Acquisition de données sous Linux RTAI

Y. Morère

Résumé

Cet article rend compte de l'installation et la configuration d'un système d'aquisition embarqué sur un fauteuil électrique intelligent VAHM http://www.lasc.univ-metz.fr/. Ce système est basé sur une distribution GNU/Linux et un noyau temps réel RTAI. Les carte d'acquisition Measurement Computing (PCI-QUAD04 et PCI-DDA8/12) sont gérées par un module noyau et le système comedi.

Table des matières

1 Introduction

Le but de cette première partie est de présenter l'installation d'un noyau temps réel sur une distribution linux. En effet le noyau linux n'est pas temps réel par défaut. Il se veut assez généraliste pour permettre une utilisation dans le plus grand nombre d'application.

Dans une seconde partie nous verrons l'installation de Comedi et l'utilisation de module, et la gestion du port série sou linux, afin de pouvoir obtenir toutes les informations issues des capteurs du fauteuil.

1.1 Contexte du projet

La dernière décennie a vue le développement de nombreux projets concernant les fauteuils intelligents dont l'objectif consiste à ouvrir son utilisation à des personnes n'ayant pas les facultés physiques, voire mentale, à commander un fauteuil classique.

L'organe de commande privilégié est le joystick qui permet d'imprimer au fauteuil une vitesse linéaire et angulaire. Selon la pathologie, certaines personnes ne sont pas capables d'utiliser un tel organe.

Un aspect essentiel dans l'acceptation d'un fauteuil intelligent est sa simplicité d'utilisation, si possible proche de la commande d'un fauteuil classique et l'adaptabilité aux exigences et habitudes de l'utilisateur. Les travaux développés ici ont pour objectif la création d'une structure de commande d'un fauteuil telle que celle-ci s'adapte à l'environnement de manière transparente pour l'utilisateur. De même, les divers comportements sont calqués sur ses contraintes propres.

L'application des techniques de la robotique mobile pour assister les personnes handicapées dans le déplacement de leur fauteuil électrique promet de réelles et utiles applications. De par le monde, de nombreux projets ont soit déjà abouti ou sont actuellement en cours de développement.

Chacun de ces projets améliore un fauteuil du marché en lui offrant un ensemble de fonctionnalités d'assistance à la conduite. La plupart des fauteuils proposés sont destinés à une utilisation en environnement d'intérieur. Globalement, les architectures de commande, les capteurs et les commandes disponibles sont très voisines. Seules les méthodes utilisées et les résultats obtenus conduisent à quelques différences.

L'intelligence des commandes est essentiellement basée sur la disponibilité d'informations concernant à la fois la localisation du fauteuil et l'environnement l'entourant. Les données provenant des codeurs incrémentaux permettent de naviguer en maintenant une orientation constante. Dans des environnements encombrés d'obstacles, les données de localisation et les données provenant des capteurs de perception doivent être fusionnées afin de suivre une direction donnée. L'observation permanente de l'environnement permet d'effectuer des commandes référencées capteurs comme le suivi de mur ou le suivi d'espace libre. Le choix de la meilleure commande à exécuter dans un contexte donné n'est souvent pas évident pour une personne non familiarisée aux machines autonomes. Nous proposons dans ce papier une méthode permettant au système d'assister la personne handicapée dans le choix de la commande ou de l'imposer pour des contextes particuliers. Pour cela, les données de perception de l'environnement local sont associées à une base de connaissances d'experts en robotique en utilisant une méthode de raisonnement par cas. De ce fait, l'utilisateur ne s'occupe plus que de deux actions :
▷ Commencer le déplacement vers une direction donnée.
▷ Arrêter le fauteuil.

2 Installation

Notre choix se porte sur RTAI (Real Time Application Interface) http://www.rtai.org/, un projet cousin européen de RTLinux.

RTAI n'est pas un systèmes d'exploitation temps réel comme peut l'être VXworks ou QNX. RTAI est basé sur un noyau linux et permet d'avoir un noyau complètement préemptible.

En effet linux est un noyau standard à temps partagé et fournit de bonnes performances dans un système multitâche et multi-utilisateur. Mais il ne possède aucun support du temps réel.

Il est donc nécessaire de modifier les sources du noyau, en ce qui concerne la gestion des interruptions, et la politique d'ordonancement.

Il s'agit d'un noyau temps réel qui va se placer à coté du noyau Linux. Le noyau temps réel organise l'odonnancement des demandes temps réelles, et délègue au noyau Linux, les tâches non prioritaires.

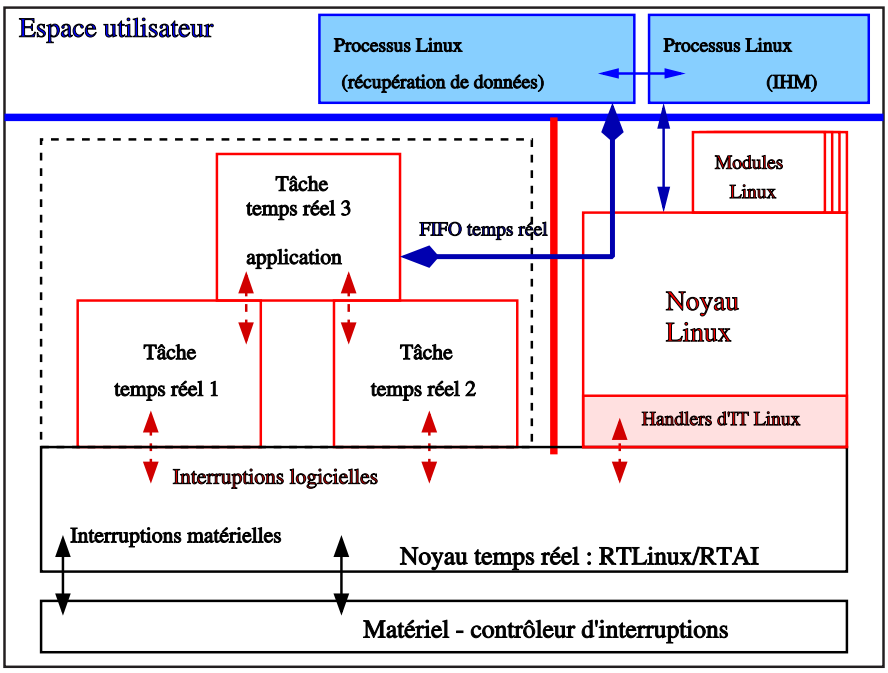

FIG. 1 – Intégration d'un noyau temps réel dans une système Linux

L'installation se déroule en deux temps. Il faut tout d'abord patcher le noyau Linux, afin de pouvoir intégrer le temps réel. Ensuite on compile les utilitaires temps réels de la distribution RTAI, ici une "vesuvio", afin de pouvoir profiter de l'ordonnanceur temps réel.

On va partir d'un noyau 2.6.8 vanilla (téléchargé sur `ftp://ftp.kernel.org`), mais les sources d'un noyau linux Debian conviennent aussi.

Pour des raisons pratiques, toutes la suite se fera en utilisateur `root`.

On commence par décompresser les sources du noyau dans les répertoires `/usr/src`. Ce n'est pas obligatoire, mais il s'agit du chemin par défaut ou les programmes et RTAI vont chercher les sources du noyau par défaut.

```
vahm2:/usr/src# cd /usr/src
vahm2:/usr/src# tar xjf linux-2.6.8.1.tar.bz2
vahm2:/usr/src# ln -s linux-2.6.8.1 linux
```

On crée en suite un lien symbolique vers ce répertoire.

