Tout d'abord, on peut respecter un peu plus le protocole de communication en ajoutant un délai entre le signal STROBE qui active le latch et l'émission du signal /ACK. La simple connexion de /STROBE et /ACK est cavalière et peut entraîner des problèmes en pilotant le port à haute vitesse. Vous pouvez très simplement en faire l'essai en envoyant des données au montage à l'aide de `cat`. Déconnectez ensuite la ligne /ACK du 74LS04. Non seulement l'écriture sur l'entrée dans `/dev/usb` est bloquée, mais vous ne pouvez pas la relancer en interrompant le processus avec ^C et en relançant l'envoi. Le périphérique USB est toujours en attente de l'accusé réception des dernières données. Seule solution, connecter par intermittence la ligne /ACK à la masse pour en changer plusieurs fois son état. On dépile ainsi les données en attente. Là, nous sommes dans le « bricolo » dans toute sa grandeur et le système le signal clairement : `usblp0: on fire`, `write error: Appel système interrompu`.

Une meilleure solution consisterait donc à utiliser un microcontrôleur en lieu et place des deux circuits logiques de manière à véritablement contrôler le flux de données et utiliser le signal /ACK judicieusement.

La déconnexion de la ligne BUSY est moins catastrophique. En effet, du fait de sa fonction, BUSY nous permet de mettre en attente le flux des données. Déconnectez BUSY, patientez, reconnectez... les opérations suivent leur court.

CONCLUSION

Les informations données dans cet article ont été testées avec deux adaptateurs USB différents. Le premier, acheté chez un détaillant, est un produit d'entrée de gamme sans marque. Celui-ci est identifié par Linux sous la désignation `067b:2305 Prolific Technology, Inc. PL2305 Parallel Port`. Le second est un modèle un peu plus ancien, détecté comme `047e:1001 Agere Systems, Inc. (Lucent) USS720 Parallel Port`. Comme vous pouvez le voir, il intègre la puce USS720, mais a néanmoins été utilisé ici avec le module `usblp`.

Le comportement des deux adaptateurs est différent. Le Prolific Technology semble plus lent et ne renvoie pas les mêmes informations que l'adaptateur Agere Systems. En effet, la lecture de l'état avec `LPGETSTATUS` sur le premier retourne simplement `LP_PSELECD`, alors que le second retourne en plus `LP_OFFL` signifiant que l'imprimante serait Offline. Il apparaît cependant que cela ne gêne en rien le fonctionnement du montage, puisque le `cat /dev/urandom` redirigé vers l'entrée `/dev/usb` correspondante fonctionne parfaitement.

Il semble donc que les constructeurs prennent quelques libertés qui ne dérangent pas le support `usblp`. Mais je ne saurais garantir que ce *hack* fonctionnera avec tous les adaptateurs existants. Si vous rencontrez des problèmes, assurez-vous que le montage est correct et essayez, si possible, avec un autre modèle. J'ai également remarqué que toutes les broches du port Centronic référencées comme étant des masses ne le sont pas toujours. Le multimètre est votre ami.

Enfin, dernière recommandation. En cas de comportement étrange, connectez éventuellement la broche 31 (/INIT) aux entrées 13 (SLCT) et 32 (/ERROR). Avec la mise à la masse de BUSY et PAPEROUT ainsi que la connexion /STROBE sur /ACK cela constitue un « *Null Parallel Adapter* » permettant à certains vieux logiciels MS/DOS de tester le port parallèle.

Denis Bodor :: db@ed-diamond.com :: lefinnois@lefinnois.net

Denis Bodor :: db@ed-diamond.com :: lefinnois@lefinnois.net

OUBLIEZ LCDPROC ET PASSEZ AU GRAPHIQUE COULEUR

Voici un titre bien racoleur, mais les faits sont là. Le matériel et des solutions permettant de piloter sous Linux un écran graphique couleur existe. Ceci pour un faible coût aussi bien en argent qu'en termes d'efforts.

Loin de moi l'idée de discréditer le projet LCDproc. Ce projet, je le rappelle, offre un support logiciel sur les systèmes UNIX permettant de piloter des afficheurs LCD au format texte. Ce type de solution, économique, permet d'afficher en façade d'une machine, par exemple, toutes sortes d'informations très simplement. Force est de constater, qu'aujourd'hui, des périphériques plus évolués, graphiques, de bonne résolution et apportant la couleur, sont de plus en plus abordables.

Jusqu'à présent, ces périphériques se résumaient souvent à des modules d'affichage généralement utilisés dans les téléphones mobiles. Achetés sous forme de pièces détachées, ils restaient souvent délicats à mettre en œuvre, puisqu'il fallait dialoguer directement avec le contrôleur LCD. Les problèmes étaient donc tout aussi bien logiciels (apprentissage des commandes, gestion du *timing*, etc.) que matériels (régulation de l'alimentation, gestion des signaux sur plusieurs lignes, connectique, etc.). Cependant, on voit maintenant de plus en plus de modules arriver sur le marché, de l'embarqué en particulier, offrant une connectivité de plus haut niveau et des protocoles bien plus accessibles et standards.

LE MODULE ULCD DE 4D SYSTEM

Le produit testé pour cet article est l'une des solutions simples et économiques aujourd'hui disponibles. Le périphérique d'affichage produit par 4D *system*, le uLCD (prononcez micro LCD) est un module intégré comprenant :

- un afficheur LCD 65K couleur de 128 x 128 pixels et de 1,5 pouces de diagonale. Celui-ci est rétro-éclairé par une LED blanche dont l'intensité lumineuse peut être contrôlée par logiciel.
- un module de contrôle s'architecturant autour d'un PIC 16F6490 et offrant une connectivité série. Le module se charge de la gestion de l'écran et de l'alimentation de l'ensemble. Celui-ci est alimenté en +5 volts, mais les signaux série RX/TX fonctionnent en 0/+3.3 volts.

C'est surtout du côté du protocole de communication que tout cela devient intéressant. En effet, celui-ci est d'une grande simplicité. Le code contenu dans le microcontrôleur PIC offre un certain nombre de facilités :

- contrôle du contraste et de la gestion de l'écran (extinction, rétro-éclairage) ;
- primitives graphiques basiques comme le tracé de cercles ou de lignes et le remplissage de zones de couleur (rectangulaires) ;
- gestion du contenu de l'écran (couleur de fond, effacement) ;
- contrôle en lecture et écriture de chaque pixel de l'écran ;
- affichage de texte avec trois tailles de polices différentes avec gestion du fond ;
- ajout et utilisation de caractères définis par l'utilisateur.

INTERFAÇAGE SÉRIE AVEC UN PC

De manière générale, un port série est une interface de communication sérielle. Entendez par là que, par opposition à une interface parallèle, les bits sont transmis un par un, les uns à la suite des autres sur une seule ligne de signal. Il faut ensuite distinguer deux types d'interfaces série. La première dispose au minimum d'une ligne de signal et d'une ligne d'horloge. A chaque fois qu'un bit est prêt à être émis, un signal d'horloge est déclenché. Les *Serial Peripheral Interface* ou SPI sont habituellement conçues de la sorte avec :

- une ligne du maître vers l'esclave (*Master Out - Slave In* ou MOSI)
- une ligne de l'esclave vers le maître (*Master In - Slave Out* ou MISO)
- une ligne d'horloge (*System CLocK* ou SCLK)

Les liaisons i2c (voir hors-série numéro 23) fonctionnent également de cette manière. Il s'agit de liaisons série dites « de type synchrone », car un signal dédié permet de synchroniser la communication.

L'autre type de liaison est asynchrone. Il n'y a pas de signal d'horloge. Maître et esclave utilisent leur propre horloge locale ou générateur de baud. La synchronisation se fait

par la détection d'un bit de *start* et les données sont transmises et reçues avec une vitesse et une fréquence fixes. Les liaisons asynchrones sont généralement plus délicates à mettre en œuvre matériellement, car elles imposent l'utilisation d'une base de temps fiable et stable. Généralement, dans le cas d'un microcontrôleur, un quartz devient alors indispensable, même si le circuit intègre une horloge (Atmel AVR par exemple) sous la forme d'un circuit RC.

Les ports communément appelés « port série » sur PC sont des liaisons asynchrones normalisées RS232. Cette norme fixe également les tensions utilisées par l'interface. C'est là le seul problème que nous devons résoudre dans le cas de l'écran uLCD. En effet, la logique de communication est prise en charge par le microcontrôleur intégré à l'écran, ainsi que par l'interface du PC.

La problématique est la suivante :

- Un périphérique disposant d'une interface série où 0 logique = 0 volt et 1 logique = 3.3 volts.
- Une interface PC ou 0 logique = entre +3 et +12 volts, et 1 logique = entre -3 et -12 volts. Dans la pratique, sur les PC modernes, on trouve plus souvent des tensions entre –5 volts et +5 volts.

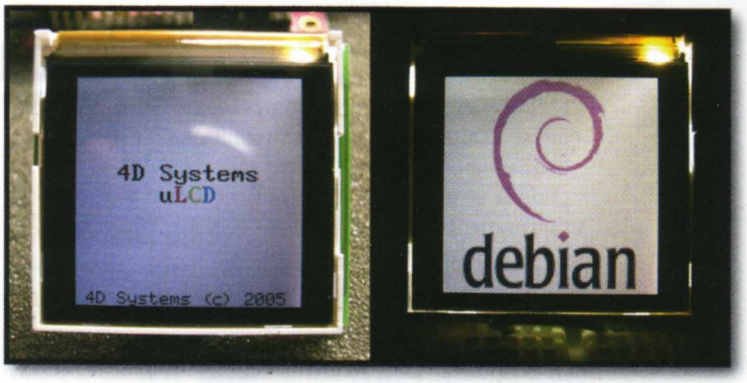
Des circuits spécialisés comme le MAX3232 permettent l'interfaçage rapide. Cependant, il est relativement difficile de trouver le composant qui n'est pas aussi populaire que le MAX232 (conversion RS232/TTL). De plus, une solution plus économique est possible via l'utilisation des transistors NPN et de résistances.

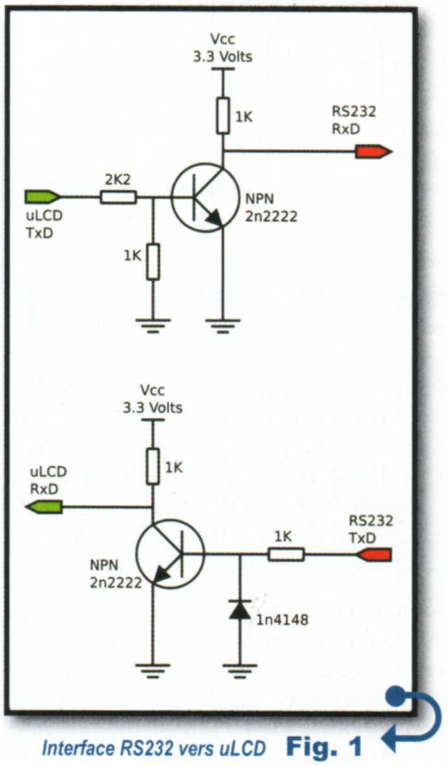
Le schéma donné en figure 1 détaille le montage utilisé ici. Il utilise de simples transistors NPN en régime de commutation (ici des 2n2222, mais on peut également utiliser des BC337). Le principe de fonctionnement est relativement trivial.

Dans le sens RS232 vers uLCD, lorsque le transistor est non saturé, le périphérique uLCD voit Vcc en entrée. La diode 1N4148 permet d'éviter de mettre une tension négative à la base du transistor (ce qui le détruirait à plus où moins court terme) lorsque qu'un 1 logique se présente. Avec un 0 logique (tension positive), le 2n2222 est saturé et le collecteur présente une tension nulle par rapport à la masse.

Dans le sens contraire, uLCD vers RS232, le montage est encore plus simple. Lorsque RxD du périphérique uLCD est à la masse, le transistor n'est pas saturé et le collecteur présente une tension correspondant à Vcc par rapport à la masse. Lorsque l'uLCD présente un 1 logique (+3.3 volts), le 2n2222 est saturé et le collecteur ne présente plus de tension par rapport à la masse.

Notez que ce montage ne respecte pas les normes, puisque l'interface RS232 attend une tension négative. Cependant, dans la pratique, une tension nulle est vue comme un 1 logique. C'est une liberté par rapport à la





Interface RS232 vers uLCD **Fig. 1**

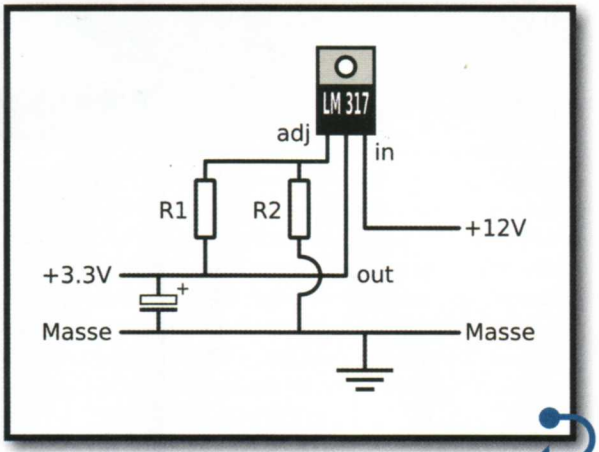
norme qui est, semble-t-il, déjà prise par les fabricants d'interfaces RS232.

Il existe une autre solution pour interfacer l'écran sans utiliser de MAX3232. L'astuce consiste à utiliser un MAX232 permettant la conversion des signaux RS232 vers TTL (0/+5V) et inversement. La documentation du périphérique uLCD précise que pour interfacer l'écran avec un microcontrôleur alimenté en +5 volts, il suffit de placer une résistance de 1000 ohms sur la ligne RX et une autre sur TX. On peut donc parfaitement procéder de la même manière avec le MAX232.

ALIMENTATION

L'autre difficulté concernant la mise en œuvre de l'écran uLCD réside non pas dans son alimentation (les +5 volts sont présents partout dans un ordinateur), mais dans la tension utilisée pour les signaux : 3.3 volts.

Il nous faut, en effet, générer la tension présentée aux bornes des résistances connectées aux collecteurs des transistors. Nous utiliserons ici un régulateur de tension LM317. Très courant, ce composant est fréquemment utilisé dans les alimentations stabilisées. Il dispose d'une broche accueillant la tension d'entrée (in), une autre pour la tension de sortie (out) et une troisième pour l'ajustement (adj).



Mise en œuvre du régulateur de tension LM317 **Fig. 2**

La figure 2 présente le montage du régulateur. On alimente en +12 volts prélevé, par exemple, sur un connecteur d'alimentation interne du PC (fil jaune). La tension à obtenir est ajustée grâce à deux résistances R1 et R2 suivant le calcul out = 1.25*(1+(R2/R1)). Dans le cas présent, R2 vaut 2200 ohms et R1 1330 ohms (1000+330), ce qui nous permet d'obtenir une tension de 3.317 volts.

Notez la présence du condensateur (environ 0.1 uF) en sortie, permettant de nettoyer le courant. Ce n'est pas obligatoire, mais toutefois fortement conseillé. Pour votre montage d'alimentation régulée, pensez à vérifier la documentation de votre LM317. En effet, le brochage change en fonction du *packaging* du composant. Une inversion de polarité signifie souvent la destruction du composant.

Dans tous les cas, avant connexion, utilisez votre multimètre pour vérifier. Mieux vaut perdre quelques minutes en mesure que quelques dizaines d'euros de composants.

INTERFAÇAGE USB

Le revendeur chez qui j'ai acheté mes uLCD (Dontronics) vend également, en option, un adaptateur USB appelé « module Micro-USB ». Il s'agit d'un convertisseur USB/Série miniaturisé construit autour d'une puce CP2101. Celle-ci est parfaitement supportée sous Linux.

La particularité de ce module coûtant quelques 17 euros est de s'interfacer parfaitement avec l'écran uLCD, et ce, aussi bien mécaniquement qu'électriquement. En effet, le circuit à l'arrière de l'écran dispose d'un emplacement prévu pour le module Micro-USB.

En dehors de son utilisation avec l'écran uLCD, le module présente quelques avantages comme celui de fournir une alimentation régulée en 3.3 volts pour le montage auquel il se connecte. Mais les limitations sont également au rendez-vous, puisque les signaux RX/TX sont eux aussi en 3.3 volts. Idéal pour l'uLCD, il n'en ira pas nécessairement de même pour un autre montage.

Bien entendu, si vous souhaitez contrôler l'écran via le bus USB, n'importe quel adaptateur pourra faire l'affaire. Assurez-vous simplement des tensions fournies par l'adaptateur et de celles utilisées pour les signaux. Des modules USB/Série TTL se trouvent facilement pour quelques 15 euros sur les sites d'enchères en ligne.

ACCÈS AU PORT SÉRIE SOUS LINUX

Comme vous le savez sans doute, les ports série sont facilement accessibles sous Linux via une entrée dans `/dev`. Toute l'infrastructure noyau est déjà en place et, pour accéder aux ports, un simple code utilisateur est suffisant.

Comme les choses sont bien faites et grandement facilitées pour le développeur, le support de la configuration du port série est présent dans la `libc`. Inutile donc d'utiliser les en-têtes du noyau (c'est mal), comme c'est le cas, par exemple, pour le bus i2c ou encore de recourir à des bibliothèques externes (comme pour l'USB).

Avant d'utiliser un port série sous Linux, il est nécessaire de le configurer ou, du moins, de s'assurer que sa configuration courante correspond à nos attentes. Deux modes d'utilisation sont possibles pour le programmeur. Le premier mode, appelé « canonique », est destiné à être utilisé pour les terminaux série. Dans ce mode, dit « orienté ligne », les données sont transmises ligne par ligne. Un *buffer* est rempli et peut être édité jusqu'à l'arrivé d'un caractère CR (*Carriage Return*), d'une fin de fichier (EOF) ou LF (*Line Feed*).

Ce mode est parfait pour des terminaux, mais inapproprié pour la communication avec un modem ou tout autre périphérique série asynchrone manipulant les données caractère par caractère, comme le module uLCD.

Il nous faut donc nous tourner vers le second mode dit « *raw* » ou tout simplement non canonique. Là, nous ne travaillons plus ligne par ligne, mais avec des paquets de caractères. L'écriture sur le port comme la lecture sont déterminées par un nombre de caractères défini et une base de temps permettant de provoquer un *timeout*.

Dans les deux modes, la communication peut être synchrone ou asynchrone. Attention, nous sommes ici à un certain niveau d'abstraction du matériel et cela n'a rien à voir avec le caractère asynchrone de la norme RS232. Dans la communication, ou plus exactement la lecture synchrone, un appel à `read` sera bloquant jusqu'à ce que la

lecture soit satisfaite ou que le délai d'attente soit écoulé. En asynchrone, l'appel à read retournera immédiatement et un signal sera émis lorsque l'opération sera terminée. Ceci impose donc l'installation d'un gestionnaire de signaux dans votre programme. Dans le cadre de cet article, nous ferons au plus simple et utiliserons une lecture synchrone.

La configuration du mode, du caractère synchrone ou non, de la vitesse, du format des données et des autres fonctionnalités se fait par le remplissage d'une structure de type termios. Nous commençons donc par déclarer deux structures de ce type :

```
struct termios oldtio,newtio;
```

oldtio est destiné à recevoir une sauvegarde de la configuration courante. En effet, une application accédant à un port série se doit de laisser la configuration dans l'état d'origine lorsqu'elle se termine. newtio est, bien sûr, notre structure contenant la configuration que nous allons composer. Une structure termios contient les membres suivants :

```
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 contrôle */
```

Nous commençons donc par configurer les modes d'entrée :

```
memset(&newtio, 0x00, sizeof(newtio));

/* ignorer les signaux BREAK en entrée */
newtio.c_iflag = IGNBRK;
/* pas de XON/XOFF
newtio.c_iflag &= ~(IXON | IXOFF | IXANY);
```

Nous passons ensuite aux modes de contrôle :

```
/* vitesse */
newtio.c_cflag |= BAUDRATE;
/* Pas de contrôle de flux hardware (RTS/CTS) */
newtio.c_cflag &= ~CRTSCTS;
/* Valider la réception */
newtio.c_cflag |=  CREAD;
/* 8 bits de données, pas de parité */
newtio.c_cflag &= ~(PARENB | CSIZE);
newtio.c_cflag |= CS8;
/* Ignorer les signaux de contrôle du modem */
newtio.c_cflag |= CLOCAL;
/* pas d'echo */
newtio.c_cflag &= ~ECHO;
```

BAUDRATE est défini par ailleurs sous la forme #define BAUDRATE B115200.

Le mode non canonique est ici implicitement entendu, puisque la structure est entièrement mise à zéro et que le drapeau ICANON du membre c_lflag n'est pas positionné. Parmi les éléments importants de cette configuration, on remarquera la désactivation de tout contrôle de flux aussi bien logiciel que matériel ou encore la configuration du format des données en 8 bits sans parité (8N1).

Deux éléments du tableau c_cc sont très importants pour la suite des opérations :

```
newtio.c_cc[VTIME] = 50;
newtio.c_cc[VMIN]  = 0;
```

L'entrée VTIME définit le délai, en dixièmes de secondes, pour une lecture. Au-delà de ce délai, si les données attendues ne sont pas arrivées, un timeout sera provoqué. En plaçant cette valeur à 0, vous indiquez que le *timer* n'est pas utilisé. VMIN indique le nombre minimum de caractères à recevoir avant qu'une lecture soit satisfaite. Ici, nous définissons 0. Avec newtio.c_cc[VTIME] = 50, ceci permet d'indiquer que la lecture est satisfaite si un seul caractère est reçu ou que le délai de 5 secondes est écoulé.

Une fois la configuration définie, nous pouvons la mettre en place. Pour cela, nous commençons par accéder au pseudo-fichier :

```
#define MODEMDEVICE "/dev/ttyS0"
[...]

if ((fd = open(MODEMDEVICE,O_RDWR|O_NOCTTY)) < 0) {
  fprintf(stderr,"Open Error : %s (%d)\n",
      strerror(errno),errno);
  exit(EXIT_FAILURE);
}
```

Nous récupérons ensuite la configuration courante dans oldtio :

```
if(tcgetattr(fd,&oldtio) != 0) {
  fprintf(stderr,"tcgetattr Error : %s (%d)\n",
      strerror(errno),errno);
  exit(EXIT_FAILURE);
}
```

Nous éliminons ensuite toutes les écritures sur l'objet fd qui ne sont pas encore transmises, ainsi que les données reçues mais pas encore lues. Ceci permet de « *flusher* » toutes les données en attente avant le changement de configuration :

```
if(tcflush(fd, TCIOFLUSH) != 0) {
  fprintf(stderr,"tcflush Error : %s (%d)\n",
      strerror(errno),errno);
  exit(EXIT_FAILURE);
}
```

Enfin, avec tcsetattr, nous reconfigurons le port à notre convenance en utilisant la structure que nous venons de remplir :

```
if (tcsetattr(fd,TCSANOW,&newtio) != 0) {
  fprintf(stderr,"tcsetattr Error : %s (%d)\n",
      strerror(errno),errno);
  exit(EXIT_FAILURE);
}
```

Suite à la configuration, l'écriture et la lecture sont simplissimes :

```
unsigned char *data
unsigned char buf[255];
int res=0;

if(write( fd, data, taille ) != taille){
  fprintf(stderr,"Write Error : %s (%d)\n",
      strerror(errno),errno);
}

if((res = read(fd,buf,255)) < 0) {
  fprintf(stderr,"Read Error : %s (%d)\n",
      strerror(errno),errno);
}
```

DIALOGUE AVEC L'ULCD

A présent, nous savons communiquer via le port série, mais encore faut-il « parler » uLCD. La documentation du module écran est des plus claires et le protocole tout autant. Les commandes destinées au module sont composées d'un seul caractère suivi éventuellement d'arguments dépendant de la nature de la commande.

La logique du module ne contient qu'un contrôle d'erreur très limité. Le module attend commande et arguments en un seul flot de données sans aucun séparateur. Lorsque la commande complète est envoyée, le module accuse réception en retournant le caractère 0x06. Ceci indique que le module a reçu et interprété correctement la commande. Si la commande reçue est incomplète, le module attendra indéfiniment le reste des arguments. Si le flot de données est complet mais incorrect, le module le signalera en retournant la valeur 0x15. Le plus souvent, en cas d'erreur de communication, c'est la commande suivante qui complète la précédente et débouche sur l'émission de 0x15 par le module. Aucune gestion de délai n'est incluse, une commande incomplète ne pourra donc être détectée avant l'envoi d'autres données.

Pour faciliter l'envoi de données au module, nous allons utiliser une fonction spécialement conçue :

```
unsigned char sendlcd(int fd, unsigned char *data, int taille)
{
  int res=0;
  unsigned char buf[255];

  if(write( fd, data, taille ) != taille){
    fprintf(stderr,"Write Error : %s (%d)\n",
        strerror(errno),errno);
    return(0xFF);
  }

  buf[0]=0x00;
  if((res = read(fd,buf,255)) < 0) {
    fprintf(stderr,"Read Error : %s (%d)\n",
        strerror(errno),errno);
    return(0xFF);
  }

  if(res == 0 ) {
    fprintf(stderr, "sendlcd - Rien lu\n");
    return(0xFF);
  }

  return(buf[0]);
}
```

Le fil des opérations est relativement facile à suivre. La fonction prend en argument le descripteur de fichier du port série préalablement configuré, un pointeur sur les données à envoyer et la taille de ces données. L'envoi sur le port série se fait avec une simple écriture via `write`. On lit ensuite le port pour s'assurer que le module a bien exécuté notre commande. N'oubliez pas, nous avons configuré le port de manière à ce que l'appel à `read` soit bloquant jusqu'à ce que nous ayons obtenu un octet ou que le délai de 5 secondes soit écoulé. Si `read` retourne une valeur négative, c'est que l'opération de lecture a échoué et nous retournons alors une valeur arbitrairement choisie de 0xFF. De la même manière, si nous n'avons rien lu (`res == 0`), nous le signalons et retournons la même valeur. Enfin, nous retournons le premier octet de la valeur lue, si tout semble bien se passer. A la charge du code ayant appelé la fonction `sendlcd` d'analyser la valeur de retour.

Notez que cette fonction ne permet pas de gérer toutes les commandes destinées au module. Ceci concerne en particulier la fonction de lecture des pixels. La commande de lecture retourne alors deux octets correspondants aux 16 bits de couleur du pixel lu.

Pour simplifier les opérations, nous définissons une macro :

```
#define SENDLCD(fd,str) sendlcd(fd, str, (sizeof(str)/
sizeof(*str)))
```

Ainsi, nous n'avons pas à déterminer nous-même, à chaque fois, la taille des données à envoyer.

CONTRÔLE DE L'AFFICHAGE

Les commandes à envoyer à l'écran se composent de suites de caractères. La première commande à impérativement envoyer après la mise sous tension du module est l'ordre de détection automatique de la vitesse. Le module uLCD est, en effet, capable de supporter toute une gamme de vitesses allant de 300 bauds à 128 K. Nous avons défini

par ailleurs dans la configuration du port une vitesse de 115200 (`#define BAUDRATE B115200`).

La commande d'auto-évaluation de la vitesse ne se compose que d'un seul caractère 0x55 ("U"). Nous définissons donc notre première commande sous la forme d'un tableau d'un seul élément :

```
unsigned char lcdautobaud[1] = { 0x55 };
```

Ceci peut paraître surprenant, mais comme nous allons définir d'autres commandes plus importantes (jusqu'à 5 caractères), autant garder une certaine cohérence dans la syntaxe. Pour envoyer la commande, nous utilisons la macro précédemment définie :

```
if((res=SENDLCD(fd, lcdautobaud )) != 0x06) {
  fprintf(stderr,"...no ACK ! (0x%.02X)\n",res);
}
```

A l'usage, on remarque que cette commande, lorsqu'elle est utilisée immédiatement après la mise sous tension, ne retourne jamais 0x06. Ce n'est pas vraiment un problème, même si cela reste cosmétiquement gênant. La seconde commande la plus intéressante est celle qui provoque l'effacement de l'écran. Là encore, il s'agit d'un seul caractère :

```
unsigned char lcderase[1] = { 0x45 };
```

Autre élément de contrôle de l'affichage, nous pouvons déterminer l'intensité du retro-éclairage. La LED blanche incluse dans l'afficheur peut, en effet, prendre trois valeurs : éteinte, niveau 1, niveau 2. Par défaut, à la mise sous tension, la LED est au niveau 1. L'intensité lumineuse est contrôlable via la commande 0x59 ("Y") prenant en charge plusieurs éléments de configuration. Deux arguments suivent, le premier est élément de configuration 0x00, pour le contrôle du rétro-éclairage, 0x01 pour l'allumage/extinction de l'écran LCD, 0x02 pour le contraste et, enfin, 0x03 pour l'allumage/extinction de l'alimentation LCD.

Dans le cas du contrôle de rétro-éclairage, le dernier argument est l'intensité, de 0x00 pour aucun à 0x02 pour un éclairage maximum. Il nous suffit donc de déclarer :

```
unsigned char lcdbacklight[3] = { 0x59, 0x00, 0x00 };
```

Puis, nous changeons `lcdbacklight[2]` avant utilisation de la macro :

```
lcdbacklight[2]=0x02;
if((res=SENDLCD(fd, lcdbacklight )) != 0x06) {
  fprintf(stderr,"...no ACK ! (0x%.02X)\n",res);
}
```

Idem pour le contraste :

```
unsigned char lcdcontrast[3] = {0x59, 0x02, 0x17};
```

Puis :

```
lcdcontrast[2]=0x28;
if((res=SENDLCD(fd, lcdcontrast )) != 0x06) {
  fprintf(stderr,"...no ACK ! (0x%.02X)\n",res);
}
```

Attention! : La documentation précise que, avant de déconnecter le module uLCD, il est impératif de l'éteindre avec la suite `{ 0x59, 0x03, 0x00 }`. Cette commande désactive les *boosters* de voltage interne, les amplificateurs de courant et coupes de rétro-éclairage. Cette commande d'extinction peut également servir, dans le cas d'utilisation du module avec un système embarqué, afin de réduire la consommation électrique. Si le module n'est pas débranché, il peut alors être rallumé avec `{ 0x59, 0x03, 0x01 }`. Toutefois, de nombreux essais ont montré que le risque d'endommager le module par un débranchement sauvage restait limité. Les expérimentations sont par ailleurs confirmées par différents messages d'employés de Dontronics sur la page consacrée au module.

AFFICHAGE D'IMAGE

Je l'ai dit en introduction, le module uLCD intègre toute une collection de fonctionnalités sous la forme de primitives graphiques (texte, ligne, boîte, cercle). Mais, ce qui nous intéresse ici est l'affichage d'images pixel par pixel. De plus, il serait déplacé de paraphraser inutilement la documentation du module qu'il est possible de télécharger en PDF, même avant achat sur le site de Dontronics.

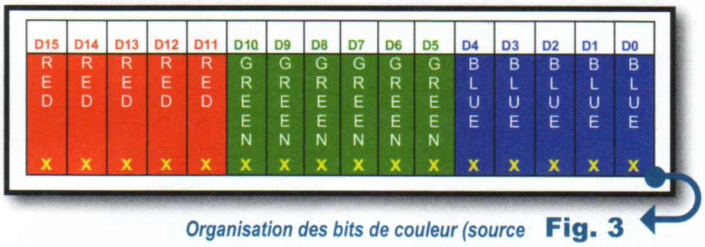
Il serait facile de placer sur l'écran LCD des pixels codés « en dur ». Facile, mais malheureusement peu pratique. Nous allons donc utiliser des données provenant d'un fichier graphique. Pour cela, nous délèguerons la tâche de lecture et de décodage du format de fichier à la bibliothèque GD.

Grâce à la cette bibliothèque, nous pouvons très facilement lire les pixels d'une image PNG, par exemple :

```
FILE *in;
gdImagePtr im=NULL;
int imgh, imgl;

if((in = fopen("image.png", "rb")) == 0) {
  fprintf(stderr,"Open img Error : %s (%d)\n",
      strerror(errno),errno);
  exit(EXIT_FAILURE);
}

if((im = gdImageCreateFromPng(in)) != NULL) {
  fclose(in);
  imgh = gdImageSY(im);
  imgl = gdImageSX(im);
  if(imgh > 128 || imgl > 128) {
    fprintf(stderr,"Image must be <= 128x128\n");
    return(EXIT_FAILURE);
  }
}
```

La fonction `gdImageCreateFromPng` nous permet d'obtenir un pointeur de type `gdImagePtr` vers une image utilisable. Nous pouvons donc immédiatement obtenir les dimensions de l'image et réagir si celle-ci dépasse la taille de l'écran (128x128).

Une fois la dimension connue, nous pouvons boucler ligne par ligne et colonne par colonne pour obtenir les trois composantes de bases, rouge, vert et bleu. Mais nous nous heurtons à un problème : l'image PNG est en 24 bits, un octet par couleur alors que les pixels de l'écran LCD ne disposent que de 16 bits de couleur.



Organisation des bits de couleur (source documentation uLCD) **Fig. 3**

La figure 3 montre la répartition des bits de couleur dans le double octet. Nous avons 5 bits de rouge, 6 de vert et 5 de bleu. Il nous faut donc adapter les couleurs RVB 24 bits de l'image à ce schéma :

```
long int c;
int r,g,b;
int mr,mg,mb;
unsigned char b1, b2;

c = gdImageGetPixel(im, l, h);
r = gdImageRed(im,c);
g = gdImageGreen(im,c);
b = gdImageBlue(im,c);
mr = r*32/256; /* reduction 8b / 5b */
mg = g*64/256; /* reduction 8b / 6b */
mb = b*32/256; /* reduction 8b / 5b */
b1 = (mr << 3) | ((mg >> 3) & 7);
b2 = ((mg << 5) & 224) | mb;
```

La fonction `gdImageGetPixel` nous permet de récupérer les informations sur un pixel de l'image en lui passant en argument le pointeur vers l'image ainsi que les coordonnées du pixel visé. On utilise ensuite `gdImageRed`, `gdImageGreen`, et `gdImageBlue` pour connaître la valeur des trois couleurs pour le pixel. De simples opérations arithmétiques et logiques ainsi que des décalages de bits nous permettent ensuite d'obtenir les deux caractères décrivant la couleur du pixel à afficher.

Pour afficher l'image, il ne nous reste plus qu'à boucler pixel par pixel pour modifier, avant utilisation de la macro, les valeurs du tableau correspondant à la commande d'affichage de pixel :

```
unsigned char putpix[5] = { 0x50, 0x00, 0x00, 0x00, 0x00 };
int h, l;
```

```
for(h=0; h<imgh; h++) {
  for(l=0; l<imgl; l++) {
    [...]
    putpix[1]= l;
    putpix[2]= h;
    putpix[3]= b1;
    putpix[4]= b2;
    SENDLCD(fd, putpix);
  }
}
```

L'opération d'affichage d'une image complète sous cette forme prend beaucoup de temps. En effet, les quelques 15640 pixels mettent environ 16 secondes pour remplir l'écran avec un débit configuré à 115200 bauds. Il doit sans doute être possible d'accélérer l'affichage (en utilisant le mode asynchrone ?). Un rapide calcul sommaire indique que nous envoyons 128*128*5 octets et que nous recevons, de plus, 128*128 accusés de réception du module uLCD. Ce qui nous donne (128*128*5+128*128)*8 bits, soit 786432. Avec un débit théorique de 115200, l'affichage devrait mettre moins de 7 secondes, soit deux fois moins de temps.

OPTIMISATION DE L'AFFICHAGE

Comment accélérer l'affichage d'une l'image ? La réponse tient en une seule expérimentation et en une seule commande à envoyer au module uLCD : celle du changement de couleur de fond. En effet, le microcontrôleur PIC ayant à sa charge l'interface entre nos directives et l'écran, à proprement parler, dispose d'une gestion basique des couleurs d'avant et d'arrière-plan. Il est ainsi possible de remplir l'écran d'une couleur qui sera considérée ensuite comme le fond. Les directives d'affichage de texte et de formes laissent le fond dans l'état.

La commande de changement du fond (blanc à la mise sous tension du module) tient en trois caractères :

```
unsigned char fond[3] = { 0x42, 0x00, 0x00 };
```

Les deux arguments déterminent la couleur selon la syntaxe 16 bits que nous connaissons déjà. Un rapide essai via notre macro montre clairement que le remplissage, délégué au microcontrôleur est bien plus rapide que l'affichage des pixels (environ une seconde). L'idée est donc d'utiliser cette fonctionnalité pour choisir une couleur de fond qui nous permettra d'éviter de dessiner tous les pixels. Il nous suffit de déterminer la couleur dominante de l'image, remplir le fond et sauter tout les pixels ayant une couleur identique.

Nous faisons donc une première passe sur les données de l'image :

```
int couleurs[256][256];
```

```
for(cy=0;cy<256;cy++) {
  for(cx=0;cx<256;cx++) {
    couleurs[cx][cy]=0;
  }
}

for(h=0; h<imgh; h++) {
  for(l=0; l<imgl; l++) {
    c = gdImageGetPixel(im, l, h);
    r = gdImageRed(im,c);
    g = gdImageGreen(im,c);
    b = gdImageBlue(im,c);
    mr = r*32/256; /* reduction 8b / 5b */
    mg = g*64/256; /* reduction 8b / 6b */
    mb = b*32/256; /* reduction 8b /  5b */
    b1 = (mr << 3) | ((mg >> 3) & 7);
    b2 = ((mg << 5) & 224) | mb;
    couleurs[b1][b2]++;
  }
}
```

Nous utilisons un tableau bidimensionnel dont chaque cellule est préalablement initialisée à 0. On parcourt ensuite l'image et incrémente les cellules correspondant aux intersections entre le premier et le second caractère de couleur 16 bits. Il ne nous reste ensuite plus qu'à déterminer quelle cellule contient la valeur maximale pour connaître la couleur ayant le plus d'occurrences :

```
unsigned char fond1 = 0x00;
unsigned char fond2 = 0x00;
int max=0;

for(cy=0;cy<256;cy++) {
  for(cx=0;cx<256;cx++) {
    if(couleurs[cx][cy]>max) {
      fond1 = cx;
      fond2 = cy;
      max=couleurs[cx][cy];
    }
  }
}
```

En sortie de boucles, nous obtenons `max` contenant le nombre d'occurrences ainsi que `fond1` et `fond2` correspondant à la couleur concernée. Enfin, il nous suffit de réécrire la boucle d'affichage en prenant en considération les nouvelles informations en conditionnant l'affichage avec un simple `if(b1==fond1 && b2==fond2)` auquel cas, éventuellement, nous incrémentons un compteur en guise d'information complémentaire.

Les performances obtenues sont, bien entendu, en relation directe avec l'image. Avec une icône, le changement est flagrant. Nous sautons une grande majorité de pixels. Avec une photo, les résultats sont plus mitigés mais existants, même si nous parcourons deux fois les données de l'image.

D'autres optimisations sont possibles et facilement implémentables. Je parle, par exemple, du fait de centrer l'image si ses dimensions sont inférieures à celles de l'écran. Un simple calcul de division et une modification des coordonnées d'affichage et le tour est joué.

PERSPECTIVES

Nous nous sommes limités ici à une utilisation très précise du module uLCD. Le projet à la base de l'article était une sorte de cadre photo numérique miniature. Celui-ci était destiné à prendre place sur le bureau et à afficher, en boucle, une série d'images placées dans un répertoire. Ce type de cadre photo numérique est disponible dans le commerce pour quelques 200 euros. Ils disposent souvent d'écran LCD de 5 à 10 pouces et affichent des images placées sur des cartes mémoires de type SD ou MMC.

Ici, nous avons un écran plus petit et surtout moins cher (quelques 30 euros chez le détaillant australien Dontronics). De plus, comme il ne s'agit pas d'un produit manufacturé, l'étendue des possibilités est bien plus vaste. Nous pouvons, par exemple, envisager, en surimpression des photos, l'affichage de données textuelles indiquant le nombre de mails à lire, la charge mémoire ou CPU, etc.

En sortant du cadre d'utilisation purement décoratif, là encore, les possibilités sont nombreuses : indication de charge d'un serveur, horloge, syndication RSS, relevé de températures...

VERSION 2

Avant de conclure, il me paraît intéressant de signaler qu'il existe d'autres versions du module. Plus récentes, elles sont également plus riches en fonctionnalités. La première inclut une mémoire flash de 1Mb permettant de stocker des images. Cette fonctionnalité permet également d'afficher une séquence de plusieurs images enregistrées pour créer des animations. Bien entendu, le prix n'est plus le même et la note passera de 30 à plus de 45 euros.

Enfin, nous avons la toute dernière génération de modules. Elle se décline sur la base des deux précédents modules (avec ou sans mémoire flash), mais la différence, de taille, réside dans l'écran. Il ne s'agit plus de LCD, mais de la technologie OLED. Les OLED ou *Organic Light-Emitting Diodes* permettent de produire des images plus lumineuses, plus contrastées et plus fines. L'écran lui-même est plus mince, consomme moins de courant et l'angle de vision est bien plus large. Bien entendu, le prix augmente, mais reste raisonnable avec ses quelques 60 euros.

Pour l'heure, seule la version OLED avec les 1 Mb de mémoire flash est disponible. On notera que Dontronic affiche clairement la couleur en spécifiant que l'augmentation de prix est strictement limitée à la différence de coût entre LCD et OLED. Vous l'aurez compris, ces versions n'ont pas encore fait l'objet d'essais (mais la tentation est grande).

Denis Bodor :: db@ed-diamond.com :: lefinnois@lefinnois.net

J.-M. Friedt <friedtj@free.fr>, É. Carry, <milou@achelem.org>,
Association Projet Aurore, Besançon

PRISES DE VUES AUTOMATIQUES

Nous proposons diverses approches permettant des prises de vues à intervalles de temps prédéfinis afin de suivre l'évolution à long terme d'évènements lents dont la dynamique est difficile à appréhender au quotidien. Nous présenterons diverses solutions fournissant des puissances de calcul décroissantes et donc des fonctionnalités de plus en plus simples, mais surtout diminuant à chaque étape la consommation électrique et donc permettant une autonomie plus grande.

«... it is well known that a vital ingredient of success is not knowing that what you're attempting can't be done. »

Terry Pratchett, *Discworld*, vol. 3 (Equal Rites)

Nous irons ainsi de la carte d'acquisition sur PC à l'appareil photographique numérique (APN) contrôlé par USB et RS232 pour finalement atteindre une solution de très faible consommation où l'APN est totalement contrôlé par un microcontrôleur réveillé par une horloge temps réel dédiée à ce type d'applications.

1. INTRODUCTION

Dès l'apparition des premières webcams, avec la disponibilité d'outils de capture d'images en ligne de commande, la possibilité de réaliser des prises de vues à intervalles de temps réguliers afin de faire des films accélérés est apparue évidente [1].

Les webcams ont cependant été dans un premier temps limitées à de la prise de vue statique en tons de gris et surtout à une résolution médiocre. Nous nous sommes alors tournés vers l'utilisation de cartes d'acquisition vidéo associées à une caméra [1]. Une originalité de notre approche a été l'utilisation d'un relais pour sélectionner deux vues d'un même évènement. Cette solution est d'autant plus attractive que le prix des cartes d'acquisition vidéo ne cesse de chuter, que des outils en ligne de commande sont disponibles pour une insertion dans un script, et finalement, que la prise de vue peut être contrôlée par un événement au lieu d'une prise de vue périodique.

Ces solutions ont cependant l'inconvénient majeur d'immobiliser un ordinateur pendant la durée de l'acquisition, avec la consommation électrique associée. Nous nous sommes donc plus récemment tournés vers les appareils photo numériques (APN) qui fournissent une bien meilleure résolution pour un coût actuellement inférieur à 100 euros. Nous présenterons deux approches à l'utilisation d'APN : la première utilisant le contrôle par ports USB ou RS232 ne nécessitant pas de modification de l'appareil, et la seconde nécessitant de démonter l'appareil photo afin de simuler l'appui des différentes touches qu'utiliserait un opérateur humain pour effectuer la prise de vue.

2. LA CARTE D'ACQUISITION VIDÉO SUR PC

La solution la plus triviale à la prise de vue consiste à connecter un périphérique capable d'acquérir une image – webcam ou carte d'acquisition vidéo – et de lancer périodiquement une commande de prise de vue. Une application plus intéressante de cette approche justifiant la consommation électrique énorme d'un PC et l'immobilisation d'un ordinateur pendant la durée de l'expérience est le déclenchement de la prise de vue sur un événement. Le programme `motion` disponible à http://www.lavrsen.dk/twiki/bin/view/Motion/WebHome permet de déclencher la prise de vue en fonction d'un événement, qu'il s'agisse de l'écoulement d'un intervalle de temps prédéfini ou lorsqu'un mouvement est détecté dans le champ de vision de la caméra. Il se connecte à tout périphérique vidéo suivant la norme `video4linux`.

