

Figure 5. Affichage sur LCD tactile

Le principe est ici exactement le m me que pour la fonction I2Cwrite avec l'appel   la fonction ioctl avant la fonction read.

### Pilotage par touchscreen

Nous rappelons que le but de l'application est de pouvoir piloter des relais d'une carte d'extension I2C   l'aide d'un  cran LCD tactile. Le m canisme est celui d'un va-et-vient, donc   chaque nouvel appui sur le LCD, le relais concern  sera tant t ON, tant t OFF.

Dans la boucle while(1) du main, il nous faut donc p riodiquement inspecter l' cran tactile, et modifier l'affichage si besoin tout en transmettant l'ordre d'activation de relais en I2C (Listing 10).

La premi re chose qui peut surprendre, ce sont les calculs li s aux coordonn es X et Y. En effet, bien souvent les  crans tactiles ont besoin d' tre calibr s, pire encore dans notre cas, les coordonn es X et Y ont besoin d' tre invers es (cas d'une rotation de l' cran du mode portrait   paysage). Nous n'avons pas voulu alourdir le code avec une  tape pr liminaire de calibrage, aussi nous effectuons le calcul directement avec une mise   l' chelle par rapport   notre  cran 320x240. On d finit de la m me fa on des zones graphiques

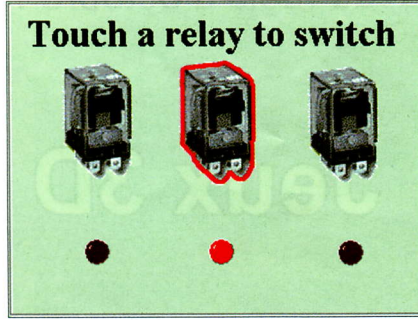


Figure 6. Ajout de texte en police fixe

pour les diff rents objets (Widgets Relais) et on appelle la fonction ToggleRelay selon l'appuie qui a  t  d tect . Bien entendu, il serait pr f rable de cr er un widget qui examine lui-m me l'appuie d tect  pour d terminer l'action   suivre. Nous serions alors davantage dans le cas d'une programmation orient e objet o  le widget poss de les donn es et les fonctions qui d finissent les actions et d cisions du widget.

Ceci peut  tre r alis  en C pour tous ceux qui se sentent   l'aise avec les pointeurs de fonctions (inclure alors le pointeur dans la structure widget et initialiser le pointeur dans la fonction main).

Enfin derni re  tape, qui n'est pas  vidente en soit mais indispensable : vider le tampon du driver de touchscreen ! En effet la plupart des drivers stockent la liste des appuis qui ont  t  d tect s, et ceci dans le but de pouvoir tracer l'historique du stylet sur le LCD (dans le cas d'un logiciel de dessin par exemple).

Dans notre cas il faut flusher cet historique pour  tre pr t   d tecter le prochain appui.

### Entr es t l commandes

Pour terminer il faut  tre capable de lire les entr es t l commandes en provenance du p riph rique I2C   l'adresse 0x23. Juste avant de flusher le buffer du touchscreen on peut en profiter pour lire les entr es t l commandes (Listing 11). La fonction SwitchLedsON est similaire   celle des fonctions li es au relais (Listing 12). Encore une fois nous n'avons ici besoin que de modifier l'identifiant de l'image et non pas de recharger l'image compl tement en m moire.

### Application finale

Finalemnt, l'application fonctionne simplement de la fa on suivante :

- lorsqu'on appuie sur une image d'un relais, celui-ci se colle ou se d colle alternativement

- lorsqu'une entr e t l commande survient, la LED correspondante s'allume

Pour compl ter l'application, on pourrait ajouter par exemple du son, en envoyant un fichier wav   l'interface /dev/sound/dsp par exemple.

Une am lioration simple avec la librairie Nano-X serait l'ajout d'un texte en haut de l' cran (Figure 6) comme ceci. Il suffit pour cela de cr er une variable font et de l'associer aux variables MainWindow et gc (Listing 13). La police Times35 est incluse de fa on native dans le nano-X que nous utilisons, mais mieux encore nous pourrions utiliser une police TrueType en chargeant par exemple la police times.ttf.

N'oubliez pas pour cela de linker la librairie libttf.a   votre projet. Enfin, cerise sur la g teau, nous pourrions aussi faire de l'affichage distant, c'est- -dire de lancer le serveur Nano-X sur une station ou un syst me embarqu  distant. En effet, Nano-X est un serveur avec lequel on communique par socket. Il peut donc  tre utilis  sur la m me plate-forme comme sur 2 plate-formes distantes. Vous pourrez donc ainsi afficher le contenu graphique du Frame Buffer sur un PC distant pour lequel nano-X aura  t  recompil  en natif.

### Conclusion

Nous esp rons que cet article vous aura convaincu de la simplicit  qu'il est tr s simple de r aliser une application graphique embarqu e. Les outils logiciels qui existent au jour d'hui sur de telles cibles permettent de r aliser rapidement des applications, voir m me dans certains cas, de porter des applications pour PC gr ce   l'utilisation d'une librairie SDL adapt e... mais nous verrons tout cela dans un prochain article !  



### Sur le r seau

- Site de uClinux, version embarqu e de Linux pour microcontr leurs : [www.uclinux.org](http://www.uclinux.org)
- Site officiel du serveur nano-X : [www.microwindows.org](http://www.microwindows.org)
- Les cartes de d veloppement SIMTEC : <http://www.simtec.co.uk>
- Forum fran ais li  au d veloppement d'applications embarqu es Linux : <http://www.pragmatux.net>



### Concernant l'auteur

Ing nieur en  lectronique et informatique, Xavier Montagne a travaill  en France et aux Etats-Unis sur des projets de d fense et de super-calculateurs. Sp cialiste en syst mes temps r els et en Linux embarqu , il est aujourd'hui directeur technique de la soci t  Pragmatec. Auteur du noyau temps r el PICos18 et intervenant aupr s d' coles d'ing nieurs et de formations universitaires, il croit avant tout en l'efficacit  des solutions GPL et Open Sources.