On récupère les sources de RTAI sur `http://www.rtai.org/modules.php?name=Downloads&d_op=viewdownload&cid=1`. Il s'agit de la distribution `vesuvio` dans notre cas.

```
vahm2:/usr/src# tar xjf rtai-3.1.tar.bz2
```

Ensuite on applique le patch au noyau.

```
vahm2:/usr/src/linux# patch -p1 < ../rtai-3.1/rtai-core/arch/i386/patches/hal7-2.6.8.1.p
```

Puis afin de compiler le noyau sans trop changer votre distribution, il suffit de copier la configuration de votre noyau actuel.

```
vahm2:/usr/src/linux# cp /boot/config-2.6.8-2-686 .config
```

Ensuite on configure la noyau par

```
vahm2:/usr/src/linux# make menuconfig
```

pour la configuration avec `ncurses` ou

```
vahm2:/usr/src/linux# make gconfig
```

pour la configuration avec interface graphique version Gnome ou

```
vahm2:/usr/src/linux# make kconfig
```

pour la configuration avec interface graphique version Kde.
Il faut faire attention aux choses suivantes :
▷ `Adeos` est selectionné (`Adeos Support -> Adeos Support`)
▷ `Loadable module support -> Module versioning support` est désactivé
▷ `Kernel hacking -> Compile the kernel with frame pointers` est désactivé

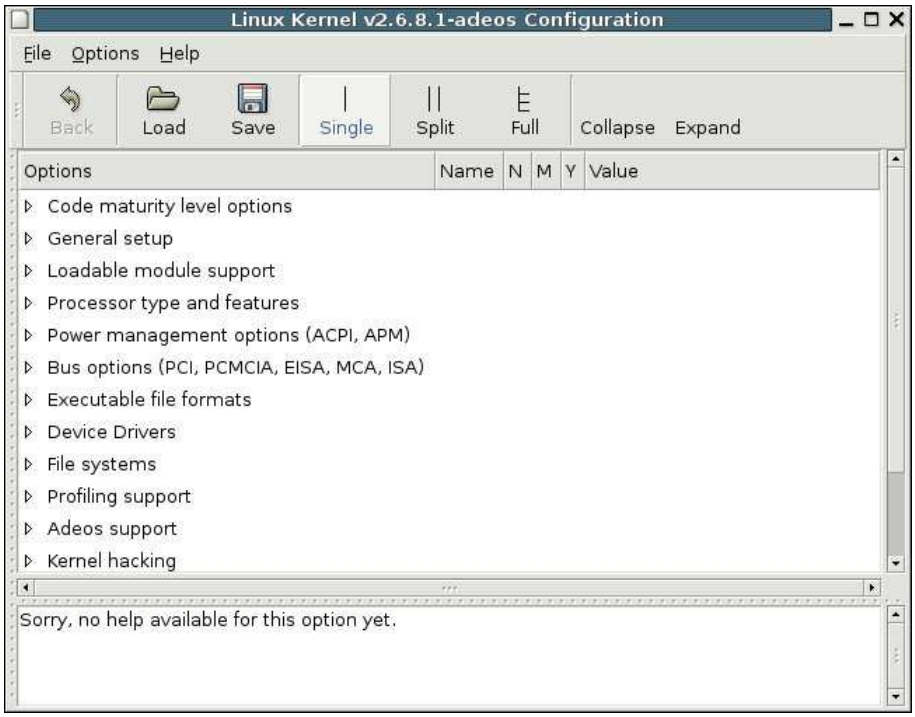

FIG. 2 – Configuration du noyau version Gnome

On peut alors compiler le noyau pour une version 2.6 cela donne :

```
vahm2:/usr/src/linux# make
vahm2:/usr/src/linux# make modules_install
vahm2:/usr/src/linux# mkinitrd -o initrd.img-2.6.8.1-adeos /lib/modules/2.6.8.1-adeos
vahm2:/usr/src/linux# make install
make[1]: « arch/i386/kernel/asm-offsets.s » est à jour.
  CHK     include/linux/compile.h
Kernel: arch/i386/boot/bzImage is ready
sh /usr/src/linux-2.6.8.1/arch/i386/boot/install.sh 2.6.8.1-adeos arch/i386/boot/bzImage
```

```
In order to use the new kernel image you have just installed, you
will need to reboot the machine.  First, however, you will need to
either make a bootable floppy diskette, re-run LILO, or have GRUB
installed.

Checking for ELILO...No

GRUB is installed. To automatically switch to new kernels, point your
default entry in menu.lst to /boot/arch/i386/boot/bzImage-2.6.8.1-adeos
vahm2:/usr/src/linux#
vahm2:/usr/src/linux# mkinitrd -o initrd.img-2.6.8.1-adeos /lib/modules/2.6.8.1-adeos
vahm2:/usr/src/linux# cp initrd.img-2.6.8.1-adeos /boot
```

On peut alors compiler le noyau pour une version 2.4 cela donne :

```
vahm2:/usr/src/linux# make clean && make dep && make bzimage
vahm2:/usr/src/linux# make modules_install
vahm2:/usr/src/linux# mkinitrd -o initrd.img-2.6.8.1-adeos /lib/modules/2.6.8.1-adeos
vahm2:/usr/src/linux# cp
make[1]: « arch/i386/kernel/asm-offsets.s » est à jour.
  CHK     include/linux/compile.h
Kernel: arch/i386/boot/bzImage is ready
sh /usr/src/linux-2.6.8.1/arch/i386/boot/install.sh 2.6.8.1-adeos arch/i386/boot/bzImage
In order to use the new kernel image you have just installed, you
will need to reboot the machine.  First, however, you will need to
either make a bootable floppy diskette, re-run LILO, or have GRUB
installed.

Checking for ELILO...No

GRUB is installed. To automatically switch to new kernels, point your
default entry in menu.lst to /boot/arch/i386/boot/bzImage-2.6.8.1-adeos
vahm2:/usr/src/linux#
vahm2:/usr/src/linux# mkinitrd -o initrd.img-2.6.8.1-adeos /lib/modules/2.6.8.1-adeos
vahm2:/usr/src/linux# cp initrd.img-2.6.8.1-adeos /boot
```

Il faut ensuite reconfigurer votre lilo ou grub. Pour lilo :

```
image=/boot/vmlinuz-2.6.8.1-adeos
        label="Linux-2.6.8.1RT"
        root=/dev/hda4
        initrd=/boot/initrd.img-2.6.8.1-adeos
        read-only
```

Pour grub :

```
title           Debian GNU/Linux, kernel 2.6.8.1 real time
root            (hd0,3)
kernel          /boot/vmlinuz-2.6.8.1-adeos root=/dev/hda4 ro
initrd          /boot/initrd.img-2.6.8.1-adeos
savedefault
boot
```

Le noyau est maintenant installé. Nous allons compiler RTAI.

```
vahm2:/usr/src/linux#cd ../rtai3.1
vahm2:/usr/src/linux#make menuconfig ou gconfig ou kconfig
vahm2:/usr/src/linux#make
vahm2:/usr/src/linux#make install
```

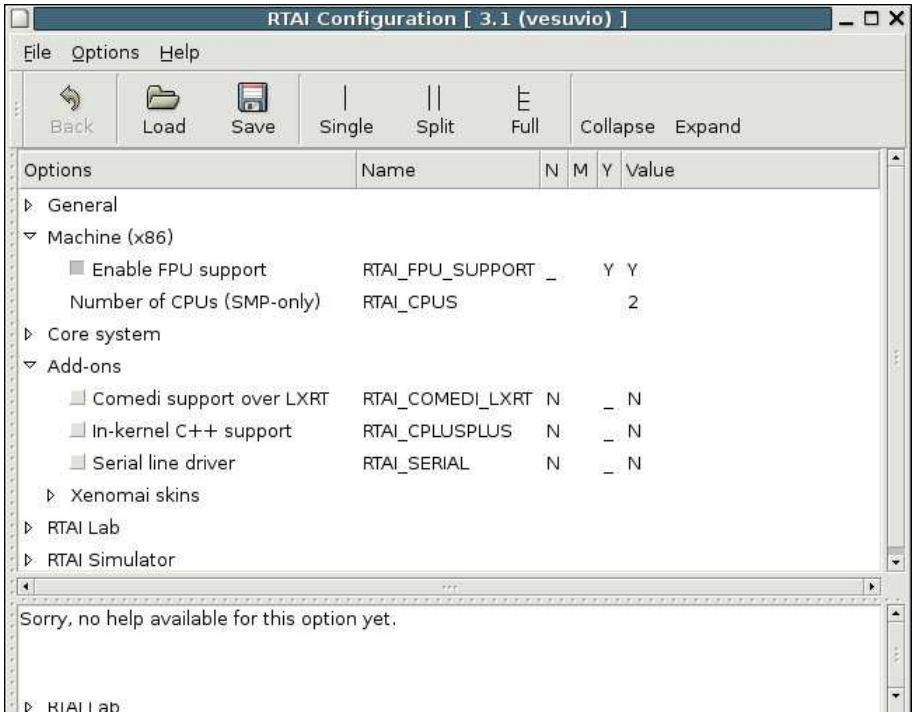

FIG. 3 – Configuration de RTAI version Gnome

Si vous désirez utiliser les portes séries de votre PC, il faut les activer `Add-ons -> Serial line driver`. Si vous voulez juste faire un test de RTAI, laissez la configuration par défaut.