`motion` est capable de gérer indépendamment plusieurs périphériques vidéo au moyen de fichiers de configuration distincts. Nous nous placerons dans le cas d'une webcam USB en mode détection de mouvement et d'une caméra analogique en prise de vue à intervalles réguliers.

Pour commencer, nous allons détailler le fichier de configuration général `/etc/motion.conf` présenté dans le tableau 1.

TABLEAU 1

```
daemon off
quiet on

thread /etc/motion/cam0.conf
thread /etc/motion/cam1.conf

# Image size in pixels
width 640
height 480
```

```
framerate 25
quality 85
auto_brightness off
```

Le fichier /etc/motion/motion.conf *permettant de lancer des captures depuis plusieurs sources différentes.*

`daemon off` nous permet de ne pas lancer `motion` en mode démon et ainsi de pouvoir affiner les réglages de la prise de vue. `quiet on` évite d'avoir un « *beep* » à chaque détection de mouvement, suivant le sujet observé, cela peut être ennuyeux (souris, cambrioleur...). Nous définissons ensuite les deux fichiers de configurations de nos caméras, puis nous donnons les caractéristiques communes voulues pour nos films et images.

Dans notre exemple, nous utilisons une webcam Philips ToUCam II avec les pilotes de Luc Saillard [2] : nous passons ainsi d'une résolution 160x112 pixels à une résolution 640x480 et augmentons le débit d'images. Une fois la webcam fonctionnelle, nous utilisons un logiciel de visualisation comme `camstream` afin de positionner la caméra correctement. Il est important de noter que lors de l'utilisation d'une caméra USB 1.1, on consomme la quasi-totalité de la bande passante du contrôleur USB. Il est donc illusoire de vouloir brancher deux webcams autrement qu'en utilisant de l'USB2 ou en ajoutant un contrôleur additionnel. Nous rentrons maintenant dans le vif du sujet avec la configuration de la caméra décrite dans le tableau 2.

TABLEAU 2

```
#device de la webcam USB
videodevice /dev/video2

#nombre de pixels changés
threshold 4000

#Temps minimum en seconde de fin d'évènement
gap 10

#répertoire de stockage des données
target_dir /home/milou/motion/cam0/

#mode film active
ffmpeg_cap_new on

#tampon rotatif d'image avant capture
pre_capture 5

#nombre d'images ajoutées en fin de capture
post_capture 10

#codec de sortie
ffmpeg_codec mpeg4

#nom des films de sortie
ffmpeg_filename cam0-%v-%Y%m%d%H%M%S
```

```
#enregistrement des images du mouvement
output_normal on

#texte supplémentaire incruste sur l'image
text_left cam0
```

Le fichier /etc/motion/cam0.conf *appelé par* motion.conf *présenté auparavant (Tab. 1). Ce port USB est connecté à une webcam.*

Il y a deux aspects à regarder dans cette configuration : le seuil de détection d'un mouvement et le stockage des données. L'option cruciale de `motion` est le `threshold`, qui définit le nombre de pixels changeant entre 2 images, indiquant s'il y a ou non un mouvement. Cette valeur est à régler avec soin et dépendra de la résolution de l'acquisition (nombre total de pixels disponibles) et du type de mouvement à détecter (détecter une souris ou un éléphant dans la même pièce). La méthode usuelle est de désactiver tous les filtres de débruitage, régler un premier niveau de détection puis d'activer et régler les uns après les autres les différents filtres disponibles (afin par exemple d'éliminer le mouvement des feuilles d'un arbre). Il est aussi possible de n'afficher que les pixels du mouvement avec les options `ffmpeg_cap_motion` et `output_motion` ou encore de dessiner une boîte autour de la zone de mouvement grâce à l'option `locate` (Fig. 1).

Souris prise sur le fait : le cadre sur l'image est indicateur de la zone où motion *a détecté le mouvement.*

Fig. 1

Maintenant que le mouvement est correctement détecté, il faut le stocker. `motion` permet de générer automatiquement des fichiers vidéo des mouvements détectés à l'aide de `ffmpeg` et de prendre des images de ces mouvements afin de pouvoir faire du post-traitement (faire de la détection de forme ou des mesures par exemple). Nous avons paramétré les deux solutions.

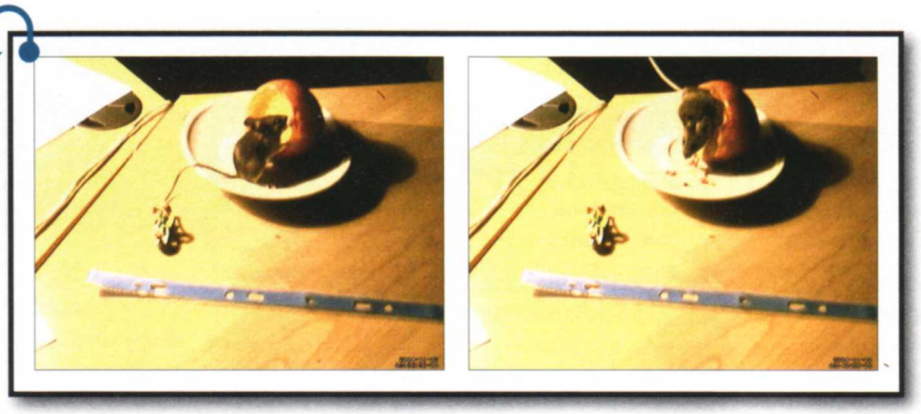


Fig. 2

Souris à table

Nous activons la génération de vidéo par l'option `ffmpeg_cap_new on` et configurons le nom des fichiers générés et leur codec par les options `ffmpeg_filename` et `ffmpeg_codec` (à noter que seuls des codecs de type `mpeg4` sont disponibles). Il est possible de donner un don de prémonition à votre caméra ! En fait, le réglage de l'option `pre_capture` va générer un tampon rotatif d'images qui sera placé au début de la vidéo pour donner un effet de pause juste avant le début du mouvement. Il faut utiliser avec parcimonie cette option, une valeur au-delà de 5 risquera de faire perdre les premières images du mouvement... À l'opposé, il est intéressant d'utiliser `post_capture` pour ajouter des images après la fin du mouvement et ainsi rendre fluide la fin de la vidéo.

Il peut être intéressant d'enregistrer toutes les images composant le mouvement pour pouvoir les réutiliser pour du traitement d'image (ou pour les réutiliser dans un article : Fig. 2). Pour cela, nous activons l'option `output_normal` à `on`. Bien entendu, `motion` dispose de plusieurs options pour configurer le nom de ces images, leur qualité, leur orientation ou encore leur format. Nous verrons avec l'exemple suivant comment traiter et générer simplement des vidéos à partir de ces images.

`motion` incruste d'office la date et l'heure dans un coin de l'image et vous laisse toute liberté pour ajouter du texte, le supprimer ou en placer dans les quatre coins de l'écran. Nous avons ici sagement placé une indication sur le numéro de caméra dans un coin. Les films issus de cet exemple sont disponibles sur **http://projetaurore.assos.univ-fcomte.fr/media/video/**.

Nous nous intéresserons maintenant à la configuration d'une caméra analogique au travers d'une carte d'acquisition vidéo et son utilisation pour prendre des images à intervalles de temps réguliers.

Nous avons utilisé un caméscope VHS SECAM auquel nous avons supprimé la mise en veille lorsqu'il n'y a pas de cassette et nous l'avons connecté à une carte PCI d'acquisition vidéo Pinnacle PCTV à base de circuit BT878. Chaque type de carte d'acquisition met à disposition un certain nombre de types d'entrées analogiques, comme par exemple l'entrée tuner, S-VHS ou composite. Il faudra donc indiquer à `motion` dans son fichier de configuration le type d'entrée utilisée, sa norme (PAL, SECAM ou NTSC), puis activer l'acquisition à intervalles de temps réguliers.

TABLEAU 3

```
# carte pctv bt878
videodevice /dev/video1

# entree composite, pour le s-video choisir 2
input 1

#norme SECAM
norm 2

# repertoire de stockage
target_dir /home/milou/motion/cam1/

#intervalle de temps entre images de la video
ffmpeg_timelapse 60

#mode de prise de vue (chaque heure, jour, mois, etc...)
ffmpeg_timelapse_mode manual

#motif pour le nom de la video
timelapse_filename cam1-%v-%Y%m%d-timelapse

#intervalle de temps entre chaque image en seconde
snapshot_interval 60

#motif du nom des images
snapshot_filename cam0-%v-%Y%m%d%H%M%S-snapshot

text_left cam1
```

Le fichier `/etc/motion/cam1.conf` appelé par `motion.conf` présenté auparavant (Tab. 1). Ce port est connecté à une carte d'acquisition vidéo capturant les images depuis un caméscope.

Le type d'entrée est configuré à travers l'option `input` qui, dans notre cas (signal composite), est placée à `1`. Il est à noter que le type par défaut est `8`, celui nécessaire pour une webcam USB. Notre caméscope étant de norme SECAM, nous positionnons l'option `norm` à `2`.

(1) Voir http://www.gnu.org/software/coreutils/faq/#Argument-list-too-long pour un diagnostic du problème.

Il nous reste maintenant à configurer la prise de vue. Nous avons deux choix possibles. Le premier est d'utiliser les capacités de `motion` à prendre des images à intervalles réguliers et de les concaténer automatiquement dans un fichier vidéo au format `mpeg1`. Ceci est rendu possible par l'option `ffmpeg_timelapse_mode` qui permet de prendre des images toutes les heures, tous les jours, tous les lundis, etc. Et il est possible de le mettre en mode manuel. Nous pouvons alors définir un temps adapté au sujet filmé. Nous avons filmé une vue depuis notre local. Pour avoir le mouvement des nuages, nous avons choisi un intervalle de 60 secondes. Cette méthode n'est pas la plus efficace, le `mpeg1` généré est d'une qualité assez décevante. C'est pourquoi nous proposons une deuxième solution où nous enregistrons une image toutes les minutes et nous générons nous même plus tard la vidéo (attention, vous pouvez assez vite saturer votre disque dur !). Ceci nous permet par exemple de supprimer les vues prises la nuit qui ne sont pas forcément très palpitantes (sauf quand la Lune se lève dans l'axe de la caméra). Nous avons donc renseigné les options `snapshot_interval` et `snapshot_filename` de notre fichier de configuration.

Une fois l'acquisition effectuée, nous nous retrouvons avec un certain nombre de fichiers JPEG à traiter. Nous allons pour cela utiliser les deux couteaux suisses de la vidéo que sont `transcode` [3] et `mencoder` [4]. Nous allons tout d'abord générer un fichier contenant la liste ordonnée des `jpeg` à traiter (d'où l'avantage de correctement générer les noms des fichiers).

```
ls *.jpg > list
```

Puis, nous allons utiliser cette liste avec transcode :

```
transcode -i list -x imlist,null --use_rgb \
-z -H 0 -g 640x480 -y ffmpeg,null \
-F mpeg4 -o film.avi
```

`-i list` indique le fichier contenant la liste des images. `-x imlist,null` indique les modules d'importation vidéo et audio à utiliser. `--use_rgb` est utilisé, car le module vidéo d'importation d'image `imlist` ne supporte pas le YUV. `-z` permet de retourner la vidéo tête en bas (il arrive que `transcode` retourne les images). `-H 0 -g 640x480` indique de ne pas tenter d'autodétection de la taille des images : `transcode` échoue pour une liste d'images. C'est pourquoi nous lui imposons la taille avec `-g`. Finalement, `-y ffmpeg,null -F mpeg4` définit le module d'exportation (ici `ffmpeg`) et le codec utilisé, et `-o film.avi` indique évidemment le nom du fichier résultant.

Il est donc possible de générer un fichier `list` qui ne corresponde qu'à une partie des images enregistrées. Par exemple, nous avons laissé allumée la caméra 6 jours d'affilés et nous voulions enlever les images de la nuit. Imaginons par exemple que nous voudrions éliminer les images de la nuit du 5 septembre au 6 septembre en considérant la nuit de 21h à 6h du matin :

```
rm cam1-0?-200609052[1-3]* && \
rm cam1-0?-200609060[0-5]*
```

Il est donc facile de *scripter* la génération du fichier `list`.

Deux remarques tout de même, l'encodage d'une vidéo où s'alternent des images de nuits et de jours nous a posé quelques problèmes. En effet, au moment de la transition nuit/jour, l'image se pixelise. Ceci est sûrement dû à la méthode de compression utilisée. Une parade possible est de générer indépendamment les différentes transitions et de joindre ensuite les fichiers vidéo à l'aide de `mencoder`

```
mencoder -ovc copy -oac copy -o \
out.avi file1.avi file2.avi
```

De plus, vous pouvez avoir généré un grand nombre d'images et obtenir l'erreur `bash: /bin/ls: Liste d'arguments trop longue`. Dans ce cas (1), il faut passer par `find` et `xargs` :

```
find . -name 'cam*.jpg'|xargs ls>list
```

Le film généré avec cet exemple est disponible sur **http://projetaurore.assos.univ-fcomte.fr/media/video/**

Pour conclure cette partie, `motion` est capable de bien plus encore, comme par exemple de faire du *tracking* (il peut piloter un moteur pour suivre une personne avec la caméra) ou encore de la caméra IP. Le principal inconvénient de cette première solution est d'immobiliser un PC pour la durée de l'expérience, avec le bruit et la consommation électrique associés. Nous avons par exemple constaté qu'un ordinateur basé sur un Pentium III cadencé à 800 MHz et équipé d'un disque dur de 2,1 GB et de 256 MB de RAM consomme 180 mA sous 220 V lors des acquisitions vidéo, soit une consommation constante d'environ 40 W.

3. L'APPAREIL PHOTO NUMÉRIQUE AVEC PORT USB

Les appareils photographiques numériques (APN) récents, équipés de ports USB, possèdent généralement deux modes de fonctionnement, l'un les faisant apparaître comme unité de stockage de masse (accessible alors par le module `usb-storage` des noyaux 2.4 et 2.6) et l'autre permettant non seulement de rapatrier les fichiers contenant les images, mais aussi l'envoi de commandes telles que l'activation du flash ou l'ordre de prise de vue (mode nommé PTP). Ce second mode nous intéresse pour la prise de vues automatique à intervalles de temps prédéfinis.

Peu de systèmes informatiques présentent des ports USB maîtres (*USB host*), la majorité des microcontrôleurs possédant un port USB n'étant qu'esclave (type clavier, souris ou webcam par exemple). Cependant, l'immobilisation d'un ordinateur possédant un port USB et la consommation électrique associée ne saurait être justifiée pour une expérience de plus de quelques jours. Aussi proposons nous de contrôler un APN par son port USB depuis la carte FOX commercialisée par ACME *via* Lextronic en France (http://www.lextronic.fr/fox/fox.htm). Nous verrons que la consommation totale de ce système est de l'ordre du watt (250 mA sous 5 V pour l'ordinateur, excluant la consommation de l'APN) : sans prétendre à une application totalement autonome, la consommation est nettement plus raisonnable que celle d'un ordinateur personnel (Fig. 3).

La carte FOX fournit deux ports USB maîtres, ainsi qu'une interface Ethernet contrôlés par un processeur ETRAX (architecture CRIS) cadencé à 100 MHz, capable de faire fonctionner un système GNU/Linux complet depuis ses 16 MB de RAM et 4 MB de mémoire flash. Elle s'alimente par une tension régulée sous 5 V. La principale difficulté que nous allons aborder ici est la *cross-compilation* d'outils à destination du processeur embarqué CRIS depuis un PC équipé d'un processeur compatible Intel.

Circuit de contrôle d'un APN en mode PTP par USB. Les connecteurs RCA blanc et jaune présents sur le câble lié à l'APN ne sont pas utilisés dans cette application.

Fig. 3

L'outil que nous désirons compiler est `gphoto2` (http://www.gphoto.org/) qui fournit une interface en ligne de commande pour communiquer en mode PTP avec une large gamme d'APN.

Une rapide étude de l'arborescence issue d'une compilation de `gphoto2` sur PC montre qu'il nous sera impossible de stocker l'ensemble des programmes associés dans les 4 MB de mémoire flash de la carte FOX. Aussi déciderons-nous dès maintenant de compiler le programme sur une clé USB et d'exécuter ultérieurement depuis ce support. En effet, par défaut, la carte FOX est fournie avec un noyau capable de communiquer sur le port USB avec les périphériques de stockage de masse tels que les clés.

L'étude de l'arborescence de la carte FOX nous montre que les points de montage des périphériques se situent dans `/mnt` avec une sous-arborescence numérotée, à côté de `/mnt/flash` qui fournit un petit espace de mémoire non volatile embarquée. Nous monterons ultérieurement la clé USB contenant notre programme dans `/mnt/1` sur la carte FOX. Il faut donc dupliquer ce point de montage `/mnt/1` sur le PC sur lequel se fera la cross-compilation et y monter la clé USB que nous utiliserons sur la carte embarquée.

La cross-compilation de `gphoto2` et des librairies associées pour la carte FOX nécessite :

1. Les outils de compilation à destination du processeur ETRAX tels que disponibles à http://www.acmesystems.it/?id=701 (fichier `cris-dist_1.63-1_i386.deb`),
2. Un environnement de compilation d'une nouvelle image à flasher dans la carte FOX tel que disponible sur la même page (fichiers `devboard-R2_01-distfiles.tar.gz` et `devboard-R2_01.tar.gz`
3. Les archives de gphoto2 (`gphoto2-2.2.0.tar.gz`), les librairies associées (`libgphoto2-2.2.1.tar.gz` et `libusb-0.1.7.tgz`) ainsi que les outils associés (`popt-1.7.tar.gz`). Mieux vaut éviter la distribution de `libusb` mise à disposition sur le site web d'ACME qui semble manquer de fonctionnalités.

De façon générale, la cross-compilation d'un programme à destination d'un processeur d'architecture CRIS depuis un PC (processeur compatible Intel) s'obtiendra par l'incantation :

```
./configure --host=cris-axis-linux-gnu \
--without-exif --prefix=/mnt/1/gphoto \
--with-drivers=ptp2
make && make install
```

qui signifie que nous allons utiliser les outils de compilation `cris-axis-linux-gnu` (placés par défaut dans le répertoire `/usr/local/cris`) afin de compiler des programmes qui

seront placés dans le répertoire /mnt/1/gphoto), emplacement auquel nous aurons pris soin auparavant de monter la clé USB qui nous servira de support (en fait la clé est montée dans /mnt/1 et contient un répertoire gphoto). Les deux options --with-drivers=ptp2 et --without-exif sont spécifiques à gphoto mais seront simplement ignorées pour les autres compilations.

Dans l'ordre :

1. Exécuter le script d'initialisation des chemins init_env dans le répertoire contenant l'archive permettant de générer l'image GNU/Linux à destination du processeur CRIS.
2. Nous compilons popt-1.7.tar.gz.
3. Nous compilons libusb-0.1.7.tgz.
4. Nous définissons les variables nécessaires pour la suite des opérations :export LIBUSB_CFLAGS=-I/usr/local/cris/include et export LIBUSB_LIBS="-L/mnt/1/gphoto/lib/ -fPIC -lusb" pour la libusb et de même pour popt (export POPT_LIBS="-L/mnt/1/gphoto/lib/-lpopt" ...
5. Nous compilons libgphoto2-2.2.1.tar.gz.
6. Finalement nous compilons gphoto2-2.2.0.tar.gz.
7. En l'état, gphoto2 ne fonctionne pas sur la carte FOX, car gphoto2 recherche par défaut les librairies définissant les ports par lesquels il peut communiquer ainsi que les protocoles supportés dans le même répertoire que l'emplacement des librairies lors de la compilation (dans notre cas /usr/local/cris/lib). Nous allons donc recompiler une nouvelle image pour la carte FOX tel que décrit à http://www.acmesystems.it/?id=701 (par ./install && ./configure && make). Cependant, nous allons modifier l'image placée dans devboard-R2_01/target/cris-axis-linux-gnu en ajoutant un lien symbolique de /usr/local/cris vers /mnt/1/gphoto afin que gphoto2 trouve les librairies dans le répertoire escompté. En effet, l'exécution de gphoto2 en l'état avec l'option debug nous montre que :

```
[root@axis /mnt/1/gphoto/bin]222# ./gphoto2 --capture-image
0.006519 main(2): gphoto2 2.2.0
0.011867 main(2): gphoto2 has been compiled with the following
options:
0.017128 main(2):  + cris-axis-linux-gnu-gcc (C compiler used)
0.022528 main(2):  + popt (for handling command-line
parameters)
0.027708 main(2):  + no exif (for displaying EXIF information)
[...]
0.044238 main(2): libgphoto2 2.2.1
0.045132 main(2): libgphoto2 has been compiled with the
following options:
0.045845 main(2):  + cris-axis-linux-gnu-gcc (C compiler used)
0.046609 main(2):  + no EXIF (for special handling of EXIF
files)
0.047389 main(2):  + no /proc/meminfo (adapts cache size to
memory available)
0.048337 main(2): libgphoto2_port 0.6.1
0.049219 main(2): libgphoto2_port has been compiled with the
following options:
0.050023 main(2):  + cris-axis-linux-gnu-gcc (C compiler used)
0.050716 main(2):  + USB (libusb, for USB cameras)
0.051555 main(2):  + serial (for serial cameras)
[...]
0.069425 gphoto2-port-info-list(2): Using ltdl to load io-
drivers from '/usr/loc
al/cris/lib/libgphoto2_port/0.6.1'...
0.075444 gphoto2-port-info-list(2): Called for filename '/usr/
local/cris/lib/lib
gphoto2_port/0.6.1/ptpip'.
[...]
```

L'image résultante est réassemblée par make fimage suivi de boot_linux -F exécuté après avoir fermé le *jumper* JP8 et effectué un *reset* (ou une remise sous tension) pour flasher la nouvelle image en mémoire non volatile.

8. Pour des raisons que nous ne saurions expliquer, le port disk crée une erreur de segmentation lors de la recherche des périphériques par gphoto2. Nous effaçons donc toute référence à ce port : rm -rf /mnt/1/gphoto/lib/gphoto2_port/0.6.1/disk* (il reste alors les ports ptpip, serial pour le RS232 et usb qui nous intéresse ici).
9. Finalement la commande gphoto2 --list-ports identifie les ports accessibles pour communiquer avec un APN :

```
Devices found: 7
Path                    Description
-----------------------------------------------
ptpip:                  PTP/IP Connection
serial:/dev/ttyS0       Serial Port 0
serial:/dev/ttyS2       Serial Port 2
serial:/dev/ttyS3       Serial Port 3
usb:                    Universal Serial Bus
usb:001,007             Universal Serial Bus
usb:001,008             Universal Serial Bus
```

La présence des ports usb et des deux périphériques associés (APN et clé USB) nous garantit le bon fonctionnement de l'application.

Une archive des programmes et librairies compilés et prêts à être placés sur une clé USB est disponible à http://jmfriedt.free.fr.

À ce point, les commandes classiques disponibles avec gphoto2 sous PC sont accessibles, par exemple gphoto2 --capture-image pour demander à l'APN de prendre une image. Une telle ligne de commande peut par exemple être appelée périodiquement depuis un *crontab* (http://www.acmesystems.it/?id=58).

4. L'APPAREIL PHOTO NUMÉRIQUE AVEC PORT RS232

Nous avons vu que la carte FOX consomme encore près de 250 mA sous 5 V : un accumulateur ou pile alcaline classique ne fournissant qu'une capacité de l'ordre de 3 à 10 A.h, l'autonomie de ce montage serait au mieux d'une demi-journée à deux jours, sans grand intérêt pour un montage devant être capable de fonctionner en l'absence d'alimentation externe. Avant l'apparition des APN avec ports USB, une large gamme d'appareils photo avec des résolutions honorables (de l'ordre du Mpixel) ont été produits avec un port série. L'application en ligne de commande pour transmettre des informations à certains de ces appareils est photopc.

Nous avons ici réalisé nos expériences avec un Olympus C860L connecté par un câble série réalisé suivant la description disponible à http://www.camerahacker.com/Olympus_C-2500L/serial_pin-out.html.

Cet appareil se démonte par ailleurs facilement pour accéder à un *switch* sous le capot mobile en face avant, dont la fermeture active la mise en marche de l'appareil : nous verrons plus bas comment commander un tel interrupteur de façon logicielle (section 5).

TABLEAU 4

```
/* Minimal protocol for taking pictures
   with the Olympus Camedia C860L
   Initial RS232 communication baud rate
   MUST be 19200 (init + set speed) */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
/* declaration of bzero() */
#include <string.h>
#include <fcntl.h>
#include <termios.h>

#define BAUDRATE B19200
#define RS_DEVICE "/dev/ttyS1"
#define DEBUG printf

struct termios oldtio,newtio;
extern struct termios oldtio,newtio;

int init_rs232()
{int fd;
 fd=open(RS_DEVICE, O_RDWR | O_NOCTTY );
 if (fd <0) {perror(RS_DEVICE); exit(-1); }
 /* save current serial port settings */
 tcgetattr(fd,&oldtio);
 /* clear struct for new port settings */
 bzero(&newtio, sizeof(newtio));
/* _no_ CRTSCTS */
 newtio.c_cflag = BAUDRATE | CS8 | CLOCAL | CREAD;
 newtio.c_iflag = IGNPAR;
 newtio.c_oflag = ONOCR;
 /* inter-character timer unused */
 newtio.c_cc[VTIME] = 0;
 /* blocking read until 1 character arrives */
 newtio.c_cc[VMIN] = 1;
 tcflush(fd, TCIFLUSH);tcsetattr(fd,TCSANOW,&newtio);
 return(fd);
}

void free_rs232(int fd)
/* restore the old port settings */
{tcsetattr(fd,TCSANOW,&oldtio);close(fd);}

int main(int argc,char **argv)
{int fd,k;char buf[255],
    buf1[255]={0x1B,0x53,0x06,0x00,0x00,
      0x11,0x02,0x00,0x00,0x00,0x13,0x00},
            // cmd17=setspeed, arg=2=19200 bauds
    buf2[255]={0x1b,0x43,0x03,0x00,0x02,
      0x02,0x00,0x04,0x00};
            // cmd02=snaphshot
 fd=init_rs232();
 // init comm
 buf[0]=0x00;write(fd,buf,1);DEBUG("cmd(%x)",(buf[0]&0xff));
 // answers 0x15
 read(fd,buf,1); DEBUG("-> %x\n",(buf[0]&0xff));
 // baud rate=19200
 write(fd,buf1,12);DEBUG("cmd(%x)",(buf1[5]&0xff));
 // ans: 0x06 (ACK)
 read(fd,buf,1); DEBUG("-> %x\n",(buf[0]&0xff));
// for (k=1;k<3;k++) {
   // take pict
   write(fd,buf2,9);DEBUG("snapshot(%x)",(buf2[5]&0xff));
   // answers 0x06 (ACK)
   read(fd,buf,1); DEBUG("-> %x\n",(buf[0]&0xff));
   // ans: 0x05 (done)
   read(fd,buf,1); DEBUG("-> %x\n",(buf[0]&0xff));
//   printf("Waiting 5 seconds ... ");sleep(5);
   printf("done\n"); // }
 free_rs232();
 return(0);
}
```

Programme de contrôle d'un appareil photo numérique muni d'un port RS232 depuis GNU/Linux, fonctionnel sous uClinux.

L'auteur de photopc a non seulement fourni un mode verbeux dans lequel le logiciel informe l'utilisateur de toutes les transactions sur le port RS232, mais a aussi rédigé une excellente description du protocole de communication (http://photopc.sourceforge.net/protocol.html) que nous pouvons alors implémenter sur tout microcontrôleur faible consommation muni d'un port série.

Cette solution a dans un premier temps été implémentée sous GNU/Linux telle que présenté dans le tableau 4, et

(2) http://www.arcturusnetworks.com/coldfire5272.shtml

sa fonctionnalité a été validée sous uClinux lors d'une utilisation sur la carte uCdimm5272 [6] à base de Coldfire 5272 commercialisée par Arcturus Networks (2), puis dans un microcontrôleur compatible 8051 (code dans le tableau 5).

La seule subtilité dans cette dernière implémentation du code est le passage du cœur du processeur à une cadence élevée de 16 MHz afin de pouvoir générer les signaux de communication à 19200 bauds.

TABLEAU 5

```
        .area code (ABS)
        .org 0h0000


RSTirq: ljmp    MAIN ; 3 bytes
IE0irq: nop nop nop nop nop nop nop nop  ; commence en 03
TF0irq: nop nop nop nop nop nop nop nop  ; commence en 0B
IE1irq: nop nop nop nop nop nop nop nop
TF1irq: nop nop nop nop nop nop nop nop
TIirq:  nop nop nop nop nop nop nop nop  ; RI+TI interrupt
TF2irq: nop nop nop nop nop nop nop nop
ADCirq: nop nop nop nop nop nop nop nop
ISPirq: nop nop nop nop nop nop nop nop
PSMirq: nop nop nop nop nop nop nop nop
TIIirq: nop nop nop nop nop nop nop nop
WDSirq: nop nop nop nop nop nop nop nop nop nop nop nop nop


MAIN:
    MOV     0hd7,#0h00  ; 16.77 MHz core clock for 19200
                ; baud rate but higher current consumption

    MOV     RCAP2H,#0hFF    ; config UART for 19200 baud
                            ; (16.77 MHz core)
    MOV     RCAP2L,#-27     ; 19200
    MOV     TH2,#0hFF
    MOV     TL2,#-27
    MOV     SCON,#0h52
    MOV     T2CON,#0h34



;----------------- Camedia Init ---------------------------
init_cam:
  mov   A,#0
  lcall SENDCHAR    ; init comm
  lcall GETCHAR     ; answers 0h15
  mov   dptr,#cmd17
  mov   r0,#12
  lcall snd_str     ; baud rate=19200
  lcall GETCHAR     ; ans: 0h06 (ACK)
  ret


noflash:
    mov dptr,#cmd_noflash
    mov r0,#12
    lcall   snd_str
    lcall   GETCHAR
    ret
camedia:
    mov dptr,#cmd02
    mov r0,#9
    lcall   snd_str     ; take pict
    ;lcall  GETCHAR     ; answer 0h06 (ACK)
    ;lcall  GETCHAR     ; ans: 0h05 (done)
    ret
;----------------- END OF Camedia Init ------------------------------



;_____________________________________________________________________
SENDCHAR:       ; sends ASCII value contained in A to UART
        JNB     TI,SENDCHAR     ; wait til present char gone
        CLR     TI              ; must clear TI
        MOV     SBUF,A
        RET

;_____________________________________________________________________
GETCHAR:        ; waits for a single ASCII character to be received
                ; by the UART.  places this character into A.
iciget: JNB     RI,iciget
        MOV     A,SBUF
        CLR     RI
        RET

;_____________________________________________________________________
; dptr contains the starting point of the string to be sent
; R0 contains the length of the string, will be zero upon end of func.
snd_str:
    clr A
    movc    A,@A+dptr
    lcall   SENDCHAR
    inc     dptr
    djnz    r0,snd_str
    ret


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Minimal protocol for taking pictures with the Olympus Camedia C860L
; Initial RS232 communication baud rate MUST be 19200 (init + set speed)
                                ;/ cmd17=setspeed, arg=2=19200 bauds
cmd17: .db 0h1B,0h53,0h06,0h00,0h00,0h11,0h02,0h00,0h00,0h00,0h13,0h00
                                ;/ cmd02=snaphshot
cmd02: .db 0h1b,0h43,0h03,0h00,0h02,0h02,0h00,0h04,0h00
cmd_noflash:
       .db 0h1b,0h43,0h06,0h00,0h00,0h07,0h02,0h00,0h00,0h00,0h09,0h00
; command is always 1B 43 (except 53 for first packet in session)
; followed by 2 bytes=length and ending with checksum (2 bytes)
; http://photopc.sourceforge.net/protocol.html
```

Programme de contrôle d'un appareil photo numérique muni d'un port RS232 depuis un microcontrôleur de type ADuC814 compatible 8051. Les séries d'octets utilisées pour l'initialisation et le déclenchement d'une prise de vue sont visibles en fin de code (labels `cmd17`, `cmd02` *et* `cmd_noflash`*). La transaction entre le microcontrôleur et l'APN est bidirectionnelle : l'appareil acquitte toute commande émise par le microcontrôleur.*

5. SIMULATION DE L'APPUI DE TOUCHES SUR APPAREIL PHOTO NUMÉRIQUE

Les appareils munis d'un port série étant malheureusement obsolètes, nous désirons conserver une solution utilisant ce microcontrôleur faible consommation, mais compatible avec les APN les plus récents. La solution consiste alors, en l'absence de maître USB, de simuler l'appui de touches. Il faut pour cela démonter l'APN afin de souder des fils sur les divers interrupteurs des touches : *cette opération annulera bien entendu toute garantie sur l'appareil démonté.*

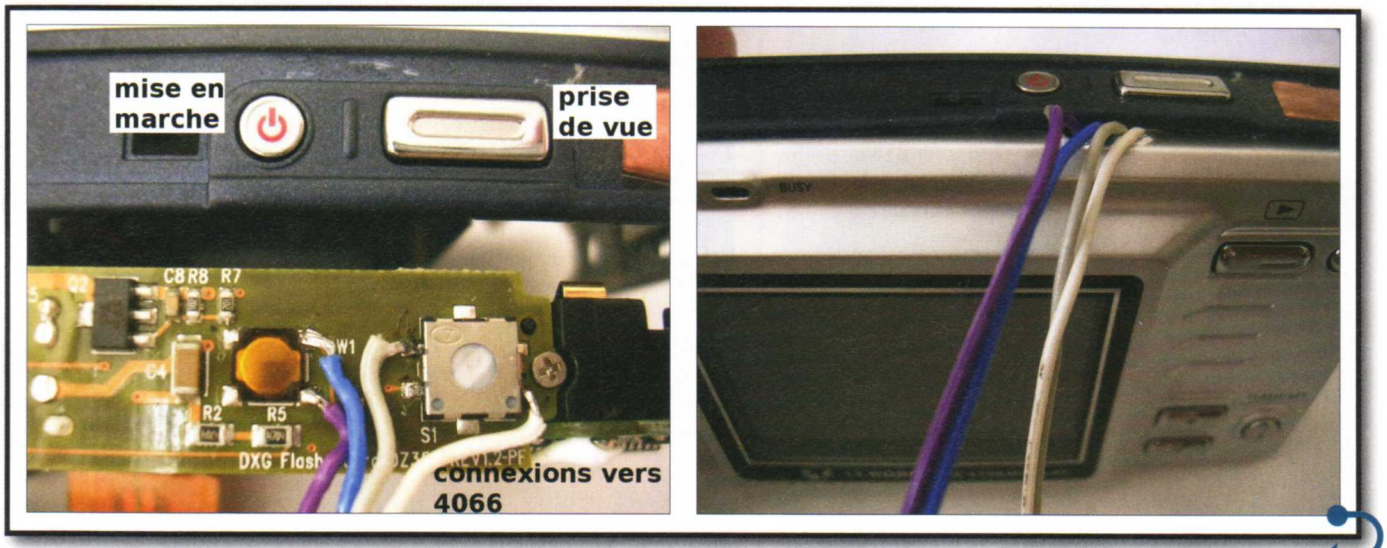
Le composant que nous utiliserons pour simuler l'appui d'une touche – qui se traduit par la fermeture d'un contact électrique – est le relais analogique CMOS 4066. Ce composant, alimenté par une tension entre 3 et 15 V, fournit des paires de pattes dont la connexion entre elles est commandée numériquement. Un 4066 fournit quatre switchs et permet donc de simuler l'appui de 4 touches, par exemple les boutons de mise en marche et de prise de vue.

générale être adapté à tout instrument contrôlé par un interrupteur alimenté sous une tension faible (<15 V) afin d'automatiser la mise en marche d'instruments prévus pour un opérateur humain (tel que nous l'avons validé sur une micropipette motorisée dans le cadre de l'automatisation d'une expérience dangereuse pour un manipulateur humain). (Voir tableau 5, page précédente.)

Nous verrons plus bas (label `fin` dans le tableau 9) que la simulation d'appui d'une touche se résume alors au passage d'un signal numérique issu du microcontrôleur, et connecté à une broche de commande du 4066, du niveau bas au niveau haut (afin de court-circuiter les deux broches associées à cette commande), attendre un délai compatible avec celui qu'attendrait un opérateur (fonction `delai` dans le tableau 8), et finalement ouvrir l'interrupteur du 4066 en repassant le signal de commande au niveau bas.

6. LE CONTRÔLE DU TEMPS

Dans toutes les solutions citées précédemment, l'unité de calcul est continuellement en fonctionnement et consomme inutilement de l'énergie en attendant d'atteindre



Connexion sur les interrupteurs de mise en marche (gauche) et de prise de vue (droite) d'un APN – ici un IT Works DSC551 – en vue d'une commande numérique par un 4066. Éviter de toucher le condensateur du flash lors de ces opérations de soudure afin de ne pas subir une décharge électrique désagréable. **Fig. 4**

Nous avons utilisé pour nos expériences les APN les plus simples (sans partie mobile afin de mieux résister aux environnements climatiques extrêmes) avec une résolution intéressante (5 Mpixels) et les moins chers disponibles sur le marché : l'IT Works DSC551 est disponible pour 89 euros chez Darty (Fig. 4). Le concept a par ailleurs été validé sur un appareil réflexe haut de gamme Canon EOS 20D : dans ce cas, le 4066 court-circuite deux fils issus du déclencheur souple vendu séparément (connecter au 4066 la tresse et la lame externe du déclencheur souple). Le principe de cette expérience peut de façon

la condition de prise de vue. Une façon plus efficace de gérer l'énergie consiste à ne réveiller l'unité de calcul que lorsque le moment de prise de vue est venu, sous réserve que la séquence d'initialisation (*boot*) ne soit pas plus gourmande en énergie que l'attente : cette condition sera toujours vérifiée pour des captures d'images à intervalles de temps réguliers espacés dans une journée.

L'horloge temps-réel Maxim DS1305 (3) fournit un calendrier et une horloge comme tous les autres composants de ce type, mais propose, en plus, deux alarmes associées à deux signaux d'interruption (actifs bas) permettant de

(3) Références 795987 et 9726209 pour respectivement 4,87 et 8,18 euros chez Farnell.
(4) Voir à ce sujet la note d'application SLAA076A de Texas Instruments, focus.ti.com/lit/an/slaa076a/slaa076a.pd
(5) http://www.ntp.org/

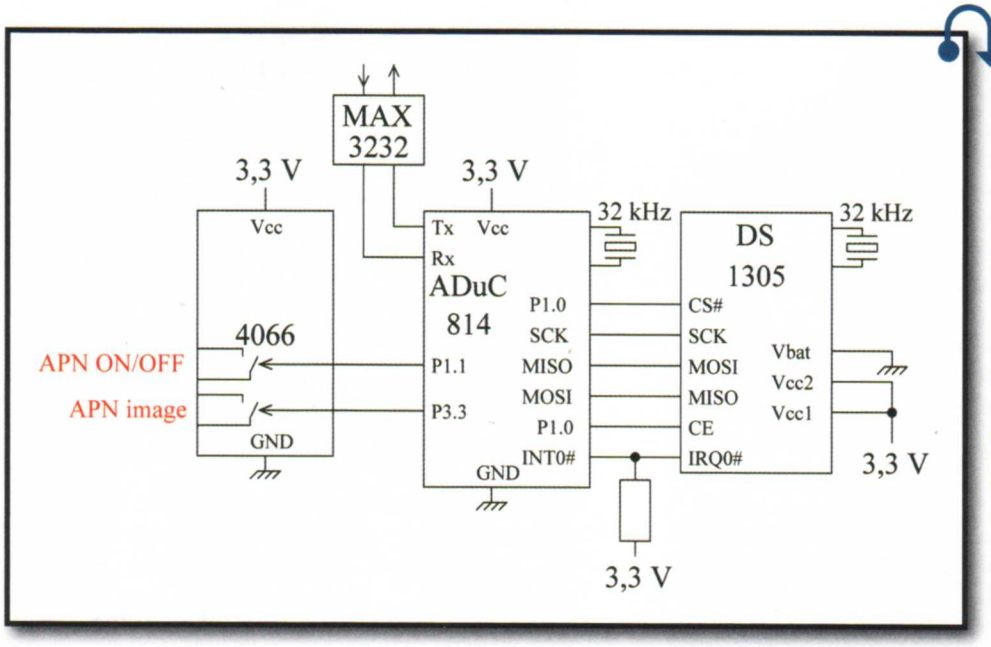


Fig. 5

Schéma de principe du circuit centré sur le microcontrôleur ADuC814 chargé de programmer la RTC par son port SPI et de commander l'allumage et le déclenchement de prise de vue de l'APN par simulation de l'appui des touches appropriées. Le microcontrôleur est la plupart du temps en mode veille pour être réveillé aux moments appropriés par l'horloge temps-réel qui déclenche une interruption matérielle lorsqu'un de ses registres d'alarme est égal à l'heure courante.

réveiller un périphérique lorsqu'une condition d'égalité entre un registre d'alarme et l'heure courante est vérifiée. De plus, une horloge temps réel permet d'aisément générer des signaux à des intervalles de temps longs (qui peuvent se compter en heures ou en jours) difficiles à générer avec les compteurs internes d'un microprocesseur (4), et ce, avec une précision égale à celle du diapason à quartz sur lequel l'oscillateur de l'horloge est asservi.

Notre stratégie sera donc la suivante (Fig. 5) : à l'allumage, un microcontrôleur reçoit de l'utilisateur les paramètres de programmation de l'horloge temps réel (*Real Time Clock* : RTC) et initialise l'horloge avec ces données. La mise en mode sommeil du microcontrôleur réduit alors considérablement sa consommation (dans son mode de sommeil le plus profond, l'oscillateur est arrêté et la consommation associée aux transitions d'état dans le processeur est alors à son minimum, avec seulement la rétention des informations dans les registres et la RAM). La génération d'une impulsion en sortie d'alarme de la RTC déclenche une interruption matérielle du processeur qui se réveille alors pour exécuter la fonction suivant l'instruction de mise en veille – fonction de prise de vue et de réinitialisation de la RTC avec la programmation de l'alarme pour le réveil suivant – avant de se rendormir.

Afin d'atteindre des autonomies de l'ordre de l'année sur une pile, la consommation du circuit doit être réduite à son strict minimum en sélectionnant une RTC de faible consommation (typiquement inférieure au mA), prendre soin de réduire le rapport cyclique réveil/sommeil du processeur et de réduire sa consommation lors des phases d'inactivité, éviter les diodes et autres résistances de tirage de valeur trop faible qui induisent des courants de fuite inutiles, et finalement désactiver les composants inutiles. Dans cette dernière classe d'optimisation, nous avons par exemple découvert qu'un convertisseur de niveaux logiques MAX3232 consomme 5 mA même inactif : nous avons donc pris soin d'ajouter un jumper sur son alimentation pour le déconnecter après la phase de programmation du microcontrôleur et d'initialisation de la RTC. Une solution plus élégante est l'utilisation du MAX3222 qui possède une broche de désactivation.

Nous proposons dans cet exemple d'illustrer l'utilisation du microcontrôleur ADuC814 pour simuler l'appui des touches de mise en marche et de prise de vue d'un APN, sa mise en mode faible consommation et son réveil sur le déclenchement d'une interruption externe. Ce microcontrôleur est de plus équipé de ports RS232 et SPI pour la réception des consignes de prises de vues fournies par l'utilisateur et la communication avec la RTC respectivement.

Une utilisation efficace pour des prises de vues synchrones par plusieurs montages commandés par des RTC distinctes nécessite de sélectionner un algorithme de synchronisation. Nous proposons d'initialiser toutes les RTC avec l'heure de l'ordinateur servant à programmer les paramètres dans les circuits situés en des points distincts géographiquement (l'horloge de plusieurs ordinateurs pouvant être synchronisée par `ntp` (5) ou GPS). Chaque jour, les RTC ont pour consigne de se réveiller à une heure prédéterminée pour amorcer la séquence de prises de vues qui se poursuit à intervalles de temps réguliers dans la journée pour s'achever après un nombre prédéterminé d'images prises. Ainsi, nous garantissons que chaque jour tous les appareils se réveillent à la même heure, et ne prennent que des photos lors de conditions d'illumination pertinentes en fonction des réglages des instruments de prise de vue (uniquement de jour ou uniquement de nuit par exemple).

Lors de la mise sous tension, le microcontrôleur s'initialise par la routine visible dans le tableau 6. Nous allons utiliser l'interruption matérielle INT0# afin de réveiller le microcontrôleur de son sommeil (faible consommation électrique) dans lequel il se place entre deux prises de vues : l'emplacement à l'adresse de gestion de cette interruption (ISR en 0x03) contient donc simplement un retour d'interruption (`RTI`) qui, au déclenchement de l'interruption, permettra au programme de continuer séquentiellement son exécution à l'instruction qui suit la mise en veille du microcontrôleur. Nous prenons donc soin d'activer cette interruption (bit EX0 du registre *Interrupt enable* `EX0=1`) ainsi que les interruptions en général (*Global Interrupt Enable* `EA=1`). De plus, le port RS232 est initialisé à 9600 bauds puisque nous l'utiliserons pour recevoir les paramètres d'initialisation de la RTC.

TABLEAU 6

```
       PCON    = 0h87
       battery = P1.1  ; mise en marche de la camera
 onoff = P3.3  ; declenchement de l'APN
       RTCcs   = P1.0

       .area code (ABS)
       .org 0h0000

RSTirq: ljmp    MAIN ; 3 bytes
IE0irq: reti nop nop nop nop nop nop nop    ; commence en 03
TF0irq: nop nop nop nop nop nop nop nop     ; commence en 0B
IE1irq: nop nop nop nop nop nop nop nop
TF1irq: nop nop nop nop nop nop nop nop
TIirq:  nop nop nop nop nop nop nop nop     ; RI+TI interrupt
TF2irq: nop nop nop nop nop nop nop nop
ADCirq: nop nop nop nop nop nop nop nop
ISPirq: nop nop nop nop nop nop nop nop
PSMirq: nop nop nop nop nop nop nop nop
TIIirq: nop nop nop nop nop nop nop nop
WDSirq: nop nop nop nop nop nop nop nop nop nop nop nop nop

MAIN:
;    MOV     0hd7,#0h00  ; 16.77 MHz core clock for 19200
               ; baud rate but higher current consumption

    MOV     RCAP2H,#0hFF    ; config UART for 9600 baud (2 MHz core)
    MOV     RCAP2L,#-7      ; 9600
    MOV     TH2,#0hFF
    MOV     TL2,#-7
    MOV     SCON,#0h52
    MOV     T2CON,#0h34

    SETB    0hAF            ; enable interrupts -- EA=0hAF
    SETB    0hA8            ; enable EX0 interrupts - EX0=0hA8

    mov     0h9c,#0h01  ; cfg814=0h9C
    mov     0hF8,#0h3D  ; SPICON=0hF8

    clr     RTCcs
    clr     battery     ; open relay
    clr     onoff    ; open relay
```

Une fois ces tâches d'initialisation réalisées, nous passons à la phase de configuration de la RTC. Un PC se charge de prendre l'heure de sa propre horloge pour la transférer au circuit : l'heure du PC est obtenue par `time_t t;time(&t);` puis est convertie en un format acceptable par la RTC de la forme année, mois et jour par la fonction `tm=localtime(&t);` qui renvoie un `struct tm *tm;`. Cette structure de données contient alors les informations que nous transférons au microcontrôleur, à savoir `tm->tm_min`, `tm->tm_hour`, `tm->tm_mday`, `tm->tm_mon+1`, `tm->tm_year-100`. Les fonctions de communication par le port SPI (tableau 8) vont nous permettre de transférer ces informations de l'ADuC814 à la RTC, non sans avoir au préalable pris soin de faire passer le signal d'activation de la RTC nommé CE (*Attention!* : Ce signal a la particularité d'être actif au niveau haut) de bas à haut afin d'annoncer le début d'une transaction. Cette opération est développée dans la fonction `fillram` du code présenté dans le tableau 7, code dans lequel nous avons mis sous forme de commentaire les fonctions d'initialisation des mois, jours... minutes afin d'alléger la présentation. Toutes ces fonctions sont en fait identiques à la fonction de définition de l'année courante présentée dans son intégralité, en remplaçant simplement le registre de destination dans la RTC par le registre explicité dans le commentaire.

Nous passons donc comme paramètres à la RTC le triplet d'octets représentant la date, le triplet correspondant à l'heure actuelle et le triplet correspondant à l'horaire de réveil. Suivent l'intervalle de temps entre deux prises (heure et minute), puis le nombre de photos à prendre chaque jour. Ces six derniers octets sont stockés dans un petit espace mémoire (RAM) fourni par la RTC toujours maintenu sous tension. Ces valeurs de référence seront conservées inchangées pour consultation à chaque déclenchement de l'alarme. En effet, rappelons que nous avons choisi de débuter les séquences de prises de vues à une certaine heure prédéterminée chaque jour, puis d'incrémenter d'un intervalle d'heures/minutes un nombre prédéterminé de fois au cours de cette journée. À la fin de la journée, les registres d'alarme et le compteur de prises de vues déjà effectuées doivent être réinitialisés en prévision de la nouvelle journée à venir : c'est le rôle de la fonction `fillinitval` du code présenté dans le tableau 7. Cette fonction cherche les valeurs dans la RAM de la RTC et les transfère dans le registre d'alarme0 de la RTC, et initialise le compteur d'images déjà prises (emplacement mémoire 0xA5), compteur qui est décrémenté à chaque prise de vue jusqu'à atteindre 0, indiquant la fin de la journée.

Initialisation de l'ADuC814 avec la définition d'un vecteur d'interruption associé à l'interruption matérielle servant à réveiller le microcontrôleur par un signal issu de la RTC.

TABLEAU 7

```
;------------------ RTC access routines -----------------------
----
; placer 0x80 dans les champs qui doivent etre executes a chaque
; fois

    MOV P1,#0hFE
    LCALL   fillram ; recoit les params de config de l'utilisateur
            ; 5 chars = heure:min ini, heure:min inc, nb/jour
    setb    RTCcs
    MOV     A,#0h8F ; fill control register : 8F <- 0x05
    LCALL   SENDSPI
    MOV     A,#0h05 ;
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs
    MOV     A,#0h8A ; fill day: take pict every day
    LCALL   SENDSPI
    MOV     A,#0h80 ;
    LCALL   SENDSPI
    clr     RTCcs

    lcall   fillinitval
    setb    RTCcs
    MOV     A,#0h07     ; read alarm0 register to restart
    LCALL   SENDSPI
    LCALL   READSPI

    clr     RTCcs
    clr     battery     ; open relay
    clr     onoff    ; open relay

; SUITE DU PROGRAMME PRINCIPAL ICI : APRES CA ROUTINES DE RTC
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; get from RS232 port the current time/date, alarm start and time
; interval -> RAM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
fillram:    ; we store restart time every day (hour:min) : A0 A1
            ; time interval between pictures (hour:min)  : A2 A3
            ; number of picture taken per day (one char) : A4
    ;;;;;; current year ;;;;;;;;;;;;;;;;;
    setb    RTCcs
    MOV     A,#0h86 ; fill RAM
    LCALL   SENDSPI
    LCALL   GETCHAR
    LCALL   BIN2BCD
    LCALL   SENDSPI ; current year
    clr     RTCcs

;;; IDEM pour registre #0h85 lu sur RS232 = current month
;;; IDEM pour registre #0h84 lu sur RS232 = current day
;;; placer 1 dans registre #0h82 = current day of week
;;; IDEM pour registre #0h82 lu sur RS232 = current hours
;;; IDEM pour registre #0h81 lu sur RS232 = current minutes

    setb    RTCcs ;;;;;;;;;;;;;;; hours ;;;;;;;;;;;;;;;;;
    MOV     A,#0hA0 ; fill RAM
    LCALL   SENDSPI
    LCALL   GETCHAR
    LCALL   SENDSPI ; init time hour
    clr     RTCcs

    setb    RTCcs ;;;;;;;;;;;;;;; minutes ;;;;;;;;;;;;;;;;
    MOV     A,#0hA1 ; fill RAM
    LCALL   SENDSPI
    LCALL   GETCHAR
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs ;;;;;;;;;;;;;;; interval ;;;;;;;;;;;;;;;
    MOV     A,#0hA2 ; inc hour
    LCALL   SENDSPI
    LCALL   GETCHAR
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs ;;;;;;;;;;;;;;; interval ;;;;;;;;;;;;;;;
    MOV     A,#0hA3 ; inc min
    LCALL   SENDSPI
    LCALL   GETCHAR
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs ;;;;;;;;;;;;;;; number/day ;;;;;;;;;;;;
    MOV     A,#0hA4 ; pict/day
    LCALL   SENDSPI
    LCALL   GETCHAR
    LCALL   SENDSPI
    clr     RTCcs
    RET         ; END of fillram

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; set alarm 0 to the initial time and the counter to its
; initial value
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
fillinitval:
    setb    RTCcs
    MOV     A,#0h20 ; read init hour
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs

    LCALL   BIN2BCD
    mov     R0,A

    setb    RTCcs
    MOV     A,#0h89 ; fill hours
    LCALL   SENDSPI
    mov     A,R0
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCCs
    MOV     A,#0h21 ; read init minutes
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs

    LCALL   BIN2BCD
    push    acc
```

```
    setb    RTCcs
    MOV     A,#0h88 ; fill minutes
    LCALL   SENDSPI
    pop acc
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs
    MOV     A,#0h87 ; fill seconds
    LCALL   SENDSPI
    mov     A,#00
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs
    MOV     A,#0h24 ; read init number
    LCALL   SENDSPI
    LCALL   READSPI
    push    acc
    clr     RTCcs
    setb    RTCcs
    MOV     A,#0hA5 ; fill number
    LCALL   SENDSPI
    pop acc
    LCALL   SENDSPI
    clr     RTCcs
    ret     ; fillinitval
```

Communication par port SPI entre l'ADuC814 et la RTC DS1305 lors de son initialisation : transfert de la date et heure actuelles et des consignes de réveil.

TABLEAU 8

```
;_____________________________________________________
SENDCHAR:       ; sends ASCII value contained in A to UART
       JNB      TI,SENDCHAR     ; wait til present char gone
       CLR      TI              ; must clear TI
       MOV      SBUF,A
       RET
;_____________________________________________________
SENDSPI:    ; sends the content of A to the SPI port
    MOV     0hF7,A          ; trigger data transfer:
SPIDAT=0hF7
icispi: JNB 0hFF,icispi
    CLR 0hFF        ; clear ISPI
    RET
;_____________________________________________________
READSPI:    ; sends the content of A to the SPI port
    MOV A,#0hff         ; generate clocks for reception
    LCALL   SENDSPI

    mov     A,0hF7      ; MOV A,SPIDAT: read resulting value
    CJNE    A,#0hff,readend ; we only continue if the MMC
                        ; answers '0x01'
    SJMP    readspi
readend:RET
;_____________________________________________________
GETCHAR:        ; waits for a single ASCII character to be
                ; received
```

```
                ; by the UART.  places this character into A.
icigèt: JNB     RI,iciget
        MOV     A,SBUF
        CLR     RI
        RET
;_____________________________________________________
BIN2BCD:
        push    b                       ; Save B
        mov     b,#10                   ; Divide By 10
        div     ab                      ; Do Divide
        swap    a                       ; Move Result To High Of A
        orl     a,b                     ; OR In Remainder
        pop     b                       ; Recover B
        ret                             ; Return To Caller
;_____________________________________________________
BCD2BIN:
        push    b                       ; Save B
        push    acc                     ; Save Value
        anl     a,#0hf0                 ; Keep High Bits
        swap    a                       ; Get To Low
        mov     b,#10                   ; 10 Decimal
        mul     ab                      ; Multiply
        mov     b,a                     ; Store In B
        pop     acc                     ; Recover BCD Value
        anl     a,#0h0f                 ; Keep Low Nybble
        add     a,b                     ; Add In High
        pop     b                       ; Recover B
        ret                             ; Return To Caller
;_____________________________________________________
delai:  ; si R1=#60 alors on attend 17 secondes
DLY2:   MOV     R2,#0hff        ; Set up delay loop0
DLY1:   MOV     R3,#0hff        ; Set up delay loop1
dly:    DJNZ    R3,dly          ; Dec R3 & Jump here until R3
is 0
        DJNZ    R2,DLY1         ; Dec R2 & Jump DLY1 until R2
is 0
        DJNZ    R1,DLY2
        RET                     ; Return from subroutine
```

Quelques routines de communication par protocoles RS232 et SPI nécessaires pour l'échange d'informations avec l'utilisateur et la RTC respectivement.

Finalement, le dernier point qui nous manque est la définition de la nouvelle alarme lorsque le microcontrôleur se réveille, mais que la fin de journée n'est pas encore atteinte. Dans ce cas, il nous faut lire l'horaire courant (heure/minutes), les incrémenter des intervalles de temps définis par l'utilisateur, prendre soin de compenser les dépassements de capacité (heure>24 et surtout minutes>60) et stocker les résultats dans les registres d'alarme0. Ces fonctions sont implémentées dans le code du tableau 9. La seule subtilité de ce code est la conversion de l'information stockée au format BCD dans la RTC vers un format binaire que nous avons choisi d'utiliser pour la gestion de nos opérations arithmétiques, le résultat étant converti de nouveau en format BCD après avoir effectué les additions et compensé les dépassements (fonctions BCD2BIN et BIN2BCD dans le code du tableau 8).

TABLEAU 9

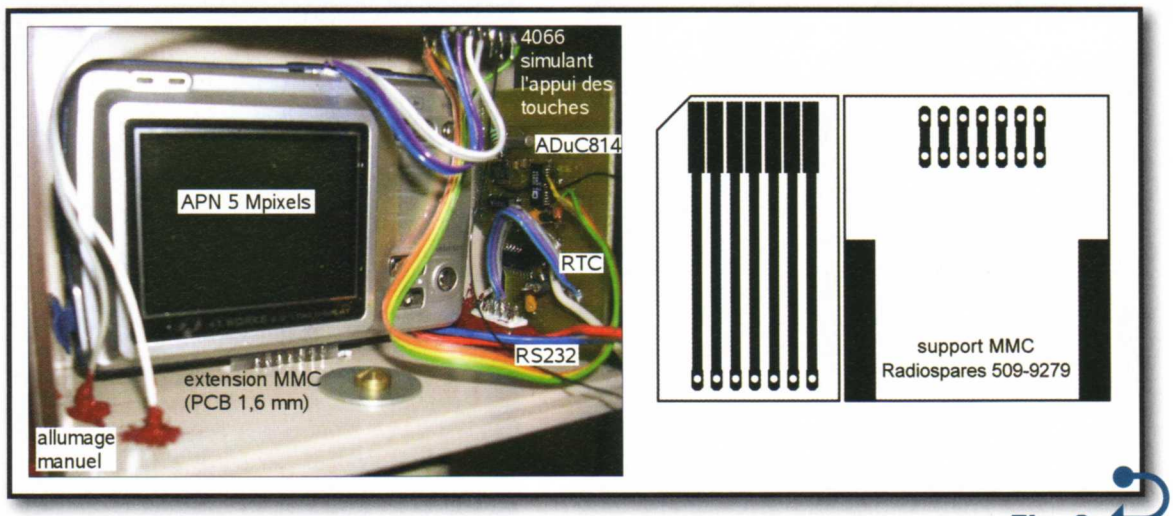
```
fin:
    mov     PCON,#0h23  ; sleep, enable wakeup triggered by INT0
    clr     onoff    ; desactive prise de vue

    LCALL   reinit ; on reinitialise tout de suite le RTC

    setb    battery     ; allume l'APN
    mov     r1,#2
    LCALL   delai       ; duree d'appui sur l'interrupteur de
                        ; mise en route
    clr     battery
    mov     r1,#5
    LCALL   delai       ; attente pour l'ajustement de la
                        ; luminosite
;    LCALL   dumpall

    setb    onoff     ; appui sur la prise de vue
    mov     r1,#2      ; #60=17 seconde
    LCALL   delai       ; duree d'appui sur la prise de vue
    clr     onoff
    mov     r1,#2      ; attente de sauvegarde de la prise de
                        ; vue
    LCALL   delai       ;

    setb    battery     ; appui sur l'interrupteur de mise en
                        ; marche APN
    mov     r1,#3
    LCALL   delai       ;
    clr     battery     ; on a fini d'etteindre l'APN
    SJMP    fin

; lire heure/min actu et ajouter delta heure/min
; compenser pour des min > 60 et des heures > 24
; decrementer compteur et si compteur == 0 alors mettre heure a
; init val
reinit:setb RTCcs
    MOV     A,#0h07 ; read alarm0 register to restart
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs

    setb    RTCcs
    MOV     A,#0h25 ; number of pictures taken so far
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs
    dec     a    ; DECREMENTER ET RESTOCKER
    jnz     continue
    lcall   fillinitval
    ret
continue:       ; non nul donc on n'est pas en fin de journee
    push    acc
    setb    RTCcs
    MOV     A,#0hA5 ; number of pictures still to be taken
    LCALL   SENDSPI
    pop     acc
    LCALL   SENDSPI ; store
    clr     RTCcs

    setb    RTCcs
    MOV     A,#0h22 ; read inc hour
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs
    mov     R0,A

    setb    RTCcs
    MOV     A,#0h02 ; read current hour
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs
    LCALL   BCD2BIN

    ADD     A,R0
    MOV     R0,A
    SUBB    A,#24   ; C is set if A<24
    JNC     finh
    MOV     A,R0    ; restore old value of A since result is <0
finh:LCALL   BIN2BCD
    push    acc
    setb    RTCcs
    MOV     A,#0h89 ; fill alarm hours
    LCALL   SENDSPI
    pop     acc
    LCALL   SENDSPI
    clr     RTCcs

    setb    RTCcs
    MOV     A,#0h23 ; read inc min
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs
    mov     R0,A

    setb    RTCcs
    MOV     A,#0h01 ; read current min
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs
    LCALL   BCD2BIN

    ADD     A,R0
    MOV     R0,A
    SUBB    A,#60   ; C is set if A<60
    JNC     corrigeh
    MOV     A,R0    ; restore old value of A since result is <0
finm:
    LCALL   BIN2BCD
    push    acc
    setb    RTCcs
    MOV     A,#0h88 ; fill alarm min
    LCALL   SENDSPI
    pop     acc
    LCALL   SENDSPI
    clr     RTCcs
    RET

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
corrigeh:
    push    acc
    setb    RTCcs
    MOV     A,#0h09 ; read alarm hour
    LCALL   SENDSPI
    LCALL   READSPI
    clr     RTCcs
```

```
    LCALL   BCD2BIN
    INC A
    mov     R0,A
    SUBB    A,#24   ; C is set if A<24
    JNC     finhc
    MOV     A,R0    ; restore old value of A since result is <0
finhc:
    LCALL   BIN2BCD
    push    acc
    setb    RTCcs
    MOV     A,#0h89 ; fill alarm hours
    LCALL   SENDSPI
    pop     acc
    LCALL   SENDSPI
    clr     RTCcs
    pop     acc
    sjmp    finm
```

Routine définissant la nouvelle alarme après une prise de vue : s'il ne s'agit pas de la dernière photo de la journée, l'horaire courant est incrémenté des intervalles en heures et en minutes prédéfinis pour déterminer les nouvelles valeurs des registres d'alarme, sinon le décompte des prises de vues est réinitialisé et l'horaire de première prise de vue le lendemain est placé dans les registres d'alarme.

L'ADuC814 ne présente que peu d'options pour sélectionner un mode de faible consommation d'énergie : étant donné qu'aucune activité n'est requise du processeur entre deux prises de vues, nous avons activé le mode de consommation dit *power down*. Dans ce mode, toutes les horloges du microcontrôleur sont arrêtées, les sorties numériques et registres internes conservent leurs positions, et nous activons le réveil par déclenchement de l'interruption matérielle INT0#. Ces résultats s'obtiennent en définissant `PCON=0x23` après la prise de vue. Un microcontrôleur dédié aux applications fonctionnant sur batteries tel que le Texas Instruments MSP430 offre beaucoup plus de précision dans la désactivation des périphériques pour finalement atteindre une consommation encore plus faible.

Le résultat de ce travail est un instrument relativement compact – à peine plus grand que l'appareil photo numérique qui sert aux prises de vues – capable de fonctionner en totale autonomie pendant près d'un an sur une pile de capacité 11 A.h. L'APN n'étant allumé que quelques secondes chaque jour, sa consommation moyenne est négligeable devant celle du circuit de contrôle (ADuC814 en mode veille et RTC). Une amélioration possible de ce circuit est de découpler au maximum les sources d'énergie afin de pallier au mieux les défaillances potentielles d'une pile : l'appareil de prises de vues peut être alimenté totalement indépendamment des autres circuits, et nous n'avons pas pris avantage ici de la possibilité d'utiliser une batterie de sauvegarde sur la RTC afin de ne pas perdre les données en cas de coupure de l'alimentation principale. En effet, en cas de dysfonctionnement de la batterie alimentant le microcontrôleur, celui-ci se met directement en attente des commandes sur le port RS232 et interrompt donc la séquence de prises de vues. Il nous faudrait donc encore modifier le contenu de la RAM de la RTC afin d'y mettre un indicateur pour que l'ADuC814 poursuive son activité de déclenchement de l'APN si la RTC a déjà été initialisée. Enfin, découpler les alimentations permettrait éventuellement de placer des accumulateurs sur les alimentations qui ne sont pas stratégiques (l'APN par exemple) afin de les recharger par panneaux solaires, sans mettre en péril le fonctionnement de l'ensemble du montage en cas de décharge des accumulateurs puisque l'électronique de contrôle continue à fonctionner sur piles. Diverses stratégies



L'APN modifié monté dans son boîtier étanche, avec le circuit électronique de contrôle visible sur la droite. Il ne manque que la pile pour fermer la boîte. Notez l'extension de la carte mémoire MMC réalisée en circuit imprimé de 1,6 mm d'épaisseur selon le schéma joint (la largeur du circuit imprimé remplaçant la MMC doit être de 24 mm), qui permet de placer la carte mémoire à l'extérieur du boîtier contenant l'APN et donc de remplacer la carte de stockage des images sans toucher à l'orientation de l'appareil photo. Les deux fils à gauche du boîtier débouchent vers l'extérieur et sont montés en parallèle du 4066 : leur court-circuit avec un objet conducteur permet la mise en marche ou l'extinction de l'APN lors de l'installation pour régler l'orientation en visant grâce à l'image visible sur l'écran LCD.

Fig. 6

de redondance des alimentations sont alors envisageables pour rendre ce montage totalement autonome pour une durée idéalement illimitée, utilisant notamment l'option de recharge de batterie du DS1305.

CONCLUSION

Nous avons abordé diverses méthodes de déclenchement automatique d'instruments permettant la capture d'images, avec des contraintes de consommation électrique décroissantes, mais de modification à l'appareil photo numérique croissante. Nous sommes partis d'un PC équipé d'une carte d'acquisition vidéo pour une capture périodique d'images ou lorsqu'un événement se produit dans le champ de vision, pour ensuite déclencher un APN par les ports prévus à cet effet (USB ou RS232), pour finalement simuler l'appui de touches par des interrupteurs analogiques commandés par microcontrôleur. Cette dernière solution ne consomme que quelques centaines de microwatts et offre le plus de perspectives pour une utilisation à long terme d'un circuit autonome installé en milieu hostile où la présence de l'opérateur n'est qu'épisodique.

REMERCIEMENTS

La carte FOX a été gracieusement prêtée par le laboratoire FEMTO-ST/LPMO de Besançon. L'ensemble des composants Analog Devices (microcontrôleurs ADuC814 et convertisseurs de niveaux RS232 ADM3202 et ADM3222) ont été fournis gratuitement dans le cadre du programme d'offres d'échantillons d'Analog Devices (www.analog.com).

RÉFÉRENCES

[1] Projet Aurore, « Films accélérés : méthodes d'acquisition et traitements », Bulletin de l'Union des Physiciens n°842, mars 2002, p. 543, disponible à http://jmfriedt.free.fr/bup_films.pdf, et un exemple d'animation obtenu en 1997 par Connectix Quickcam noir&blanc qui y est cité à http://friedtj.free.fr/jupitereclipse.mpg

[2] Pilotes pwc pour webcam USB : http://www.saillard.org/linux/pwc/

[3] est disponible à http://www.transcoding.org/cgi-bin/transcode

[4] est disponible à http://www.mplayerhq.hu/design7/news.html

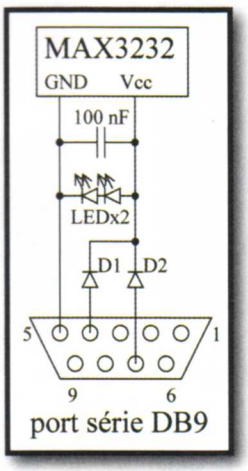
[5] Une façon triviale de passer d'une source de courant à une autre tout en chargeant une batterie, le tout uniquement avec des composants passifs : http://www.cq-vhf.com/Switching.pdf

[6] FRIEDT (J.-M.), GUINOT (S.) & CARRY (E.), « Introduction au Coldfire 5272 », GNU/Linux Magazine France n° 73, juin 2005, pp. 26-33.

J.-M. Friedt <friedtj@free.fr>, É. Carry, <milou@achelem.org>,
Association Projet Aurore, Besançon

BONUS

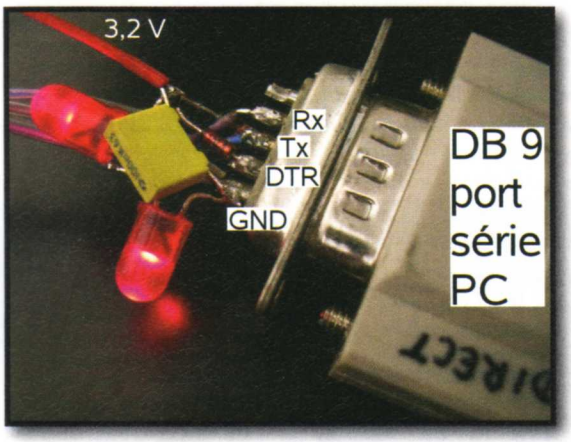
ALIMENTATION PAR LE PORT SÉRIE



Une astuce pour éviter de consommer inutilement de l'énergie en alimentant le convertisseur de tensions RS232 (composant MAX3232) est d'extraire le courant nécessaire du port série du PC. Ainsi ce composant n'est alimenté de l'extérieur que lors de la phase de programmation et ne consomme pas de courant inutilement lors de la phase d'utilisation du microcontrôleur. Nous utiliserons pour cela les deux broches de contrôle de flux du port série, DTR et RTS.

Nous proposons un montage excessivement simple à base de quelques composants passifs. Les rôles de ces composants sont :

- Deux diodes D1 et D2 (D1=D2=1N4148 par exemple) ne sont passantes que lorsque les broches DTR et RTS sont au niveau haut, et bloquantes si une ou les deux broches sont au niveau bas. Nous évitons ainsi tout risque de court-circuit entre les deux broches de contrôle si une erreur de programmation nous faisait élever la tension d'une seule des deux broches.



- La tension de sortie d'un port série varie entre ±5 V et ±12 V. Or notre MAX3232 ne supporte qu'une tension d'alimentation entre 3 et 5 V, et le microcontrôleur alimenté en 3,3 V désire recevoir des signaux autour de sa tension d'alimentation. Nous plaçons donc deux diodes électroluminescentes en sortie de D1 et D2 afin de limiter la tension vue par le MAX3232. En effet, une diode électroluminescente (LED) présente une tension de seuil de l'ordre de 1,5 à 1,6 V. Comme toute diode, elle ne conduit pas le courant sous ce seuil, et devient passante au-delà avec une résistance très faible. Ainsi, tout dépassement de la tension seuil de 2×1,6=3,2 V se traduit par une circulation de courant et donc l'allumage des diodes. La protection en courant de l'UART du PC limite le courant de sortie, ce qui se traduit par une tension alimentant le MAX3232 de l'ordre de 3,2 V. Une protection additionnelle qui ne fait pas appel à la limitation de courant de l'UART serait de placer une résistance en série avec les deux LED. L'alternative aux deux LED en série est l'utilisation d'une diode Zener (seuil à 3,3 V) placée en inverse.

- Finalement, afin de limiter les fluctuations de courant lors de l'utilisation du MAX3232, un petit condensateur (~100 nF) absorbe toute fluctuation brusque de tension.

D'un point de vue logiciel, le passage au niveau haut des broches DTR et RTS du port série s'obtient sur un PC par :

```
#include <sys/io.h> /* for glibc */
int main() {ioperm(0x3f8,6,1); outb(3,0x3FC);return(0);}
```

Abonnez - vous !

11 Numéros de Linux Magazine

1 an de bonne lecture, bien UNIX, bien technique... Bref...

LES 3 BONNES RAISONS DE VOUS ABONNER !

- Ne manquez plus aucun numéro
- Recevez Linux Magazine chaque mois chez vous, ou dans votre entreprise
- Economisez 15,20 €/an ! (soit plus de 2 magazines offerts !)

- Des offres de couplage sont disponibles
- Retrouvez les Tarifs étrangers hors France Métro sur www.ed-diamond.com

BON D'ABONNEMENT À REMPLIR ET À RETOURNER À (OU PHOTOCOPIE)

*DIAMOND Éditions - LINUX MAGAZINE - BP 20142 - 67603 SÉLESTAT Cedex

11 Numéros de Linux Magazine à 53€

Offre France Métro

Soit

Votre Linux Magazine à 4,82€

(Tarif au numéro dans le cadre d'un abonnement France Métro)

Pour les tarifs étrangers, consultez notre site : www.ed-diamond.com

LES 4 FAÇONS DE VOUS ABONNER !

- Par courrier postal en nous renvoyant le bon ci-dessous.
- Par le Web, sur notre site : www.ed-diamond.com.
- Par téléphone (paiement C.B.) entre 9h-12h & 14h -17h au 03 88 58 02 08.
- Par Fax au 03 88 58 02 09 C.B. et/ou bon de commande administratif

❍ Oui, je souhaite m'abonner à Linux Magazine pour 11 numéros

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon réglement :

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions**

☐ Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

☐ Pour avoir un suivi par e-mail de vos abonnements, merci de nous indiquer votre adresse e-mail** :

**En application des articles 27 et 34 de la loi dite «Informatique et libertés» n° 78-17 du 6 janvier 1978, vous disposez d'un droit d'accès

Offres collectionneurs

Les anciens numéros !

TOUJOURS DISPONIBLES !

BON DE COMMANDE

À REMPLIR ET À RETOURNER À (OU PHOTOCOPIE)

*DIAMOND Éditions - LINUX MAGAZINE - BP 20142 - 67603 SÉLESTAT Cedex

Bon de commande Linux Magazine

Référence	Prix / N°s	Qté.	Total
Linux Magazine 77 Systèmes de fichiers chiffrées	6,40 €		
Linux Magazine 78 Bluetooth	6,40 €		
Linux Magazine 79 Sécurisation du noyau avec PAX	5,95 €		
Linux Magazine 80 Run in memory	5,95 €		
Linux Magazine 81 Comment fonctionnent les générateurs	5,95 €		
Linux Magazine 82 eCos, une autre solution libre pour	5,95 €		
Linux Magazine 83 Greylist Eliminez le SPAM à la racine	5,95 €		
Linux Magazine 84 Déploiement de hotspots Wifi sécurisés	5,95 €		
Linux Magazine 85 Firewall: Netfilter & NuFW	6,20 €		
Linux Magazine 86 Serveur SMTP: Routage des mails avec Postfix	6,20 €		

Bon de commande Linux Magazine Hors Série

LM HS 09 Installer son serveur Web à la maison	5,95 €		
LM HS 10 Installation de votre serveur internet	5,95 €		
LM HS 12 Firewall votre meilleur ennemi Acte 1	5,95 €		
LM HS 13 Firewall votre meilleur ennemi Acte 2	5,95 €		
LM HS 15 GIMP et la Photo	5,95 €		
LM HS 16 Kernel (1)	5,95 €		
LM HS 17 Kernel (2)	5,95 €		
LM HS 18 Haute Disponibilité	5,95 €		
LM HS 19 The Gimp 2.0	5,95 €		
LM HS 20 PHP 5	5,95 €		
LM HS 21 Recyclez vos PC	6,40 €		
LM HS 22 GIMP et le Web	6,40 €		
LM HS 23 Linux et electronique	6,40 €		
LM HS 24 Linux Embarqué	6,40 €		
LM HS 25 Linux Embarqué 2	6,40 €		
LM HS 26 Spécial The GIMP	6,40 €		
		TOTAL	
		Frais de port France	+ 3,81 €
		Frais de port Etranger	+ 5,34 €
		TOTAL	

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon réglement :

☐ **Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions****

☐ **Paiement par carte bancaire :**

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

LES 4 FAÇONS DE COMMANDER !

- Par courrier postal en nous renvoyant le bon ci-dessous.
- Par le Web, sur notre site : www.ed-diamond.com.
- Par téléphone (paiement C.B.) entre 9h-12h & 14h -17h au 03 88 58 02 08.
- Par Fax au 03 88 58 02 09 C.B. et/ou bon de commande administratif

Denis Bodor :: db@ed-diamond.com :: lefinnois@lefinnois.net

NOTIFICATION DE MAIL À TUBE NIXIE

Vous vous souvenez peut-être du notificateur de mail sur port parallèle présenté dans GLMF 62 de juin 2004. Il s'agissait, à l'époque, d'indiquer le nombre de messages non lus dans une mbox via un double afficheur numérique à LED connecté au port parallèle. En voici la version tube.

Les afficheurs à LED sont monnaie courante dans beaucoup d'applications allant du distributeur à café au radioréveil en passant par toute une gamme d'équipements industriels. Ils se trouvent remplacés de plus en plus fréquemment et petit à petit par différents autres systèmes d'affichages : écrans LCD alphanumériques, écrans LCD graphiques couleur, afficheurs OLed ou encore véritables écrans LCD.

Mais bien avant la popularisation des technologies LED, un autre type d'afficheur constituait la seule manière de présenter rapidement une valeur numérique directement compréhensible par un humain : le tube Nixie.

Attention ! Cet article met en œuvre des composants utilisant des tensions potentiellement mortelles. Assurez-vous de prendre toutes les mesures nécessaires au travail avec des hautes tensions. Bien que le montage intègre une isolation galvanique, le risque d'électrocution reste présent. Si vous ne vous sentez pas en mesure de travailler avec des hautes tensions, demandez l'assistance ou la supervision d'une personne qualifiée.

Le premier brevet concernant les tubes électroniques (*vacuum tubes*) permettant un affichage numérique date des années 1920. C'est cependant en 1954 que la société Haydu Brothers Laboratories (rachetée par la suite par Burroughs Corporation) fabrique les premiers tubes Nixie (pour, selon la rumeur, « Numeric Indicator eXperimental No. 1 » ou NIX 1). Peu après, ces tubes sont utilisés avec l'un des tout premiers ordinateurs, l'UNIVAC 1101, via un périphérique d'affichage appelé le « Trochotron ». Utiliser ce type de composants sur un ordinateur moderne peut être vu, en quelque sorte, comme un hommage aux machines de la préhistoire informatique.

D'un point de vue purement technologique, un tube Nixie est une ampoule contenant du gaz néon, une anode (+) et plusieurs cathodes (-) en forme de chiffres. Contrairement aux tubes utilisés encore de nos jours pour certains amplificateurs, les tubes Nixie n'ont pas besoin d'être chauffés pour fonctionner. Il s'agit de tubes dits « à cathode froide ». Les chiffres présents dans les tubes ne deviennent pas lumineux, car ils sont chauffés, comme dans une ampoule classique, mais sous l'effet d'une émission thermoïonique. Sans entrer dans le détail, c'est le flux d'électrons de la cathode vers l'anode qui provoque l'émission de lumière dans le gaz néon. Le déplacement d'électrons entre la cathode et l'anode est également connu sous le nom d'effet Edison.

La complexité, l'histoire et l'ingéniosité des tubes Nixie sont autant de choses qui expliquent que leur mise en œuvre est aujourd'hui une étape incontournable pour tout bidouilleur. On trouve ainsi bon nombre de pages Web décrivant la construction d'horloges à tubes Nixie ainsi que des sites spécialisés dans le commerce de ces tubes qui ne sont, à ma connaissance, plus fabriqués aujourd'hui.

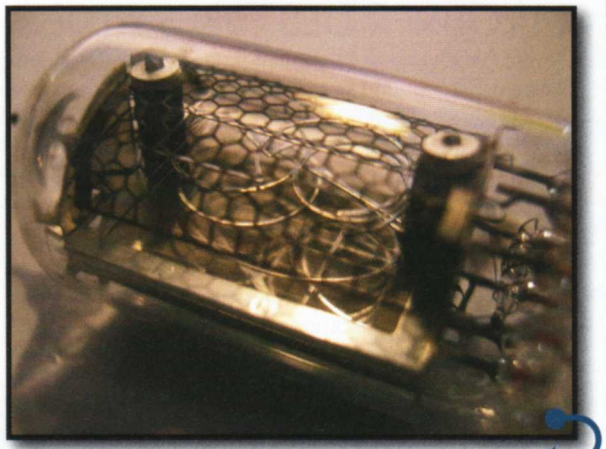
TROUVER DES TUBES NIXIE

Devant l'engouement de certains pour ces tubes, quelques filières de distribution se sont mises en place. Des sociétés spécialisées dans le commerce de vieux composants vendent des tubes de toutes provenances dans leurs boutiques en ligne. Les boutiques en question étant souvent situées à l'étranger, le coût de revient des tubes n'est souvent pas négligeable. Heureusement, il arrive tantôt que des tubes soient vendus sur les sites d'enchères en ligne. Habituellement, les lots se constituent de six pièces, et sont destinés à la fabrication d'horloge. Dans le cadre de cet article, seul un tube sera utilisé, mais rien n'empêche l'extension du montage.

Il existe plusieurs types de tubes. Les premiers essais pour cet article ont été faits avec des tubes Valvo/Dario ZM1032 (Figure 1). Leur mise en œuvre est plus complexe que celle des tubes russes IN-1 finalement choisis. En effet, il faut non seulement alimenter le tube entre l'anode et la cathode, mais également une grille (ou écran) séparant les cathodes paires et impaires. Vous l'aurez compris, ce tube utilise deux anodes. Une seule broche cathodique correspond à deux chiffres. Les différentes tentatives d'utilisation simplifiée de ce tube (alimentation cathode/anode) découlaient sur l'allumage de deux chiffres. Il s'est avéré rapidement que sans alimentation de la grille de

Un tube ZM1032 **Fig. 1**

séparation, aucun résultat probant ne serait obtenu. De plus, d'après une documentation française d'époque, ce tube est destiné à une utilisation en courant alternatif, ce qui exclut l'utilisation de transistors de pilotage et complique grandement la mise en œuvre. Ces tubes ont été écartés dans le cadre du présent article pour ces raisons, mais fonctionnent toutefois très bien.



Détail du ZM1032. On distingue la grille (écran) placée derrière les chiffres impairs au premier plan. **Fig. 2**

Tout ceci vous permet de constater qu'il ne faut pas acheter à l'aveuglette, même si on pense faire une affaire. Payer sur un site de ventes aux enchères quelques 40 euros pour 6 tubes inutilisables (ou trop complexes à mettre en œuvre) n'est pas très intéressant. Pensez à poser des questions et vous documenter sur le modèle de tube que vous convoitez.

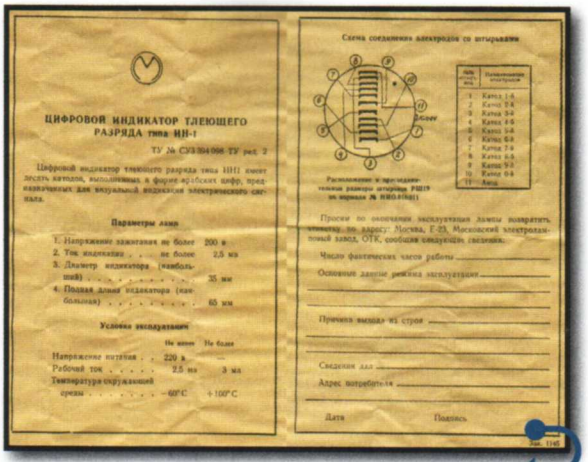
TUBES IN-1

L'objet de cet article concerne l'utilisation d'un tube russe IN-1 relativement facile à mettre en œuvre. Le brochage est constitué d'une anode et de dix cathodes. Les tubes sont relativement volumineux (35 mm de diamètre pour 70 mm de haut), avec une lecture par le haut et une hauteur de chiffre de 18 mm. C'est visible de loin, parfait pour le notificateur de mail à construire.

Un tube Sovtec IN-1 **Fig. 3**

Les caractéristiques importantes du tube sont les suivantes :

- Tension d'alimentation : 170VDC. Il s'agit de la tension (continue) à maintenir entre l'anode et la cathode pour l'allumage d'un chiffre.
- Tension de maintien : 133 VDC. Une fois le chiffre allumé, il n'est plus nécessaire de maintenir une tension de 170 volts. En effet, jusqu'à un minimum de 133 volts, nous pouvons faire baisser la tension sans que le chiffre ne s'éteigne. Ceci ne sera pas utilisé dans le cadre de cet article.
- Courant cathode : 2.5 mA. C'est le courant qui traversera le circuit anode/cathode. Comme vous pouvez le constater, il est très faible. En prenant en compte la tension, on constate ainsi que le tube avec un chiffre allumé ne consomme que 170*0.0025 = 0.42 watt (0.33 watt avec la tension de maintien).



ЦИФРОВОЙ ИНДИКАТОР ТЛЕЮЩЕГО РАЗРЯДА типа ИН-1

Параметры лампы

Условия эксплуатации

Схема соединения электродов со штырьками

L'une des documentations originales en russe disponibles sur le Web **Fig. 4**

ALIMENTATION DU TUBE

Toute la difficulté de mise en œuvre de tubes Nixie réside dans la manipulation des hautes tensions. 170 volts est non seulement une tension mortelle pour un être humain, mais cela représente semble-t-il une difficulté non négligeable au niveau technique. Comment obtenir une telle tension ?

Obtenir une tension de 18, 12 ou 5 volts est un jeu d'enfant. Il suffit de mettre en œuvre un bloc d'alimentation standard et de réguler avec un composant adéquat (type 7805). Avec les tensions de l'ordre de 170 volts, nous serions obligés de construire une alimentation complète à grand recours de transformateurs et/ou de circuits logiques (alimentation à découpage). Bon nombre de pages Web consacrées aux tubes et horloges Nixie proposent des schémas d'alimentation.

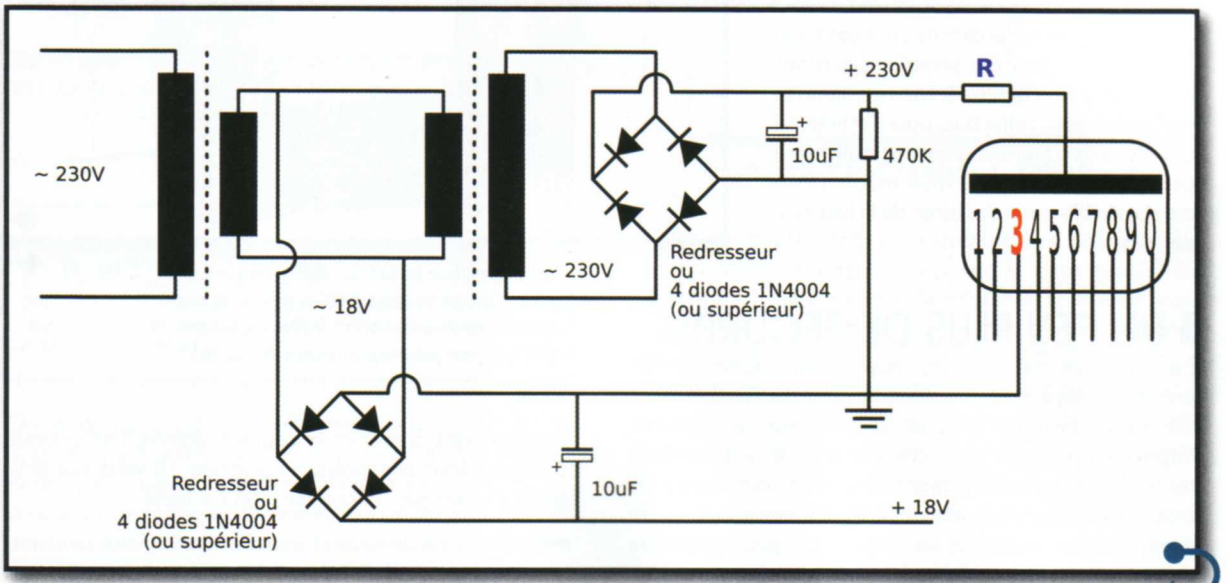
Heureusement, les tubes Nixie possèdent une caractéristique importante : l'utilisation d'un très faible courant. De plus, ici, nous n'avons pas l'intention d'alimenter plus d'un chiffre dans un seul tube. Un grand nombre de paramètres sont donc connus et parfaitement fixes. Inutile de réguler l'alimentation du tube, puisque celle-ci est constante.

Observez le schéma en figure 5 en faisant, pour l'heure, abstraction de toute la moitié gauche. Le groupe de quatre diodes 1N4004 permet de redresser le courant qui est alternatif. En sortie, nous obtenons un courant continu, mais irrégulier. La correction est le travail du condensateur placé en sortie de l'étage de redressage. Notez la présence d'une résistance en parallèle permettant de vider doucement le condensateur après déconnexion. Ceci vous permettra d'éviter une électrocution (brève, mais douloureuse) en cas de contact avec le circuit, même celui-ci débranché.