S'il n'y a aucune erreur, vous pouvez alors redémarrer en choisissant le nouveau noyau avec le support temps réel.

Une fois que tout est bien démarré, il est possible de tester RTAI à l'aide de la commande :

```
# cd /usr/realtime/testsuite/kern/latency/
# ./run
```

Maintenant vous devriez voir les sorties min et max des temps de latence

Remarque : Si vous utilisez une Fedora 2 et que vous obtenez des "kernel panic", mieux vaut utiliser une version CVS de RTAI.

3 Comedi

Cette partie rend compte de l'installation des librairies et utilitaires du projet `comedi` http://www.comedi.org.

Pour l'installation de comedi, une des premières choses à vérifier est la version de autogen ; Afin que tout se passe bien il faut la version 1.7 minimum.

```
yann@vahm2:~/realtime$ tar xzf comedi.tar.gz
yann@vahm2:~/realtime$ cd comedi
yann@vahm2:~/realtime/comedi$ ./autogen.sh
yann@vahm2:~/realtime/comedi$ make
yann@vahm2:~/realtime/comedi$ make check
yann@vahm2:~/realtime/comedi$ su
vahm2:/home/yann/realtime/comedi# make install
vahm2:/home/yann/realtime/comedi# make dev
vahm2:/home/yann/realtime/comedi# ls /dev/comedi*
```

Installation de comedilib

```
yann@vahm2:~/realtime/comedi$ cd ../comedilib
yann@vahm2:~/realtime/comedilib$ ./autogen.sh
yann@vahm2:~/realtime/comedilib$ ./configure
yann@vahm2:~/realtime/comedilib$ make
yann@vahm2:~/realtime/comedilib$ su
yann@vahm2:~/realtime/comedilib# make install
yann@vahm2:~/realtime/comedilib# make test
```

Tout est installé, il faut maintenant configuré la/les cartes qui se trouve sur le PC.
D'après le message https://mail.rtai.org/pipermail/rtai/2004-October/008854.html

```
The "depmod" command can't solve all the dependencies of the comedi.o (or
comedi.ko) module. You have to install

rtai_hal
rtai_lxrt

from the /usr/realtime/modules directory before any comedi driver.

In your case:

insmod /usr/realtime/modules/rtai_hal.ko
insmod /usr/realtime/modules/rtai_lxrt.ko

modprobe ni_atmio
```

Comme on a compilé avec RTAI, il faut charger les modules temps réels avant le module de la carte d'acquisition.
Sinon on reçoit les messages d'erreurs suivants :

```
vahm2:/home/yann/realtime/comedi# modprobe cb_pcidda
WARNING: Error inserting comedi (/lib/modules/2.6.8.1-adeos/comedi/comedi.ko): Unknown s
WARNING: Error inserting 8255 (/lib/modules/2.6.8.1-adeos/comedi/8255.ko): Unknown symbo
FATAL: Error inserting cb_pcidda (/lib/modules/2.6.8.1-adeos/comedi/cb_pcidda.ko): Unkno
vahm2:/home/yann/realtime/comedi#
```

et avec un dmesg

```
comedi: Unknown symbol rt_request_srq
comedi: Unknown symbol rtai_domain
comedi: Unknown symbol rt_shutdown_irq
comedi: Unknown symbol rt_enable_irq
comedi: Unknown symbol rt_pend_linux_srq
comedi: Unknown symbol rt_umount
comedi: Unknown symbol rt_startup_irq
comedi: Unknown symbol rt_free_srq
comedi: Unknown symbol rt_mount
comedi: Unknown symbol rt_request_irq
comedi: Unknown symbol rt_release_irq
8255: Unknown symbol comedi_buf_put
8255: Unknown symbol comedi_event
8255: Unknown symbol range_unipolar5
8255: Unknown symbol comedi_driver_unregister
8255: Unknown symbol comedi_driver_register
cb_pcidda: Unknown symbol subdev_8255_init
cb_pcidda: Unknown symbol subdev_8255_cleanup
cb_pcidda: Unknown symbol comedi_driver_unregister
cb_pcidda: Unknown symbol comedi_driver_register
```

Voici, en opérant dans le bon ordre :

```
vahm2:/home/yann/realtime/comedi# insmod /usr/realtime/modules/rtai_hal.ko
vahm2:/home/yann/realtime/comedi# insmod /usr/realtime/modules/rtai_lxrt.ko
vahm2:/home/yann/realtime/comedi# modprobe cb_pcidda
```

et `dmesg` donne

```
Adeos: Domain RTAI registered.
RTAI[hal]: 3.1 mounted over Adeos 2.6r7/x86.
RTAI[hal]: compiled with gcc version 3.3.5 (Debian 1:3.3.5-8).
RTAI[malloc]: loaded (global heap size=131072 bytes).
RTAI[sched_lxrt]: loaded (LxrtMode 0).
RTAI[sched_lxrt]: timer=periodic (8254-PIT).
RTAI[sched_lxrt]: standard tick=1000 hz, CPU freq=2392536000 hz.
RTAI[sched_lxrt]: timer setup=2010 ns, resched latency=2688 ns.
comedi: version 0.7.69 - David Schleef <ds@schleef.org>
rt_pend_tq: RT bottom half scheduler initialized OK
```

Tout est donc installé convenablement en mémoire.
Pour les enlever ces modules il faut opérer dans l'ordre inverse.

```
vahm2:/home/yann/realtime/comedi# rmmod cb_pcidda
vahm2:/home/yann/realtime/comedi# rmmod rtai_lxrt
vahm2:/home/yann/realtime/comedi# rmmod  8255
vahm2:/home/yann/realtime/comedi# rmmod comedi
vahm2:/home/yann/realtime/comedi# rmmod rtai_hal
vahm2:/home/yann/realtime/comedi#
```

Un fois que tous les modules sont chargés convenablement, (un petit `lsmod` devrait arranger tout cela), on passe à la configuration de la carte.

Il faut maintenant configurer `comedi` pour qu'il puisse trouver la carte et ainsi dialoguer avec elle. Pour cela il faut utiliser le programme `comedi_config` et lui passer les bons paramètres.

Je vous renvoie à l'adresse http://www.comedi.org/doc/x1672.html, pour retrouver la manière dont se configure votre carte. Dans notre cas, il s'agit de la carte supportée par le module `cb_pcidda.o`.

La commande est composée de plusieurs chose :

```
comedi_config -v /dev/comedi0 cb_pcidda 2,13
```

Nous alons expliquer les différents paramètres de la commande précédente :

- ▷ L'option `-v` permet de passer en mode verbeux, afin d'obtenir un maximum d'information lors de l'utilisation de la commande ;
- ▷ `/dev/comedi0` permet de sélectionné le fichier spécial comedi que l'on va utiliser avec la carte ;
- ▷ `cb_pcidda` permet de spécifié quel module, nous allons utilisé en correspondante avec `/dev/comedi0` afin de réaliser nos opérations d'acquisition ;
- ▷ finalement nous donnons les paramètres les plus important à `comedi_config`, qui lui permettent de trouver la carte d'aquisition relative au module utilisé pour la lier au fichier spécial `/dev/comedi0`. Ces deux paramètres peuvent être retrouver grâce à la commande `lspci` comme montrer dans la suite.