Arrive ensuite notre tube Nixie connecté à l'anode via.... une simple résistance (notée R dans le schéma). Comment est-ce possible ? Où sont les circuits complexes et transformateurs ? Il ne s'agit pas d'une erreur, mais tout simplement de l'application de la loi d'Ohm (eh oui, encore !).

Une résistance (ou deux) est suffisante pour deux raisons :

- Contrairement à un circuit habituel, nous connaissons précisément les caractéristiques techniques du circuit. Inutile donc de réguler le courant et la tension en fonction du circuit. Rien ne bouge.
- Le courant passant dans le tube Nixie est très faible. Faire chuter la tension ne demande pas à la résistance une grande dissipation en d'énergie.



Schémas de principe de l'alimentation pour un tube Nixie **Fig. 5**

Appliquons la loi d'Ohm :

```
U = R.I

U = 230-170 = 60 V
I = 2.5mA = 0.0025 A

60 = R x 0.0025
60/0.0025 = R
R = 24000 Ohm = 24K Ohm

P = U.I
P = 60 * 0.0025
P = 0.15 Watt
P < 1/4 Watt
```

Le calcul le montre clairement, pour faire chuter la tension de 230 V à 170 V, nous utilisons une résistance de 24 K ohm. Celle-ci devra dissiper quelques 150 milliwatts, ce qui est bien inférieur à ce que permettent la plupart des résistances dites « quart de watt ». Bien entendu, ce calcul n'est valable que pour un tube IN-1 et une tension de départ de 230 VDC.

En principe, nous pouvons donc simplement redresser le courant « secteur », placer une résistance adéquate et alimenter notre tube. En principe, oui, cela fonctionne et dans les faits également. Il faut toutefois prendre en compte quelques éléments :

- La tension « secteur » est normalement de 230 VAC (Alternatif), mais, dans la pratique, ce n'est pas toujours le cas.
- La tension du courant continu redressé n'est pas égale à la tension en courant alternatif (230 VAC redressé = 230 VDC)
- Il y a danger de mort en cas de contact avec l'un ou l'autre pôle, car le montage n'est pas isolé.

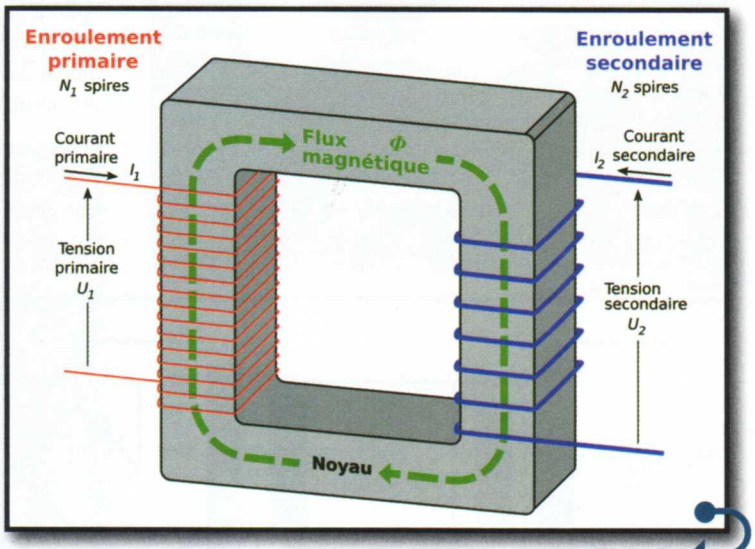
Les deux premiers points ne sont pas aussi critiques qu'il pourrait sembler. Les tubes offrent une certaine tolérance et, dans les nombreux essais effectués pour l'article, j'ai dépassé plus d'une fois la tension d'allumage de 20 ou 30 volts. Un multimètre reste indispensable pour s'assurer de la justesse des calculs.

UN PEU PLUS DE SÉCURITÉ

Le montage est simple, trop simple. Comme dit précédemment, il n'offre aucune sécurité. Ainsi, si vous entrez en contact avec un conducteur, un courant important pourrait vous traverser en direction de la terre. C'est particulièrement vrai pour tout ce qui se trouve avant le redresseur, mais également pour le pôle positif de tout le circuit. Le risque n'est pas négligeable et nous pouvons le réduire de manière importante sans trop de difficultés.

Considérez maintenant la figure 5 dans son ensemble. De gauche à droite, nous avons :

- La connexion au secteur (230 VAC).
- Le premier transformateur délivrant 2 fois 9 volts. Nous avons ici un primaire et deux secondaires isolés. Il nous suffit de relier les deux secondaires en série pour obtenir une tension de sortie de 18 volts alternatifs. Le courant obtenu en sortie du transformateur est isolé du secteur. Le fait de toucher l'un ou l'autre conducteur n'est pas suffisant pour le passage de courant dans le corps. Le courant ne peut circuler que d'un connecteur à l'autre. Le risque se limite ainsi au fait de toucher les deux connecteurs en même temps (attention, cela reste donc mortel !).
- En sortie du premier transformateur, nous en trouvons un second parfaitement identique. Cette fois, le sens primaire/secondaire est inversé. Le transformateur ne fournit plus de 18 VAC à partir du 230 VAC, mais l'inverse. En théorie, avec un transformateur parfait, nous obtenons ainsi la même tension qu'en entrée, mais le courant est isolé de la terre. On parle d'isolation galvanique. Vous pouvez toujours être électrocuté par la tension de sortie, mais il faut toucher les deux connecteurs (je me répète, mais c'est important). En sortie du premier transformateur, se trouve également un redresseur (pont de diodes) transformant là encore le courant alternatif en courant continu. Nous obtenons, en principe 18 volts continus après filtrage par un condensateur adapté.
- En sortie du second transformateur, nous trouvons un redresseur, un condensateur et une résistance en parallèle déjà décrits plus haut dans l'article.



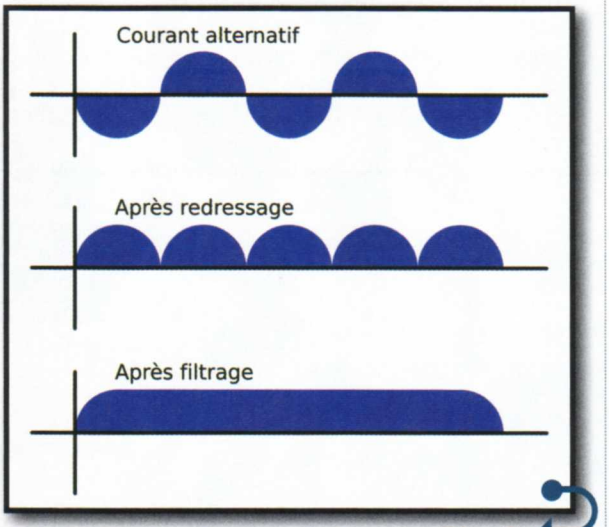
Transformateur parfait ou idéal, sans aucune perte. On voit clairement qu'en dehors de ses qualités de transformation, il offre également une isolation galvanique (source Wikipédia).

Fig. 6

La différence entre ce montage complet et la simple connexion est la sécurité apportée par l'isolation galvanique. Accessoirement, nous obtenons également une tension d'alimentation plus basse.

Nous pouvons ainsi utiliser ces 18 volts pour connecter un régulateur 7812 et un 7805 permettant respectivement d'obtenir des tensions régulées de 12 et 5 volts pour toute la logique de contrôle du tube.

Le rôle des condensateurs est très important dans ce montage. Ils ont pour utilité de nettoyer la tension, car, en sortie des transformateurs et après redressage, celle-ci n'est pas « propre » (Figure 7). La caractéristique principale du condensateur est de s'opposer à tout changement rapide de tension. Plus vous utilisez de condensateurs, plus la tension sera stable. La tension supportée par le condensateur n'est pas importante à partir du moment où elle est supérieure à la tension du montage. Pour davantage de filtrages, placez plusieurs condensateurs par ordre décroissant de capacité (Farad).



Variation de tension après redressage du courant alternatif. Un condensateur est clairement nécessaire. **Fig. 7**

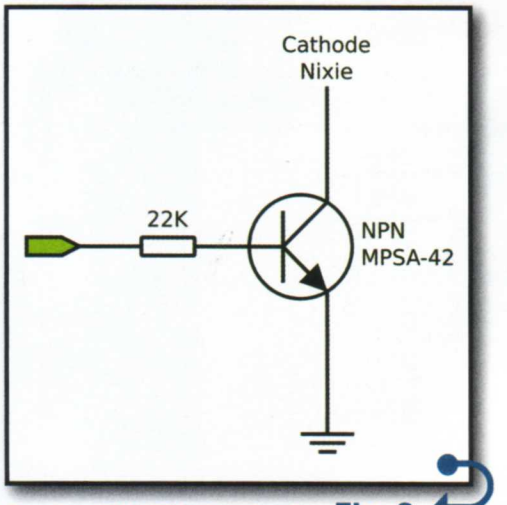
Note : La puissance d'un transformateur s'exprime en VA (voltampère). Un transformateur de 5 VA fournissant 2 fois 9 volts pourra fournir, si les deux secondaires sont reliés en série, 5/(2x9) = 0.278, soit 278 milliampères. Avec des secondaires connectés en parallèle, nous obtenons deux fois plus d'ampères (556 milliampères), mais deux fois moins de tension (9 volts).

On remarquera que les transformateurs ne sont ni précis ni strictement identiques, même pour un même modèle. Ainsi, en fonction de la perte et de divers autres facteurs, la tension de sortie avant redressage ne sera sans doute pas identique à la tension secteur en entrée. Il conviendra donc de tester à l'aide d'un voltmètre la tension réelle en sortie. Cette tension ne peut être mesurée que si le transformateur dispose d'une charge. Si vous mesurez simplement la tension aux bornes de sortie avec votre multimètre, la valeur lue ne sera pas correcte car aucun courant ne traverse le circuit. Placez une résistance de forte valeur (470K ou 1M) en guise de charge et mesurez la tension aux bornes de cette dernière, puis refaites les calculs.

CONTRÔLE DES CHIFFRES

A présent, nous savons comment alimenter le tube avec un minimum de sécurité et disposons par la même occasion d'une source de courant régulée à 5 volts pour alimenter des circuits logiques. Pour pouvoir piloter l'allumage de chaque chiffre, plusieurs solutions existent, de la plus barbare (relais) à la plus subtile et sure (opto-triac) en passant pas la plus simple : l'utilisation de transistors.

Piloter une tension de 170 volts n'est pas à la portée de n'importe quel transistor. Celui-ci doit être en mesure de supporter un Vce (tension entre le collecteur et l'émetteur) très important. C'est du côté des transistors utilisés dans les téléviseurs que nous trouvons notre bonheur avec le MPSA-42. Il est, en effet, capable de supporter un Vce de 300 volts et est relativement courant et économique.

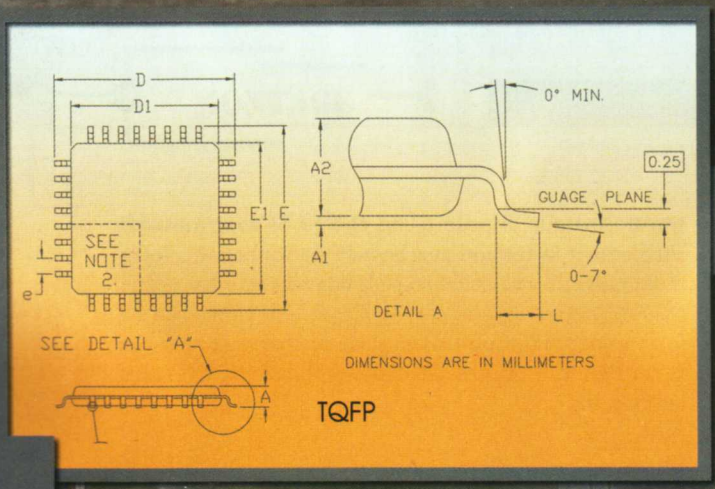


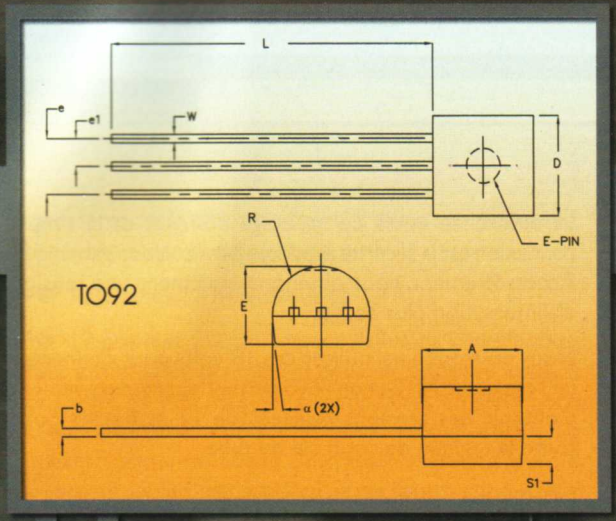
Transistor NPN MPSA-42 en situation **Fig. 8**

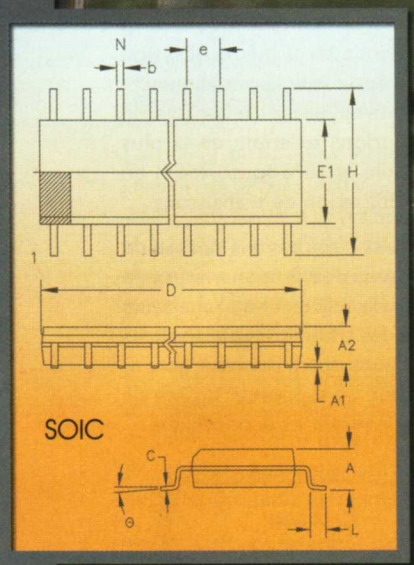
La figure 8 présente le transistor en situation. La résistance limitant le courant sur la base est ici de 22 K ohms, ce qui permet de piloter le transistor avec une tension de 5 volts typique des circuits logiques courants. Reportez-vous au précédent hors-série consacré à l'électronique (numéro 23) pour en savoir plus sur le calcul à opérer. De nombreux sites didactiques expliquent également l'une des bases de connaissance les plus importantes après la loi d'Ohm.

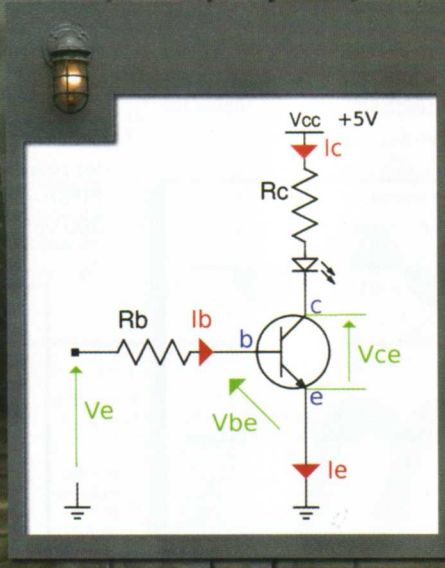
Nous connectons ainsi chaque cathode du tube au collecteur d'un transistor et la masse sur l'émetteur. La présence de 5 volts sur la base via la résistance provoquera la saturation et le passage du courant. Le chiffre s'allume.

Source Maxim Dallas



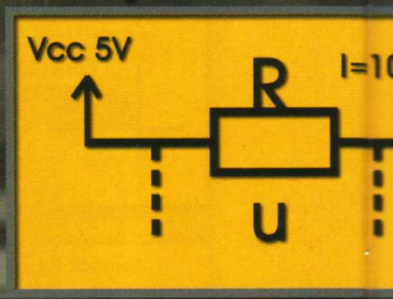


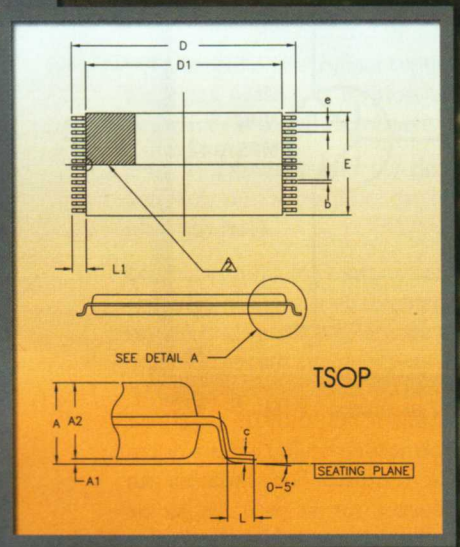


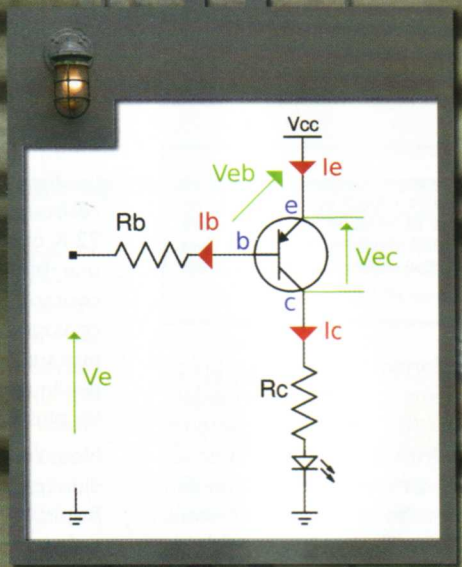


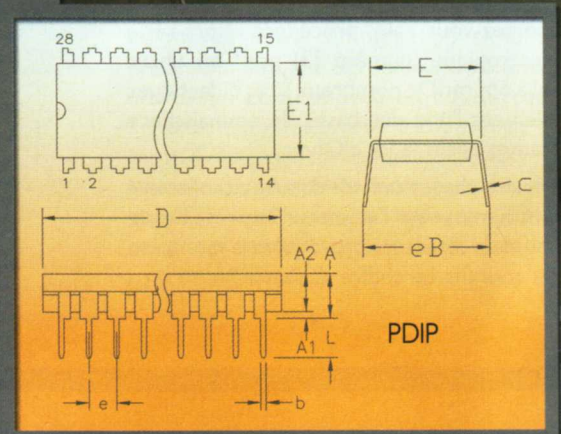
Loi d'O

U=R

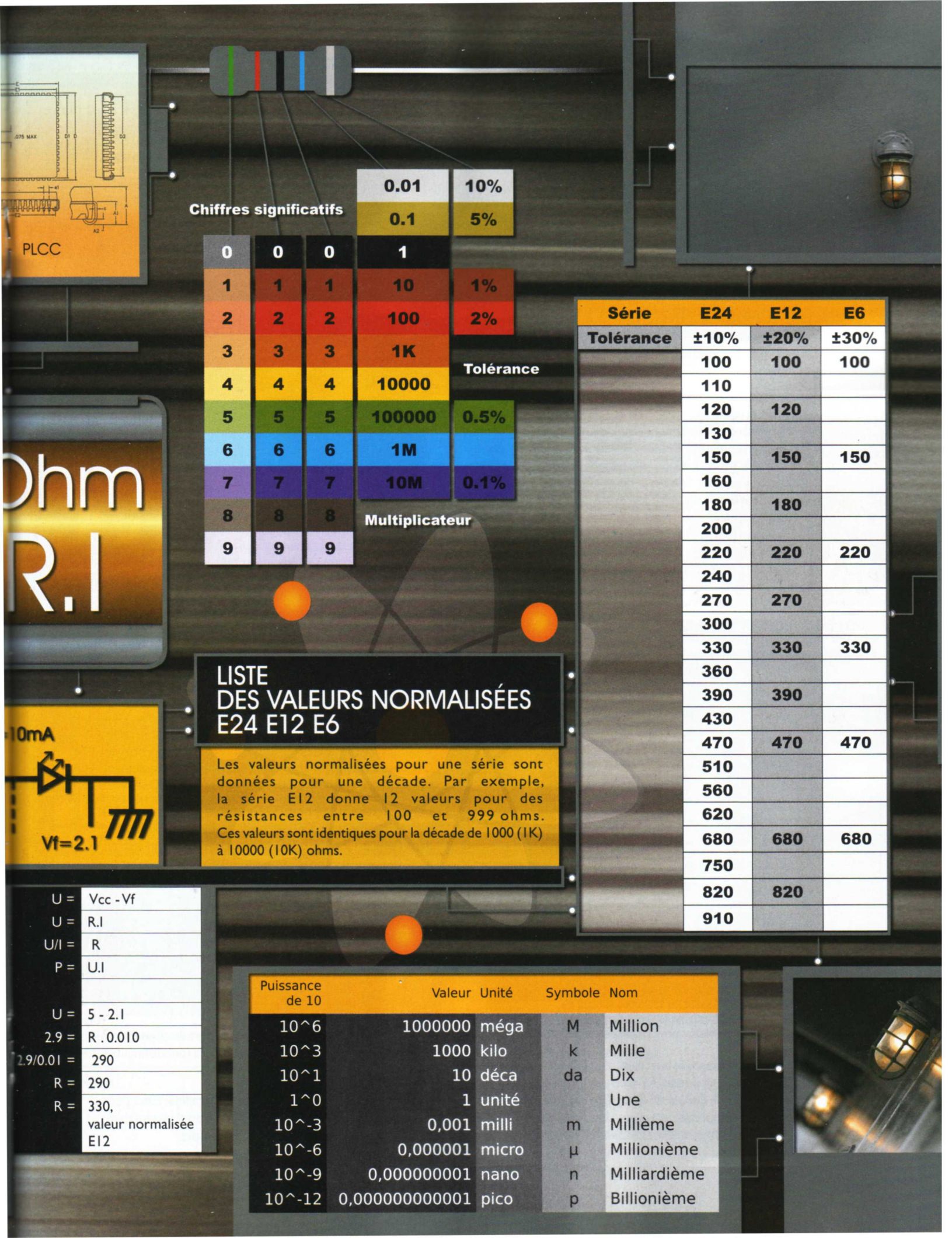








U =	tension aux bornes de la résistance.
Vcc =	tension d'alimentation.
R =	valeur de la résistance en ohms.
I =	courant traversant le circuit en ampères. Donné par la *datasheet* (documentation fabricant) de la LED. Ici 10 mA, soit 0.010 A.
Vf =	tension aux bornes de la LED. Donnée par la datasheet. Ici 2.1 volts.
P =	puissance dissipée en watts par la résistance.



0	0	0	1
1	1	1	10
2	2	2	100
3	3	3	1K
4	4	4	10000
5	5	5	100000
6	6	6	1M
7	7	7	10M
8	8	8	
9	9	9	

Série	E24	E12	E6
Tolérance	±10%	±20%	±30%
	100	100	100
	110		
	120	120	
	130		
	150	150	150
	160		
	180	180	
	200		
	220	220	220
	240		
	270	270	
	300		
	330	330	330
	360		
	390	390	
	430		
	470	470	470
	510		
	560		
	620		
	680	680	680
	750		
	820	820	
	910		

LISTE DES VALEURS NORMALISÉES E24 E12 E6

Les valeurs normalisées pour une série sont données pour une décade. Par exemple, la série E12 donne 12 valeurs pour des résistances entre 100 et 999 ohms. Ces valeurs sont identiques pour la décade de 1000 (1K) à 10000 (10K) ohms.

U =	Vcc - Vf
U =	R.I
U/I =	R
P =	U.I
U =	5 - 2.1
2.9 =	R . 0.010
2.9/0.01 =	290
R =	290
R =	330, valeur normalisée E12

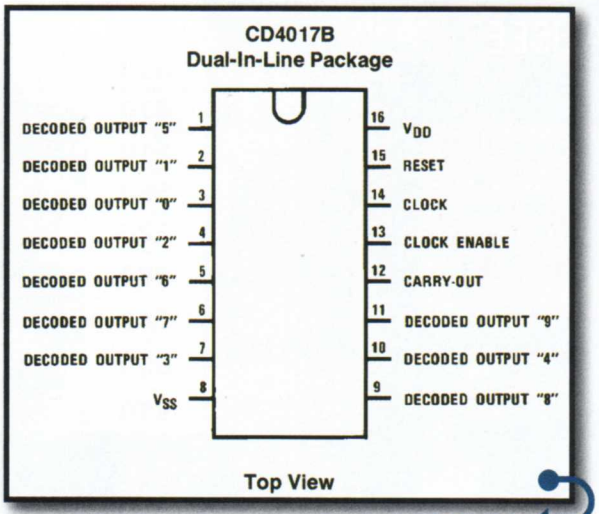
Puissance de 10	Valeur	Unité	Symbole	Nom
10^6	1000000	méga	M	Million
10^3	1000	kilo	k	Mille
10^1	10	déca	da	Dix
1^0	1	unité		Une
10^-3	0,001	milli	m	Millième
10^-6	0,000001	micro	µ	Millionième
10^-9	0,000000001	nano	n	Milliardième
10^-12	0,000000000001	pico	p	Billionième

LOGIQUE DE CONTRÔLE

Nous en arrivons donc à la partie concernant le contrôle de l'allumage ou, autrement dit, à la logique du montage. Là encore de nombreuses solutions existent :

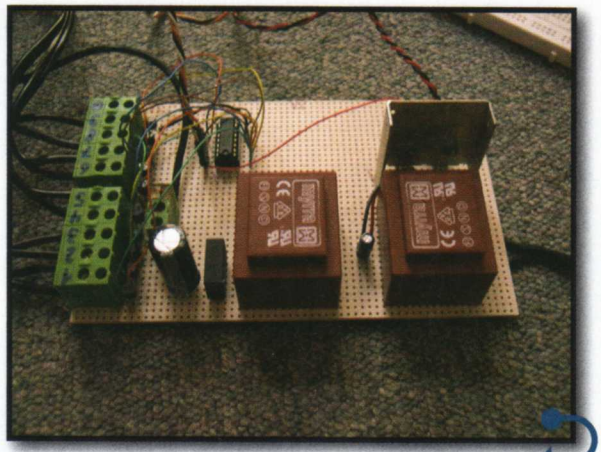
- Directement relier les bases des transistors au port parallèle d'un PC. Avec 8 bits de données plus quelques bits de contrôle, le port parallèle peut faire l'affaire.
- Un décodeur 4 vers 16, comme le 74154, qui, avec une valeur présentée sur 4 bits, permettra d'activer l'une des 16 sorties du composant.
- Un compteur à 10 étages comme le CD4017 qui se pilote avec deux signaux, *reset* et *clock* et qui dispose de 10 sorties.

C'est cette dernière solution qui sera choisie. Notez que la première présente un risque. En effet, en cas d'erreur de programmation du port, il est parfaitement possible que deux transistors ou plus soient saturés. Dans cette situation précise, la valeur de la résistance sur l'anode du tube n'est plus correcte, car ce n'est plus 2.5 mA qui traverseront le circuit, mais davantage. La seconde solution présente également un désavantage mais de moindre importance. Là encore, une erreur dans la programmation du code sur PC risque de changer la donne. En activant une sortie où aucun transistor n'est connecté, le tube ne sera pas alimenté. Ceci peut être souhaitable, mais, personnellement, je n'aime pas l'idée d'avoir un montage utilisant des hautes tensions et qui puisse sembler éteint alors qu'il ne l'est pas.



Le circuit CMOS CD4017 est un compteur piloté par deux signaux et fournissant les 10 sorties pilotant les transistors MPSA-42 (source datasheet Fairchild) **Fig. 9**

La figure 9 présente le brochage d'un CD4017. L'ensemble est relativement simple, +5 volts sur VDD, VSS à la masse ainsi que *Clock Enable* et *Carry Out* (la retenue) en l'air. Pilotez le composant est aisé. Une impulsion sur reset et la broche 3 est active. Ensuite, de brèves impulsions sur Clock permettent d'incrémenter les sorties. Au bout de 10 impulsions, la broche 3 est à nouveau à l'état haut (ainsi que Carry Out).



Le montage complet sur platine. On voit les deux transformateurs dos à dos au premier plan (à droite), le redresseur contenant le pont de diodes (à gauche), le condensateur de filtrage (la résistance est dessous) et les transistors MPSA-42. A l'arrière-plan, se trouve le CD4017. A droite, un gros radiateur cache les régulateurs 7812 et 7805. Sous la platine, se trouve le pont de diodes permettant le redressage et, coté composant, entre les transformateurs, le condensateur de filtrage pour le 18 volts. **Fig. 10**

CODE

J'ai déjà traité à de nombreuses reprises de la manière de piloter les sorties du port parallèle d'un PC sous Linux (cf. HS 23). Le présent numéro traite également du sujet par l'intermédiaire d'adaptateurs USB/Imprimante. Découvrons donc d'autres horizons en nous penchant sur un autre système Unix libre qu'est FreeBSD.

L'architecture du support du port parallèle sous FreeBSD est relativement proche de celle de Linux et tout à fait dans l'esprit Unix (modèle en couches). Voici un extrait de la commande `dmesg` sur un FreeBSD 6.1 :

```
ppc0: <Parallel port> at port 0x378-0x37f irq 7 on isa0
ppc0: Generic chipset (NIBBLE-only) in COMPATIBLE mode
ppbus0: <Parallel port bus> on ppc0
plip0: <PLIP network interface> on ppbus0
lpt0: <Printer> on ppbus0
lpt0: Interrupt-driven port
ppi0: <Parallel I/O> on ppbus0
```

`ppc0` est le port parallèle lui-même (support du *chipset*), sur lequel se greffe le support pour plusieurs fonctionnalités via l'interface générique `ppbus0` :

- `plip0` : Le réseau IP via le port parallèle. Ceci peut offrir des fonctionnalités réseau minimum pour une machine dépourvue d'interface réseau par exemple.

Offres de couplage

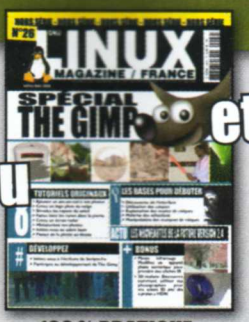
Lisez-vous RÉGULIÈREMENT :

100 % Linux

et / ou

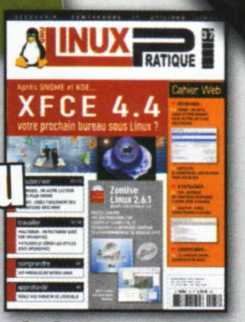
100 % Sécurité

et / ou



100 % PRATIQUE

et / ou



Apprivoisez votre pingouin !

Si OUI, alors ces offres d'abonnement à tarif préférentiel vous sont destinées...

En kiosque(1) ~~109,60 €~~ **79 €**

soit une économie de 27.60 €

En kiosque(3) ~~157,60 €~~ **105 €**

soit une économie de 49,60 €

En kiosque(2) ~~116,20 €~~ **83 €**

soit une économie de 33,20 €

En kiosque(4) ~~180,20 €~~ **129 €**

soit une économie de 61,30 €

(1) Pour 11 N°s Linux Magazine + 6 N°s Linux Mag HS - (2) Pour 11 N°s Linux Magazine + 6 N°s MISC - (3) Pour 11 N°s Linux Magazine + 6 N°s MISC + 6 N°s Linux Mag. HS - (4) Pour 11 N°s Linux Magazine + 6 N°s MISC + 6 N°s Linux Mag. HS + 6 N°s Linux Pratique

OFFRE DE COUPLAGE À REMPLIR ET À RETOURNER À (OU PHOTOCOPIE)

*DIAMOND Éditions - LINUX MAGAZINE - BP 20142 - 67603 SÉLESTAT Cedex

☐ **OUI, je m'abonne et désire profiter des offres spéciales de couplage**

Référence de l'offre :	Prix	Qté.	Total
☐ 11 N°s Linux Mag. + 6 N°s Linux Mag HS	79 €		
☐ 11 N°s Linux Mag. + 6 N°s MISC	83 €		
☐ 11 N°s Linux Mag. + 6 N°s MISC + 6 N°s Linux Mag HS	105 €		
☐ 11 N°s Linux Mag. + 6 N°s MISC + 6 N°s Linux Mag HS + 6 N°s Linux Pratique	129 €		
OFFRES VALABLES UNIQUEMENTS EN FRANCE MÉTRO.		TOTAL	

Pour les tarifs étrangers, consultez notre site : www.ed-diamond.com

LES 4 FAÇONS DE VOUS ABONNER !

- Par courrier postal en nous renvoyant le bon ci-dessous.
- Par le Web, sur notre site : www.ed-diamond.com.
- Par téléphone (paiement C.B.) entre 9h-12h & 14h -17h au 03 88 58 02 08.
- Par Fax au 03 88 58 02 09 C.B. et/ou bon de commande administratif

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon réglement :

☐ Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions**

☐ Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

Boostez votre collection !

Avez-vous L'ÂME du COLLECTIONNEUR ?

Vous recherchez un magazine en particulier? Allez sur www.ed-diamond.com pour voir le sommaire détaillé de chaque magazine et ensuite... Boostez votre collection avec les "POWER PACKS x5", soit 5 Linux Magazine pour 15€ et les "POWER PACKS x10", soit 10 Linux Magazine pour 25€ à choisir dans la liste ci-dessous :

LES 4 FAÇONS DE COMMANDER !

- Par courrier postal en nous renvoyant le bon ci-dessous.
- Par le Web, sur notre site : www.ed-diamond.com.
- Par téléphone (paiement C.B.) entre 9h-12h & 14h -17h au 03 88 58 02 08.
- Par Fax au 03 88 58 02 09 C.B. et/ou bon de commande administratif

Choisissez vos numéros dans le tableau ci-dessous*

*Seuls les numéros ci-dessous sont disponibles pour une commande de POWER PACKS par x5 et x10

N°06 GNOME - The Gimp	N°35 QoS et iproute : optimisation et contrôle du trafic IP	N°57 Maîtrisez la gestion... Slots & Signaux ... des événements en C++
N°07 Dopez Linux	N°36 Linux embarqué : Le projet mGlinux	N°58 Djbdns enfin une alternative viable à BIND !
N°08 Le futur résolument objet	N°37 L'impression sous Linux	N°59 Zopix, Créez un CD "Live" Zope en 10 minutes !
N°09 Prêt pour le jeu !	N°38 Le desktop Shell : Enlightenment	N°60 JBoss serveur d'applications J2EE OpenSource
N°10 The HURD : 100% GNU	N°39 Sécurité : Patchez votre noyau !	N°61 Découvrez MySQL 5 et les procédures stockées
N°11 Exclusif : l'avenir de G.N.O.M.E	N°40 MySQL : la base de donnée OpenSource	N°62 Créez votre OS, principe et implémentation
N°12 NT et Linux : Guerre ou complément ?	N°41 Steganographieou l'art de la dissimulation de données	N°63 Les threads : kernel 2.6 et 2.4
N°13 Cryptage : la clé de la sécurité	N°42 Développez vos pilotes de périphérique	N°64 Adamoto
N°14 XFree 4.0 : le futur à notre portée	N°43 Administrez facilement votre réseau SNMP	N°65 Théorie et pratique : Supervision avec Nagios
N°15 Passez à la vitesse supérieure	N°44 Comprenez NetBios pour Maîtriser l'interopérabilité windows GNU Linux	N°66 Créez votre Distribution Live
N°16 OpenSources : Est-ce suffisant?	N°45 Cohabitation : UnDNS Bind dans un reseau Windows 2000	N°67 C# .NET
N°17 Linux : Système embarqué	N°46 Debian : Utilisez Samba avec le support ACL	N°68 Le crash disque vous guette
N°18 Spécial interview : l'avenir de Linux	N°47 GNUstep : le petit frère de Mac OS X ?	N°69 La réponse de Sun à Linux ? SOLARIS 10
N°19 Dossier spécial : Postgre SQL 7.0	N°48 Caudium, votre prochain serveur Web !	N°70 Découvrez et comprenez la technologie GRID
N°21 Le protocole Internet du 21e siècle : IPv6	N°49 Après MySQL & PostgreSQL SAP DB : La base de données libre & puisante	N°71 Présentation et installation du Hurd
N°22 Le multi-hreading : Une manière moderne de programmer le Multitâche	N°50 Créez un album Photo avec PHP ...et sans MySQL	N°72 Services Web... C/C++ et gSOAP
N°23 Débugger sous Linux	N°51 Boostez votre site Web avec XML grâce à XSLT, CSS & XPath	N°73 Compression théorie algorithmes et programmation
N°24 Palm et Linux	N°52 Linux Temps réel où en est-on aujourd'hui ?	N°74 VFS : Système de fichiers virtuel
N°25 Kernel 2.4.0	N°53 Linux sur PDA : Linux dans votre poche !	N°75 Tuning de code
N°26 <Dossier> XML </Dossier>	N°54 Maîtrisez LVM	N°76 Algorithmes évolutionnistes
N°27 Les systèmes de fichiers journalisés	N°55 Intelligence Artificielle : Principes & programmation de jeux de stratégie classique	
N°28 Scripting : la force d'Unix	N°56 Développez vos applications Mozilla avec XPFE & XPCOM	
N°29 L.F.S. Linux From Scratch		
N°30 Le chiffrement des données		
N°31 VPN et tunneling		
N°32 Changez de coquille		
N°34 XSL - FO :TeX Killer ?		

NUMÉROS LINUX MAGAZINE ÉPUISÉS

N°01, N°02, N°03, N°04, N°05, N°20, N°33.

BON DE COMMANDE POWER PACKS À REMPLIR ET À RETOURNER À (OU PHOTOCOPIE)

*LINUX MAGAZINE - BP 20142 - 67603 SELESTAT CEDEX

	OUI, je désire acquérir un POWER PACK X5	1er 1PP* X5	2éme 2PP* X5	3éme 3PP* X5
Cochez ici ▲ POWER PACKS X5	1, Linux Magazine N°			
	2, Linux Magazine N°			
	3, Linux Magazine N°			
	4, Linux Magazine N°			
	5, Linux Magazine N°			
	Total par série de POWER PACKS X5 :	15 €	30 €	45 €
	OUI, je désire acquérir un POWER PACK X10	1er 1PP* X10	2éme 2PP* X10	3éme 3PP* X10
Cochez ici ▲ POWER PACKS X10	1, Linux Magazine N°			
	2, Linux Magazine N°			
	3, Linux Magazine N°			
	4, Linux Magazine N°			
	5, Linux Magazine N°			
	6, Linux Magazine N°			
	7, Linux Magazine N°			
	8, Linux Magazine N°			
	9, Linux Magazine N°			
	10, Linux Magazine N°			
	Total par série de POWER PACKS X10 :	25 €	50 €	75 €
	Les Hors Séries et numéros spéciaux sont exclus des POWER PACKS. Montant TOTAL 15€ + 3,81€ de frais de port. Le TOTAL s'élève à 18,81€ pour l'achat d'un POWER Pack x5.	TOTAL :		
		Frais de port :		+3,81€
	SEULEMENT EN FRANCE MÉTROPOLITAINE !	TOTAL :		

1 Voici mes coordonnées postales

Nom :

Prénom :

Adresse :

Code Postal :

Ville :

2 Je joins mon réglement :

Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions**

Paiement par carte bancaire :

N° Carte :

Expire le : Cryptogramme Visuel : Voir image ci-dessous

Date et signature obligatoire : 200

- `lpt0` concerne la gestion d'imprimante, un peu comme `lp0` sous Linux (voir article sur USB/imprimante).
- `ppi0` est l'interface qui nous intéresse nous permettant de contrôler les lignes du port depuis l'espace utilisateur. A noter que la page de manuel est judicieusement titrée « *user-space interface to ppbus parallel 'geek' port* ».

FreeBSD permet l'accès direct aux lignes du port parallèle via le pseudo-fichier `/dev/ppi0` à l'instar de Linux et son `/dev/parport0`. Il nous suffit donc d'ouvrir le fichier et d'utiliser un certain nombre d'`ioctl` :

```
#include <fcntl.h>
#include <errno.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

#include <dev/ppbus/ppi.h>
#include <dev/ppbus/ppbconf.h>

int main(int argc, char **argv)
{
  int i;
  int fd;
  char *filename;
  u_int8_t valon;
  u_int8_t valoff;

  filename = strdup("/dev/ppi0");
  valon = 0x01;
  valoff = 0x00;

  if ((fd = open(filename, O_RDWR)) < 0) {
    fprintf(stderr,"Open Error : %s (%d)\n",
      strerror(errno),errno);
    exit(EXIT_FAILURE);
  } else {
    free(filename);
  }

  for(i=0;i<+64;i=1) {
    ioctl(fd, PPISDATA, &valon);
    printf("tic ! (%u)\n",i);
    usleep(100000);
    ioctl(fd, PPISDATA, &valoff);
    printf("tac ! (%u)\n",i);
    usleep(100000);
  }

  if(close(fd) < 0) {
    fprintf(stderr,"Close Error : %s (%d)\n",
      strerror(errno),errno);
    exit(EXIT_FAILURE);
  }

  return(EXIT_SUCCESS);
}
```

Le code parle de lui-même et un simple coup d'œil à `ppi.h`, `ppbconf.h` ainsi qu'aux pages de manuel de `ppi` et `ppbus` vous apprendra tout ce qu'il faut savoir.

CONCLUSION

Cet article ne fait que présenter les bases de l'utilisation de tubes Nixie. Nous l'avons vu, il existe plusieurs modèles et tous possèdent leurs caractéristiques propres. C'est donc chaque fois une nouvelle aventure. En termes de mise en œuvre, là aussi, il y a de quoi faire. Partant des explications données ici, vous pouvez créer un afficheur avec plus de tubes par exemple. Vous pouvez également vous pencher sur d'autres systèmes d'alimentation comme celles dites « à découpage ».

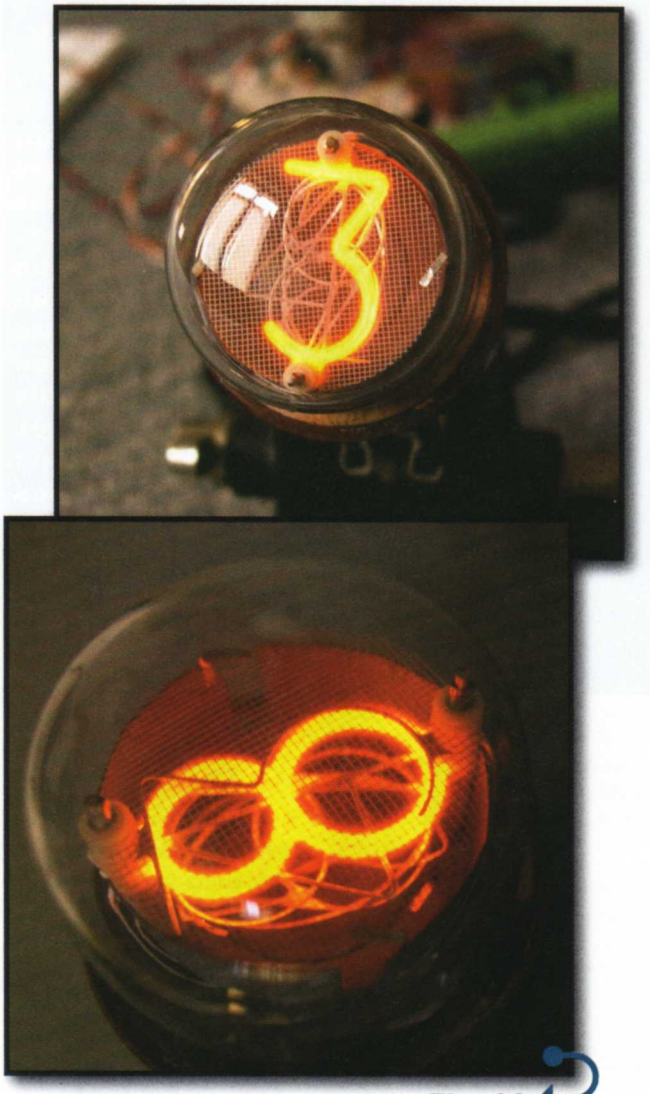


Fig. 11 *Notre tube IN-1 en action. Comment résister à la douce lumière émise ? Ce n'est en rien comparable aux afficheurs LED ou LCD. Seuls les VFD (Vacuum fluorescent display) peuvent éventuellement rivaliser, et encore...*

Il ne vous reste qu'à investir un peu de temps et vous disposerez d'un périphérique qui fera baver d'envie le pire JackyPC de votre entourage, auquel vous prendrez un malin plaisir à raconter que ce n'est compatible qu'avec un système « non sale » ;)

J.-M. Friedt <friedtj@free.fr>, É. Carry, <milou@achelem.org>,
Association Projet Aurore, Besançon

ACQUISITION ET DISSÉMINATION DE TRAMES GPS À DES FINS DE CARTOGRAPHIE LIBRE

Nous avions présenté en mars un circuit électronique permettant à moindre coût d'acquérir des traces de points GPS. Le traitement et la validation des données ainsi obtenues, restaient cependant des tâches relativement fastidieuses, puisque la phase de comparaison des données propriétaires (www.mapquest.com) avec nos données passait par une phase de programmation Postscript délicate (recherche des facteurs d'homothétie et de translation des points afin de les ajuster sur les cartes). Nous allons ici présenter un certain nombre d'améliorations au circuit électronique pour en réduire les dimensions et la consommation électrique, et surtout les outils appropriés au partage des traces GPS acquises et à leur présentation : partage avec UPCT, fusion avec les données disponibles dans Google Earth et Google Maps, utilisation de la librairie M_MAP de Matlab et de GRASS pour la gestion géographique d'information.