  `0000:02:0d.0 ffff: Measurement Computing PCI-DDA08/12 (rev 02)`

  C'est cette ligne qu'il s'agit d'analyser. On cherche donc les paramètres de la carte Measurement Computing PCI-DDA08/12. Pour cela il faut décoder les 3 premiers champs de la ligne séparés par des `:`. Un `man` de la commande `lspci` pous indique que cette ligne est formatée de la manière suivante : `[[<bus>] :][<slot>][.[<func>]]`. Les `0000` représente le numéro du bus PCI, le `02` représente le slot PCI, et `0d` en héxadécimal soit 13 en décimal, la fonction. Ce sont ces deux derniers paramètres qu'il faut renseigner dans notre cas. Même si la documentation de comedi nous dit qu'il s'agit du bus et du slot (soit 0 et 2 dans notre cas).

```
Configuration options:
  [0] - PCI bus of device (optional)
  [1] - PCI slot of device (optional)
  If bus/slot is not specified, the first available PCI
  device will be used.

  Only simple analog output writing is supported
```

Voici ce que donne la commande `lspci` sur la machine de test :

```
vahm2:/home/yann# lspci
0000:00:00.0 Host bridge: Intel Corp. 82845 845 (Brookdale) Chipset Host Bridge (rev 11)
0000:00:01.0 PCI bridge: Intel Corp. 82845 845 (Brookdale) Chipset AGP Bridge (rev 11)
0000:00:1e.0 PCI bridge: Intel Corp. 82801 PCI Bridge (rev 05)
0000:00:1f.0 ISA bridge: Intel Corp. 82801BA ISA Bridge (LPC) (rev 05)
0000:00:1f.1 IDE interface: Intel Corp. 82801BA IDE U100 (rev 05)
0000:00:1f.2 USB Controller: Intel Corp. 82801BA/BAM USB (Hub #1) (rev 05)
0000:00:1f.3 SMBus: Intel Corp. 82801BA/BAM SMBus (rev 05)
0000:00:1f.4 USB Controller: Intel Corp. 82801BA/BAM USB (Hub #2) (rev 05)
0000:00:1f.5 Multimedia audio controller: Intel Corp. 82801BA/BAM AC'97 Audio (rev 05)
0000:01:00.0 VGA compatible controller: Silicon Integrated Systems [SiS] 315PRO PCI/AGP
```

```
0000:02:08.0 Ethernet controller: Intel Corp. 82801BA/BAM/CA/CAM Ethernet Controller (re
0000:02:0c.0 ffff: Measurement Computing PCI-QUAD04 (rev 02)
0000:02:0d.0 ffff: Measurement Computing PCI-DDA08/12 (rev 02)
```

Et voici le résultat de la commande configuration :

```
vahm2:/home/yann# comedi_config -v /dev/comedi0 cb_pcidda 2,13
configuring driver=cb_pcidda 2,13,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
vahm2:/home/yann# more /proc/comedi
comedi version 0.7.69
format string: "%2d: %-20s %-20s %4d",i,driver_name,board_name,n_subdevices
 0: cb_pcidda            pci-dda08/12            3
cb_pcidda:
 cb_pcidda
8255:
 8255
vahm2:/home/yann#
```

Puis la commande de test que l'on redirige vers un fichier car la réponse est assez longue :

```
comedi_test -v -f /dev/comedi0 > test_dda
```

Le contenu du fichier `test_dda` :

```
test_dda
I: Comedi version: 0.7.69
I: Comedilib version: unknown =)
I: driver name: cb_pcidda
I: device name: pci-dda08/12
I:
I: subdevice 0
I: testing info...
rev 1
I: subdevice type: 2 (analog output)
  number of channels: 8
  max data value: 4095
  ranges:
    all chans: [-10,10] [-5,5] [-2.5,2.5] [0,10] [0,5] [0,2.5]
I: testing insn_read...
rev 1
E: comedi_do_insn: Invalid argument
I: testing insn_read_0...
E: comedi_do_insn: Invalid argument
I: testing insn_read_time...
rev 1
comedi_do_insn: -1
W: comedi_do_insn: errno=22 Invalid argument
W: comedi_do_insn: returned -1 (expected 3)
read time: -1065571902 us
I: testing cmd_no_cmd...
got EIO, good
```

```
I: testing cmd_probe_src_mask...
not applicable
I: testing cmd_probe_fast_1chan...
not applicable
I: testing cmd_read_fast_1chan...
not applicable
I: testing cmd_write_fast_1chan...
not applicable
I: testing cmd_logic_bug...
not applicable
I: testing cmd_fifo_depth_check...
not applicable
I: testing cmd_start_inttrig...
not applicable
I: testing mmap...
not applicable
I: testing read_select...
not applicable
I: testing bufconfig...
buffer length is 0, good
I:
I: subdevice 1
I: testing info...
rev 1
I: subdevice type: 5 (digital I/O)
  number of channels: 24
  max data value: 1
  ranges:
    all chans: [0,5]
I: testing insn_read...
rev 1
comedi_do_insn returned 1, good
I: testing insn_read_0...
comedi_do_insn returned 0, good
I: testing insn_read_time...
rev 1
comedi_do_insn: 3
read time: 7 us
I: testing cmd_no_cmd...
got EIO, good
I: testing cmd_probe_src_mask...
not applicable
I: testing cmd_probe_fast_1chan...
not applicable
I: testing cmd_read_fast_1chan...
not applicable
I: testing cmd_write_fast_1chan...
not applicable
I: testing cmd_logic_bug...
not applicable
```

```
I: testing cmd_fifo_depth_check...
not applicable
I: testing cmd_start_inttrig...
not applicable
I: testing mmap...
not applicable
I: testing read_select...
not applicable
I: testing bufconfig...
buffer length is 0, good
I:
I: subdevice 2
I: testing info...
rev 1
I: subdevice type: 5 (digital I/O)
  number of channels: 24
  max data value: 1
  ranges:
    all chans: [0,5]
I: testing insn_read...
rev 1
comedi_do_insn returned 1, good
I: testing insn_read_0...
comedi_do_insn returned 0, good
I: testing insn_read_time...
rev 1
comedi_do_insn: 3
read time: 7 us
I: testing cmd_no_cmd...
got EIO, good
I: testing cmd_probe_src_mask...
not applicable
I: testing cmd_probe_fast_1chan...
not applicable
I: testing cmd_read_fast_1chan...
not applicable
I: testing cmd_write_fast_1chan...
not applicable
I: testing cmd_logic_bug...
not applicable
I: testing cmd_fifo_depth_check...
not applicable
I: testing cmd_start_inttrig...
not applicable
I: testing mmap...
not applicable
I: testing read_select...
not applicable
I: testing bufconfig...
buffer length is 0, good
```

4 Mise en oeuvre de Comedi

La partie suivante est tirée du rapport de projet de maîtrise EEA 2005 de SCHERER Damien et ZGRZENDEK Christophe.

4.1 Gestion de PCIDDA à l'aide de COMEDI

COMEDI signifie COntrol and MEasurement Device Interface. IL s'agit en fait d'une communauté de programmeurs "libres" écrivant des routines de services permettant une plus grande facilité d'utilisation des cartes d'acquisition sous linux. On y trouve : des développeurs de drivers et des développeurs logiciels censés améliorer la fiabilité des librairies.

A notre grande chance, les drivers de PCI DDA ont déjà été écrits par un utilisateur de COMEDI. Par conséquent notre seule ambition pour cette carte est de la faire fonctionner et de pouvoir commander le moteur à partir du PC distant, ce qui serait une preuve définitive de la bonne marche du driver.

L'installation de COMEDI nécessite celle de autogen version 1.7 minimum et la copie des sources du noyau linux (dans notre cas un 2.4) Ceci effectué, on pourra commencer l'installation de COMEDI et de comedilib sur la machine. On notera d'ailleurs que ce dernier est un paquet source de debian et que son installation se fait donc en un seul click.

Bien que le programme COMEDI soit installé, nous n'avons pas encore accès à la carte pci-dda. Pour cela nous devons lier le fichier du pilote à un fichier d'entrée/sortie *comedi* La commande `comedi_config` a pour rôle cette configuration.

```
comedi_config -v /dev/comedi0 cb_pcida 2,13
```

Nous allons expliquer les différents paramètres de la commande précédente :
- ▷ L'option `-v` permet de passer en mode verbeux, afin d'obtenir un maximum d'information lors de l'utilisation de la commande.
- ▷ `/dev/comedi0` permet de sélectionner le fichier spécial COMEDI que l'on va utiliser avec la carte.
- ▷ `cb_pcidda` permet de specifier quel module nous allons utiliser en correspondance avec `/dev/comedi0`
- ▷ `2,13`, les champs 2 et 13 correspondent respectivement au slot pci et sa fonction. Ces paramètres peuvent être retrouvés à l'aide de `lspci`.

La commande `./board_info` permet de récupérer de nombreuses informations sur la carte (nombre de canaux, nombre de subdevice ?) Nous reviendrons plus en détail sur cette commande dans la suite de cette documentation. Son bon fonctionnement permet d'assurer la bonne liaison avec COMEDI.

Désormais la carte pci-dda est opérationnelle. Toute écriture ou lecture transitera par le fichier `comedi0`. Comme nous le verrons bientôt, il reste encore à savoir se servir des services d'interfaçage et connaître la syntaxe de communication.

5 Compilation driver pci_quad04

5.1 Gestion de PCI QUAD

Il n'existe pas de driver COMEDI pour la carte PCI QUAD. Une des solutions pourrait être de l'écrire, la deuxième serait de le programmer sans passer par COMEDI. Nous découvrirons bientôt qu'aucune des solutions envisagées ne va être réalisée.

Notre étude s'est tout d'abord portée sur la programmation modulaire d'un driver. En effet, bien que la souplesse de COMEDI soit un atout considérable, l'écriture du driver semble assez spécialisée et la documentation forcément plus rare que la programmation modulaire. Lors de la programmation de drivers, il est nécessaire d'accéder à certaines fonctions du noyau. On ne programme alors plus en utilisateur mais en super utilisateur. Cette programmation est peu confortable et nécessite une grande précaution sous peine de se retrouver face à une erreur du type `kernel panik` qui pourrait nécessiter un reformatage complet du système. Linux est très commode dans ce domaine car il permet, au lieu de reprogrammer le noyau, de coder un module capable de s'insérer dans ce dernier.

On évite ainsi un redémarrage du système. L'insertion d'un module se réalise à l'aide de `insmod`. La commande `lsmod` permet de lister les modules en mémoire et `rmmod` de les retirer.

Etudions par exemple le module `mhello.c`

```
#include <linux/module.h>

int init_module(void)
{
printk("<1>Hello, world\n");
return 0;
}

void cleanup_module(void)
{
printk("<1>Goodbye cruel world\n");
}
```

`printk` est l'équivalent de printf dans l'espace noyau. La compilation est assez particulière. Il s'agit de faire correspondre la version des headers avec celle du noyau :

```
gcc -O -DMODULE -D__KERNEL__ -c mhello.c
```

Au chargement du module, nous aurons donc :

```
# insmod mhello
# dmesg | tail -1
Hello, world
```

Au déchargement, on a :

```
# rmmod mhello
# dmesg | tail -1
Goodbye cruel world
```

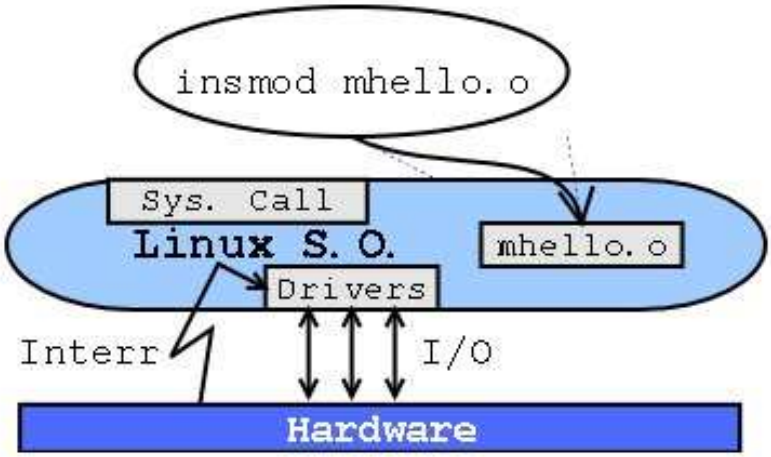

FIG. 4 – Insertion d'un module noyau

La programmation du driver demande une connaissance approfondie du matériel. Lors d'un échange de courriel à propos de la configuration d'un registre de notre carte PCI-QUAD, nous avons par hasard rencontré John Conner (conner@empiredi.com) ayant déjà réalisé ce driver mais qui ne l'avait pas mis en ligne. Pourquoi donc réinventer la roue ? Si la procédure de test s'avère concluante, nous décidons d'utiliser le travail déjà fait.

Ce pilote est tout d'abord testé sur un PC personnel. La carte est enfichée dans un emplacement PCI libre, puis on démarre le système sous Linux. Après quelques soucis de compilations dus aux compatibilités de version headers / noyau, nous finissons par parvenir à nos fins.

```
gcc -D__KERNEL__ -I/usr/src/kernel-headers-2.4.27-2-386/include -DMODULE -Wall-O2 -o pci
```

La commande `insmod pciquad` nous permet d'insérer le module dans le noyau sans problème. Un programme python permet de tester la bonne marche du module

```
#!/usr/bin/python
#
# Test interface to the PCI or CIO QUAD04 board.
#
#  Copyright (C) 2003, John Conner (conner@empiredi.com)
#
#
#  This program is free software; you can redistribute
#  it and/or modify it under the terms of the GNU General
#  Public License as published by the Free Software
#  Foundation; either version 2 of the License, or (at your
#  option) any later version.
#
#  This program is distributed in the hope that it will be
#  useful, but WITHOUT ANY WARRANTY; without even the implied
#  warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
#  PURPOSE.  See the GNU General Public License for more
```

```
#  details.
#
#  You should have received a copy of the GNU General Public
#  License along with this program; if not, write to the
#  Free Software Foundation, Inc., 59 Temple Place,
#  Suite 330, Boston, MA  02111-1307  USA
#
# Copyright 2003 by Empire Digital Instruments and
#                   ImageMap Inc. , All Rights Reserved.
#
"""
A simple minded test of the quad04 drivers. Reads counter 1
every 0.1 second and prints the value. If an encoder is
attached the count can be seen to go up and down as the
shaft is turned back and forth.

The xxx_quad04 dirvers are installed using the following
commands as root:
    # insmod pci_quad04.o
        or
    # insmod cio_quad04.o

It should be possible to have one of each board in the system
and drivers for both installed. This program will find the
cio_quad04 in that case.

"""
# $Id: test_1.py,v 1.3 2003/12/12 18:09:43 conner Exp $
rcsid = "$Id: test_1.py,v 1.3 2003/12/12 18:09:43 conner Exp $"
#

import sys, time

cio_cntr_name = '/proc/cio_quad04/cntr_1'
pci_cntr_name = '/proc/pci_quad04/cntr_1'

def get_cntr_1():
    try:
        # first try to open the cio_quad04 counter 1
        file_1 = open( cio_cntr_name, 'r' )
    except:
        try:
            # try to open the pci_quad04 counter 1
            file_1 = open( pci_cntr_name, 'r' )
        except:
            print "Unable to open a quad04 counter."
            print "Be sure the driver is installed."
            sys.exit()

    line = file_1.readline()
```

```
    file_1.close()
    return line

while 1:
    line = get_cntr_1()
    print line,
    time.sleep(0.1)
```

Cette version du driver est prévue pour un noyau 2.4. Quelques modifications sont necessaires pour le faire fonctionner avec un noyau 2.6.

Bien que le driver soit opérationnel, il comporte quelques défauts.

Il ne s'agit pas d'un driver de type COMEDI. Pour lire une donnée, il est nécessaire d'ouvrir à chaque fois le fichier `cntr_1` dans lequel le driver écrit la donnée du canal 1. Ce driver ne gère pas le temps réel.

Ces désavantages sont largement pondérés par le gain de temps, de développement et de simplicité. Pour que ce driver gère le temps réel, il serait nécessaire de le faire évoluer vers un pilote de type COMEDI. Ceci est du domaine du possible mais il serait faux de penser que cela ne nécessite que quelques changements mineurs. L'écriture de gestionnaires de périphériques COMEDI est très spécialisée est diffère du fonctionnement de notre driver.