1 RÉCEPTEURS GPS UTILISÉS

Nos objectifs dans la réalisation de ce projet sont de quatre ordres : réduire les coûts par rapport aux récepteurs commerciaux, réduire les dimensions, réduire la consommation afin d'atteindre une autonomie maximale, et maîtriser l'ensemble des éléments du projet, tant sur le plan matériel que logiciel.

Un récepteur GPS est un composant excessivement complexe dont la réalisation dépasse largement les compétences des auteurs. Nous avons donc acquis des modules de réception OEM auprès de Lextronic (www.lextronic.fr) : Laipac UV40 basé sur un composant Sony (1) et Laipac PG31 (2). Le premier module, plus ancien, coûtait une centaine d'euros et consomme de l'ordre de 30 mA. La baisse soudaine de son prix laisse supposer qu'il sera bientôt en rupture de stock. Le second module coûte 83 euros/unité, consomme de l'ordre de 70 mA, mais surtout propose le mode WAAS (États-Unis)/EGNOS (Europe) qui améliore considérablement la résolution au sol en informant notamment le récepteur au sol de satellites défectueux.

Les antennes incluent un préamplificateur et donc doivent non seulement inclure un connecteur adapté (SSMT ou MMCX), mais aussi supporter la tension fournie par le récepteur GPS. Les anciennes antennes vendues par Lextronic (Laipac Tech GLP-1) étaient légèrement plus volumineuses que les antennes actuelles (Laipac Tech AT-65), mais capables de soulever une masse deux fois plus importante, 980 grammes pour les premières contre 500 g pour les secondes. Ce commentaire apparemment anodin a cependant une implication importante pour notre application de GPS mobile fixé à un vêtement : les nouvelles antennes nécessitent un système de fixation externe (couture, velcro) alors que les anciennes antennes pouvaient se fixer au moyen d'une pièce métallique magnétisable sous le vêtement. Un test préalable à tout achat d'une nouvelle antenne peut donc s'avérer nécessaire.

2 AMÉLIORATIONS DU SYSTÈME D'ACQUISITION

Un certain nombre d'améliorations ont été apportées au système précédemment développé : nous allons donc brièvement décrire le montage expérimental d'acquisition et de stockage des trames GPS en insistant sur les nouveautés visant à rendre ce système fiable et utilisable.

Le matériel se résume toujours aux mêmes composants : un récepteur GPS OEM fournissant des trames par RS232, un microcontrôleur recevant ces trames et se chargeant de les stocker dans une carte mémoire de stockage de masse non volatile de type MultiMediaCard (MMC). Nous n'entrons pas ici dans les détails du protocole de communication entre ces divers éléments qui a déjà été présenté précédemment [1]. Une amélioration fondamentale cependant est le passage à des composants ne nécessitant tous qu'une alimentation de 3,3 V (contre 5 V auparavant : le récepteur GPS utilisé et décrit en détail plus bas est plus compact que le Motorola Oncore. Le MAX232 chargé de

(1) www.lextronic.fr/laipac/uv40.htm/
(2) www.lextronic.fr/laipac/tf30.htm
(3) 0,80 euros chez Lextronic ou Farnell, réf. 9755349.

générer les tensions compatibles avec le protocole RS232 est remplacé par un MAX3232) et par conséquent d'un régulateur de tension fonctionnant avec une tension d'entrée aussi basse que 3,8 V (LE33CZ (3)). Ainsi, nous pouvons alimenter ce circuit au moyen de 4 accumulateurs NiMH de 1,2 V placés en série, solution moins chère et moins dangereuse que l'utilisation d'accumulateurs LiPo.

La première contrainte à résoudre était la nécessité de reprogrammer le microcontrôleur entre les phases d'acquisition et de restitution des trames GPS. Une modification du circuit de gestion de la communication RS232 nous permet d'identifier si un câble de communication série relie le montage à un PC : nous ajoutons une résistance de *pull-up* entre la broche d'entrée de la communication RS232 du MAX3232 et la tension haute en sortie de la pompe de charge de ce composant (broche 2, Fig. 1).

ceux présentés auparavant, mais consommant nettement moins de courant et surtout beaucoup moins encombrants. De plus, ces récepteurs fournissent des trames NMEA (décrites à http://www.gpsinformation.org/dale/nmea.htm par exemple) au lieu d'un flux binaire. L'avantage de cette communication ASCII est double : d'une part, n'importe quel programme de communication sur port série peut être utilisé pour la récupération des données (nous utilisons `minicom` sous GNU/Linux), et, d'autre part, l'identification et l'élimination de trames erronées est possible et permet donc d'obtenir des traces propres. En effet, l'entropie [2] d'une trame ASCII est plus faible que celle d'un flux binaire (en d'autres termes, la redondance d'information est plus élevée puisqu'une valeur de latitude ou longitude est encodée sur 11 octets au lieu des 8 octets dans un flux binaire) et il est donc possible

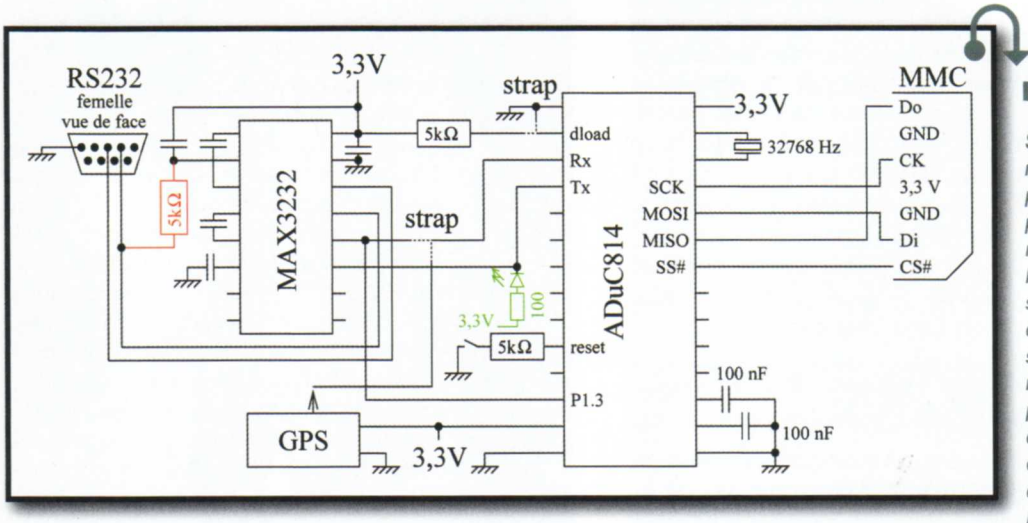


Fig. 1

Schéma du circuit utilisé : un microcontrôleur est connecté d'une part à un récepteur GPS et, d'autre part, à une carte de stockage de masse non volatile MultiMediaCard avec laquelle il communique par protocole synchrone SPI. Une résistance de tirage (en rouge sur la figure) soudée après la programmation du microcontrôleur (afin de ne pas en perturber la communication) permet d'identifier la présence d'un câble entre ce circuit et un PC et donc de déterminer si le microcontrôleur doit initialiser une phase d'acquisition de trames GPS (absence de câble) ou de restitution des données (présence du câble détectée sur la broche P1.3/ADC1). La diode sur la ligne Tx de la liaison RS232 du microcontrôleur (en vert sur la figure) s'allume à chaque écriture d'un bloc de 512 octets sur la MMC (transmission de l'adresse du bloc nouvellement occupé) : cette capacité à valider le bon fonctionnement du montage en cours d'utilisation est apparue fondamentale à l'usage. Deux séries de straps connectent la broche DLOAD (2) soit à l'alimentation (mode programmation), soit à la masse (mode exécution une fois le programme flashé dans le microcontrôleur) ; et le port RS232 du microcontrôleur est soit connecté au PC (phase de programmation), soit au récepteur GPS (phase utilisation).

Ainsi, la polarité en l'absence de câble connecté (mode acquisition de données) est de l'ordre de +12 V, tandis qu'un câble connecté à un PC impose une tension de l'ordre de -12 V (mode restitution des données). La sortie correspondante du MAX232 est dérivée de l'entrée de communication série RxD du microcontrôleur vers une broche généraliste dont le niveau est testé au lancement du programme pour sélectionner le mode de fonctionnement approprié (notez que le MAX232 se comporte comme un inverseur en interne). Ainsi, nous flashons une seule fois un programme dans le microcontrôleur chargé des deux modes de fonctionnement, la bonne routine étant appelée en fonction de la présence ou l'absence d'un câble.

La seconde modification majeure consiste en l'utilisation de récepteurs GPS plus récents et un peu plus coûteux que

d'identifier par des a priori sur la forme des chaînes de caractères attendues les erreurs de transmission. Dans notre cas, nous éliminons toute série de données qui, après extraction du fichier NMEA, tel que décrit plus bas, n'est pas de la forme

```
^[0-9][0-9][0-9][0-9]\.[0-9][0-9][0-9][0-9]\ [0-9][0-9][0-9][0-9][0-9]\.[0-9][0-9][0-9][0-9]\ ....
```

c'est-à-dire une latitude formée de 4 chiffres suivis de 4 décimales et une longitude formée de 5 chiffres suivis de 4 décimales, le tout suivi d'au moins 4 caractères formant l'altitude.

En termes de taille de fichiers, le passage d'une sauvegarde en mode binaire à une sauvegarde en ASCII se traduit bien entendu par une occupation mémoire plus importante,

même si le récepteur Motorola Oncore (format binaire) sauve plus de télémétrie sur les satellites visibles que ne le fait le format NMEA. Ainsi, nous avons constaté que le format binaire du Motorola Oncore génère de l'ordre de 572 KB/h tandis que le récepteur Laipac PG31 génère de l'ordre de 800 KB/h au format NMEA et le module Laipac UV40 (récepteur GPS Sony) génère de l'ordre de 1300 KB/h au format NMEA. Dans tous les cas, nous constatons qu'une carte MMC de 128 MB telle que couramment disponible aujourd'hui pour moins de 20 euros contient plus d'une semaine de traces à raison de 14 heures d'acquisitions continues par jour.

Un dernier point concernant la récupération des données stockées sur une carte MMC de grande taille concerne la séparation des trajets distincts en fichiers que nous pourrons traiter indépendamment et contenant chacun une trace continue. En effet, il est probable qu'en l'absence de possibilité de recharger les batteries, nous privilégions l'accumulation de traces successives plutôt qu'utiliser l'énergie disponible à récupérer les traces accumulées dans la journée. Nous constatons que les récepteurs GPS fournissent systématiquement quelques trames d'initialisation lors de leur mise sous tension : dans le cas qui nous intéresse, nous obtenons des informations du type `$TOW: 0` ou `$CHNL: 12` uniquement dans l'en-tête et jamais dans les trames NMEA qui suivent. L'outil `csplit` nous permet de séparer une grosse archive de traces successives accumulées sur plusieurs jours en de petits fichiers contenant chacun une trace continue :

```
csplit -f trace fichier.nmea /\$TOW/ {*}
```

coupe le fichier `fichier.nmea` tel que récupéré de la carte MMC en de petits fichiers `traceXX` (`XX` s'incrémente pour chaque nouveau fichier) à chaque occurrence de la chaîne de caractères `$TOW`.

2.1 CIRCUIT UTILISÉ

Reprenons ici la procédure de programmation de l'ADuC814 telle que nous l'avions décrite auparavant [1]. Afin de permettre l'acquisition de traces multiples au cours d'un même trajet entre lesquelles le récepteur GPS est éteint pour économiser les batteries, nous stockons, dans une zone de la mémoire non volatile du microcontrôleur (adresses $BC à $BF), l'adresse du dernier bloc écrit sur MMC. Cette zone mémoire est actualisée après chaque écriture de 512 octets sur MMC. Lors de la mise en marche du circuit, une des premières tâches du microcontrôleur est de charger l'adresse du dernier bloc accédé et d'initialiser avec cette valeur le compteur d'adresse : un problème d'initialisation se pose donc. Nous avons écrit un programme chargé de mettre à zéro la mémoire non volatile de l'ADuC814 (fonction `wrEE` du code présenté dans le tableau 1) et qui, de plus, nous sert à valider la connectique entre le microcontrôleur et la MMC indépendamment de tout récepteur GPS.

Ce programme est présenté dans le tableau 2 et est disponible à http://jmfriedt.free.fr/.

TABLEAU 1

```
;_____________________________________________________________
HEX2ASCII:      ; converts A into the hex character
                ; representing the
                ; value of A's least significant nibble

        ANL     A,#0h0F
bbb:    CJNE    A,#0h0A,bbb+3
        JC      IO0030
        ADD     A,#0h07
IO0030: ADD     A,#0h30         ; '0'
        RET
;_____________________________________________________________
SENDCHAR:       ; sends ASCII value contained in A to UART
        JNB     TI,SENDCHAR     ; wait til present char gone
        CLR     TI              ; must clear TI
        MOV     SBUF,A
        RET
;_____________________________________________________________
SENDSPI: ; sends the content of A to the SPI port
 MOV    0hF7,A          ; trigger data transfer: SPIDAT=0hF7
icispi: JNB 0hFF,icispi
 CLR 0hFF  ; clear ISPI
 RET
;_____________________________________________________________
READSPI: ; sends the content of A to the SPI port
 mov    B,#8  ; max of 8 'FF' answers, otherwise continue ...
retry1: MOV A,#0hff         ; generate clocks for reception
 LCALL SENDSPI

 mov    A,0hF7  ; MOV A,SPIDAT: read resulting value
 CJNE   A,#0hff,readend ; we only continue if the MMC answers
                        ; NOT 0
        DJNZ    B,retry1
 ; SJMP readspi
readend:RET
;_____________________________________________________________
READSPIVAL:     ; sends the content of A to the SPI port
        MOV     A,#0hff         ; generate clocks for
                                ; reception
        LCALL   SENDSPI
        mov     A,0hF7          ; MOV A,SPIDAT: read resulting
                                ; value
        RET
;_____________________________________________________________
GETCHAR:        ; waits for a single ASCII character to be
received
                ; by the UART.  places this character into A.
iciget: JNB     RI,iciget
        MOV     A,SBUF
        CLR     RI
        RET
;_____________________________________________________________
DELAY:                                  ; Delays by 100ms * A
                                        ; 100mSec based on
2.097152M
                                        ; Core Clock
                                        ; i.e. default ADuC814
 Clock

                MOV     R1,A            ; Acc holds delay
```

```
variable
 DLY0:          MOV     R2,#0h22        ; Set up delay loop0
 DLY1:          MOV     R3,#0hFF        ; Set up delay loop1
ici:            DJNZ    R3,ici          ; Dec R3 & Jump here
until R
                DJNZ    R2,DLY1         ; Dec R2 & Jump DLY1
until R
                DJNZ    R1,DLY0         ; Dec R1 & Jump DLY0
until R
                RET                     ; Return from
subroutine
;_____ LIRE L'ADRESSE DE DEMARRAGE EN FLASH
rdEE:  mov      EADRL,#0h00     ; EEPROM page (1..159)
       mov      ECON,#0h01      ; read page
       mov      A,EDATA1
       LCALL    SENDCHAR
       mov      MEM0,A
       mov      A,EDATA2
       LCALL    SENDCHAR
       mov      MEM1,A
       mov      A,EDATA3
       LCALL    SENDCHAR
       mov      MEM2,A
       mov      A,EDATA4
       LCALL    SENDCHAR
       mov      MEM3,A
       ret
;_____ ECRIRE L'ADRESSE COURANTE EN FLASH
wrEE:   mov     EADRL,#0h00     ; EEPROM page (0..159)
        mov     EDATA1,MEM0
        mov     EDATA2,MEM1
        mov     EDATA3,MEM2
        mov     EDATA4,MEM3
        mov     ECON,#0h05      ; erase page
        mov     ECON,#0h02      ; write page
 ret


;_____ EFFACER L'ADRESSE COURANTE EN FLASH
rmEE:   mov     EADRL,#0h00     ; EEPROM page (0..159)
        mov     EDATA1,#00
        mov     EDATA2,#00
        mov     EDATA3,#00
        mov     EDATA4,#00
        mov     ECON,#0h05      ; erase page
        mov     ECON,#0h02      ; write page
 ret
```

Quelques fonctions de base utiles dans tout programme en assembleur 8051 pour ADuC814, utilisées dans les programmes qui suivent.

Une fois la mémoire non volatile initialisée, il nous reste à transférer le programme de l'application *sans modifier la mémoire non volatile* contenant l'adresse de départ de la MMC. Pour ce faire, il est fondamental de programmer la plage programme de l'ADuC814 par la commande 'C' et non 'A' qui effacerait la mémoire non volatile pour y stocker des valeurs aléatoires. La conséquence de cette réinitialisation erronée serait une adresse de départ non nulle de la MMC, et surtout une adresse qui peut ne pas être dans la plage accessible par la MMC et donc un plantage du programme d'application qui se verrait dans l'impossibilité de stocker les trames GPS lues.

TABLEAU 2

```
SS      = 0hb5          ; P3.5 drives slave device's SS pin
        LED     = 0hb3
        CHAN    = 0             ; convert this ADC input channel (0 thru
                                ; 6)

        MEM0    = 0h30          ; RAM locations: MMC address and a
                                ; counter
        MEM1    = 0h31
        MEM2    = 0h32
        MEM3    = 0h33
        CNT     = 0h34

        EADRL=0hc6
        ECON=0hb9
        EDATA1=0hbc
        EDATA2=0hbd
        EDATA3=0hbe
        EDATA4=0hbf

        .area code (ABS)
        .org 0h0000
RSTirq: ljmp    MAIN ; 3 bytes  ,
IE0irq: nop nop nop nop nop nop nop nop
TF0irq: nop nop nop nop nop nop nop nop
IE1irq: nop nop nop nop nop nop nop nop
TF1irq: nop nop nop nop nop nop nop nop
TIirq:  nop nop nop nop nop nop nop nop
TF2irq: nop nop nop nop nop nop nop nop
ADCirq: setb LED reti nop nop nop nop nop
ISPirq: nop nop nop nop nop nop nop nop
PSMirq: nop nop nop nop nop nop nop nop
TIIirq: nop nop nop nop nop nop nop nop
WDSirq: nop nop nop nop nop nop nop nop nop nop nop nop nop

MAIN:   MOV     RCAP2H,#0hFF ; config UART for 9600 baud
        MOV     RCAP2L,#-7   ;
        MOV     TH2,#0hFF
        MOV     TL2,#-7
        MOV     SCON,#0h52
        MOV     T2CON,#0h34

        setb    ss
        mov     0h9c,#0h01      ; cfg814=0h9C
        mov     0hF8,#0h3D      ; SPICON=0hF8, CPHA=CPOL=1=>0h3D

        lcall rmEE              ; erase current flash mem adress
        lcall rdEE              ; erase current flash mem adress

;-- START INITALIZING MMC: send 10 times 0xff ----------------
        MOV     A,#0h0ff        ; send 10 times $ff with SS#=hi
        mov     R0,#0h0a
init:   LCALL   SENDSPI
        DJNZ    R0,init
        NOP
        NOP
        NOP

;-- now reset MMC: send 0x40/00/00/00/00/0x95=cmd(00) -------------
        clr     ss              ;
        MOV     A,#0h40         ; RESET MMC
        LCALL   SENDSPI         ;

        MOV     A,#0h00         ; RESET MMC: send 4 times '0x00'
```

```
        mov     R0,#0h04        ;
rst:    LCALL   SENDSPI         ;
        DJNZ    R0,rst
        MOV     A,#0h95         ; ... and send CRC
        LCALL   SENDSPI
        NOP
        NOP
        NOP

loop2:  LCALL   READSPI
        LCALL   SENDCHAR
        CJNE    A,#0h01,loop2   ; we only continue if the MMC answers
                                ; '0x01'

cmd1:   setb    ss
        MOV     A,#0hff
        LCALL   SENDSPI

        clr     ss              ;
        MOV     A,#0h41         ; command 0x01 & 0x40
        LCALL   SENDSPI
        MOV     A,#0h00
        LCALL   SENDSPI
        MOV     A,#0            ; A,#0hc0
        LCALL   SENDSPI
        MOV     A,#0            ; A,#0hff
        LCALL   SENDSPI
        MOV     A,#0h00
        LCALL   SENDSPI
        MOV     A,#0hff
        LCALL   SENDSPI

loop3:  LCALL   READSPI
        LCALL   SENDCHAR
        CJNE    A,#0h01,SPIok   ; we continue as long as MMC answers
                                ; '0x01'
        SJMP    cmd1            ; otherwise MMC should answer 0x00: SPI
                                ; mode OK
                                ; AT THIS POINT MMC IS IN SPI MODE
SPIok:  setb    ss
        MOV     A,#0hff
        LCALL   SENDSPI

        mov     MEM0,#0         ; RAM location 0, 1, 2 and 3 store the
                                ; address
        mov     MEM1,#0
        mov     MEM2,#0
        mov     MEM3,#0
contadc:mov     A,MEM1
        LCALL   SENDCHAR
        mov     A,MEM2
        LCALL   SENDCHAR
        LCALL   MMCwriteBlock   ; write block at address 0
        inc     MEM1
        inc     MEM1

        mov     A,MEM1
        JNZ     contadc         ; jmp if accumulator is not 0
        inc     MEM2
        mov     A,MEM2
        JNZ     contadc
        inc     MEM3
        mov     A,MEM3
        JNZ     contadc

theend: SJMP    theend
```

```
;_____________________________________________________
; 1/ write command + 4 bytes writing address, multiple of 512,
; MSB firs
; 3/ write $FF (CRC) + wait until answer is 0
; 5/ write $FE, followed by 512 bytes and 2x$FF as CRC
; 6/ read answer and check that it finishes with 5 (ie
; answer=$X5)
; 7/ read while 0 -> continue when answer is non-0 (*** might
; be $FF
; care of not locking at point 7 ***)
MMCwriteBlock:
        clr     ss
        MOV     A,#0h58         ; command 0d24=0x18 & 0x40
        LCALL   SENDSPI
        MOV     A,MEM3          ; hi byte first ...
        LCALL   SENDSPI
        MOV     A,MEM2
        LCALL   SENDSPI
        MOV     A,MEM1
        LCALL   SENDSPI
        MOV     A,MEM0          ; ... and lo byte last
        LCALL   SENDSPI
        MOV     A,#0hff         ; CRC
        LCALL   SENDSPI

        LCALL   READSPI         ; should be 0x00
        LCALL   SENDCHAR

        MOV     A,#0hfe         ; answer for writing data
        LCALL   SENDSPI
        MOV     A,#0hff         ; DELETE ALL DATA
        mov     R1,#0h02
writ3:  mov     R0,#0hff
writ2:  LCALL   SENDSPI         ; read value
        DJNZ    R0,writ2
        DJNZ    R1,writ3

        MOV     A,#0hff         ; CRC (0xff)
        LCALL   SENDSPI         ; x2
        LCALL   SENDSPI
        LCALL   READSPI         ; write response: should
                                ; finish with 5=ACK

lpwrit: LCALL   READSPI         ; read SPI ...
        CJNE    A,#0h00,endwrit ; ... while answer is 0x00
        SJMP    lpwrit
endwrit:setb    ss
        MOV     A,#0hff         ; CRC (0xff)
        LCALL   SENDSPI
        RET
```

Programme en assembleur 8051 pour ADuC814 chargé de communiquer avec une MultiMediaCard, d'effacer la mémoire non volatile de données du microcontrôleur, puis de remettre à $FF tous les emplacements mémoire de la MMC. Les 3 octets de poids fort de l'adresse d'un bloc effacé avec succès sont transférés sur le port série (9600, N81) : l'affichage rapide de successions de 3 octets est donc une indication du bon fonctionnement de ce programme qui valide donc les connexions électriques entre l'ADuC814 et le MAX3232, d'une part, la MMC, d'autre part. Les fonctions appelées dans ce programme sont incluses dans le tableau 1.

TABLEAU 3

```
        SS      = 0hb5          ; P3.5 drives slave device's
                                ; SS pin
        LED     = 0hb3
        CHAN    = 0             ; convert this ADC input
                                ; channel (0 thru 6)

        MEM0    = 0h30          ; RAM locations: MMC address
                                ; and a counter
        MEM1    = 0h31
        MEM2    = 0h32
        MEM3    = 0h33
        dmpMEM0 = 0h34          ; RAM locations: MMC address
                                ; and a counter
        dmpMEM1 = 0h35
        dmpMEM2 = 0h36
        dmpMEM3 = 0h37

        EADRL=0hc6
        ECON=0hb9
        EDATA1=0hbc
        EDATA2=0hbd
        EDATA3=0hbe
        EDATA4=0hbf

        .area code (ABS)
        .org 0h0000
RSTirq: ljmp    MAIN ; 3 bytes
IE0irq: nop nop nop nop nop nop nop nop
TF0irq: nop nop nop nop nop nop nop nop
IE1irq: nop nop nop nop nop nop nop nop
TF1irq: nop nop nop nop nop nop nop nop
TIirq:  nop nop nop nop nop nop nop nop   ; RI+TI interrupt
TF2irq: nop nop nop nop nop nop nop nop
ADCirq: setb LED reti nop nop nop nop nop
ISPirq: nop nop nop nop nop nop nop nop
PSMirq: nop nop nop nop nop nop nop nop
TIIirq: nop nop nop nop nop nop nop nop
WDSirq: nop nop nop nop nop nop nop nop nop nop nop nop nop

MAIN:
        MOV     RCAP2H,#0hFF ; config UART for 9600 baud
        MOV     TH2,#0hFF
        MOV     SCON,#0h52
        MOV     T2CON,#0h34

        MOV     0hEF,#0h80   ; power up ADC -- ADCCON1=0hEF
        SETB    0hAF         ; enable interrupts -- EA=0hAF
        SETB    0hAE         ; enable ADC interrupt --
                             ; EADC=0hAE

        MOV     P1,#0hF7     ; P1.3=digital input
        CLR     LED          ; turn the LED off/no AD
                             ; conversion occured yet
        setb    ss
        mov     0h9c,#0h01      ; cfg814=0h9C
        mov     0hF8,#0h3D      ; SPICON=0hF8,
                                ; CPHA=CPOL=1=>0h3D

;-- START INITALIZING MMC: send 10 times 0xff ----------------

[...]
SPIok:  setb    ss
        MOV     A,#0hff
        LCALL   SENDSPI
        LCALL   rdEE
        ; on teste le port serie : si niveau haut alors ... sinon ...
        jnb     P1.3,rs232inactif       ; branch if bit is 0 (see also
                                        ; jb)
        LJMP    xfer

rs232inactif:
        MOV     RCAP2L,#-14
        MOV     TL2,#-14

contadc:mov     A,MEM1
        LCALL   SENDCHAR
        mov     A,MEM2
        LCALL   SENDCHAR
        LCALL   MMCwriteBlock   ; write block at address 0
        lcall   wrEE            ; sauver l'adresse qui vient d'être
                                ; remplie
        inc     MEM1
        inc     MEM1
        mov     A,MEM1
        JNZ     contadc         ; jmp if accumulator is not 0
        inc     MEM2
        mov     A,MEM2
        JNZ     contadc
        inc     MEM3
        mov     A,MEM3
        JNZ     contadc

theend: SJMP    theend

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MMCwriteBlock:
        clr     ss
        MOV     A,#0h58         ; command 0d24=0x18 & 0x40
        LCALL   SENDSPI
        MOV     A,MEM3          ; hi byte first ...
        LCALL   SENDSPI
        MOV     A,MEM2
        LCALL   SENDSPI
        MOV     A,MEM1
        LCALL   SENDSPI
        MOV     A,MEM0          ; ... and lo byte last
        LCALL   SENDSPI
        MOV     A,#0hff         ; CRC
        LCALL   SENDSPI

        LCALL   READSPI         ; should be 0x00
        LCALL   SENDCHAR

        MOV     A,#0hfe         ; answer for writing data
        LCALL   SENDSPI
        mov     R1,#0h02
writ3:  mov     R0,#0h00
writ2:  LCALL   GETCHAR ; ... OR store value received from RS232 port
        CJNE    A,#0h40,pasA    ; ajout : on lit ADC si on a '@'=0x40
        JNB     LED,premier     ;   LED=1 => conversion done, read 2nd
                                ; byte
 ; rque : on peut continuer comme ca avec d'autres compteurs pour
                                ; d'autres ADC
        MOV     A,0hD9          ;   ADCDATAL=0hD9
        CLR     LED             ;   reset converter
        SJMP    sauve           ;   and save value
premier:;CLR    LED             ;   ... OR read ADC: turn the LED off
        MOV     0hD8,#CHAN      ;   select channel to convert -- ADCCO
        SETB    0hDC            ;   innitiate single ADC conversion --
```

```
                        ;   ADC ISR is called upon completion
iciadc: JNB     LED,iciadc
        MOV     A,#0hDA         ;   ADCDATAH=0hDA
        sjmp    sauve           ;   garde LED a 1 pour le 2eme octet
pasA:   clr     LED             ; remet a 0 le compteur de sauvegarde
ADC
        ;LCALL   SENDCHAR       ;   ici A contient le ADC au lieu de @
sauve:  LCALL   SENDSPI         ; read value
        ;MOV     A,#0hD9         ; ADCDATAL=0hD9
        ;LCALL  SENDSPI         ; read value
        ;MOV     A,#0h10         ; Delay length
        ;LCALL   DELAY           ; delay 100ms
        DJNZ    R0,writ2
        DJNZ    R1,writ3

        MOV     A,#0hff         ; CRC (0xff)
        LCALL   SENDSPI         ; x2
        LCALL   SENDSPI
        LCALL   READSPI         ; write response: should finish with
5=ACK

lpwrit: LCALL   READSPI         ; read SPI ...
        CJNE    A,#0h00,endwrit ; ... while answer is 0x00
        SJMP    lpwrit
endwrit:setb    ss
        MOV     A,#0hff         ; CRC (0xff)
        LCALL   SENDSPI
        RET
```

Première partie du programme en assembleur 8051 pour ADuC814 chargé de recevoir les trames GPS transmises sur le port RS232 (4800 bauds) et de les stocker dans une MultiMediaCard, ou de restituer vers un PC par RS232 (9600 bauds) les informations accumulées sur la MMC. Notez que ce circuit et le programme associé servent d'enregistreur de données universel et ne se restreignent pas à l'enregistrement d'informations issues d'un récepteur GPS. Nous avons volontairement laissé en commentaire les fonctions d'enregistrement des données issues de convertisseurs analogique-numérique pour stockage sur la MMC dans la fonction MMCwriteBlock. Les fonctions appelées dans ce programme sont incluses dans le tableau 1, tandis que dans un souci d'allègement de la présentation, l'initialisation de la MMC, identique à celle présentée dans le tableau 2, a été remplacée par les symboles "[...]" après la 45ème ligne de texte dans ce code.

TABLEAU 3

```
xfer:
        MOV     RCAP2L,#-7
        MOV     TL2,#-7
mov     A,#0haa
lcall   SENDCHAR
mov     A,#0haa
lcall   SENDCHAR
mov     A,MEM1
lcall   SENDCHAR
mov     A,MEM2
lcall   SENDCHAR
mov     A,MEM3
lcall   SENDCHAR
```

```
        mov     dmpMEM0,#0      ; RAM location 0, 1, 2 and 3
                                ; store the address
        mov     dmpMEM1,#0      ; 64 or 0
        mov     dmpMEM2,#0
        mov     dmpMEM3,#0

        mov     A,MEM1
        jnz     plus2
        mov     A,MEM2
        jnz     plus2
        mov     A,MEM3
        jnz     plus2
        mov     MEM3,#0h01      ; si on est arrive ici,
                                ; MEM1=MEM2=MEM3=0 ie
                                ; on a fait un reset trop vite
                                ; => dump 16 MB

plus2:  inc     MEM1            ; necessairement paire donc
                                ; +1 != 0
        inc     MEM1            ; on lira juskau bloc suivant
        mov     A,MEM1
        JNZ     incmem2
        inc     MEM2
        mov     A,MEM2
        JNZ     incmem2
        inc     MEM3
incmem2:

litout: LCALL   MMCreadBlock    ; read block at address 0x00

        inc     dmpMEM1
        inc     dmpMEM1
        lcall   compare
        mov     A,dmpMEM1       ; set bit Z
        JNZ     litout
        inc     dmpMEM2
        lcall   compare
        mov     A,dmpMEM2       ; set bit Z
        JNZ     litout
        inc     dmpMEM3
        lcall   compare
        sjmp    litout          ; au cas ou` ...

compare:mov     A,MEM3
        CJNE    A,dmpMEM3,pasegal
        mov     A,MEM2
        CJNE    A,dmpMEM2,pasegal
        mov     A,MEM1
        CJNE    A,dmpMEM1,pasegal
        lcall   rmEE                    ;
MEM2{2,3}=dmpMEM{2,3} : fini
        mov     A,#0h45         ; E
        LCALL   SENDCHAR        ;
        mov     A,#0h4E         ; N
        LCALL   SENDCHAR
        mov     A,#0h44         ; D
        LCALL   SENDCHAR
theend2: SJMP    theend2;       fin de dumpall
pasegal:ret
;-------------------------------------------------------------------
MMCreadBlock:
        clr     ss              ;
        MOV     A,#0h51         ; command 0d17=0x11 & 0x40
        LCALL   SENDSPI
        MOV     A,dmpMEM3         ; hi byte first ...
        LCALL   SENDSPI
        MOV     A,dmpMEM2
```

(4) Disponible chez Farnell pour 2,08 euros/pièce.

```
        LCALL   SENDSPI
        MOV     A,dmpMEM1
        LCALL   SENDSPI
        MOV     A,dmpMEM0          ; ... and lo byte last
        LCALL   SENDSPI
        MOV     A,#0hff         ; CRC
        LCALL   SENDSPI

        LCALL   READSPI         ; should be 0x00
        LCALL   READSPI         ; should be 0xfe

        mov     R1,#0h02        ; here we read 512=2x256
                                ; values
read1:  mov     R0,#0h00        ;
read0:  LCALL   READSPIVAL      ; read value, even if it is
                                ; 0xFF => we keep
        LCALL   SENDCHAR        ; all values and cannot use
                                ; READSPI here
        DJNZ    R0,read0
        DJNZ    R1,read1

        LCALL   READSPIVAL              ; read CRC
                                        ; POURQUOI PAS ?
        LCALL   READSPIVAL              ; read CRC
                                        ; POURQUOI PAS ?
        setb    ss

        RET
```

Seconde partie du programme en assembleur 8051 pour ADuC814 (à concaténer au code présenté au tableau 3) chargé de restituer vers un PC par RS232 (9600 bauds) les informations accumulées sur la MMC. Les fonctions appelées dans ce programme sont incluses dans le tableau 1.

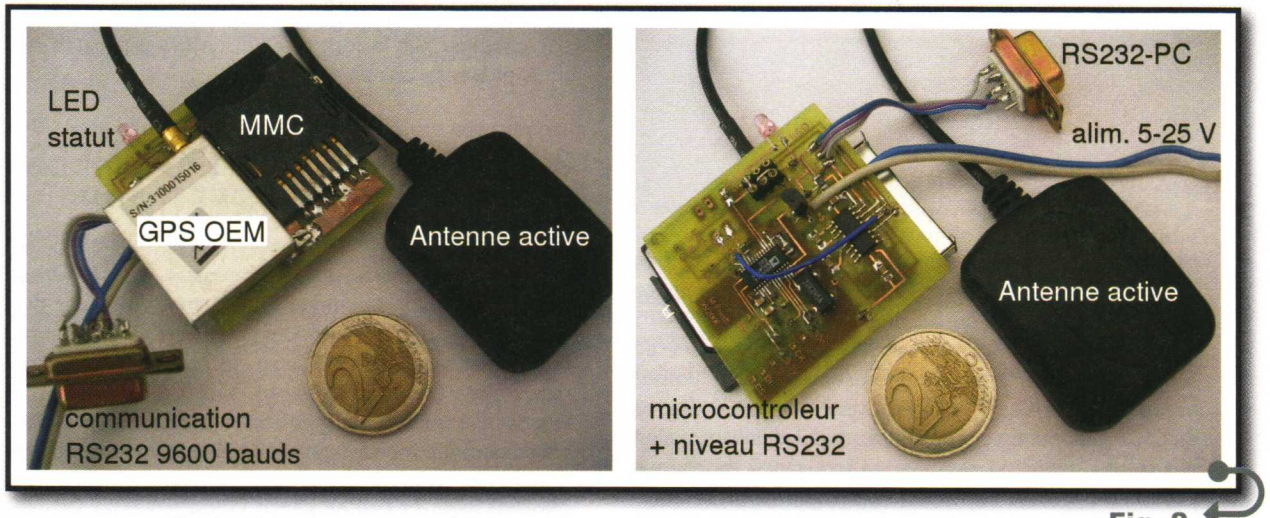
Le programme d'application, présenté dans les tableaux 3 et 4, et disponible à http://jmfriedt.free.fr, contient deux fonctions principales appelées selon la présence ou l'absence d'un câble RS232 : en l'absence du câble (label `rs232inactif`) toute trame RS232 à 4800 bauds est capturée et stockée sur la MMC (fonction `MMCwriteBlock` qui envoie sur la MMC les données issues de la routine `GETCHAR` au niveau du label `writ2`) ; ou restitue les informations stockées sur la MMC à la vitesse de 9600 bauds si le câble est présent (label `xfer`). Dans ce second cas, la lecture commence à l'adresse 0x00000000 et continue jusqu'à atteindre l'adresse du dernier bloc accédé lors de la phase d'enregistrement, tel que stocké en mémoire non volatile. Une fois toutes les données transférées au PC, la mémoire non volatile du microcontrôleur est réinitialisée afin de permettre la capture d'une nouvelle trace (fonction `rmEE`).

Nous avons choisi d'embarquer le convertisseur de niveaux MAX3232 (4), car sa consommation est négligeable devant celle du récepteur GPS et il garantit la disponibilité d'un circuit totalement autonome pour transférer les données accumulées (Fig. 2).

2.2 LES TRAMES ISSUES DES RÉCEPTEURS GPS

Notre première tâche après transfert des trames brutes au format NMEA sur le PC est d'extraire les informations pertinentes et les convertir en un format utilisable.

Seules les lignes commençant par les caractères `$GPGGA` vont nous intéresser ici, les autres lignes contenant



Circuit utilisé au cours de ces expériences. Le fil bleu additionnel sert à la détection de la présence ou absence d'une connexion RS232 avec un PC, déterminant si le microcontrôleur lance le programme d'acquisition de données depuis le récepteur GPS (absence de connexion) ou le programme de restitution des trames stockées sur la MMC (présence du câble). La diode électroluminescente (LED) fournissant le statut de fonctionnement du récepteur est apparue à l'usage comme une option fondamentale pour déterminer si le circuit fonctionne correctement : la diode s'allume à chaque écriture sur le port RS232, notamment après chaque stockage d'un bloc de 512 octets lors de l'acquisition des trames.

Fig. 2

principalement de la télémétrie sur les satellites visibles le long du parcours. Cette ligne GPGGA contient notamment la latitude, la longitude et l'altitude du récepteur GPS : nous allons extraire ces informations au moyen de grep, puis cut. Nous constatons cependant que de nombreux points sont aberrants : nous attribuons ces erreurs au stockage des informations sur MultiMediaCard. En effet, cette dernière place en mémoire tampon 512 octets avant de physiquement les écrire en mémoire non volatile. Cette écriture prend un certain temps, pendant lequel nous plaçons le microcontrôleur en attente d'un acquittement de la carte confirmant la bonne écriture des données. Or l'UART de l'ADuC814 ne peut conserver que le dernier octet reçu sur le port série : tout nouvel octet écrase le précédent. Nous attribuons donc les lignes erronées aux caractères perdus lors du stockage des trames en mémoire non volatile de la MMC. Ces points, relativement rares, mais nettement visibles au tracé des parcours, sont éliminés par un filtrage suivant un a priori fort sur la forme des points tel que mentionné auparavant : 4 chiffres et 4 décimales pour la latitude, 5 chiffres et 4 décimales pour une longitude.

Nous voulons donc ne conserver que les informations de latitude, longitude et altitude (champs 3, 5 et 10 respectivement de la trame GPGGA), retirer les virgules et les lignes contenant encore des lettres (par exemple à cause d'une erreur d'enregistrement des trames qui accole une lettre à ce qui devrait être un nombre), ne conserver que les nombres contenant 4 ou 5 chiffres avant la virgule et 4 décimales, et finalement retirer les trames qui contiennent encore la valeur ASCII 0xff qui peut se trouver dans les trames du fait de notre implémentation du protocole de sauvegarde sur MultiMediaCard. Ces opérations se résument dans la ligne suivante :

```
grep -a GGA ${nom}.nmea | cut -d, -f3,5,10 | \
sed 's/,/ /g' | grep -v "[A-Z]" | \
grep "^[0-9][0-9][0-9][0-9]\.[0-9][0-9]"\
"[0-9][0-9]\ [0-9][0-9][0-9][0-9][0-9]"\
"\.[0-9][0-9][0-9][0-9]\ ...." | \
grep -v $'\xff' | grep "^4"
```

Nous avons ajouté une hypothèse concernant la localisation des traces : nous supposons que la latitude commence par un 4. Ceci est dû au fait que divers récepteurs GPS fournissent des informations variables au cours de l'initialisation : par exemple les récepteurs Sony utilisés donnent 35 degrés, 37.8333 minutes Nord et 139 degrés, 44.6667 minutes Est comme valeurs par défaut, position que nous n'avons pu identifier, se trouvant à mi-chemin entre les sièges de Sony et de NEC (5) à Tôkyô. Cette hypothèse sur la localisation des traces devra être adaptée pour l'utilisation des informations de cet article à des pays autres que la France.

Une fois les trames appropriées sélectionnées, il nous faut les convertir en une information utilisable pour le tracé : au lieu de degrés suivis de minutes (entre 0 et 60) et de fractions de minutes tels que fournis par NMEA, il nous faudrait des degrés suivis de fractions de degrés (entre 0 et 1) afin de déjà tracer les trames sous gnuplot ou Matlab/Octave. Nous avons pour cela écrit un petit programme de conversion fonctionnant sous Matlab/Octave nommé deg2dec.m :

```
function sortie=deg2dec(entree)
sortieX=floor(entree(:,1)/100);
sortieX=sortieX+((entree(:,1)/100)-sortieX)/60*100;
sortieY=floor(entree(:,2)/100);
sortieY=sortieY+((entree(:,2)/100)-sortieY)/60*100;
sortie=[sortieY sortieX entree(:,3)];
```

Nous conservons systématiquement la sortie en format ASCII (fonction save -ascii de octave de cette fonction comme base de travail ultérieur. Par exemple, ce programme transforme :

```
4803.0632 150.5188 173.7
4803.0826 150.5236 173.5
4803.102 150.5285 173.4
4803.1214 150.5336 173.2
4803.1408 150.5388 173.1
```

en

```
1.84198 48.0510533333333 173.7
1.84206 48.0513766666667 173.5
1.84214166666667 48.0517 173.4
1.84222666666667 48.0520233333333 173.2
1.84231333333333 48.0523466666667 173.1
```

après inversion des latitudes et longitudes telle que requise par exemple par le format KML de Google Earth que nous présenterons plus loin.

3 LES BASES DE DONNÉES UPCT ET OPENSTREETMAP

Un des formats acceptables par la base de données du projet UPCT (6) est celui utilisé par gpsman. Bien que relativement bien documenté, le format de fichier doit être *scrupuleusement* respecté pour être acceptable par le serveur web qui se charge de rapatrier les points des traces GPS. Nous avons, avec l'aide de Michel Bondaz [3] qui nous a gracieusement fourni un fichier correctement formaté, pu écrire un convertisseur de données brutes binaires issues d'un récepteur Motorola Oncore vers un format ASCII compris par le serveur d'UPCT (Fig. 3). La principale subtilité tient au respect scrupuleux des tabulations qui ne peuvent en aucun cas être remplacées par des espaces (dans la plus pure tradition du Makefile ou des formats de données pour programmes Fortran). Nous obtenons ainsi un formatage qui se décrit comme suit :

(5) http://www.jref.com/gallery/showphoto.php/photo/1312
(6) http://www.upct.org

- une série de commentaires commençant par un % de la forme :

```
% Written by GPSManager 18-Mar-2006 15:06:57 (CET)
% Edit at your own risk!
```

- une définition du format et de la norme de projection des données :

```
!Format: DMS 1 WGS 84
!Creation: n
```

- finalement la création d'une trace sous la forme :

```
!T: ACTIVE LOG      width=2 colour=#8b0000  GD310:display=~|Z
    01-Jan-2006 00:00:00    N47 14 53.77    E5 59 19.87 0
    01-Jan-2006 00:00:00    N47 14 53.79    E5 59 19.86 0
!TS:
    01-Jan-2006 00:00:00    N47 14 53.78    E5 59 19.83 0
    01-Jan-2006 00:00:00    N47 14 53.78    E5 59 19.81 0
...
```

où « ... » marque bien sûr la suite de points à inclure dans la trace. Le point fondamental ici est la présence d'une tabulation entre `!T:` et le nom de la trace (ici `ACTIVE LOG`). La date et l'heure ne sont incluses que pour respecter le format, mais n'ont aucune importance dans la validité des données qui sont,elles, fournies sous la forme de degré, minute, seconde et fraction de seconde d'angle.

Nous avons décidé d'automatiser la génération de ces fichiers destinés à UPCT plutôt que d'utiliser la fonction `Import` de `gpsman`, car, d'une part, nous n'arrivions pas à la faire fonctionner, mais surtout nous pouvons inclure ainsi nos propres filtres destinés à épurer les acquisitions de valeurs aberrantes. Nous faisons appel à une séquence de 3 opérations pour transformer nos fichiers au format NMEA en un fichier UPCT :

- Soit un fichier à traiter dont le nom est supposé avoir pour extension `.nmea` tel que obtenu dans un script *bash* par :

```
#!/bin/bash nom=`basename $1 .nmea`
```

- De ce fichier, nous allons extraire uniquement les trames contenant le terme « GPGGA » tel que expliqué auparavant (section 2.2).

- Nous avons constaté pour le moment que cette ligne épure la trace de tout point aberrant. Il nous reste maintenant à convertir ces données de la forme degré-minute-fraction de minutes en un format intelligible par UPCT. Pour cela nous faisons appel à `octave` qui exécute le script suivant en incluant la commande `octave upct.m` où `upct.m` contient :

```
% ici prend en entrée la sortie de go.csh sur un fichier nmea
%function upct(nom)
%eval(['load ',nom,'.upct']);
%eval(['x=',nom,';']);
%eval(['clear ',nom]);
load x
mx=deg2dec(x);
a=dec2deg(mx(:,1));
b=dec2deg(mx(:,2));
d=ones(1,length(a))*99999;
c=[b d' a];
save -ascii nom.degres c
```

où les fonctions `dec2deg.m` et `deg2dec.m` ont respectivement pour objectif de convertir des décimales de minutes en secondes (telles que présentées auparavant, section 2.2), puis de séparer les contributions des degrés, minutes et secondes en trois colonnes distinctes. Cette seconde fonction se présente sous la forme :

```
function sortie=dec2deg(entree)
degre=floor(entree);
reste=(entree-degre)*60;
minute=floor(reste);
seconde=(floor((reste-minute)*60*100))/100;
sortie=[degre minute seconde];
```

L'intérêt de séparer ainsi ces deux fonctions apparaîtra plus tard avec la génération de fichiers KML pour Google Earth : seule la première étape sera alors nécessaire, la seconde étant spécifique à UPCT.

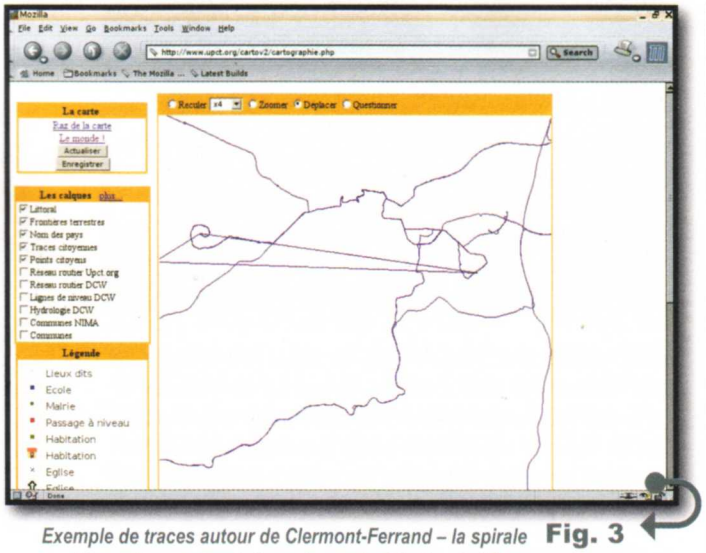
- Finalement, nous convertissons le format de sauvegarde ASCII d'Octave en un format utilisable par UPCT en éliminant les commentaires de la sauvegarde, en ajoutant une altitude nulle pour tous les points et une date permettant de respecter la convention du format de fichier :

```
grep -v ^\# nom.degres | \
sed 's/^ / 01-Jan-2006 00:00:00    N/g' | \
sed 's/$/    0/g' | \
sed 's/99999 /   E/g' > x
```

Ici encore, il faut prendre soin de placer des tabulations avant la date, entre l'heure et le N et avant les 0 et E du second et troisième `sed` respectivement.

- Finalement, nous ajoutons en début de fichier l'en-tête constant d'UPCT présentée plus tôt dans ce paragraphe (`cat entete.upct x > $nom.upct`), et incluons 2 lignes après le début de la trace la commande `!TS:` par un appel à vim (`vi -n -s upct.vi $nom.upct`) où `upct.vi` contient les commandes telles que nous les taperions sous cet éditeur :

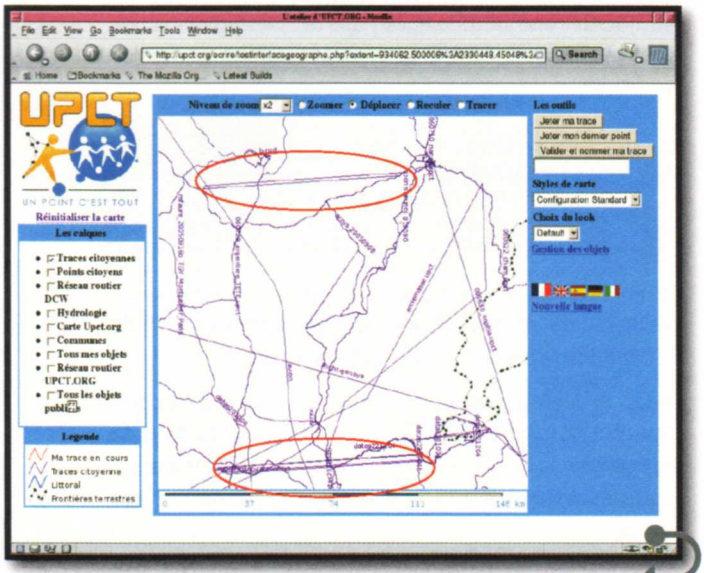
```
:8
dd
p
ZZ
```



Exemple de traces autour de Clermont-Ferrand – la spirale sur la gauche représente la route montant le Puy de Dôme – extraites de la base de données d'upct.org.

Fig. 3

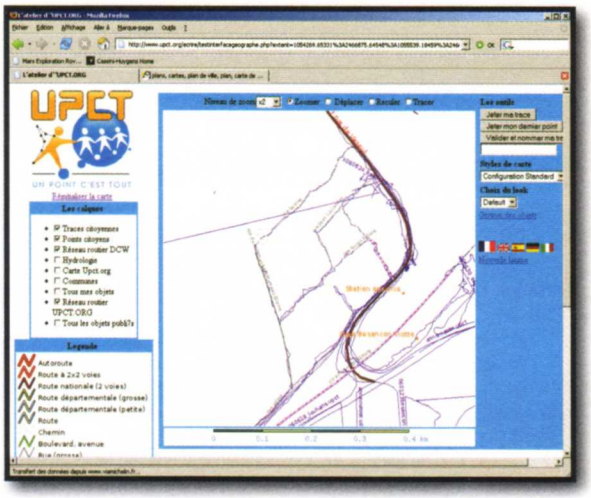
L'épuration des traces de tout point aberrant et la validation des traces sur des outils tels que Google Maps ou Google Earth nous semblent fondamentales avant de partager ces traces avec la communauté d'utilisateurs d'UPCT : en effet, la représentation des traces se fait en reliant les points entre eux. Ainsi, le moindre octet erroné lors du stockage des informations issues du récepteur se traduit par une immense trace peu esthétique et surtout sans réalité physique (Fig. 4).

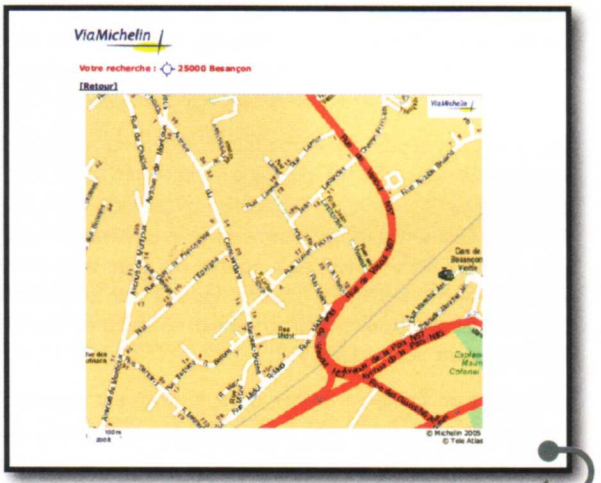


Exemple de trace erronée obtenue lors d'un trajet Besançon-Annemasse au moyen d'un récepteur Motorola Oncore. En deux endroits – indiqués ici par des ellipses rouges – la trace a été translatée pour des raisons que nous ne pouvons expliquer. La continuité de la forme de la trace semble indiquer qu'il s'agit soit d'une erreur lors du stockage des informations, soit lors de la conversion depuis le format binaire.

Fig. 4

Ces points erronés n'influent pas sur la validité du travail de cartographie, puisque nous allons voir plus bas que les traces brutes sont ensuite manuellement vectorisées pour une identification des voies de communication, mais ils diminuent la clarté de la carte UPCT dans son ensemble et rendent le travail de vectorisation plus difficile.





Exemple de vectorisation d'une région autour de Besançon et comparaison avec la base de donnée disponible auprès de www.viamichelin.fr. À titre d'illustration de l'indépendance au système d'exploitation de l'interface de vectorisation, ces captures d'écran ont été obtenues sous MS-Windows.

Fig. 5

Nous constatons donc qu'UPCT se comporte dans un premier temps comme un dépôt de traces GPS avec éventuellement la reconnaissance de quelques sites remarquables sur un trajet (points). Une fois suffisamment de traces accumulées sur un trajet donné afin d'atteindre une statistique suffisante, la seconde étape tient dans la vectorisation et l'identification des traces afin de réaliser une cartographie au sens habituel du terme. Cette seconde

(7) A titre indicatif, on peut considérer qu'en sélectionnant une fois par seconde des points séparés de 30 m, la vectorisation se fait à la vitesse de l'ordre de 100 km/h : il faudra donc sensiblement la même durée pour parcourir une route et la vectoriser sous UPCT.

phase, relativement fastidieuse, est la plus satisfaisante, puisqu'elle conclut le travail d'acquisition des traces. La suite des étapes pour cette seconde tâche se fait en :

- Se connectant à www.upct.org et sélectionnant « Dessiner sur la carte ». Activez « Traces citoyennes » et « Réseau routier UPCT.ORG ».

- Déplacez-vous jusqu'à la zone à vectoriser et zoomez jusqu'à atteindre une barre pleine échelle de 120m (×96, taille de l'unité d'échelle de 30 m).

- Activez le bouton « Tracer » et sélectionnez successivement des points sur la trace à vectoriser (7). Le principe de mise en commun des traces implique que seules les traces suivies plusieurs fois sont vectorisées afin de limiter les erreurs de biais sur la position de la route en cours de vectorisation par une statistique suffisante. Une fois le segment de trace vectorisé, introduisez le nom identifiant ce segment dans l'emplacement prévu à cet effet sous « Valider et nommer ma trace », puis concluez en cliquant sur ce bouton. La trace vectorisée est alors placée dans la liste d'objets publiables sur la carte.

- Il reste alors à définir les propriétés pour la publication de ces données : cliquez sur « Gestion des objets » pour visualiser la liste des objets en attente de publication.

- Pour chaque objet en attente de publication, sélectionnez le Type de la trace *qu'il faut valider après sélection* en cliquant sur « Enregistrer ». Le poids est indicatif de l'importance de cet axe de transport pour la population locale.

- La publication s'achève par « Demander la publication », qui met en attente la trace vectorisée d'une validation de l'administrateur du site.

UPCT ne fournit qu'une interface utilisable sur le web pour vectoriser les traces. L'avantage de cette solution est de ne pas nécessiter l'installation d'un logiciel sur la machine locale pour ajouter ses traces, d'autant plus que l'interface est rapide et souple d'emploi. Les inconvénients majeurs de cette solution sont :

- Les risques de lenteur associés aux connexions réseau. Alors que l'interface a été pensée en vue d'une connexion bas débit avec un minimum d'informations transférées à chaque ajout de point lors de la vectorisation, nous avons observé des problèmes lors de la connexion depuis un site à bande passante élevée (Renater), mais derrière une passerelle particulièrement mal configurée qui nécessite quelques secondes pour acquitter chaque nouvelle connexion (proxy de la zone étudiante de l'université de Franche-Comté à Besançon). Étant donné que l'ajout d'un nouveau point nécessite une nouvelle connexion, la séquence de tracé d'un nouveau chemin est tellement lente qu'il en devient inutilisable.

- L'impossibilité de travailler en l'absence de connexion réseau tel que dans le train par exemple, qui serait pourtant un endroit approprié pour une telle activité.

Au contraire Openstreetmap propose plusieurs logiciels à installer sur sa machine locale pour la vectorisation des traces hors connexion réseau avant d'envoyer le résultat de cette opération vers le serveur (http://wiki.openstreetmap.org/index.php/Compare_editors). Les traces sont ici superposées aux images satellites de Landsat publiquement accessibles, de moins bonne résolution en milieu urbain que les images disponibles sur les serveurs de Google (Maps et Earth), mais pouvant fournir une aide à la vectorisation si nécessaire. Le format requis par OpenStreetMap est de type XML – GPX – fourni par GPSBabel, que nous n'avons pas exploré de façon approfondie pour le moment. L'ensemble des traces vectorisées peut ensuite être téléchargé pour une utilisation locale auprès de http://wiki.openstreetmap.org/index.php/Planet.osm.

L'intérêt de ces projets est visible dans la liste de disponibilité de bases de données de routes dans le monde : alors que librement disponibles aux États-Unis, ces bases de données ne sont disponibles que pour des sommes exorbitantes en Europe (à l'exception du Danemark) et, en tout cas, inaccessibles à l'utilisateur individuel [4, p. 397].

4 AFFICHAGE DES POINTS SUR GOOGLE EARTH

Suite à la publication de notre article précédent [1], un lecteur – Benoît Rolland – nous a informé de la capacité de Google Earth à insérer des points fournis par un utilisateur dans les cartes affichées par le logiciel. Nous nous sommes donc proposés là encore d'écrire un convertisseur de formats de données, binaire pour le récepteur Motorola Oncore, et NMEA pour les autres récepteurs, vers le format *kml* compris par Google Earth. Contrairement au cas précédent, le formatage de ce fichier ASCII est très libre et son format parfaitement décrit à http://www.keyhole.com/kml/kml_tut.html qui vient d'être remis à jour à http://earth.google.com/kml/kml_21tutorial.html. Nous nous inspirerons fortement de ces exemples pour générer nos propres fichiers :

- Un en-tête que nous garderons constant pour tous les fichiers :

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">
<Placemark>
  <description>Trajet Besancon-Chamonix, 23-06-2006</
description>
  <name>Absolute Extruded</name>
```

dont nous pourrons éventuellement prendre soin de modifier la description du document.

- Une définition de point de vue et de la direction de visée initiale après chargement de la trace :

```
<LookAt>
    <longitude>6.02194166666667</longitude>
    <latitude>47.24624</latitude>
    <range>4451.842204068102</range>
    <tilt>44.61038665812578</tilt>
    <heading>-125.7518698668815</heading>
</LookAt>
```

Généralement, nous nous contentons ici de modifier de façon automatique la latitude et longitude de départ pour conserver la direction de visée fixe.

- Un certain nombre d'options de tracé du polygone représentant la trace sur les données issues de Google Earth :

```
<visibility>1</visibility>
  <open>0</open>
  <Style>
     <LineStyle> <color>ff00ffff</color> </LineStyle>
     <PolyStyle> <color>7f00ff00</color> </PolyStyle>
  </Style>
  <LineString>
  <extrude>1</extrude> <tessellate>1</tessellate>
  <altitudeMode>absolute</altitudeMode>
```

- Finalement le cœur du fichier : une suite de points de la forme {longitude,latitude,altitude} fournis sous forme décimale :

```
<coordinates>
6.02194166666667,47.24624,100
6.02194333333333,47.2462416666667,100
6.02194333333333,47.2462433333333,100
...
</coordinates> </LineString> </Placemark> </kml>
```

où ... indique bien entendu la suite des points à insérer dans la trace.

L'insertion des traces GPS dans Google Earth est d'autant plus pertinente depuis la mi-juin que Google vient de fournir à la communauté d'utilisateurs de GNU/Linux une version de son logiciel fonctionnant sur ce système d'exploitation http://earth.google.com/download-earth.html.

Les résultats sont impressionnants de précision, résultats qu'il faut attribuer à l'excellente qualité des signaux GPS, d'une part, et aux concepteurs de Google Earth (originellement KeyHole), d'autre part, pour leur alignement des images satellites sur les coordonnées GPS.

Quelques exemples d'images issues de Google Earth présentant l'intérêt en présence de reliefs remarquables de visualiser la troisième dimension. Sur toutes les figures, la trace jaune correspond à la trace acquise par notre montage et incluse par conversion au format KML. De haut en bas : arrivée dans la vallée de Chamonix par le côté suisse ; le trajet depuis Besançon vers le lac Léman pour ensuite atteindre Chamonix, mettant en évidence deux points d'altitudes aberrantes ; randonnées dans la vallée de Chamonix (chaque trace est ici encodée par une couleur différente) ; trajet sur les puys près de Clermont Ferrand.

Fig. 6

(8) http://bbs.keyhole.com/ubb/postlist.php/Cat/0/Board/EarthExternalData
(9) http://www.eos.ubc.ca/~rich/map.html a été testé sous Matlab, mais ne semble pas fonctionner en l'état sous Octave.

5 AFFICHAGE DE CARTES SUR GOOGLE EARTH

Google Earth propose, en plus d'ajouter des polygones et des points à ses cartes, de superposer en transparence des images au format JPEG. Des applications tout à fait étonnantes de ces options ont été réalisées par les utilisateurs de Google Earth, notamment pour la visualisation géographique de données accumulées et disponibles par internet (voir dans Google Earth le forum *Dynamic Data Layers* (8).

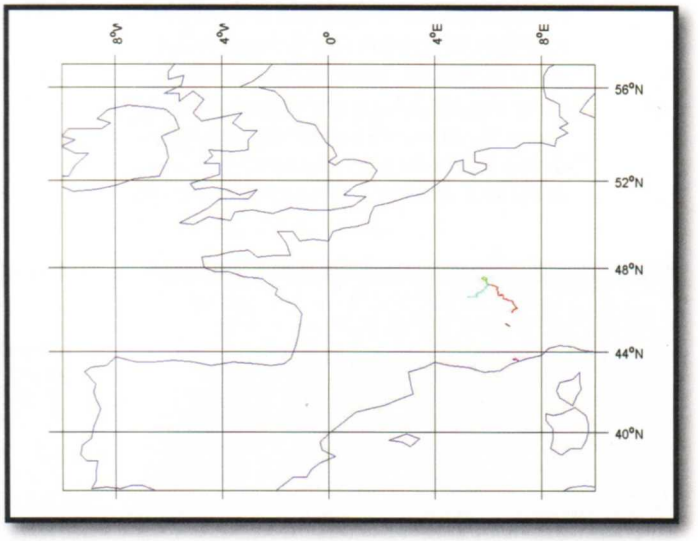
Superposer des images de cartes à Google Earth nécessite d'utiliser une projection appropriée, dite « Projection Cylindrique » (ou Plate-Carrée [5]). Nous avons utilisé une *toolbox* gratuitement disponible pour Matlab, `m_map` (9) pour inclure nos traces GPS dans une carte d'une partie de l'Europe de l'Ouest et projeter l'image résultante sur Google Earth (Fig. 7). L'intérêt est ici d'avoir accès à toute la puissance en termes de calculs matriciels et en traitement du signal de Matlab pour extraire les informations pertinentes de grandes quantités de points, par exemple, pour un réseau de capteurs géographiquement distribués. L'exemple proposé ici se borne à valider la capacité de plaquer une carte convenablement projetée, incluant nos propres points, sur le globe de Google Earth.

Nous utilisons les commandes suivantes sous Matlab pour tracer la carte :

```
path(path,'c:\matlab\m_map');
m_proj('Miller Cylindrical','longitudes',[-10
10],'latitudes',[37 57]);
m_coast; axis square;              % trace du fond de carte
load 060622_cham.dat;              % fichier long,lat,alti
issu du grep
s=deg2dec(X060622_cham);           % conversion en format
decimal
u=diff(s(:,1)); v=diff(s(:,2));    % vecteur vitesse
lat=s(1:length(s)-1,2); lon=s(1:length(s)-1,1);
hold on;m_quiver(lon,lat,u,v,'r'); % iterer pour chaque
nouvelle trace
```

Nous utilisons l'attribut `color` pour afficher l'image en transparence sur le globe de Google Earth au moyen de la balise `Icon` de la façon suivante :

```
<color> a9ffffff </color>
  <Icon>
    <href>http://mmyotte.free.fr/france_mmpap2.jpg</href>
  </Icon>
  <LatLonBox id="khLatLonBox751">
    <north>59.57</north>
    <south>34.333</south>
    <east>12.70</east>
    <west>-13.7666</west>
    <rotation>0</rotation>
  </LatLonBox>
```



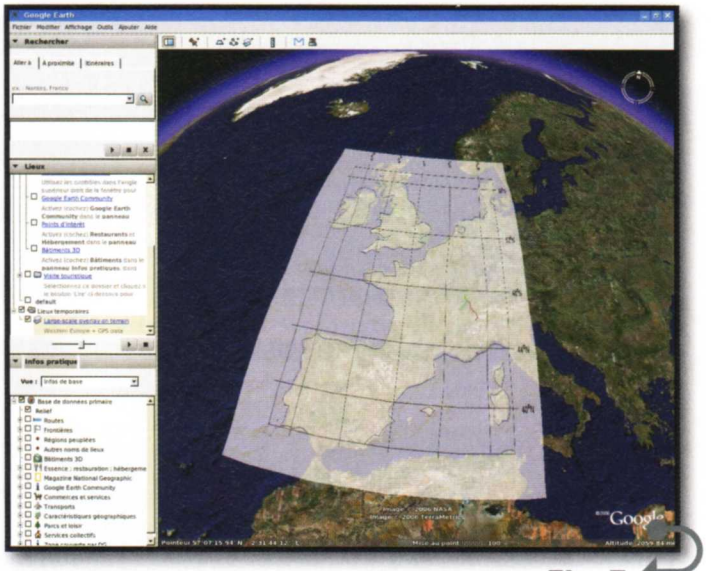


Fig. 7

Haut : image issue de la toolbox M_MAP de Matlab en projection cylindrique d'une partie de l'Europe de l'Ouest sur laquelle ont été ajoutées quelques traces GPS. Bas : projection de cette image sur le globe de Google Earth.

6 AFFICHAGE DES POINTS SUR GOOGLE MAPS

Contrairement à Google Earth qui nécessite l'installation d'un logiciel propriétaire sur sa machine locale, Google Maps fournit une interface de programmation (API) accessible en javascript : la plupart des navigateurs web sont ainsi capables de tracer une carte issue du serveur Google Maps et d'afficher la vue aérienne ainsi que la carte routière associée sur l'ordinateur local. L'inconvénient majeur de Google Maps par rapport à Google Earth est de ne pas fournir la 3ème dimension qu'est l'altitude des points, information dont nous pourrons nous passer pour de nombreuses applications.

Nous partirons de l'hypothèse que nos points sont stockés au format kml reconnu par Google Earth. La discussion qui suit sera ensuite étendue à des séries de points stockés en format NMEA. Le premier problème à résoudre est l'accès au serveur Google Maps et le rapatriement d'une carte aux coordonnées sélectionnées, munie d'une interface utilisateur minimale (changement du type d'affichage entre route et photographies aériennes, translation et homothétie de la carte). Le second problème est l'ajout de nos propres points à ces cartes.

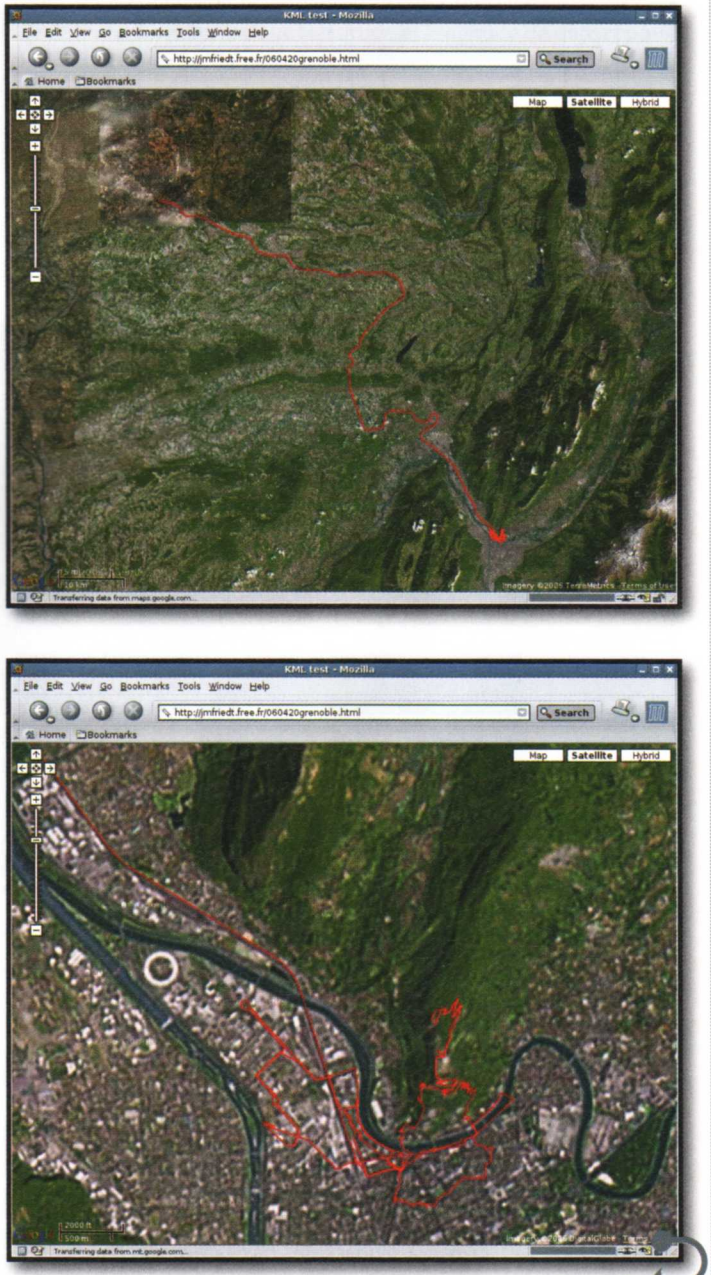


Fig. 8

Page web générée manuellement depuis un fichier contenant une série de couples {latitude,longitude} afin d'afficher ces points sur la base de données de vues aériennes Google Maps. En haut, le trajet en train de Lyon à Grenoble, en bas un zoom sur Grenoble et les trajets à pied qui y ont été effectués. L'information d'élévation n'est pas disponible dans Google Maps.

6.1 L'INTERFACE JAVASCRIPT DE GOOGLE MAPS

Afin de nous familiariser avec l'API de Google Maps, nous allons générer manuellement une page web faisant appel au serveur et ajoutant sur les cartes ainsi rapatriées nos points tracés sous forme de polygones (Fig. 8).

Google requiert, afin de permettre l'affichage de ses cartes sur une page web accessible au public, de s'enregistrer afin d'obtenir une clé associée à un domaine spécifique (dans l'exemple qui va suivre, la clé obtenue est valide pour l'ensemble du domaine **jmfriedt.free.fr** – notez que pour une utilisation locale telle qu'une page web stockée sur le disque dur de la machine exécutant l'interpréteur HTML, n'importe quelle clé convient) : la page pour ce faire est disponible à **http://www.google.com/apis/maps/signup.html**. À l'issue de cet enregistrement, nous obtenons un exemple de code qui va servir de base aux développements ultérieurs. Par exemple, pour générer une clé pour l'ensemble du site **friedtj.free.fr**, nous obtenons comme réponse :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html;
charset=utf-8"/>
    <title>Google Maps JavaScript API Example</title>
    <script src="http://maps.google.com/maps?file=api&amp;v=2&
amp;key=ABQIAAAAQ5iKlBqfwApuWu7aIzUiPRTELJePVJ4bAZX18VT1uNz_yt
5gwxTQjL4ngbxeSIY1ZypSj6avKm_2Ig"
      type="text/javascript"></script>
    <script type="text/javascript">

    //<![CDATA[

    function load() {
      if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map"));
        map.setCenter(new GLatLng(37.4419, -122.1419), 13);
      }
    }

    //]]>
    </script>
  </head>
  <body onload="load()" onunload="GUnload()">
    <div id="map" style="width: 500px; height: 300px"></div>
  </body>
</html>
```

Nous constatons dans cet exemple que nous pouvons :

- nous connecter au serveur Google Maps et en rapatrier une carte centrée sur des coordonnées fournies sous forme de latitude/longitude ;

- sélectionner un mode d'affichage et un taux de grossissement parmi un ensemble de valeurs prédéfinies ;
- ajouter à la page un certain nombre de commandes telles que la translation et l'homothétie de la carte ou le changement du type d'affichage (carte routière ou vue aérienne).

L'API Google Maps est décrite de façon très générale à http://www.google.com/apis/maps/documentation/, mais les fonctions pour l'ajout de nos propres points sont issues de lectures de codes source sur le web et notamment http://www.obviously.com/gis/gpx_loader.html.

Avant d'automatiser la génération des pages web au moyen de scripts PHP, nous allons nous familiariser avec la syntaxe en générant une page manuellement. Partant de l'en-tête précédent, nous désirons ajouter à la carte fournie par le serveur Google Maps un polygone incluant nos propres traces GPS. Pour ce faire, nous déclarons un tableau de points : `var points = new Array();` que nous allons remplir petit à petit des coordonnées des points acquis par `points.push(new GPoint(5.04382,45.6604383333333));` pour par exemple ajouter le point de coordonnées {5,044;45.660}.

L'idée la plus simple consistant à accumuler tous les points d'une trace en un seul polygone ne fonctionne pas, car l'affichage de cartes devient excessivement lent au point de s'interrompre lorsqu'un polygone contient plus d'une centaine de points, le maximum admissible avant d'interrompre l'affichage étant de l'ordre du millier de segments.

Il nous faut donc découper notre trace en plusieurs polygones que nous avons choisis de taille 60 points, ce qui revient à 1 minute de parcours par polygone : cette opération a été réalisée manuellement dans un premier temps pour cet exemple dont le but est de valider le concept, et nous le verrons inclus dans les scripts PHP de génération automatique de pages web plus bas.

La création du polygone se conclut par son affichage : `map.addOverlay(new GPolyline(points, "#FF0000",2,.75));`, puis la réinitialisation de la variable en un tableau vide avant de réitérer le processus jusqu'à la fin de la trace : `points=[]; points.push(new GPoint(5.125925,45.63738)); ....`

Une page se génère facilement à partir d'instructions `sed`, mais nous voyons déjà apparaître plusieurs problèmes qui vont être résolus plus tard :

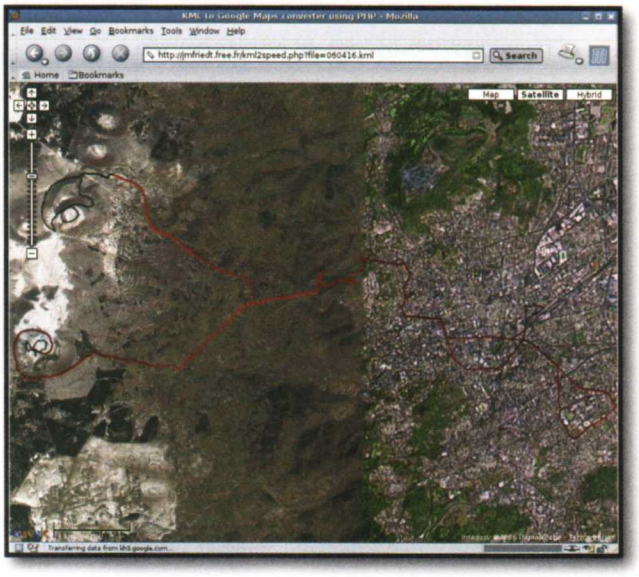
- Les pages ainsi générées sont énormes. Un parcours d'un peu plus de 5 heures à raison d'un point par seconde génère une page de 1,1 MB (qui s'affiche malgré tout raisonnablement vite dans Google Maps). Pour rapidement citer les causes de cette taille excessive : l'absence de fonctions qui réduiraient la taille de la page et l'excès de décimales dans les coordonnées.
- Nous désirons pouvoir générer les pages au vol via une interface web sans imposer à l'utilisateur l'installation d'outils avec lesquels il n'est pas nécessairement familier.

En conclusion, il nous faut retenir les commandes importantes qui sont :

- La définition de l'objet contenant la carte : `var map = new GMap(document.getElementById("map"));`.
- La définition de la visualisation des images aériennes au lieu du mode par défaut qui est le tracé de routes : `map.setMapType(G_SATELLITE_TYPE);`.
- l'ajout des commandes permettant à l'utilisateur d'interagir avec les cartes : `map.addControl(new GLargeMapControl()); map.addControl(new GMapTypeControl()); map.addControl(new GScaleControl());` ,
- `map.centerAndZoom(new GPoint(6.0, 47.0), 5);` qui définit la position du centre de l'objet `map`, ici le point de longitude et latitude {6,0;47.0} avec un grossissement de 5. En pratique, nous remplacerons ces valeurs arbitraires par la première coordonnée pertinente de la trace (point de départ de la trace).

6.2 L'AJOUT AUTOMATIQUE DES TRACES AU FORMAT KML À GOOGLE MAPS

Nous avons séquentiellement abordé deux approches à l'ajout de traces aux cartes : la génération de pages HTML contenant les points et les commandes associées à leur affichage sous forme de polygone selon l'API Google Maps (voir section précédente), et l'utilisation d'un script PHP chargé de lire un fichier disponible sur le web, contenant des données au format KML ou NMEA, et de générer la page HTML au vol. La seconde solution nous semble plus souple, mais nécessite un serveur supportant le PHP.



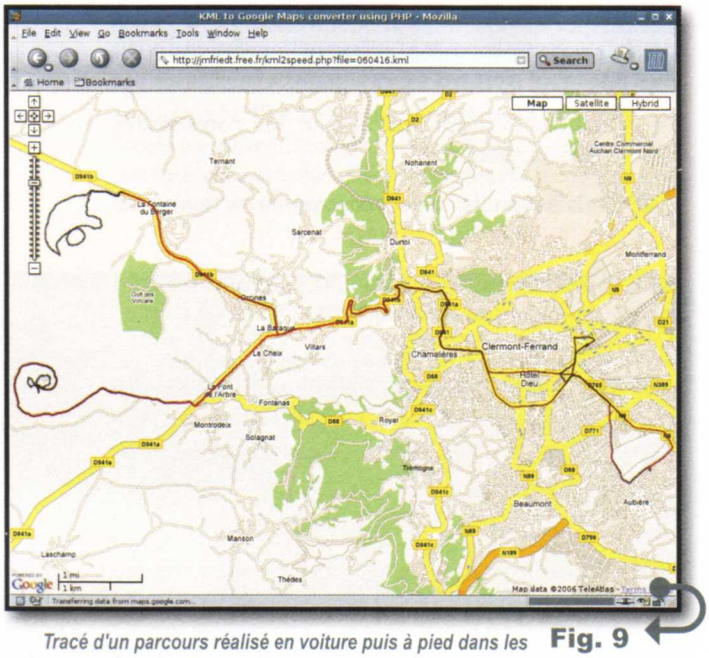


Fig. 9

Tracé d'un parcours réalisé en voiture puis à pied dans les puys près de Clermont-Ferrand : nous représentons ici par une évolution de couleurs la vitesse de déplacement du récepteur GPS, du plus foncé (le plus lent) au plus rapide (rouge). Google Maps permet soit de représenter la vue aérienne du parcours (page précédente), soit la route correspondante (ici). La moyenne de vitesse a été effectuée sur 60 points (1 minute), taille de chaque polygone tracé pour lequel une même couleur est définie.

Ayant présenté auparavant la génération de fichiers au format KML pour Google Maps qui se résume principalement en l'extraction depuis le fichier brut d'enregistrement des trames GPS (binaire pour Motorola Oncore, NMEA sinon), il nous a semblé simple de nous familiariser avec un script PHP se contentant de prendre ces séries de coordonnées et de les inclure dans l'API Google Maps pour affichage dans une interface web. Nous verrons plus tard comment améliorer ce script PHP pour lire directement des fichiers au format NMEA.

Le PHP consiste en un langage interprété du côté du serveur afin de générer dynamiquement une page : nous conservons les en-têtes imposés par l'API Google Maps tel que décrit auparavant, mais au lieu d'inclure des points au moyen de `sed`, puis d'exporter la page résultante sur un serveur web, nous allons cette fois passer comme argument au script un nom de page contenant un fichier au format KML qui sera utilisé pour générer la page web. La partie dynamique de la page est incluse entre les balises `<?php ... ?>`. Le passage de paramètre se fait dans l'URL appelant le script : par exemple **http://jmfriedt.free.fr/kml2maps.php?file=monfichier.kml** appelle le script PHP `kml2maps.php` disponible sur le serveur **jmfriedt.free.fr** en passant comme argument une variable `file` contenant le nom du fichier à traiter (supposé ici présent sur le même serveur, mais qui pourrait aussi être une URL vers un fichier disponible sur un autre serveur web). Cet argument est récupéré côté PHP dans la variable nommée `$file`. Notre script PHP doit effectuer deux tâches :

- dans un premier temps, identifier la première coordonnée pertinente du fichier KML afin de définir sur quel point centrer la carte requise auprès du serveur Google Maps (argument de la méthode `centerAndZoom()`), notamment en éliminant l'en-tête du fichier KML : `do $buffer=fgets($handle, 4096); while (strpos($buffer, "coordinates")===false);` qui signifie que l'on élimine l'en-tête du fichier (adressé par la variable `$handle`) jusqu'à atteindre la balise `coordinates`.

- Une fois la zone des coordonnées atteinte dans le fichier KML, il nous faut les récupérer et les transformer en un format acceptable pour la page web appelant le serveur Google Maps, et ce, jusqu'à atteindre une nouvelle fois la balise `coordinates` indiquant la fin de la trace : `while (strpos($buffer,"coordinates")===false) { $buffer = fgets($handle, 4096); list($lon, $lat, $alt)=explode(",", $buffer); }` et en plus périodiquement tracer le polygone et réinitialiser le tableau de points afin de ne pas dépasser la taille limite au-delà de laquelle l'affichage perd de sa fluidité. Ainsi cette fonction se résume en :

```
while (strpos($buffer,"coordinates")===false) {
  $n++;
  if ( $n == 100) {
     $n=0;
     printf("map.addOverlay(new GPolyline(points, \"#FF0000\",
2, .75));
     points=[];");
    }
  list($lon, $lat, $alt)=explode(",",$buffer);
  print "points.push(new GPoint($lon,$lat));";
  $buffer = fgets($handle, 4096);
}
```

Le script PHP dans son ensemble est proposé dans le tableau 5.