Les modifications du code source sont les suivantes :

```
yann@tuxpowered:~/Projects/testvahm/quad04$ diff pci_quad04.c pci_quad04_26.c
42d41
<
43a43
> #include <linux/moduleparam.h>
45a46,50
> #include <linux/stat.h>
>
> //#include <linux/module.h>
> //#include <linux/kernel.h>
> //#include <linux/init.h>
yann@tuxpowered:~/Projects/testvahm/quad04$
```

et le `Makefile` modifié :

```
#
#  Make file for Quadrature Encoder driver.
#
# Copyright (c) 2003 by Empire Digital Instruments
#
#
#  This program is free software; you can redistribute
#  it and/or modify it under the terms of the GNU General
#  Public License as published by the Free Software
#  Foundation; either version 2 of the License, or (at your
#  option) any later version.
#
#  This program is distributed in the hope that it will be
#  useful, but WITHOUT ANY WARRANTY; without even the implied
```

```
#  warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
#  PURPOSE.  See the GNU General Public License for more
#  details.
#
#  You should have received a copy of the GNU General Public
#  License along with this program; if not, write to the
#  Free Software Foundation, Inc., 59 Temple Place,
#  Suite 330, Boston, MA  02111-1307  USA
#
#  $Id: Makefile,v 1.2 2003/12/12 18:09:43 conner Exp $
#
#

# set this to the path to you kernel source include directory

obj-m += pci_quad04_26.o
```

Pour faire ces modifications je me suis inspiré de l'article sur la programmation de module sous linux http://www.tldp.org/LDP/lkmpg/
La compilation du module s'opère par la commande :

```
make -C /usr/src/linux-2.6.8.1 SUBDIRS=$PWD modules
```

et l'on obtient :

```
vahm2:/home/yann/Projects/inter2/src/quad04#  make -C /usr/src/linux-2.6.8.1 SUBDIRS=$PW
make: Entering directory '/usr/src/linux-2.6.8.1'
  Building modules, stage 2.
  MODPOST
make: Leaving directory '/usr/src/linux-2.6.8.1'
vahm2:/home/yann/Projects/inter2/src/quad04#
```

On insert ensuite le module par :

```
insmod ./pci_quad04_26.ko
```

et l'on peut lire des valeurs des compteurs par les commandes : lecture des valeurs :

```
vahm2:/home/yann/Projects/inter2/src/quad04# more /proc/pci_quad04/cntr_1 cntr_1 = 42557
vahm2:/home/yann/Projects/inter2/src/quad04#
```

5.2 Commandes des moteurs du fauteuil à travers le driver PCIDDA

Il s'agit de la deuxième partie de notre projet, savoir communiquer avec l'interface de type COMEDI.

5.2.1 Récupération d'informations

Pour écrire un programme se servant de fonctions COMEDI, la première chose à faire est de comprendre le fonctionnement global de la carte. Bien sûr et heureusement nous ne nous occupons désormais plus de la configuration de registres, de la gestion des interruptions mais COMEDI ne dispense pas de connaissances informatiques bas niveau. Ainsi COMEDI nous fournit un service de renseignement sur notre carte à travers la commande `./board_info` dans COMEDIlib/demo/ Cette commande nous renvoit :

```
overall info:
  version code: 0x000604
  driver name: cb_pcidda
  board name: at-mio-16e-10
  number of subdevices: 7
subdevice 0:
  type: 1 (analog input)
  number of channels: 8
  max data value: 4095
...
```

Les informations nous intéressant directement sont les suivantes :

- ▷ Le numéro de subdevice qui correspond au fonctionnement de notre carte. Dans notre cas, il sera égal à 0, ce qui signifie que nous utilisons la carte en mode analogique.
- ▷ Le numéro range qui nous indique dans quel rang sera compris notre tension de sortie. Nous choisissons 4, ce qui correspond à [0V ; 5V]. Comme nous le verrons un peu plus tard notre moteur est commandable pour des tensions comprises entre à peu prés 1V et 4V
- ▷ La référence de la masse. Ce paramètre a peu d'importance, on choisira la constante `AREF_GROUND`.
- ▷ Les numéros de canaux (`CHAN#`). Notre carte en comprend 8 en mode analogique. Cela signifie concrètement que l'on peut délivrer simultanément en sortie 8 tensions.

Ces seules informations vont nous permettre de piloter notre carte très simplement et efficacement.

5.2.2 Les fonctions fournies par COMEDI

Nous allons maintenant découvrir tout l'intérêt de COMEDI pour notre projet à travers l'étude de quelques fonctions utiles à la finalisation de notre projet. On précisera que ces fonctions ne sont disponibles qu'à la seule condition que l'on indique en début de fichier de code `#include <config.h>` (inclusion des headers COMEDI) COMEDI gère le flux de données vers la carte comme s'il s'agissait d'un simple fichier. C'est pourquoi, on ne sera pas surpris de découvrir quelques similarités avec le langage C : lors de l'ouverture de la carte et la permission d'accès aux données.

```
it=comedi_open("/dev/comedi0");
```

Cette fonction renvoi un pointeur sur fichier COMEDI dont on se sert, comme en C :, pour écrire et lire dans le fichier comedi0, donc la carte ! Penchons nous sur un dernier exemple, suite logique du précédent :

```
comedi_data_write(it,SUBDEV,CHAN0,RANGE,AREF,data);
```

it contient le pointeur sur fichier spécial COMEDI, SUBDEV est une constante définissant notre carte en tant qu'analogique, CHAN0 est égal à 0 car on écrit sur le canal 0, RANGE est définit à 4 [0V ;5V], AREF est égal à la constante `AREF_GROUND` et data contient la valeur numérique à écrire dans le registre de sortie de la carte. Passons maintenant à la pratique et étudions un petit morceau d'un programme de test de notre fauteuil.

5.2.3 Elaboration d'un programme de test

La première question à laquelle il est nécessaire de répondre avant tout codage est la suivante : Comment est branchée le moteur sur la carte d'acquisition et quelles tensions faut il lui envoyer ?

5.2.3.1 Etude physique du problème Le VAHM possède deux roues motrices et un moteur pour chaque roue. En mode automatique, ces moteurs ne sont pas commandés directement par la carte PCIDDA mais à travers un système de mise en forme et de contrôle. Par exemple, un signal de 5V en sortie de carte mettra immédiatement le fauteuil en position de sécurité. Le boîtier de commande indique un message d'erreur (J5) et le fauteuil se bloque jusqu'au redémarrage du boîtier de commande. En mode manuel, on dit que le système est court-circuité (au sens figuré bien entendu). C'est à dire que le système de direction piloté par l'ordinateur du VAHM n'est plus opérationnel mais remplacé par une commande humaine.

FIG. 5 – Boîtier de commande en mode automatique en état de fonctionnement normal

Après étude des programmes sous Windows, nous parvenons à dégager un fonctionnement physique des deux moteurs. Une charte précise et rigoureuse de tension est à respecter.

Ces règles sont les suivantes :

- ▷ On doit émettre sur le canal 0 une tension de direction gauche droite proportionnelle à l'angle pris par le fauteuil vers la gauche ou la droite
- ▷ On émet sur le canal 1 le complément de cette tension 5 - tension gauche droite
- ▷ On émet sur le canal 2 une tension 5V-tension avant arrière proportionnelle à la vitesse du fauteuil vers l'avant ou l'arrière.

▷ On émet sur le canal 3 le complément de cette tension : tension avant arrière.
Les tensions de repos sont tensions avant arrière = tension gauche droite = 2.54V Pour ces valeurs, le fauteuil ne bouge pas.