TABLEAU 5

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:v="urn:schemas-microsoft-com:vml">
<head>
<title>KML to Google Maps converter using PHP </title>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
<meta name="description" content="test - powered by Google
Maps" />
<script src="http://maps.google.com/maps?file=api&amp;v=2&am
p;
key=ABQIAAAAQ5iKlBqfwApuWu7aIzUiPRRsafyVTyOwfc9A1qQgm5QZxWMv
mxQ
```

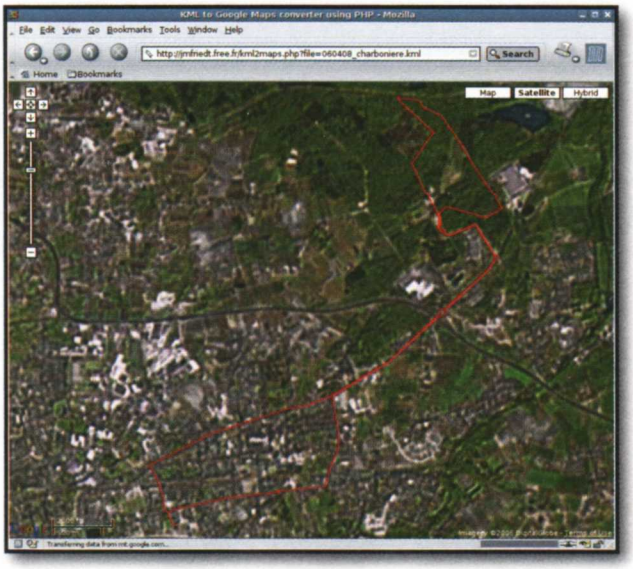
```
R2gKn15-3mqugeybg6Yf1Za_yIQ"
type="text/javascript">
</script>

<style type="text/css">
            v\:* {
            behavior:url(#default#VML);
            }
            html, body, #map
            {
              width: 100%;
              height: 100%;
            }
          body            {
            margin-top: 0px;
            margin-right: 0px;
            margin-left: 0px;
            margin-bottom: 0px;
            }

</style>
</head>

<body>
<div id="map"> </div>

<script type="text/javascript">
//<![CDATA[            // check for compatibility


if (GBrowserIsCompatible()) { // call the info window opener

  function makeOpenerCaller(i) {
     return function() {showMarkerInfo(i);};} // open an info
window
  function showMarkerInfo(i) {
     markers[i].openInfoWindowHtml(infoHtmls[i]);
  }               // create the map

     var map = new GMap(document.getElementById("map"));

     map.setMapType(G_SATELLITE_TYPE);
     map.addControl(new GLargeMapControl());
     map.addControl(new GMapTypeControl());
     map.addControl(new GScaleControl());

<?php
if ($file=="")  {$file="060416.kml";}
$handle = fopen($file, "r");   // ou http://jmfriedt.free.fr

 if ($handle) {
       do {$buffer = fgets($handle, 4096);}
          while (strpos($buffer,"coordinates")===false);
       $buffer = fgets($handle, 4096);
       list($lon, $lat, $alt)=explode(",",$buffer);

       print "map.centerAndZoom(new GPoint($lon, $lat), 5)";
       fclose($handle);
 }
?>

 if (window.attachEvent) {
    window.attachEvent("onresize", function() {this.map.
onResize()} );
 } else {
    window.addEventListener("resize",function() {this.map.
onResize()},false);
 }               // add a polyline overlay
 var points = new Array();

<?php
 if ($file=="")  {$file="060416.kml";}
 $handle = fopen($file, "r");        // au lieu de http://
jmfriedt.free.fr ...

 if ($handle) {
      do {$buffer = fgets($handle, 4096);}
         while (strpos($buffer,"coordinates")===false);
      $buffer = fgets($handle, 4096);
      $n = 0;
      while (strpos($buffer,"coordinates")===false) {
       $n++;
       if ( $n == 100) {
         $n=0;
         printf("map.addOverlay(new GPolyline(points,
\"#FF0000\", 2, .75));points=[];");
       }
       list($lon, $lat, $alt)=explode(",",$buffer);
       print "points.push(new GPoint($lon,$lat));";
       $buffer = fgets($handle, 4096);
      }
      fclose($handle);
 }
?>
   map.addOverlay(new GPolyline(points, "#FF0000", 2, .75)); }
 else {
   document.getElementById("quicklinks").innerHTML ="web
browser not compatible"
 }
//]]>
</script>
</body>
</html>
```

Script PHP pour la génération de pages web incluant les points d'un fichier KML pour superposition aux images aériennes issues du serveur Google Maps.

Il apparaît que nous pouvons ajouter des fonctionnalités à ce programme en remplaçant la couleur constante des polygones successifs par une couleur associée à une quantité physique mesurée simultanément à l'acquisition des trames GPS (par exemple notre microcontrôleur est équipé d'un capteur de température LM35 et est capable de compter les impulsions d'un compteur Geiger pour la détection de rayonnements ionisants). À titre d'illustration, nous proposons d'encoder la vitesse du véhicule portant le récepteur GPS, moyennée sur une minute, dans la couleur de chaque polygone tel que présenté sur la figure 9. Cette fonctionnalité a été ajoutée par :

```
do {
    list($lon, $lat, $alt)=explode(",",$buffer);
    $vit+=floor(sqrt(($lono-$lon)*($lono-$lon)+($lato-
$lat)*($lato-$lat))*20000);
    print "points.push(new GPoint($lon,$lat));";
    $i++;
    if ($i==120) {
//     if ($vit > 255) $vit=255;
       $couleur=sprintf("%02X0000",$vit);
```

```
  // will ignore unnecessary 0 in the color code
      print "map.addOverlay(new GPolyline(points,\
"#$couleur\",2,.75));\n";
      print "points=[];";
      print "points.push(new GPoint($lon,$lat));";
      $i=0; $vit=0;
    }
    $lono=$lon;
    $lato=$lat;
    $buffer = fgets($handle, 4096);
  } while (strpos($buffer,"coordinates")===false);
  fclose($handle);
 }
```



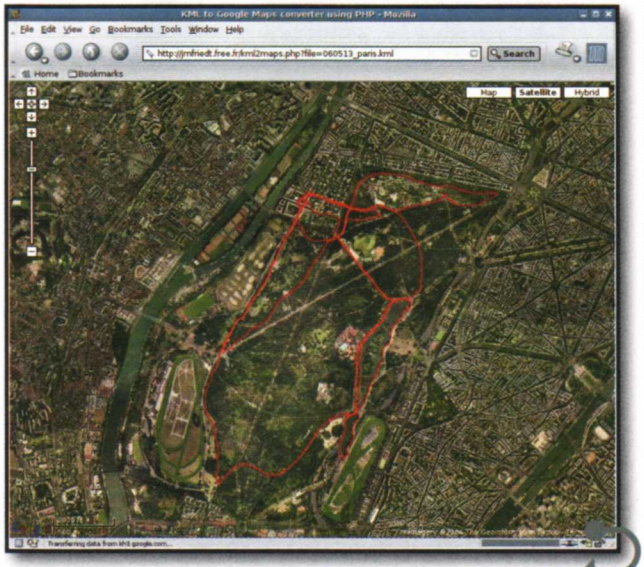


Fig. 10

Exemple de deux tracés issus de fichiers KML disponibles sur le web pour Google Earth et ici convertis de façon transparente pour l'utilisateur pour un affichage en 2D sur Google Maps (ne nécessitant donc pas une installation locale d'un logiciel dédié tel que Google Earth). En haut, le parc de Charbonnière près d'Orléans, en bas le parc de Longchamps près de Paris. Notez que le grossissement est le même, mais que la vue de Paris est disponible en haute résolution.

qui se résume donc à mémoriser le point précédent et à calculer la norme du vecteur vitesse pour chaque nouveau point, valeur qui après *moyennage* sur 1 minute est utilisée pour sélectionner la couleur du polygone. Le reste du script présenté auparavant (tableau 5) reste le même et seule la fonction principale de génération de la page web a été modifiée.

6.3 L'AJOUT AUTOMATIQUE DES TRACES AU FORMAT NMEA À GOOGLE MAPS

TABLEAU 6

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:
schemas-microsoft-com:vml">
<head>
<title>NMEA to Google Maps converter using PHP </title>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
<meta name="description" content="test - powered by Google
Maps" />

<script src="http://maps.google.com/maps?file=api&amp;v=2&amp;
key=ABQIAAAA5Q7z5crwd4oEknmiapx7sBTOx7y3ojUZHr6Em7fwe7qgiECJ2h
RRd1mQtnWmN97u8JRq0x8jxwTz8w"
  type="text/javascript">
</script>

<style type="text/css">
            v\:* {
            behavior:url(#default#VML);
            }
            html, body, #map
            {
              width: 100%;
              height: 100%;
            }
          body            {
            margin-top: 0px;
            margin-right: 0px;
            margin-left: 0px;
            margin-bottom: 0px;
            }
</style>
</head>

<body>
<div id="map"> </div>

<script type="text/javascript">
//<![CDATA[            // check for compatibility


if (GBrowserIsCompatible()) { // call info window opener for
the given index

  function makeOpenerCaller(i) {
     return function() { showMarkerInfo(i); };
  }               // open an info window
```


(10) Une façon de cacher ce problème sur Firefox est décrite à http://www.itchyhands.com/2006/01/07/fix-firefox-unresponsive-script-warnings : accédez à l'URL about:config et modifiez la variable dom.max_script_run_time à une valeur de l'ordre de 20. Cette procédure ne semble pas fonctionner avec Mozilla.

```
  function showMarkerInfo(i) {
    markers[i].openInfoWindowHtml(infoHtmls[i]);
  }                // create the map

    var map = new GMap(document.getElementById("map"));

    map.setMapType(G_SATELLITE_TYPE);
    map.addControl(new GLargeMapControl());
    map.addControl(new GMapTypeControl());
    map.addControl(new GScaleControl());

    if (window.attachEvent) {
      window.attachEvent("onresize", function() {this.map.
onResize()} );
    } else {
      window.addEventListener("resize",function() {this.map.
onResize()},false);
    }                // add a polyline overlay
    var points = new Array();
    // fonctions pour reduire la taille de la page HTML
generee
    function p(lon,lat) {points.push(new GPoint(lon,lat));}
    function o() {
        map.addOverlay(new GPolyline(points, "#FF0000", 2,
.75));points=[];}

<?php
if ($file=="") {$file="test.nmea";}
$handle = fopen($file, "r") or exit("Unable to open file!");
$n = 0;
$commence=0;
$france=1;

  do {
    do {$buffer = fgets($handle, 4096);}
       while ((strpos($buffer,'GPGGA')===false) &&
(!feof($handle)));
    if (!feof($handle)) {
      $n++;
      if ( $n == 100) {
         $n=0;
     if ($commence == 1) {
        printf("o();\n");
        $commence=0; // ne PAS faire de Polyline sans point
        }
      }
      list($rien1,$heure,$lat,$N,$lon,$E,$pasalt)=explode(",
",$buffer);
   if ((preg_match("/[0-9]{4}\.[0-9]{4}/i",$lat)) &&
       (preg_match("/[0-9]{4}\.[0-9]{4}/i",$lon))) {
      $lati=floor($lat/100);
      $loni=floor($lon/100);
      $lati=$lati+floor( (($lat/100)-$lati)/60*100*1000000)/
1000000;
      $loni=$loni+floor( (($lon/100)-$loni)/60*100*1000000)/
1000000;
      if (($lati > 40 )&&($lati<90)&&($loni>0)&&($loni<60)&&
$france!=0) {
          print "map.centerAndZoom(new GPoint($loni,
$lati),5);";
          $france=0;}
      if ($lati > 40 ) {
          $commence=1; // on a commence a tracer la courbe
          print "p($loni,$lati);";}
      }
    }
 } while (!feof($handle));
 printf("map.addOverlay(new GPolyline(points, \"#FF0000\", 2,
.75));");
 fclose($handle);
?>

   map.addOverlay(new GPolyline(points, "#FF0000", 2, .75));
   } else {
   document.getElementById("quicklinks").innerHTML="web
browser not compatible"
   }
//]]>
</script>

</body>
</html>
```

Script PHP pour la génération de pages web incluant les points d'un fichier NMEA pour superposition aux images aériennes issues du serveur Google Maps.

Finalement, la solution la plus simple pour un utilisateur est de pouvoir simplement mettre ses trames NMEA sur une page web et d'utiliser un convertisseur en PHP qui se charge de lire le fichier, le transformer au format approprié et envoyer la page résultante au client qui interprète alors le script javascript pour superposer aux cartes de Google Maps le trajet parcouru. Cette solution a cependant un inconvénient majeur en termes de bande passante : le fichier NMEA contient beaucoup d'informations inutiles pour l'application qui nous intéresse ici, et le fichier placé sur le web pour traitement est plus volumineux que nécessaire. La première tâche du script PHP sera donc de ne conserver que les trames GPGGA qui nous concernent. Nous utiliserons dans ce nouveau script PHP (tableau 6) les mêmes expressions régulières que nous avions utilisées auparavant dans les scripts shell, accessibles grâce à la fonction `preg_match()` : bien que le script PHP soit très ressemblant à celui chargé de traiter les fichiers KML, nous le proposons à nouveau ici dans son intégrité par souci de clarté. La structure de la page a changé, puisque nous ne faisons maintenant plus qu'un unique appel à PHP et l'ouverture de fichier cible associé, et la distinction entre la phase recherche de la première coordonnée (afin de centrer la carte à afficher sur la trace) et la suite du traitement du fichier s'obtient en définissant un drapeau (`france=0;` dans cet exemple). Nous avons ici dû ajouter la vérification de la validité des informations de latitude-longitude (hypothèse sur le nombre de décimales présentes) et la conversion en fractions de degrés.

Une capture d'écran du résultat de ce script est présentée sur la figure 11, page suivante.

Un second point lié à la bande passante concerne la taille de la page générée : un trajet long peut facilement générer une page de quelques mégaoctets qui, sur une connexion réseau un peu lente, engendre un *timeout* de la part du client web (10). Il est donc fondamental de viser à générer une page aussi petite que possible. Nous avons adopté 3 solutions :

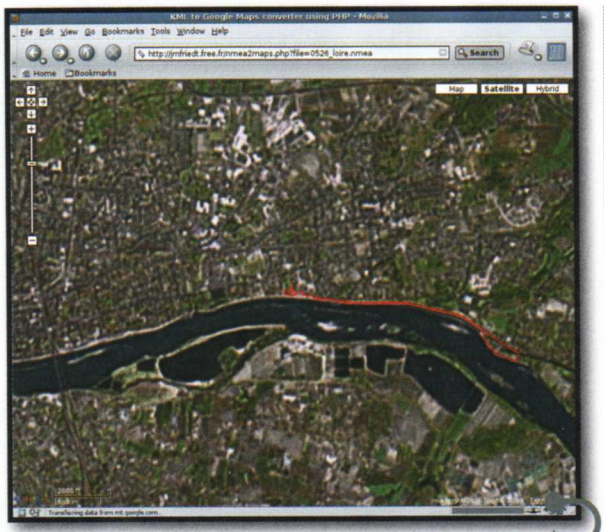


Fig. 11

Exemple d'une conversion d'un fichier brut NMEA mis à disposition sur le web pour un affichage sur Google Maps. Saint Jean de Braye et Combleux près de Orléans.

1 Utilisation de fonctions aux noms très courts appelées pour chaque nouveau point ajouté à la liste d'éléments de chaque polygone, par exemple par `function p(lon,lat) {points.push(new GPoint(lon,lat));}` et `function o() {map.addOverlay(new GPolyline(points, "#FF0000", 2, .75));points=;}` qui remplace donc respectivement 23 caractères et 64 par 1 seul, résultant en une économie considérable de bande passante.

2 Réduction du nombre de décimales pour ne garder qu'une précision significative. Par défaut, PHP nous fournit le résultat en calcul flottant d'un quotient avec 13 décimales. Un degré de latitude correspond à 40000/360=111 km. Donc, pour avoir des points définis avec une résolution de 10 cm (bien en deçà de la variance sur la position des récepteurs utilisés ici), nous nous contenterons de ne garder que 6 décimales par l'utilisation de la commande `$lat=floor($lat*1000000)/1000000;`.

3 Enfin, une amélioration que nous n'avons pas pris le temps d'implémenter du fait de nos trajets majoritairement pédestres est l'élimination de points successifs colinéaires et donc apparaissant comme alignés sur une trace. Ce calcul sera surtout efficace lors de la présence de longues lignes droites telles qu'observées lors de trajets en train : dans ce cas, 3 points *A*, *B* et *C* successifs sont alignés si l'angle (*AB*,*AC*) est inférieur à une valeur seuil XXXXX ce qui se traduit par le calcul classique du produit scalaire XXXXX. Une autre implémentation plus grossière de ce type d'algorithme est proposée dans [6, p. 140].

Nous proposons dans le tableau 7 une implémentation sous Matlab/Octave du calcul de l'angle entre 3 points successifs et l'élimination des points pour lesquels cet angle est inférieur à un seuil fourni en paramètre, ou dont la distance au point précédent est inférieure à une valeur minimum que nous avons arbitrairement sélectionnée à 2×10^-6, soit une vitesse de déplacement de 0,8 km/h à l'équateur. La figure 12 démontre que pour un choix d'angle seuil de l'ordre 0,5 degrés, un nombre substantiel de points est éliminé sans pour autant affecter l'aspect général de la trace. L'élimination des points considérés comme colinéaires permet d'éliminer de 20 à 50 % des points sans affecter notablement le rendu visuel de la trace (Fig. 12).

TABLEAU 6

```
function reduit=angle_gps(nom,ang_lim)
% prend nom_fichier[.dat] et l'angle seuil

eval(['load ',nom,'.dat']);
eval(['x=',nom,';']);
eval(['clear ',nom]);

% version Matlab
t=diff(x(:,1:2));
an=diff(angle(t(:,1)+i*t(:,2)));
a=find(abs(an)>ang_lim);
XX=x(a,1); YY=x(a,2);
reduit=[XX YY];
% fin version Matlab

% version C
for n=2:length(x)
    t(n-1,1)=x(n,1)-x(n-1,1);
    t(n-1,2)=x(n,2)-x(n-1,2);
    norm=sqrt(t(n-1,1)^2+t(n-1,2)^2);
    if (norm>2E-6) an(n-1)=acos(t(n-1,1)/norm); % 2E-6=0.8
km/h
    else an(n-1)=0;end;
end

k=1;
for n=1:length(an)-1
   if (abs((an(n+1)-an(n)))>ang_lim) r(k,:)=[x(n,1)
x(n,2)];k=k+1;end;
end
% fin version C

length(x)
length(reduit)
length(r)
reduit=r;
```

Script Matlab/Octave nommé `angle_gps.m` calculant l'angle entre 3 points successifs et chargé d'éliminer les points pour lesquels cet angle est inférieur à un seuil fourni par l'utilisateur. Ce script prend en argument un nom de fichier sans son extension (supposée être `.dat`), contenant deux ou trois colonnes avec les latitudes, longitudes et altitudes respectivement (telles qu'issues de `deg2dec.m` par exemple), et un angle seuil en radians (une valeur de 0,01 radians semble raisonnable). Deux implémentations de ce code réalisant la même opération sont proposées : une en calcul matriciel typique de Matlab, l'autre avec des boucles typiques d'un langage de type C.

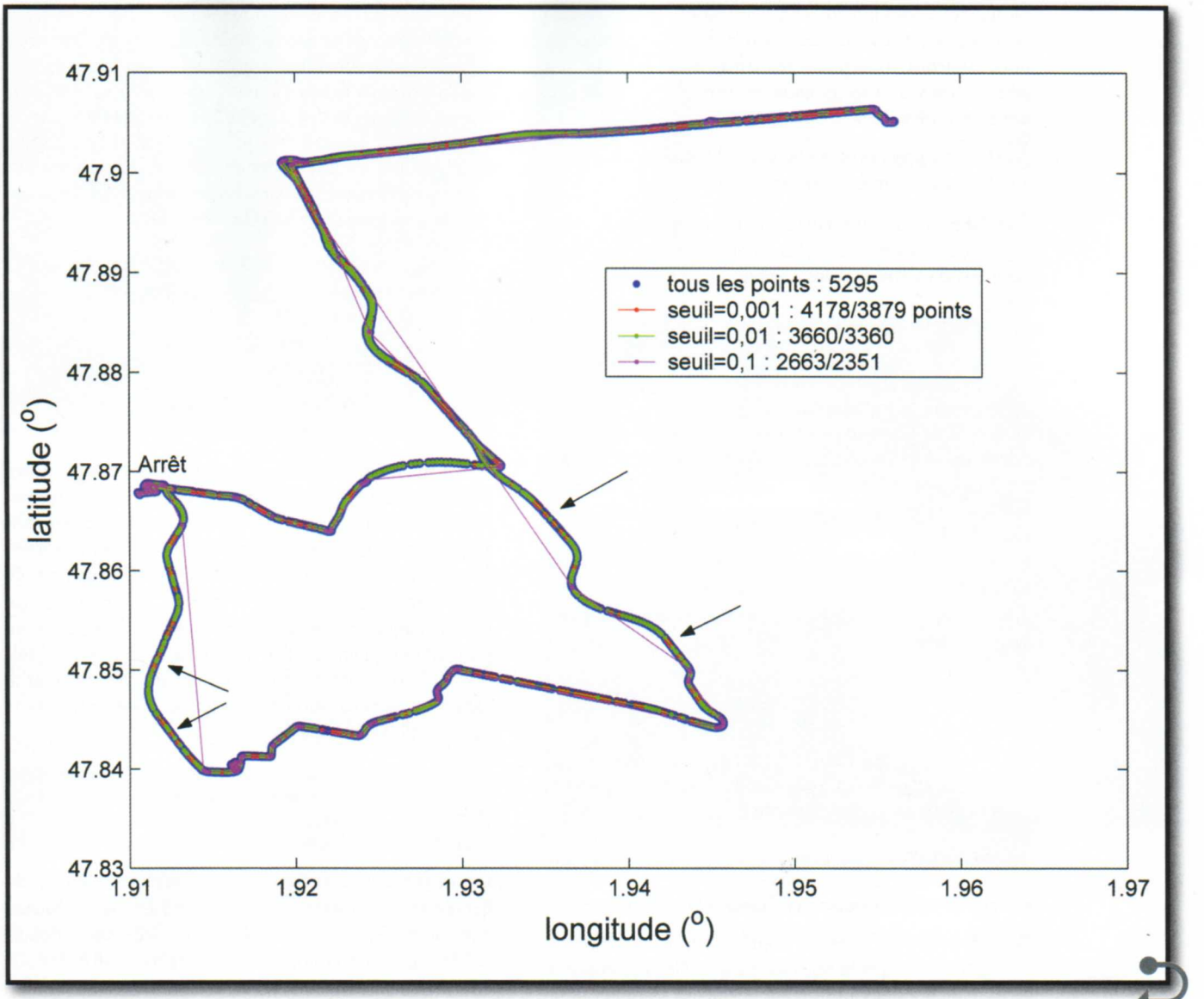


Fig. 12

Traitement d'une trace obtenue à Orléans illustrant la réduction du nombre de points affichés en fonction de l'angle seuil entre 3 points adjacents. Pour un angle seuil de 0,001 et 0,01 radians, aucune différence perceptible au niveau du tracé n'est visible par rapport à la trace contenant tous les points. Avec un angle seuil de 0,1 radians, la perte d'information est nettement visible. Les valeurs indiquées dans la légende correspondent au nombre de points restant à tracer après traitement du fichier : le premier nombre correspond à l'élimination des points considérés comme colinéaires, et le second nombre après élimination des points considérés comme trop proches (norme inférieur à 2×10^-6). Ce second traitement élimine principalement des points autour de la zone annotée par « Arrêt » dans la figure, pendant laquelle le véhicule a été stationné. Les flèches attirent l'œil sur quelques différences entre les traces issues d'un seuillage à 0,01 radians et à 0,001 radians.

7 UTILISATION DES TRACES GPS DANS GRASS

Cette présentation ne saurait se prétendre complète sans une mention au logiciel GRASS (*Geographic Resources Analysis Support System*), référence dans le domaine de la gestion géographique des données (*Geographical Information System, GIS*). GRASS suit la philosophie d'Unix en se présentant comme un ensemble de petits modules puissants appelés séquentiellement lors du traitement d'informations. GRASS s'installe trivialement sous GNU/Debian (Sarge) par `apt-get install grass` qui fournit la version 6.0.0 à la date de rédaction de cet article (début août 2006). Notre introduction aux commandes de base de ce logiciel nous a été fournie par [4, pp. 326-355] bien que la version plus récente de GRASS utilisée ici semble rendre un certain nombre de commandes présentées dans cet ouvrage obsolète.

Nous allons nous intéresser à la présentation de données acquises lors de randonnées dans la vallée de Chamonix : nous désirons superposer les traces acquises avec des informations de topographie du terrain. Nous devons donc résoudre deux problèmes :

1. Lire nos propres données dans GRASS et les incorporer aux bases de données existantes.
2. Obtenir des informations topographiques de la région qui nous intéresse, les incorporer dans la base de données de GRASS et les afficher.
3. Exporter les résultats en un format utilisable dans ce document.

Familiarisons-nous dans un premier temps avec GRASS en résolvant le premier point. Nous avons vu plus haut comment, au moyen de scripts shell et d'Octave, convertir des fichiers NMEA en des listes de données sous la forme de 3 colonnes représentant la longitude, la latitude et l'altitude de chaque point enregistré (la dernière étape avant la génération d'un fichier KML par exemple, juste avant l'ajout de l'en-tête et le remplacement des espaces par des virgules). Nous partons, par exemple, d'un fichier nommé `060729_argentiere.dat` dont le début se présente sous la forme :

```
6.92566666666667 45.9809566666667 1251.5
6.92558 45.9809533333333 1233.8
6.92557333333333 45.9809433333333 1231.2
6.92557333333333 45.9809266666667 1227.2
6.92555166666667 45.98091 1220.4
6.92552833333333 45.9808883333333 1212.8
...
```

et ainsi de suite pour un total de 34311 points.

GRASS nécessite, dans un premier temps, la définition de la zone sur laquelle nous allons travailler. En l'absence de base de données a priori sur lesquelles travailler, nous définissons une nouvelle zone vierge :

1. Pour une première utilisation, créer un répertoire dédié aux données utilisées par GRASS : dans mon cas `/home/jmfriedt/grass`.
2. Lancer GRASS par `grass60`.
3. Créer une nouvelle carte sur laquelle nous allons travailler : `Create New Location` que j'appellerai `jmfriedt` avec un Mapset du même nom. Cette carte n'existant pas, GRASS nous demande s'il faut la créer : nous désirons travailler avec des informations de Latitude-Longitude (B) avec des données au format WGS84 (le format dans lequel les données GPS sont fournies) : nous remplissons ainsi le champ `datum name`, sans transformation additionnelle. Ce mode de représentation des coordonnées spécifie aussi l'ellipsoïde associé (`ellipsoid name`) et il ne nous reste plus qu'à créer un espace de travail incluant toute la région de l'Europe qui nous intéresse. La France, par exemple, s'étend de 5 degrés Ouest à 8 degrés Est et de 42 à 52 degrés Nord. Nous utilisons ces paramètres dans le choix des dimensions initiales de la carte. La validation des paramètres se fait par ESC suivi de ENTER.
4. Une fois la carte nommée `jmfriedt` comme nom de LOCATION et de MAPSET créée, nous la validons une fois de plus par la paire ESC puis ENTER.
5. Nous observons alors le lancement d'une interface Tk et obtenons accès à une ligne de commande GRASS :

```
Welcome to GRASS 6.0.0 (2005)
...
GRASS 6.0.0 (jmfriedt):/home/jmfriedt/grass >
```

Ayant maintenant défini un terrain de jeu vierge, notre première tâche consiste à importer des données au format ASCII présentées sous forme de colonnes contenant respectivement la longitude, la latitude et un commentaire (ce commentaire dans notre cas est l'altitude). La commande pour atteindre ce but est `v.in.ascii` qui, comme son nom l'indique, convertit un fichier ASCII en vecteur de points. Un certain nombre d'arguments sont fournis tels que le nom du fichier, le nom du vecteur de sortie et le caractère de séparation des colonnes. Dans notre cas, la commande se résume en :

```
v.in.ascii input=060729_argentiere.dat output=waypoint
format=point fs=space
```

qui va générer le vecteur nommé `waypoint` à partir du fichier `060729_argentiere.dat` décrit auparavant. Notez que toutes les commandes shell sont accessibles depuis GRASS. Lors de la conversion du fichier, GRASS nous informe que le format de fichier est bien reconnu (`Maximum number of columns: 3` et qu'il a bien lu le bon nombre de points (`Number of points: 34311`) tel que nous le confirme `wc -l 060729_argentiere.dat`. Afin de tracer ces points, il nous faut définir un terminal de sortie qui sera dans un premier temps l'affichage graphique : `d.mon start=x0`. Finalement, nous traçons ce vecteur avec un certain nombre de paramètres explicites : `d.vect map=waypoint icon=basic/diamond size=8 color=green fcolor=red`. Nous pouvons zoomer en une région de la carte par `d.zoom` qui nécessite de bien lire les instructions pour en comprendre le bon fonctionnement (bouton de gauche pour le premier point, bouton du milieu pour le second point, bouton de droite pour reculer). Le résultat présenté sur la figure 13 n'a à peu près aucun intérêt par rapport à la sortie que fournirait `gnuplot` avec le même fichier. Nous allons cependant voir plus bas tout l'intérêt de ce petit exercice par la superposition de nos traces aux informations topographiques du terrain.

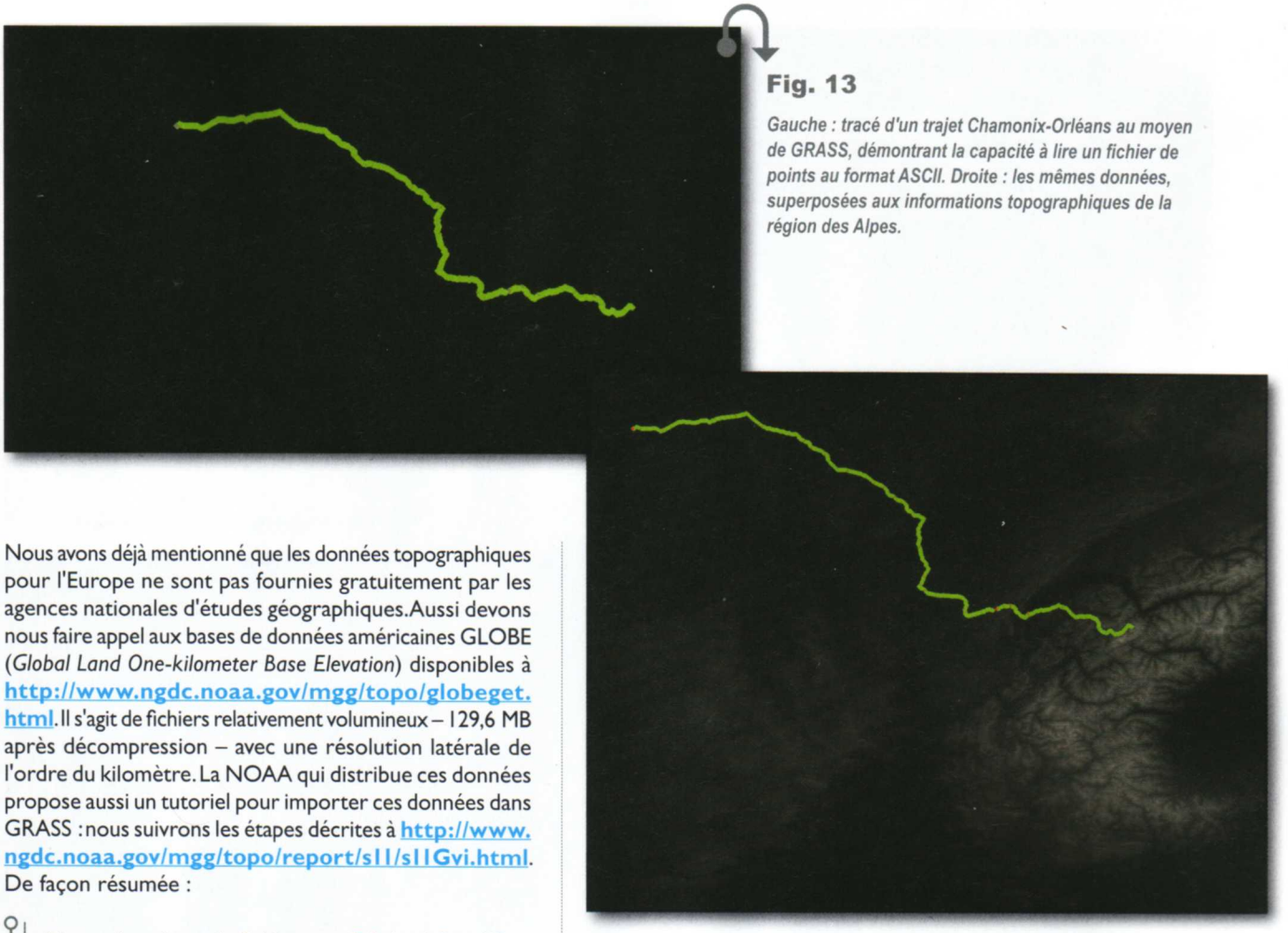


Fig. 13

Gauche : tracé d'un trajet Chamonix-Orléans au moyen de GRASS, démontrant la capacité à lire un fichier de points au format ASCII. Droite : les mêmes données, superposées aux informations topographiques de la région des Alpes.

Nous avons déjà mentionné que les données topographiques pour l'Europe ne sont pas fournies gratuitement par les agences nationales d'études géographiques. Aussi devons nous faire appel aux bases de données américaines GLOBE (*Global Land One-kilometer Base Elevation*) disponibles à **http://www.ngdc.noaa.gov/mgg/topo/globeget.html**. Il s'agit de fichiers relativement volumineux – 129,6 MB après décompression – avec une résolution latérale de l'ordre du kilomètre. La NOAA qui distribue ces données propose aussi un tutoriel pour importer ces données dans GRASS : nous suivrons les étapes décrites à **http://www.ngdc.noaa.gov/mgg/topo/report/s11/s11Gvi.html**. De façon résumée :

1. Nous récupérons le fichier `grasglob.tar` à **ftp://ftp.ngdc.noaa.gov/GLOBE_DEM/data/elev/grass/grasstar** qui contient les scripts nécessaires à l'importation de ces bases de données et le désarchivons dans notre répertoire de travail, créant ainsi une arborescence `grass/users`.

2. Nous téléchargeons les bases de données topographiques qui nous intéressent (la base de donnée incluant les Alpes est nommée G10G) à **http://www.ngdc.noaa.gov/mgg/topo/gltiles.html** et en sélectionnant la région couvrant de 0 à 50 degrés Nord et 0 à 90 degrés Est. Le fichier obtenu est `g10g.gz`.

3. Nous décompressons ce fichier et le renommons `g10g.pc`.

4. Nous changeons son *endianneess* (opération nécessaire sur un processeur Intel sur lequel ce test a été fait) par `dd if=g10g.pc of=g10g.ux conv=swab` et plaçons le fichier `g10g.ux` résultant dans le sous-répertoire `grass/user/cell` de notre répertoire de travail.

Au lancement de GRASS, nous n'allons plus créer une carte vide, mais allons charger le monde décrit dans les fichiers de configurations fournis ci-dessus. Ainsi, nous lançons `grass60` qui, cette fois, nous propose `Location: grass` et `Mapsets: user`. Ayant sélectionné ces options, nous cliquons sur `Enter GRASS`. Une nouvelle fonction, `g.region`, permet de charger une carte topographique d'une région du monde. Nous lançons donc `g.region rast=g10g.ux` qu'il faut ajuster afin de gérer correctement les valeurs sur 16 bits : `r.mapcalc 'g10g=if(g10g.ux-32768,g10g.ux-65536,g10g.ux,g10g.ux)'`. Cette opération prend environ 4 minutes sur une Pentium II cadencé à 400 MHz et possédant 128 MB de RAM, pour finalement générer la variable `g10g` contenant les données topographiques qui nous intéressent. Notez qu'un peu plus de 200 MB d'espace libre sur le disque dur sont nécessaires, puisque le contenu de la variable y est stocké. La liste des objets bitmap disponibles, `g.list type=rast`, nous montre bien cette variable dont le contenu s'affiche par `d.rast g10g`, toujours après avoir défini le terminal de sortie `d.mon start=x0`. Au lieu d'une image colorée en fonction de l'altitude, nous désirons assigner à cette carte un dégradé de gris : `r.colors map=g10g color=grey`, puis pour réafficher `d.redraw`. L'arc des Alpes est alors visible dans le coin en haut à gauche.

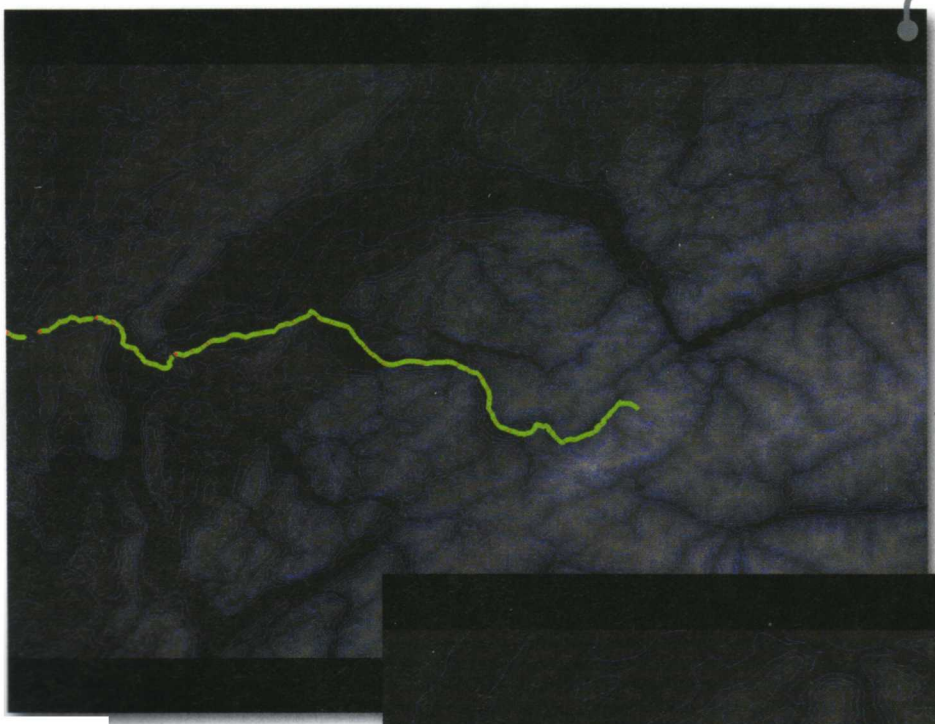
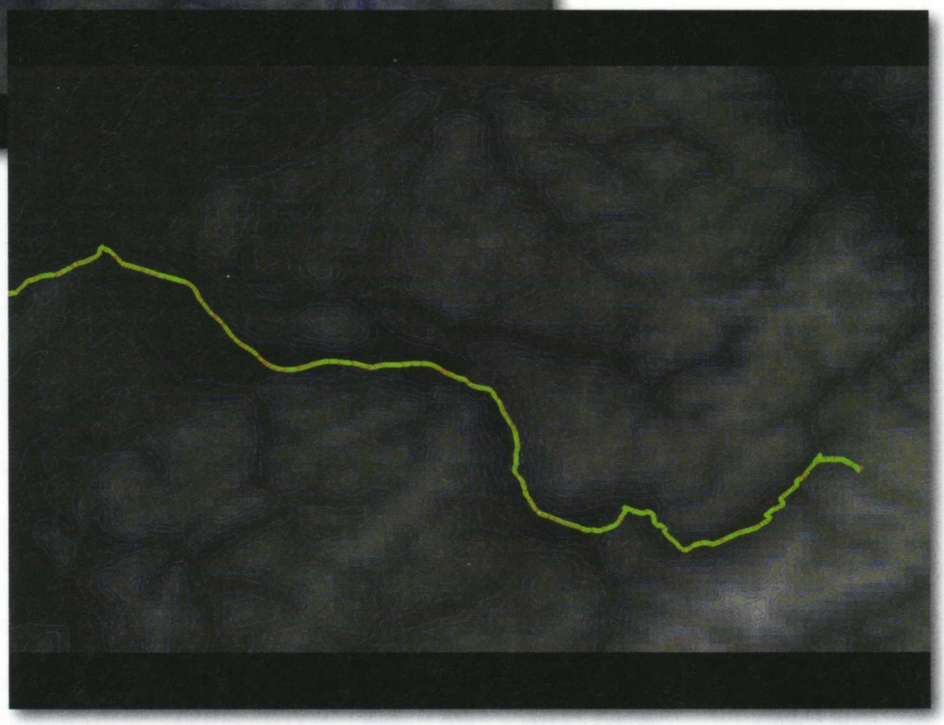


Fig. 14

Superposition d'une information topographique en tons de gris et sous forme de lignes de niveau espacées de 200 m avec les points GPS enregistrés lors d'un départ de la vallée de Chamonix en direction d'Orléans. Le zoom à droite permet de bien distinguer les différentes lignes de niveau, mais met aussi en évidence la résolution spatiale médiocre des informations topographiques.



Nous ajoutons maintenant nos données en suivant la séquence vue précédemment, et obtenons les figures telles que présentées sur la figure 13, page précédente, à droite. Afin d'exporter ces images au format PNG, nous avons remplacé le terminal graphique par la sortie dans un fichier par `d.mon start=PNG` après avoir défini quelques variables d'environnement sous GRASS :

```
export GRASS_WIDTH=1600
export GRASS_HEIGHT=1200
export GRASS_TRUECOLOR=TRUE
export GRASS_BACKGROUNDCOLOR=000000
export GRASS_PNGFILE=topographie.png
```

Le tracé s'obtient en retapant les commandes `d.vect` et `d.rast` et en achevant le tracé par `d.mon stop=PNG`. Une fois le fichier d'image obtenu, nous repassons au terminal graphique par `d.mon select=x0`.

Afin d'ajouter des courbes de niveau au fond de carte représentant l'élévation par des tons de gris, nous allons transformer les informations topographiques spatiales en données vectorielles contenant les coordonnées des lignes de niveau. La conversion d'une image bitmap dont l'intensité représente une altitude en un ensemble de courbes de niveau commençant au niveau de la mer (0) et espacées de 200 m s'obtient par : `r.contour in=g10g out=cont200 min=0 step=200` dont le résultat est placé dans la variable `cont200`. Le résultat est affiché sur la carte par `d.vect cont200 color=blue` pour donner le résultat visible sur les figures 14 et 15.

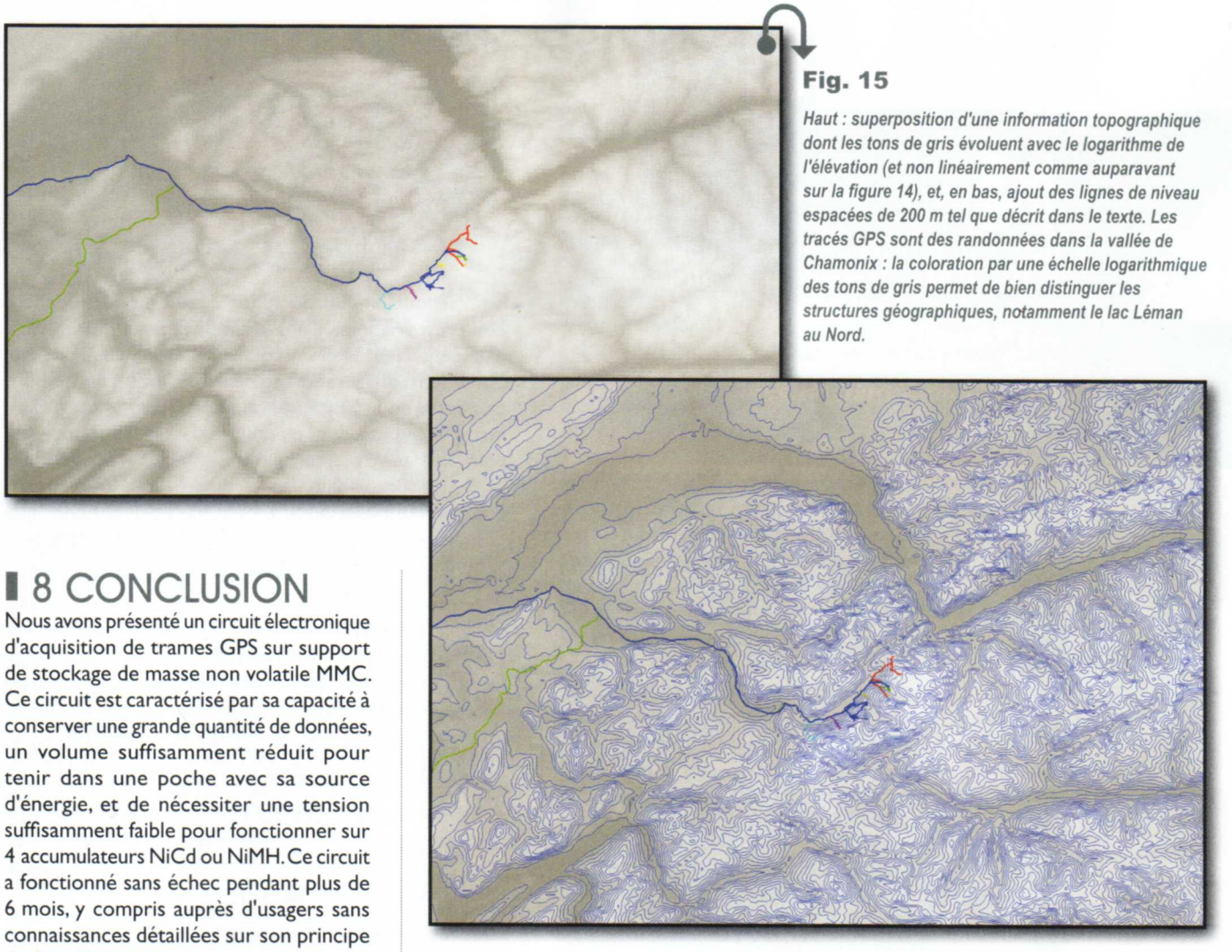


Fig. 15

Haut : superposition d'une information topographique dont les tons de gris évoluent avec le logarithme de l'élévation (et non linéairement comme auparavant sur la figure 14), et, en bas, ajout des lignes de niveau espacées de 200 m tel que décrit dans le texte. Les tracés GPS sont des randonnées dans la vallée de Chamonix : la coloration par une échelle logarithmique des tons de gris permet de bien distinguer les structures géographiques, notamment le lac Léman au Nord.

8 CONCLUSION

Nous avons présenté un circuit électronique d'acquisition de trames GPS sur support de stockage de masse non volatile MMC. Ce circuit est caractérisé par sa capacité à conserver une grande quantité de données, un volume suffisamment réduit pour tenir dans une poche avec sa source d'énergie, et de nécessiter une tension suffisamment faible pour fonctionner sur 4 accumulateurs NiCd ou NiMH. Ce circuit a fonctionné sans échec pendant plus de 6 mois, y compris auprès d'usagers sans connaissances détaillées sur son principe de fonctionnement.

Nous nous sommes efforcés de diffuser par divers moyens les données ainsi accumulées : par fusion avec des données existantes telles que fournies par Google Maps, Google Earth et GRASS, ou en partageant avec des projets de cartographie libre tels qu'UPCT. L'ensemble des outils développés pour l'acquisition et la conversion des données a été décrit afin de permettre au lecteur de les adapter à ses propres applications.

Notre souhait serait désormais que ce circuit puisse servir de base au développement de tels projets aujourd'hui restreints à un public déjà équipé de récepteurs relativement onéreux. Les lecteurs intéressés par ce montage électronique sont encouragés à contacter les auteurs puisqu'une réalisation en quantités importantes doit permettre d'abaisser les coûts d'achat des composants.

RÉFÉRENCES

[1] FRIEDT (J.-M.), CARRY (É.), « Enregistrement de trames GPS – développement sur microcontrôleur 8051/8052 sous GNU/Linux », *GNU/Linux Magazine*, n° 81, février 2006.

[2] GUIDON (Y.), « Introduction à la compression de données : mise en évidence de l'entropie », *GNU/Linux Magazine* ; n°74 (juillet/août 2005), pp. 46-61.

[3] « Interview de Michel Bondaz UPCT.ORG », GNU/Linux Magazine, N° 73.

[4] ERLE (S.), GIBSON (R.) & WALSH (J.), *Mapping Hacks – Tips and tools for electronic cartography*, O'Reilly, 2005.

[5] SNYDER (J. P.), « *Flattening the Earth – Two thousand years of map projections* », *The University of Chicago Press* (1993), pp. 178-183.

[6] GIBSON (R.) & ERLE (S.), *Google Maps Hacks – Tips and*

Denis Bodor :: db@ed-diamond.com :: lefinnois@lefinnois.net

LINUX EMBARQUÉ SUR CARTE ACME FOX

Nous le rabâchons sans cesse dans GNU/Linux Magazine France, l'embarqué est le nouveau terrain de jeu de Linux et des outils GNU. Trouver une plate-forme d'expérimentation dans ce domaine est souvent délicat. La carte présentée dans cet article s'avère être une plate-forme embarqué polyvalente, un vrai paradoxe.

Pourquoi un paradoxe ? Le monde de l'embarqué n'est pas celui de l'informatique personnelle, bureautique, ludique ou serveur. La plate-forme choisie pour construire un système embarqué est sélectionnée en fonction des besoins et d'un cahier des charges précis. Au final, le système embarqué sera utilisé pour une seule et unique tâche. Il ne fera qu'une chose, mais il la fera bien. C'est précisément l'inverse de la polyvalence. Malheureusement, lorsqu'on souhaite expérimenter dans un domaine, mieux vaut investir dans un matériel permettant le plus grand nombre d'expériences. La carte FOX développée par la société italienne ACME System se prête bien à ce type d'utilisations.

ACME FOX

Alors que l'ARM se taille une part non négligeable dans le marché de l'embarqué. Il est amusant de constater qu'ACME a préféré utiliser le processeur AXIS Etrax 100LX. Ce processeur RISC 32 bits cadencé à 100 Mhz intègre une MMU (*Memory Management Unit*). C'est donc un Linux standard qui équipe la carte et non un uCLinux. Le développement en est tout autant facilité, puisqu'on retrouve toutes les fonctionnalités habituelles dans un environnement GNU/Linux. Le processeur Etrax n'est pas très courant, mais dispose d'un portage GCC stable. C'est parfois un point qui est mis en avant en défaveur de la plate-forme par certains préférant alors se tourner vers des processeurs sur base ARM. L'Etrax est certes spécifique, mais il convient de ne pas oublier qu'AXIS, le fabricant, est actuellement leader dans le domaine des caméras IP et des serveurs d'impression. La documentation, les ressources et l'expertise concernant ce processeur ne manquent pas et ne manqueront sans doute pas dans l'avenir.

La carte FOX intègre 16 Mo de mémoire vivre (RAM) et 4 Mo de mémoire Flash. Surprenant ? En effet, nous avons là davantage de mémoire vive que de Flash. Ceci s'explique, comme c'est souvent le cas dans ce genre de situation, par la présence d'une autre solution de stockage des données et des programmes. Dans le cas de la carte FOX, il s'agit d'un contrôleur USB 1.1 couplé à un *hub* racine de deux ports (en configuration standard). Un certain nombre de pilotes pour des périphériques USB sont intégrés au système de base, dont un pour la classe *USB Storage*. On peut ainsi très facilement connecter n'importe quelle clef USB et disposer d'un espace de stockage en mémoire Flash pouvant aller jusqu'à 4 Go. Inutile de préciser que des clefs de 256 Mo ou même 512 Mo se trouvent dans le commerce pour moins de 20 euros.

Parmi les caractéristiques standards, nous retrouvons le faible encombrement (66 x 72 mm), le poids plume (37 gr), la console VT100 (en 3.3V, attention !), l'interface réseau Ethernet 10/100 ou encore la consommation de l'ordre du watt (pour la carte seule).

Mais l'une des caractéristiques les plus avenantes de la carte reste sa connectivité :

- 48 E/S génériques ;
- un bus i2c ;
- 2 contrôleurs USB 1.1 ;
- 4 ports série ;
- 4 ports IDE (8 disques) ;
- 2 ports SCSI (ou un *Wide* SCSI) ;
- 2 ports parallèles.

Bien entendu, toutes les fonctionnalités ne peuvent pas être utilisées en même temps. Par exemple, un port est commun à l'USB et `/dev/ttyS1`. `/dev/ttyS0` est, par ailleurs, dédié à la console. Il ne reste donc plus que deux ports série. Il en va de même pour les ports IDE qui supprimeront tout autre type de connectivité en étant activés. La configuration de base par défaut de la carte offre une poignée d'E/S, 2 ports série et un bus i2c.

INSTALLATION DU SDK

L'installation du SDK est parfaitement décrite sur le site d'ACME System. Je reprendrai simplement ici la procédure d'installation sur un système Debian de manière commentée afin de rendre l'article le plus complet possible. La première étape consiste à installer une version récente de NetBSD *make* (1.98) en utilisant tout simplement le système de gestion de paquets Debian :

```
$ sudo apt-get install pmake
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances... Fait
Les NOUVEAUX paquets suivants seront installés:
  pmake
[...]
Dépaquetage de pmake (à partir de .../pmake_1.111-1_i386.deb)
...
Paramétrage de pmake (1.111-1) ...
```

Nous installons ensuite la chaîne de compilation spécifique au processeur Etrax et aux plateformes AXIS. Celle-ci est disponible sous la forme d'un paquet Debian directement utilisable. Il nous suffit donc de le télécharger (47 Mo) et de l'installer très classiquement avec un `sudo dpkg -i` :

```
$ wget http://developer.axis.com/\
download/compiler/cris-dist_1.63-1_i386.deb

$ sudo dpkg -i cris-dist_1.63-1_i386.deb
[...]
Paramétrage de cris-dist (1.63-1) ...
```

Bien entendu, un paquet similaire existe pour les distributions basées sur RPM comme RedHat ou encore pour Slackware. En jetant un coup d'œil au contenu du paquet fraîchement installé (`dpkg -L`), on remarque que presque l'ensemble de l'arborescence s'est installée dans `/usr/local`. Seule la documentation trouve sa place à l'emplacement habituel, `/usr/share/doc`. Ceci peut paraître surprenant, mais je suppose que le mainteneur du paquet a souhaité éviter au maximum les conflits et aura trouvé cette solution préférable.

Nous devons maintenant installer les sources qui nous permettront de reconstruire un système complet pour la carte FOX. Pour cela, nous récupérons tout d'abord le *tarball* des fichiers de configuration propres aux différentes cartes à base d'Etrax : **http://developer.axis.com/download/devboard/R2_01/devboard-R2_01.tar.gz**.

Cette archive contient la base du SDK, mais il faut ajouter les véritables sources (GNU pour la plupart) réunies dans un grosse archive : **http://developer.axis.com/download/distribution/devboard-R2_01-distfiles.tar.gz**. Notez que ce fichier n'est pas strictement nécessaire. Il permet de simplifier la procédure en mettant à disposition une copie locale des sources. Dans le cas contraire, les scripts récupèreront eux-mêmes les archives nécessaires.

Nous commençons donc par désarchiver la base du SDK dans un sous-répertoire `FOX` de notre répertoire personnel. Le choix du nom de répertoire en question est laissé à votre convenance, bien sûr. Nous obtenons donc un `~/FOX/devboard-R2_01`. Si vous avez récupéré les fichiers sources, vous pouvez vous placer dans ce répertoire et désarchiver le tarball de quelques 110 Mo. Le sous-répertoire `devboard-R2_01/distfiles` se trouve ainsi créé et rempli.

Toujours dans `~/FOX/devboard-R2_01`, exécutez le script `./install` :

```
$ ./install
*** Install script for Axis Developer Board/Device Server SDK
***
=============================================================
 If you already have an SDK installed, you can let this script
 copy distribution files from that installation. This can
possibly
 reduce the amount of data that has to be downloaded.
 * Do you want to copy distfiles now [yn]? (default: n):
```

Le script vous demande s'il doit copier les archives sources (*distfiles*) depuis un endroit vers `./distfiles`. Si nous avez désarchivé `devboard-R2_01-distfiles.tar.gz` dans un autre répertoire que celui spécifié plus haut, répondez affirmativement. Le script vous demandera alors de préciser où se trouvent les archives en question :

```
 Enter the path to the installed SDK which contain a distfiles
directory.
 For example: "../devboard-R2_00"
 * Path:
```

Si vous avez désarchivé dans `~/FOX/devboard-R2_01/distfiles` ou que vous souhaitez télécharger les archives automatiquement, répondez négativement.

```
 * Fetching "tools/build" revision "R2_12_4"... done
 * Fetching MD5 for "distfiles/tools-build-R2_12_4.tar.gz"...
done
 * Checking MD5 for "distfiles/tools-build-R2_12_4.tar.gz"...
done
 * Unpacking "distfiles/tools-build-R2_12_4.tar.gz"... done
[...]
 Select wich product to use. This selection will affect which
 configuration files the SDK will use. Select the product in
 the list below that closest resemblels your product.

 Alternatives:
  1. devboard_82  -  "Axis Developer Board 82/83"
  2. devboard_82+  -  "Axis Developer Board 82+/83+"
  3. devboard_lx  -  "Axis Developer Board LX"
  4. devboard_lx_ide  -  "Axis Developer Board LX IDE"
  5. fox  -  "FOX BOARD by ACME systems (www.acmesystems.it)
             with MCM4+16"
  6. mcm_2_8  -  "Custom designs with MCM2+8, no additional
                 memory"
  7. mcm_4_16  -  "Custom designs with MCM4+16, no additional
                  memory"
 * Specify product (default: 2. devboard_82+):
```

Dernière étape d'installation du SDK, nous spécifions la plate-forme utilisée. Ici, il s'agit bien sûr du choix 5, la carte FOX d'ACME. Ce choix permet d'utiliser les fichiers de configuration (noyau et applications) spécifiques à la carte. De plus, cette configuration par défaut nous permet également d'embrayer immédiatement sur la compilation de notre première image.

CONSTRUCTION D'UNE IMAGE

Suite à l'exécution de la procédure complète, un message résume l'état de l'installation du SDK :

```
Configured for: fox - Install complete
+-----------------------------------------------------------+
| To setup product (Next step):                             |
|                                                           |
|                                                           |
| Optional             Required                             |
| --------------       ---------------------                |
| 1.                   2.              3.                   |
| make config      -->  ./configure  --> make               |
| make menuconfig                                           |
| make xconfig                                              |
|                                                           |
|                                                           |
| If the configuration step is skipped (the first step) the |
| default configuration will be used when building the image. |
|                                                           |
|                                                           |
| Use ./configure --help for more information about         |
| configure options.                                        |
+-----------------------------------------------------------+
```

Tout comme avec un système uCLinux, une interface de configuration *curses/dialog* permet de spécifier ses préférences d'installation. Nous verrons cela par la suite. Le message le précise, en sautant l'étape de configuration, nous pouvons construire une image système et les outils de base pour la carte FOX dans leur configuration par défaut. C'est précisément ce que nous allons faire. Pour cela, une seule commande est suffisante :

```
$ ./configure
### Selected product: "fox" ###
 * Using previously fetched packages... done
 * Fetching "modules/rules.build_modules" revision "R1_0_4"...
done
 * Fetching MD5 for "distfiles/modules-rules.build_modules-
R1_0_4.tar.gz"... done
 * Checking MD5 for "distfiles/modules-rules.build_modules-
R1_0_4.tar.gz"... done
 * Unpacking "distfiles/modules-rules.build_modules-R1_0_
4.tar.gz"... done
 * Using compiler "/usr/local/cris/bin/cris-axis-linux-gnu-
gcc" (revision "R63").
 * Fetching "packages/romfs_meta/common" revision "R1_0_1"...
done
 * Fetching MD5 for "distfiles/packages-romfs_meta-common-
R1_0_1.tar.gz"... done
 * Checking MD5 for "distfiles/packages-romfs_meta-common-
R1_0_1.tar.gz"... done
 * Unpacking "distfiles/packages-romfs_meta-common-R1_0_1.tar.
gz"... done
 * Fetching "os/linux" revision "tag--devboard-R2_01"...
[...]
------------ Remplis ./devboard-R2_01/distfiles ------------
 * Fetching MD5 for "distfiles/packages-devices-scsi-R1_0_
0.tar.gz"... done
 * Checking MD5 for "distfiles/packages-devices-scsi-R1_0_
0.tar.gz"... done
 * Unpacking "distfiles/packages-devices-scsi-R1_0_0.tar.
gz"... done
 * Fetching "packages/devices/ttyusb" revision "R1_0_0"...
done
 * Fetching MD5 for "distfiles/packages-devices-ttyusb-R1_0_
0.tar.gz"... done
 * Checking MD5 for "distfiles/packages-devices-ttyusb-R1_0_
0.tar.gz"... done
 * Unpacking "distfiles/packages-devices-ttyusb-R1_0_0.tar.
gz"... done
 * Generating "Makefile"... done
```

Le script de configuration va automatiquement chercher les éléments logiciels qu'il ne possède pas (tout cela est dépendant de l'installation du tarball de 110 Mo) et paramétrer les sources pour vous. L'ensemble du système et des outils seront préparés pour la compilation par défaut. Armez-vous de patience et compilez avec :

```
$ make
[...]
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): y
Please type root's password to make etrax100boot setuid root
or Ctrl-D to cancel.
Password: ****************
[...]
```

A un moment du processus de compilation, le script s'interrompra pour vous demander le mot de passe *root*. Le *bootloader* réseau `etrax100boot` doit, en effet, être *suid root* pour fonctionner s'il est lancé par un utilisateur standard. Une solution plus sure est d'utiliser le compte superutilisateur pour bootloader le chargeur dans la carte FOX et charger l'image du système en Flash. Cependant, cela s'avère bien plus pénible, car non prévu par les différents Makefiles et on choisira donc la méthode suid root. Notez que les problèmes de sécurité liés aux applications suid root sont normalement limités dans un tel environnement. Il s'agit ici d'une machine de développement et non d'un serveur en production. Le contexte sécuritaire n'est pas le même.

Au terme de la compilation des différents éléments logiciels et de la composition des fichiers de configuration, on trouve dans `target/cris-axis-linux-gnu` l'arborescence complète du futur système.

Commence alors la composition de l'image (le *firmware*) destinée à la mémoire Flash de la carte FOX. Le script `make` commence par créer le système de fichier CRAMFS en utilisant l'arborescence :

```
Making cramfs of /home/denis/FOX/devboard-R2_01/target/cris-
axis-linux-gnu/
Using a blocksize of 8192 bytes.
```

Nous obtenons, à ce stade, le fichier `rootfs.img`. Autre fichier pris en compte, `rescue.img` est une copie de `os/linux/arch/cris/boot/rescue/rescue.bin` complétée (*padding*) pour arriver à une taille de 0x010000 octets (65536, soit 64 Ko). Il s'agit d'un code de secours exécuté en cas de problème et qui utilise un bootloader concaténé à une `kimage`. Elle-même est le résultat de la concaténation de `vmlinux.bin` et `rootfs.img`. `vmlinux.bin` est obtenu dans `os/linux-tag--devboard-R2_01/arch/cris/Makefile` via `cris-objcopy` permettant d'*aplatir* le fichier `vmlinux` qui est une version « brute de compilation » du noyau (non *bootable* et non compressée, par opposition à `vmlinuz`). L'objectif du code contenu dans `rescue.bin` est un chargement en tout début de la mémoire flash. Il a pour but de vérifier un certain nombre de sommes de contrôle et de s'assurer que la mémoire flash n'est pas corrompue. En cas de problème, il n'est plus possible de flasher le firmware via le réseau et le code initialise alors un port série pour recevoir un chargeur et une image.

Vient ensuite la création d'une partie de l'image en mémoire flash :

```
echo -n > flash1.img
mkptable -a crisv10 -d -f ptable_dummy.img ptablespec
cat ptable_dummy.img vmlinuz rootfs.img > flash1.img
padflashimage 0x350000 flash1.img
```

Après avoir créé un fichier `flash1.img`, c'est l'outil `mkptable` qui entre en jeu pour créer une table de partition à partir du fichier `ptablespec` créé par ailleurs :

```
# Generated file, do not edit!
[ptable]
# name size csum type image-file
rescue 0x010000 ro rescue rescue.img
flash1 0x350000 ro kernel flash1.img
flash2 0x0a0000 rw jffs2 flash2.img
```

Ce fichier reprend la répartition de la mémoire Flash par rapport à son contenu. Sont spécifiées ici les tailles en hexadécimal de chaque partie. On retrouve l'image `rescue.img`, la première zone de 0x350000 octets correspondant au système de fichier CRAMFS et `flash2.img`, une zone de mémoire Flash en lecture/écriture.

On remarque que la première version du fichier `flash1.img` n'est pas viable. `mkptable` a, en effet, besoin d'un fichier valide. Pour produire l'image finale et un `ptable.img` valable, il faut tout d'abord produire un fichier « bidon » (option `-d` de `mkptable`) pour une première version de `flash1.img` intégrant `ptable_dummy.img`. Ensuite, `mkptable` est relancé et prendra en compte un fichier `flash1.img` type. Voilà pourquoi nous avons tout d'abord :

```
cat ptable_dummy.img vmlinuz rootfs.img > flash1.img
```

puis, avec vérification :

```
mkptable -a crisv10 -v -f ptable.img ptablespec
Check flash1 0x350000 ro kernel flash1.img
partition starts at 0
partition size is 3473408
partition checksum is        592203832

Check flash2 0x0a0000 rw jffs2 flash2.img
partition starts at 3473408
partition size is 655360
partition checksum is        161850587

partition table csum size is 68, sum is 1943
Updated branch-skip offset to 88
rm ptable_dummy.img
cat ptable.img vmlinuz rootfs.img > flash1.img
```

Par ailleurs, le fichier `flash2.img` est construit par `mkfs.jffs2` à partir du contenu de `target/cris-axis-linux-gnu/mnt/flash`. C'est dans ce répertoire, qui sera `/mnt/flash` sur la carte FOX, que sont placés les répertoires `etc` et `root` (pointés par un lien symbolique depuis l'arborescence en lecture seule contenue dans `flash1.img`).

Enfin, tout cela est réuni dans le fichier `fimage` :

```
Creating fimage
cat rescue.img flash1.img flash2.img > fimage
Adding hardware ID "1.00" to fimage
Adding checksum "770658951" to fimage
```

CHARGEMENT DU FIRMWARE

Si vous avez suivi les indications jusqu'ici, vous devez avoir un fichier `fimage` d'un peu plus de 4 Mo correspondant au firmware à charger dans la mémoire Flash de la carte FOX. Il existe plusieurs solutions permettant le chargement de cette image :

CHARGEMENT VIA LE RÉSEAU

C'est ma méthode typique. Elle nécessite la présence du SDK sur la machine de développement et la connexion de ma carte FOX au réseau local (par défaut, l'IP de la carte FOX est 192.168.0.90) :

```
$ . init_env
Prepending "/home/denis/FOX/devboard-R2_01/tools/build/bin"
to PATH.
Prepending "/usr/local/cris/bin" to PATH.
```

On place ensuite un cavalier sur J8 (à côté du *reset*) et on lance :

```
boot_linux -F -i fimage
Using internal boot loader: INTERNAL_NW - Network boot
(default).
Starting boot...
We're doing a flash write, this may take up to a few
minutes...
```

On procède ensuite au reset de la carte :

```
Device ID = 0xc0fa5179
This bootloader was built by denis on
 lundi 18 septembre 2006, 18:04:34 (UTC+0200).
Checksum of bootloader is 0x000a1200
Waiting for load info.
Checksum of file is 0x00001e6d
Got load info.
SET_REGISTER
0xb0000000
0x000095f8
[...]
Checksum of file is 0x294fb0e0
FLASH
0xc0004000
0x00000000
0x00400000
Found 1 x CFI at 0x80000000
No single x16 at 0x84000000
No interleaved x16 at 0x84000000
0x80000000: Erasing 0x00002000 bytes
0x80000000: Writing 0x00002000 bytes
0x80002000: No need to write
0x80004000: No need to write
0x80006000: No need to write
0x80008000: No need to write
0x8000a000: No need to write
0x8000c000: No need to write
0x8000e000: No need to write
0x80010000: Erasing 0x00010000 bytes
0x80010000: Writing 0x00010000 bytes
[...]
0x803f0000: No need to write
0x80000000: Verifying...OK
JUMP
0x00000000
END
Exiting with code 0
```

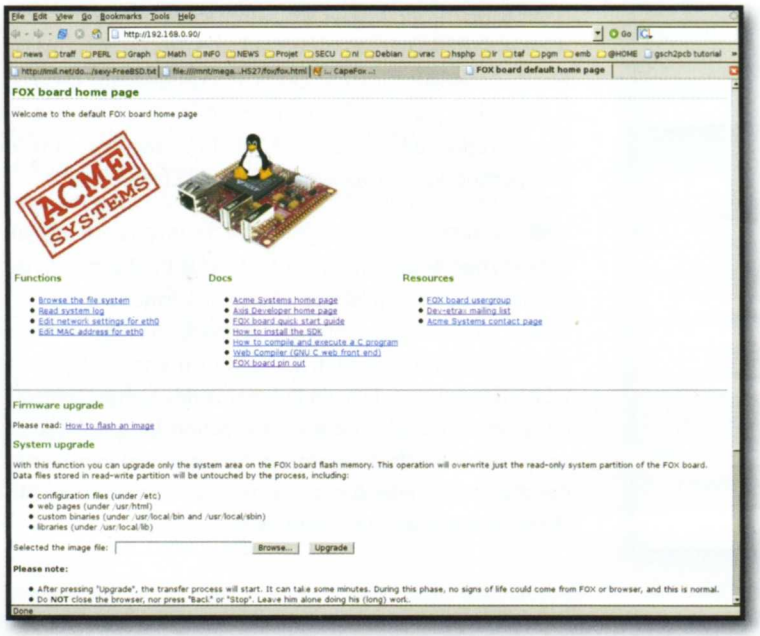
La carte est automatiquement redémarrée en mode normal. Pensez simplement à retirer le cavalier de J8 afin que la carte ne retourne pas en mode chargement lors du prochain reset.

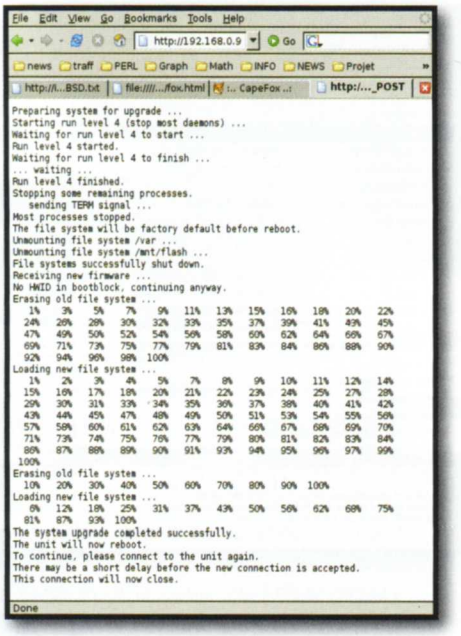
Note : Lorsque le système sur la carte FOX redémarre et que les clefs d'hôte SSH (*dropbear*) n'existent pas, celles-ci sont générées en tâche de fond. Ainsi, bien après que le système ne soit accessible par Telnet, il est possible que les connexions SSH échouent. Ce n'est pas un disfonctionnement, il n'y a pas d'autre solution que d'attendre. Le processeur Etrax est destiné à l'embarqué et non au calcul intensif.

CHARGEMENT VIA FTP

Plus simple, cette méthode de chargement ne nécessite rien d'autre qu'une image du firmware et un client FTP. La carte n'est pas nécessairement sur le même réseau que le client FTP. Il faut simplement qu'elle soit accessible :

```
$ ftp 192.168.0.90
Connected to 192.168.0.90.
220 FOX board by Acme Systems srl (brown_1_0)
 release 2.01 (sep 20 2006) rea
Name (192.168.0.90:denis): root
331 User name okay, need password.
Password: ****
230 User logged in, proceed.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put fimage flash_all
local: fimage remote: flash_all
200 Command okay.
150-Shutting down processes.
Preparing system for upgrade ...
Starting run level 4 (stop most daemons) ...
Waiting for run level 4 to start ...
Run level 4 started.
Waiting for run level 4 to finish ...
... waiting ...
Run level 4 finished.
Stopping some remaining processes.
   sending TERM signal ...
Most processes stopped.
The file system will be factory default before reboot.
Unmounting file system /var ...
Unmounting file system /mnt/flash ...
File systems successfully shut down.
150 Opening data connection.
214-Virtual target execution.
Receiving new firmware ...
No HWID in bootblock, continuing anyway.
Erasing old file system ...
   1%
   3%
   5%
[...]
Loading new file system ...
   1%
   2%
   3%
[...]
Erasing old file system ...
  10%
  20%
[...]
Loading new file system ...
   6%
  12%
The system upgrade completed successfully.
The unit will now reboot.
To continue, please connect to the unit again.
There may be a short delay before the new connection is
accepted.
This connection will now close.
214 Virtual target exit.
4194328 bytes sent in 4.54 secs (902.2 kB/s)
ftp> bye
221 Goodbye.
```





Le mot de passe par défaut de la carte est `pass` et non `acme` comme précisé sur certaines pages du site du fabricant. On notera que cette solution de chargement n'est pas sécurisée. Il suffit, en effet, qu'un attaquant « écoute » le mot de passe circulant en clair sur le réseau pour faire ce qui lui plait. Sachant que les systèmes embarqués sont parfois fortement liés à la sécurité des personnes, il est donc vivement recommandé de désactiver cette fonctionnalité. La solution de chargement via le réseau local (par un câble croisé par exemple) nécessite une intervention physique sur la carte (*jumper*+reset), ce qui renforce la sécurité.

CHARGEMENT VIA LE WEB

De base, dans sa configuration d'usine, la carte FOX intègre non seulement un serveur FTP, mais également HTTP ainsi qu'une interface CGI permettant le chargement d'une image du firmware. Pour charger une image, il suffit de pointer son navigateur sur l'adresse IP de la carte et d'utiliser le formulaire adéquat sur la page.

Encore une fois, la sécurité est ici inexistante. Aucune authentification n'est nécessaire pour mettre à jour le firmware. Il faut donc, lors de la mise en production, soit désactiver cette fonctionnalité, soit la renforcer (SSL, filtrage `iptable`, etc.).

PERSONNALISATION DE L'IMAGE

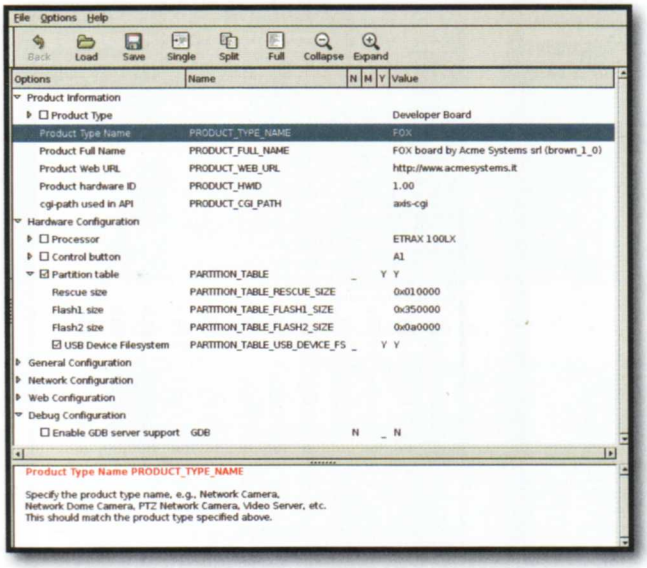
La carte, avec son firmware d'origine, dispose déjà de fonctionnalités intéressantes... trop sans doute. Ainsi, quelques fonctionnalités peuvent être indésirables comme le serveur HTTP, FTP, etc. D'autre part, vous pouvez également avoir besoin de fonctionnalité présentes uniquement dans le noyau en version 2.6. Enfin, la carte FOX disposant d'un grand nombre de ports de toutes sortes, vous pouvez adapter l'ensemble à vos besoins.

En reposant sur les sources et la configuration d'origine du SDK, l'adaptation se fera en deux parties :

- la configuration de la suite logiciel ;
- la configuration du noyau.

La configuration du SDK est simple. Il vous suffit d'avoir lu le message durant l'installation de ce dernier. Un simple `make config` (mode console), `make menuconfig` (interface curses/dialog) ou `make gconfig` (interface Gnome/GTK) vous permettront de configurer une vaste gamme d'options :

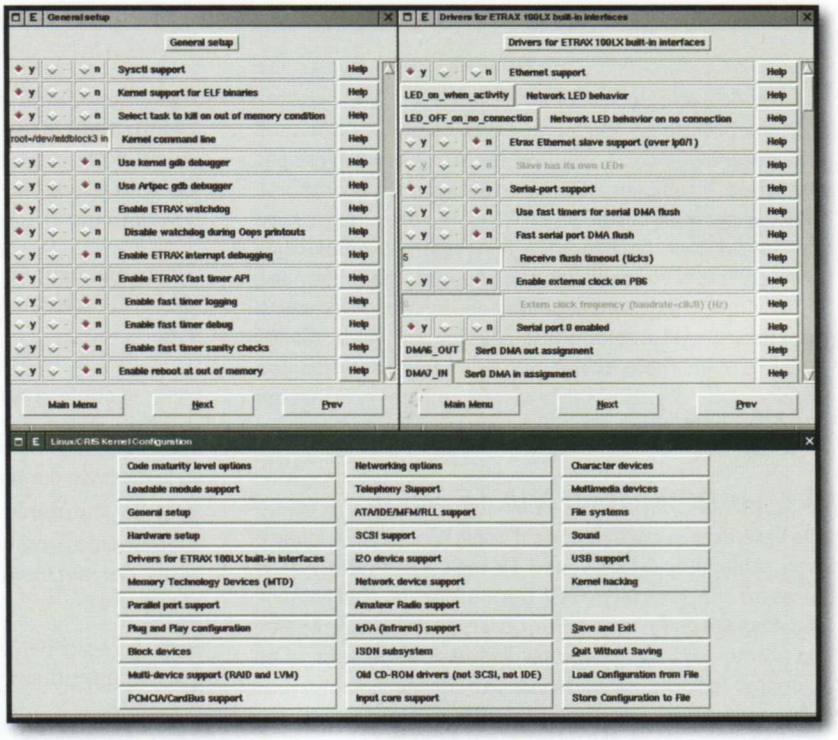
- la répartition de la mémoire flash entre les différentes parties (partitionnement) ;
- les applications à intégrer comme le client SMTP ou FTP, le serveur HTTP, PPP, l'éditeur EasyEdit (berk!) ou encore `tcpdump` ;
- les paramètres d'une partie de la configuration réseau et des services ;
- la version du noyau (2.4 ou 2.6) et le fichier de configuration à utiliser (voir ci-après) ;
- une partie des fichiers de données (exemple de pages Web pour le serveur HTTP avec formulaire de chargement du firmware) ;
- le choix de la bibliothèque C standard : la version GNU libc ou uClibc.



Hardware setup :Vous trouverez ici des fonctionnalités de très bas niveau comme la configuration de la mémoire Flash, la configuration des LED CMS soudées sur la carte, la direction par défaut des ports A et B ou encore la configuration des puces de SDRAM (intégrées dans le composant pour la version MCM de l'Etrax 100LX).

Drivers for ETRAX 100LX built-in interfaces : C'est, là, la section la plus importante pour la configuration de la carte. Ici, vous pourrez choisir les fonctionnalités à activer ou non. Bien entendu, certaines d'entre elles en désactivent d'autres (IDE vs GPIO ou encore ttyS1 vs USB). Gardez sous la main le schéma des sorties (*pinout*) de la carte FOX lors de la configuration. Les paramètres proposés par défaut correspondent (plus ou moins) à la description donnée sur le site de documentation ACME (http://www.acmesystems.it/?id=18).

Note : Remarquez que la carte est livrée en standard avec un noyau 2.4.31 et le SDK préconfiguré pour utiliser également cette version. Un noyau 2.6.12 est également inclus dans le SDK. Cependant, il est important de signaler que le sous-système USB n'est pas encore considéré comme stable (dixit le site ACME). Si vous comptez faire usage de l'USB ou, tout simplement, que vous n'avez pas strictement besoin de fonctionnalités présentes dans le noyau 2.6, restez sur du 2.4. La série 2.4 du noyau est stable, éprouvée et vérifiée. Bien que le 2.6 offre certains avantages, la stabilité et les garanties offertes par un 2.4 sont primordiales pour l'embarqué.



Après reconfiguration du SDK, il est important de ne pas oublier de relancer `./configure` afin de générer un `Makefile` adéquat.

La configuration la plus importante reste celle du noyau lui-même. Celle-ci permet d'adapter l'utilisation des lignes d'E/S à vos besoins et de définir des éléments clefs de la configuration matérielle de la carte FOX.

La configuration du noyau se fera par `make -C os/linux menuconfig` (ou `config` ou encore `xconfig`) depuis la racine du SDK. On notera que l'interface de configuration, bien que proche de celle d'un noyau classique se voit étoffée d'un certain nombre d'éléments. Les plus remarquables sont :

General setup : C'est ici que se configurent certaines fonctionnalités intéressantes pour un système embarqué comme le *watchdog* ou encore le *debugger* kgdb (voir article de P. Ficheux dans GLMF 88)

Le point important de la configuration concerne l'enregistrement des paramètres. Si vous enregistrez simplement en quittant l'interface, un simple `.config` sera créé. Mais le `Makefile` à la racine utilisera le fichier de configuration `kernelconfig` (ou `kernelconfig-2.6`). La solution consiste donc à enregistrer la configuration dans un fichier alternatif placé à la racine du SDK (`kernelconfig.truc` par exemple) et, dans la configuration du SDK, à renseigner le champ `Kernel configuration file` de la section `General Configuration`.

Enfin, pour conclure cette section, je dirais qu'il est important, voire indispensable d'adapter la configuration du noyau à vos besoins. Ceci même en dehors des paramètres propres à la carte FOX. Un simple exemple concerne le support des différents types de systèmes de fichiers. Par défaut, le support des clefs USB (`usb-storage`) est activé, mais un petit `cat /proc/filesystems` annonce clairement la couleur : pas d'Ext2. Il faut donc, soit recompiler un noyau, soit se satisfaire de FAT et ne voir dans la clef USB qu'un support de stockage de données et non d'outils/bibliothèques acceptables.

HELLO WORLD

Vous ne pensiez pas sérieusement y couper ? Bien entendu, le classique du genre est d'une simplicité affligeante, puisque l'environnement est connu tout comme le compilateur. Nous avons donc (depuis la racine du SDK et après un `. init_env`) :

```
$ mkdir apps/hello
$ cd apps/hello

$ cat > hello.c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv)
{
  printf("Coucou Monde !\n");
  return(EXIT_SUCCESS);
}
^D
```

Il nous faut ensuite créer un `Makefile` approprié. Pour cela, nous reprenons tout simplement celui déjà fourni par un utilitaire installé par défaut (`setbits` voir ci-après) avec `$ cp ../utils/setbits/Makefile .` et nous l'adaptons :

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

PROGS     = hello

INSTDIR   = $(prefix)/bin/
INSTMODE  = 0755
INSTOWNER = root
INSTGROUP = root

all: $(PROGS)

install: all
          $(INSTALL) -d $(INSTDIR)
          $(INSTALL) -m $(INSTMODE) -o $(INSTOWNER) \
        -g $(INSTGROUP) $(PROGS) $(INSTDIR)

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

On utilisera ensuite :

```
$ make cris-axis-linux-gnu
make[1]: entrant dans le répertoire
       " /home/denis/FOX/devboard-R2_01/apps/hello "
rm -f hello *.o core
cp "/home/denis/FOX/devboard-R2_01/apps/
       hello/.tmp.target-makefrag" .target-makefrag
make[1]: quittant le répertoire
       " /home/denis/FOX/devboard-R2_01/apps/hello "
```

Cette commande crée simplement un fichier `.target-makefrag` contenant le nom du profil de compilation à utiliser. Ici `cris-axis-linux-gnu` pour un Linux 2.4 et GNU libc. Jetez simplement un œil au `Makefile` à la racine du SDK (ligne 180 à 220). Vous y trouverez une brève description de chaque profil.

Il ne vous reste plus qu'à utiliser la commande `make` pour obtenir un exécutable ELF pour le processeur Etrax. Dès lors, vous pouvez faire la copie du fichier binaire `hello` dans le `/mnt/flash` (via FTP ou SSH) et l'exécuter (via Telnet ou SSH).

Après la phase d'essai et de débogage, qui peut être grandement simplifiée par une connexion Telnet et le montage d'un volume NFS, il est temps d'intégrer l'application dans la construction de l'image. Rien de plus simple. Vous n'avez qu'à éditer le `Makefile` à la racine du SDK et ajouter `apps/hello` dans `SUBDIRS`.

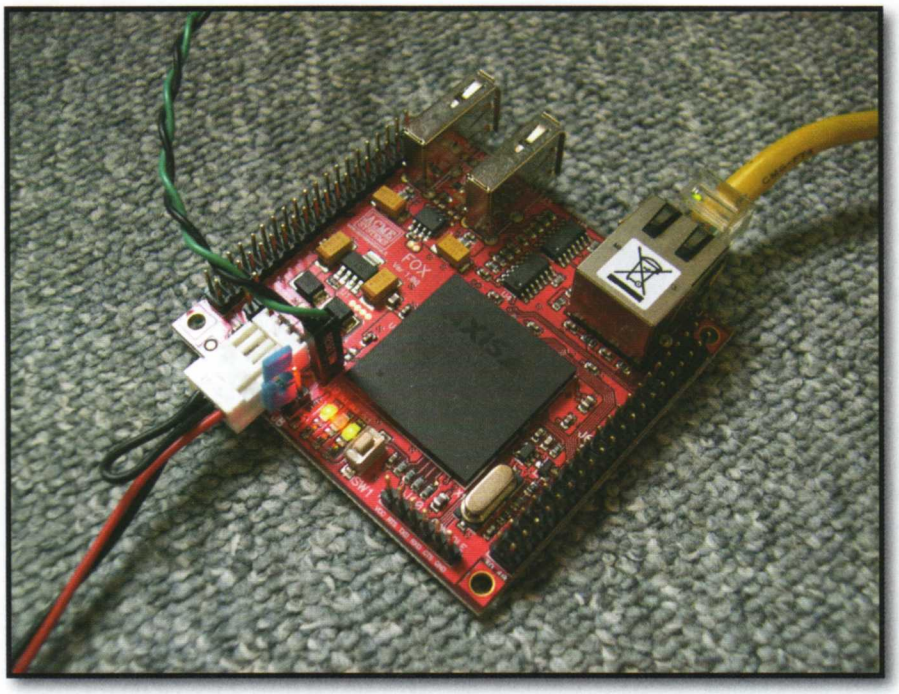
> Note : En exécutant le script `configure` à la racine du SDK suite à la reconfiguration par `make menuconfig` par exemple, vous générerez un nouveau `Makefile` et perdrez la modification de la variable `SUBDIRS`.

HELLO WORLD, VERSION LED

Dans sa configuration la plus directe, la carte FOX offre 48 E/S ce qui, techniquement, nous offre 48 possibilités pour y connecter une LED et une résistance pour nos essais. Cependant, dans la configuration par défaut, une partie des ports E/S sont utilisés pour :

- l'interrupteur et les LED CMS soudées sur la carte ;
- le bus i2c destiné à connecter une RTC ;
- les deux contrôleurs USB ;
- deux ports série.

Tout ceci peut être désactivé bien sûr, mais nous avons encore l'embarras du choix. Reportez-vous à la page http://www.acmesystems.it/?id=18 du site d'ACME pour la correspondance port/broche.



Note : Lors de l'interfaçage de montages sur les ports, n'oubliez pas que la carte est en 3.3 volts (tolérant du 5 volts en entrée). Dans tous les cas, les sorties se feront sous cette tension. Si vous interfacez de l'électronique 5 volts, utilisez simplement des *buffers* ou des *latchs*. Alimentés avec un Vcc de 5 volts, un 74LS244 (Porte *bufferisée* 8 bits 3 états) ou un 74HCT573 considèrent comme un niveau haut toutes tensions entre 2 volts et Vcc. De plus, ce type de composant peut fournir jusqu'à 70mA par sortie (selon modèle). De quoi satisfaire la plupart des usages !

Les 48 E/S sont réparties en trois ports A, B et G. La direction pour chaque bit des ports A et B est définie dans la configuration du noyau. Par ailleurs, un masque définit si une application utilisateur peut ou non changer la direction pour chaque bit. Le port G, quant à lui, est divisé en plusieurs groupes. Certaines broches sont ainsi des E/S et d'autres uniquement des entrées ou des sorties pour un même bit. Enfin, toujours pour le port G, certains bits sont configurés en entrée ou en sortie par groupe de 8 bits uniquement.

Un certain nombre d'outils sont livrés avec le SDK :

- `statusled` : C'est un petit démon chargé de faire clignoter la ou les LED CMS soudée(s) à la carte à un intervalle défini. Un simple `statusled -h` vous en apprendra plus.

- `setbits` : Il s'agit d'un utilitaire générique permettant de définir l'état d'un bit pour un port de manière arbitraire. Il suffit pour cela de lui passer en argument différentes options, comme le port à utiliser (`-p`), le numéro du bit concerné (`-b`) et l'état 0/1 (`-s`). Bien entendu, ce sont surtout les sources de cet outil qui sont intéressants et qu'on trouvera dans `apps/utils/setbits`).

- `readbits` : C'est l'équivalent de `setbits` pour la lecture. Celui-ci, exécuté sans argument, retournera quelque chose comme `111XXX1X XX01XXX1 XXXXXXX11111111111111111XX11111X` où on trouve respectivement les bits du port A, B et G. Les `X` symbolisent les bits non utilisables en lecture (dédiés à une autre utilisation). Là encore, c'est le code source qui est le plus instructif.

CONCLUSION, ROHS ET FOX LX

Ce court (sic) article n'est qu'une introduction et un tour d'horizon aux développements autour de la carte d'ACME system. Il reste beaucoup à dire et beaucoup à faire. Le site d'ACME est riche en documentation pouvant servir de point de départ même si, souvent, il ne s'agit, là encore que d'introduction au sujet. C'est finalement sur le site d'AXIS qu'on pourra pleinement satisfaire ses besoins en documentation : **http://developer.axis.com/doc/hw/mcm4_16.html**.

Il paraît également important de signaler en conclusion de cet article que la carte utilisée pour les essais n'est plus produite dans l'état. Elle n'en est pas pour autant obsolète, loin de là. L'application récente (1er juillet 2006) des directives RoHS visant à protéger l'environnement des déchets électroniques a conduit la société ACME à produire une nouvelle carte FOX, la FOX LX.

Il s'agit techniquement de la même carte, avec les mêmes caractéristiques et disposant des mêmes fonctionnalités. Seul le *packaging* est différent. La carte présentée ici utilise un processeur que l'on pourrait presque qualifier de microcontrôleur, puisqu'il intègre directement la RAM, la mémoire Flash, l'interface Ethernet, etc. dans un composant dit « MCM » (*Multi Chip Module*). Or, justement, AXIS doit revoir la conception de la puce Etrax 100LX MCM pour se conformer aux directives RoHS.

En attendant l'arrivée des nouveaux composants MCM, ACME a donc décidé de produire une carte sur la base d'un processeur Etrax 100LX et de composants annexes (RAM, Flash, etc.). La nouvelle carte FOX LX est, pour l'heure, identique fonctionnellement à l'ancienne carte, mais de futures versions se déclineront jusqu'à 64 Mo de RAM et de Flash.

Tout ce qui a donc été dit ici est parfaitement valable pour la carte FOX LX.

MISC

Multi-System & Internet Security Cookbook

http://www.miscmag.com

ACTUELLEMENT EN KIOSQUE

LES ATTAQUES IPv6

NUMÉRO 27

SOMMAIRE

DOMAINE - EMAIL - HÉBERGEMENT - CRÉATION DE SITE - E-COMMERCE - RÉFÉRENCEMENT - BACKUP

AMEN RCS PARIS: B 421 527 797. IN WEB WE TRUST : Nous croyons au web * Voir conditions sur le site www.amen.fr Conditions Générales de Vente sur www.amen.fr. Prix au 01/09/2006, modifiables sans préavis. Photo : Getty : Erik Dreyer.

1Go Pack Web Nom +

. Nom de domaine : .fr, .eu, .com, .net, .org ...
. Hébergement web : 1 Go
. Messagerie : 1 compte POP/IMAP avec 1 Go de stockage, anti-virus/anti-spam

+ 1 application pré-installée au choix

. Blog, album photo, gestion de contenu (CMS)...

12 € HT/AN soit 14,35 € TTC/AN

2Go Pack Web Mail +

. Nom de domaine : .fr, .eu, .com, .net, .org ...
. Hébergement web : 2 Go
. Messagerie : 10 comptes POP/IMAP avec 20 Go de stockage, anti-virus/anti-spam

+ 2 applications pré-installées au choix

. FAQ, blog, album photo, gestion de contenu (CMS), forum...

24 € HT/AN soit 28,70 € TTC/AN

5Go Pack Web Pro +

. Nom de domaine : .fr, .eu, .com, .net, .org ...
. Hébergement web dynamique : 5 Go
. 10 bases MySQL 4 + PHP 5, Perl 5, Python, Ruby
. Messagerie : 1000 comptes POP/IMAP avec 25 Go de stockage, anti-virus/anti-spam

+ 5 applications pré-installées au choix

. osCommerce, forum, FAQ, blog, album photo, gestion de contenu (CMS)...

70,80 € HT/AN soit 84,68 € TTC/AN

Avec plus de 260 000 noms de domaine gérés et 210 000 sites hébergés, AMEN est l'un des principaux fournisseurs européens de services et d'hébergement Internet. Notre vocation : innover pour vous offrir des services toujours plus performants.
Si vous croyez au web, vous croirez en nous.

amen
IN WEB WE TRUST

www.amen.fr

0892 55 66 77
(0,34 € TTC/mn depuis la France 9H - 19H)