5.2.3.2 Etude informatique du problème]

Nous disposons presque de tous les éléments nécessaires à une écriture du code d'un programme de test. Il reste un seul problème, comment convertir la tension de sortie de la carte en donnée numérique à inscrire dans un registre ? Une méthode serait de calculer le pas de quantification du convertisseur numérique analogique. La carte étant une 16 bits, nous aurions donc une précisions de $1/216 = ?$. On peut ainsi trouver facilement à quelle valeur numérique correspond une tension v en posant $N = v/?$. Cela dit, nous n'utiliserons pas cette méthode et une fois de plus COMEDI va bien nous simplifier la vie et donner à notre projet bien plus de flexibilité. Voici notre manière de procéder :

```
it=comedi_open("/dev/comedi0");
max_value = comedi_get_maxdata(it, subdev, chan0);
```

On commence par procéder à l'ouverture du fichier spécial COMEDI puis on récupère la valeur maximale du registre de sortie numérique N.

```
cr = comedi_get_range(it, subdev, chan0, range);
data = comedi_from_phys(tensiongd, cr, max_value);
success = comedi_data_write(it,subdev,chan0,range,aref,data);
printf("\nChannel 0 : %d",success);
printf("\n AO0=%f    Data=%d        \n  ", tensiongd, data);
```

On récupère un pointeur sur structure COMEDI contenant des informations de configuration de notre carte. La fonction `comedi_from_phys` convertit ainsi toute seule la tension analogique en valeur numérique. `comedi_data_write` écrit cette valeur dans le registre de sortie de la carte. La variable success contient 1 si l'écriture a réussi, une autre valeur sinon.

5.3 Commande du système par l'interface graphique

Après avoir réussi à commander les moteurs du fauteuil par l'intermédiaire d'un programme en C dialoguant avec le driver COMEDI de la carte `PCI_DDA`, notre objectif fut de contrôler les canaux par l'intermédiaire d'une interface graphique préexistante. Cette interface regroupe une série de champs et de boutons :
▷ Des retours d'information des capteurs ultrasons
▷ Des retours d'information des codeurs incrémentaux des moteurs
▷ Un bouton de commande binaire
▷ Un bouton de commande pour diriger les moteurs, activer ou désactiver l'affichage des codeurs ainsi que des capteurs ultrasons.

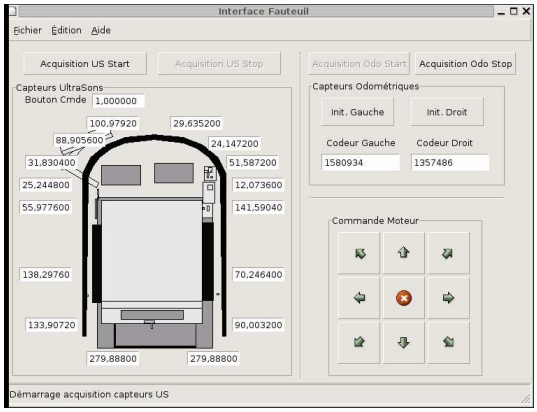

FIG. 6 – Interface de test des différents composants du fauteuil

Voici l'interface graphique compilée et exécutée. Dans la partie Capteurs Ultrasons, les différents champs affichent les distances mesurées par les capteurs. Les Capteurs Odométriques renvoient les valeurs des deux codeurs. Enfin, la partie Commande Moteur est celle qui nous intéresse. A l'aide de ces 9 boutons, nous devons contrôler les deux moteurs pour que le fauteuil aille dans la direction désirée ou s'arrête.

5.3.0.3 Principe La personne à mobilité réduite ne peut commander le fauteuil que part un seul bouton. Pour que cela soit possible et qu'il puisse aller dans tous les sens, il faut que l'interface graphique offre à intervalles de temps régulier la possibilité à la personne de changer de direction. La solution prévue est que chaque direction soit alternativement proposée à l'individu sur le fauteuil. Par simple pression sur un contact binaire, ce dernier a la possibilité de valider, ou pas, la direction proposée

5.3.0.4 Insertion du code voici un extrait du code présent dans le fichier `callbacks.c`

```
void on_ButDroit_released (GtkButton *button, gpointer user_data)                {
gdouble delta = 0.77;
direction = 3;
delta = delta * gain; //valeur Ã  ajouter Ã  la tension de repos gauche droite
tensiongd = TENSIONGD_INIT - delta;
tensionaa = TENSIONAA_INIT;
ecrire_dda( tensionaa, tensiongd);
}
```

La fonction `on_ButDroit_released ()` est appelée lorsque le bouton droit de l'interface graphique est relâché. Le code à l'intérieur des crochets est ainsi exécuté. On fixe la valeur de la tension gauche droite et avant arrière. Dans notre cas, le fauteuil se dirige à droite toute. On décide donc de faire tourner les roues dans un sens opposé en faisant varier uniquement la consigne gauche droite. Les autres boutons ont le même principe de fonctionnement mais des valeurs d'initialisation différentes selon les de déplacement.

Enfin, nous avons apporté quelques améliorations, à savoir :
▷ une "scroll bar" permettant de modifier en temps réel la vitesse du moteur
▷ une fonction assurant la surbrillance cyclique de chaque bouton.
Notre nouvelle boite de commande moteur se présente désormais ainsi :

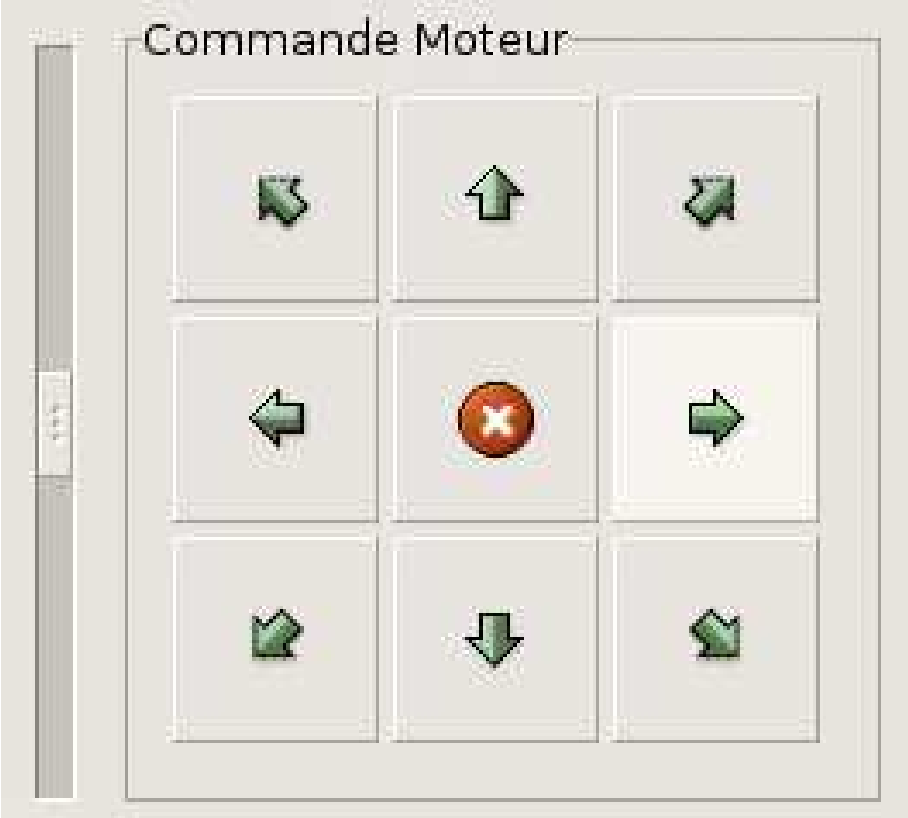

Fig. 7 – Nouveau panneau Commande Moteur

Lorsqu'on change la position de la scroll bar sur la gauche, une fonction s'exécute qui récupère la valeur de la position du curseur. Plus le curseur sera haut, plus la valeur du gain sera importante et plus la vitesse du fauteuil sera élevée quelque soit sa direction.

Enfin, penchons nous rapidement sur la fonction de mise en surbrillance des boutons.

```
gboolean aff_button(gpointer data)
{
GtkWidget * but;
num_butt ++;
if(num_butt == 10)num_butt = 1;

switch(num_butt)
{
case 1:  but = (GtkWidget *) lookup_widget (window1, "button9");
gtk_widget_set_state(but,GTK_STATE_NORMAL);
but = (GtkWidget *) lookup_widget (window1, "butstop");
gtk_widget_set_state(but,GTK_STATE_PRELIGHT);
break;

.
.
.

}
```

Cette fonction est appelée par :

```
g_timeout_add_full(G_PRIORITY_HIGH_IDLE, 750,  aff_button ,  NULL, NULL);
```

Cette fonction génère une interruption toute les 750 ms qui appelle la fonction `aff_button` ci-dessus. Cette fonction est très simple : on incrémente la valeur `num_butt` correspondant au bouton et toute les 750ms, on change de bouton.

6 Programmation du port série

Les moyens de perception extéroceptifs sont constitués d'une ceinture de 16 capteurs à ultrasons. Trois capteurs couvrent chaque coté du fauteuil et deux autres sont montés à l'arrière. Les autres capteurs jouent un rôle plus actifs durant la navigation et sont placés pour couvrir intégralement la perception de l'environnement à l'avant du fauteuil. Les capteurs à ultrasons sont contrôlés par une carte spécialisée à base de microcontrôleur 68HC11 qui met à jour la table des mesures toutes les 100 millisecondes. Cette table est lue à intervalle régulier par le calculateur embarqué, à travers son port série.

Il est donc necessaire d'effectuer une gestion du port série, afin de pouvoir obtenir les informations provenant de la ceinture de capteur.

Un très bon point de départ pour la gestion du port série sous Linux peut être consulter à l'adresse suivante : http://www.easysw.com/~mike/serial/serial.html

Les programmes suivants sont commentés et se suffisent à eux mêmes àprès la lecture de l'article précédent.

Fichier `routineus.h`

```
/*************************************************************************
 *            routineus.h
 *
 *  Thu Mar 31 15:21:07 2005
 *  Copyright  2005  Yann MORERE
 *  morere@sciences.univ-metz.fr
 *************************************************************************/

/*
 *  This program is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITN#include <comedilib.h>
ESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU Library General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program; if not, write to the Free Software
 *  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
 */
#ifndef ROUTINE_US
#define PORTSERIE "/dev/ttyS0"
```

```
#define VAHM32_NBRCAPTSUS 17
#endif

int open_port(char * nom_port);
void close_port(int fd) ;
void demarrer68HC11(int fd);
void stopper68HC11(int fd);
void lectureUS(int fd, double * ustab);
```

Fichier routineus.c

```
/*************************************************************************
 *            routineus.c
 *
 *  Thu Mar 31 15:21:07 2005
 *  Copyright  2005  Yann MORERE
 *  morere@sciences.univ-metz.fr
 *************************************************************************/

/*
 *  This program is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU Library General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program; if not, write to the Free Software
 *  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
 */


#include <stdio.h>   /* Standard input/output definitions */
#include <string.h>  /* String function definitions */
#include <unistd.h>  /* UNIX standard function definitions */
#include <fcntl.h>   /* File control definitions */
#include <errno.h>   /* Error number definitions */
#include <termios.h> /* POSIX terminal control definitions */

#include "routineus.h"
/*
* 'open_port()' - Open serial port 1.
*
     * Returns the file descriptor on success or -1 on error.
*/
```

```
int
open_port(char * nom_port)
{
int fd; /* File descriptor for the port */
struct termios options; /*options Ã  rÃ©cupÃ©rer*/

/*ouverture du port*/
fd = open(nom_port, O_RDWR | O_NOCTTY | O_NDELAY);
    if (fd == -1)
      {
    /*
* Could not open the port.
*/
perror("open_port: Unable to open Port");
      }
      else
fcntl(fd, F_SETFL, 0);

/*configuration du port*/
/* recupÃ¨re les options courante */
     tcgetattr(fd, &options);

   /*
      * Fixe la vitesse du port 9600...
      */
    cfsetispeed(&options, B9600);
     cfsetospeed(&options, B9600);

/* Fixe 8 bits, pas de bits de stop, 1 bits de paritÃ© */
   options.c_cflag &= ~PARENB; /*pas de bit de paritÃ©*/
options.c_cflag &= ~CSTOPB; /*1 bit de stop*/
options.c_cflag &= ~CSIZE; /*pour la taille des donnÃ©es*/
options.c_cflag |= CS8; /*8 bits*/
   /*options.c_cflag |= CNEW_RTSCTS;*/ /* contrÃ´le de flux matÃ©riel*/

     /* set raw input, 1 second timeout */
     options.c_cflag     |= (CLOCAL | CREAD);
     options.c_lflag     &= ~(ICANON | ECHO | ECHOE | ISIG);
/*     options.c_cc[VMIN]  = 0;
     options.c_cc[VTIME] = 10;*/

/* Output */
options.c_oflag &= ~OPOST; /*output en mode raw*/
     /* set the options */
     tcsetattr(fd, TCSANOW, &options);
/*fputs("Ouverture Port\n", stderr); */

      return (fd);
    }
```

```
void close_port(int fd)
{
close(fd);
/*fputs("Fermeture Port\n", stderr); */
}

void demarrer68HC11(int fd)
{
int n;
char buffer[1]="D"; /*dÃ©marrer*/
n = write(fd, buffer, 1);
    if (n < 0)
      fputs("write() of D failed!\n", stderr);
/*fputs("DÃ©marrage 68HC11\n", stderr);*/
sprintf(buffer,"%c",'C'); /*vitesse de capture*/
n = write(fd, buffer, 1);
    if (n < 0)
      fputs("write() of D failed!\n", stderr);
/*fputs("vitesse 68HC11\n", stderr);*/
}

void stopper68HC11(int fd)
{
int n;
char buffer[1]="F";
n = write(fd, buffer, 1);
    if (n < 0)
      fputs("write() of F failed!\n", stderr);
/*fputs("ArrÃªt 68HC11\n", stderr);*/
}

void lectureUS(int fd, double * ustab)
{
        int i, k, n;
        double usp[ VAHM32_NBRCAPTSUS ];
        double vit = (double) 34300.0;
        double basetps = (double) 32.0e-6;
        unsigned char reception[ VAHM32_NBRCAPTSUS ];
        char buffer[1];

sprintf( buffer,"%c",'E' ); /*lecture donnÃ©e*/
n = write(fd, buffer, 1);
     if (n < 0)
       fputs("write() of E failed!\n", stderr);
n = read(fd,reception,17);
        if (n < 0)
       fputs("read() of failed!\n", stderr);
//else
// fputs("read() ok !\n", stderr);
        for( i = 0; i < VAHM32_NBRCAPTSUS; i++ )
```

```
        {
                usp[i]=(double)(reception[i]);
    }
        for( i = 1; i < VAHM32_NBRCAPTSUS; i++ )
        {
                usp[i] = usp[i] * vit * basetps;
        }
        for( k = 0; k <= 3; k++ )
        {
                ustab[1+4*k] = usp[1+k];
                ustab[2+4*k] = usp[5+k];
                ustab[3+4*k] = usp[9+k];
                ustab[4+4*k] = usp[13+k];
        }
        for( i = 1; i < VAHM32_NBRCAPTSUS; i++ )
        {
                if( ustab[i] <= 10 )
                ustab[i] = 11;
        }
        ustab[0] = usp[0];
}
```

Fichier de test `test_routineus.c`

```
/*****************************************************************************
 *            test_routineus.c
 *
 *  Thu Mar 31 15:21:07 2005
 *  Copyright  2005  Yann MORERE
 *  morere@sciences.univ-metz.fr
 *****************************************************************************/

/*
 *  This program is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU Library General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program; if not, write to the Free Software
 *  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
 */


#include <stdio.h>   /* Standard input/output definitions */
```

```
#include <string.h>  /* String function definitions */
#include <unistd.h>  /* UNIX standard function definitions */
#include routineus.h

int
main (int argc, char *argv[])
{
int fport,i,j;
double dataUS[VAHM32_NBRCAPTSUS];

fport=open_port(PORTSERIE);
demarrer68HC11(fport);

for (j=0;j<10;j++)
{
lectureUS(fport, dataUS);
for (i=0;i<VAHM32_NBRCAPTSUS;i++)
{
printf("capteur %d = %lf\n",i,dataUS[i]);
}
sleep(1);
}
stopper68HC11(fport);
close_port(fport);
return 0;
}
```

C'en est fini de cet article, toutes remarques et corrections sont les bienvenues à l'adresse morere@univ-metz.fr