une image, il importe de fonctionner en 2 temps :

- Charger l'image en mémoire,
- Afficher l'image

À quoi bon avoir séparé ces 2 fonctions ? Comme vous pouvez le constater dans cet exemple, il est possible de charger en mémoire 3 images différentes, et de les afficher plusieurs fois chacune. Ne chargez donc pas autant d'images que vous en affichez... sinon votre mémoire va rapidement saturer !

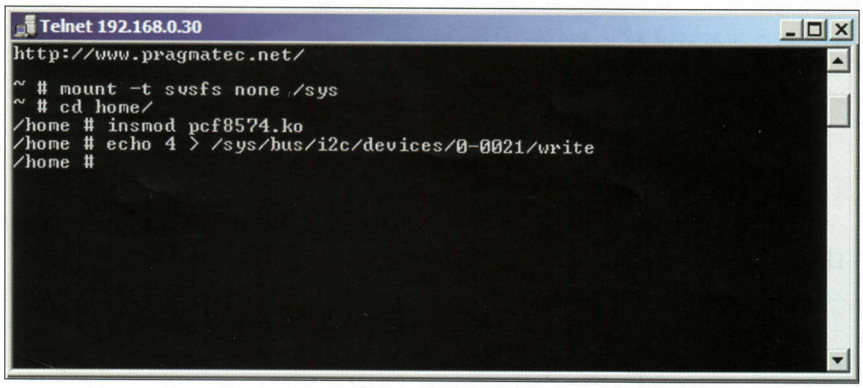

Figure 4. Pilotage du hardware depuis le sysfs

Listing 11. Lecture des télécommandes

```
input = I2CRead(I2C_INPUT) & 0x0f;
if (input & 0x01) SwitchLedON(0);
if (input & 0x02) SwitchLedON(1);
if (input & 0x04) SwitchLedON(2);
if (input == 0) SwitchLedsOFF();
/*
   We flush the TS buffer
*/
```

Listing 12. Allumage des LEDs

```
void SwitchLedON(int index) {
   Led[index].ImageId = thescreen.LedId_ON;
   GrDrawImageToFit(MainWindow, gc, Led[index].X, Led[index].
         Y, 20, 20, Led[index].ImageId);
}

void SwitchLedsOFF(void) {
   int i;

   for (i = 0; i < 3; i++)     {
      Led[i].ImageId = thescreen.LedId_OFF;
      GrDrawImageToFit(MainWindow, gc, Led[i].X, Led[i].Y,
            20, 20, Led[i].ImageId);
   }

}
```

Listing 13. Affichage du texte

```
font = GrCreateFont("Times35", 0, 0);
GrSetFontAttr(font, GR_TFKERNING | GR_TFANTIALIAS, 0);
GrSetGCFont(gc, font);
GrText (MainWindow, gc, 5, 35, "Touch a relay to switch", -1,
      GR_TFASCII|GR_TFBOTTOM);
```

Nous chargeons ici des images GIF. Elles ont le mérite d'être peu gourmande en espace mémoire et sont performante pour des images simples.

Si votre image utilise de nombreux dégradés, surtout dans des gammes de couleurs différentes, préférez plutôt un fichier BMP même si celui-ci est plus volumineux.

Nano-X gère sans difficulté les images au format GIF et BMP, voir même au format JPEG si vous avez reconstruit Nano-X avec une librairie JPEG adéquate. Attention toutefois au LCD que vous utilisez et au nombre de couleurs requis pour votre image, vous pourriez être déçu : le rendu de votre image n'est pas le même sur le LCD de votre PC et celui de votre système embarqué, surtout si ce dernier utilise un LCD à 256 couleurs...

Interface hardware

Commençons par piloter les relais de la carte d'extensions I2C. Nous utiliserons une simple fonction `toggle` qui permet de coller et de décoller un relais alternativement (Listing 7).

L'objet `Relay[n]` possède un identifiant d'image. Pour changer d'affichage, et donc d'une image d'un relais OFF à un relais ON, il suffit de changer d'identifiant, puis de réafficher l'image.

Nous rappelons à toutes fins utiles que Nano-X sépare les 2 notions de chargement en mémoire d'une image et de son affichage dans le `Frame Buffer` : l'image est chargée en mémoire depuis un nom de fichier, alors que son affichage se fait depuis un identifiant retourné par la fonction de chargement en mémoire. Ceci permet de charger une image d'icône une seule fois en mémoire et de l'afficher autant de fois que nécessaire.

La fonction `I2Cwrite` permet d'envoyer la valeur `Value` au périphérique dont l'adresse est `I2C_RELAY(0x21)` (Listing 8).

Comme la fonction `I2Write` est une fonction générique (acceptant les paramètres d'adresse de la cible et de données à transmettre), nous commençons par spécifier le périphérique concerné à l'aide de la fonction `ioctl` avant d'appeler la fonction `write` à proprement parler.

Le driver I2C générique de Linux utilise l'appel à `ioctl` pour déterminer quel est le sous-driver I2C concerné par cette action. Mais n'oublions pas qu'il nous faut aussi inspecter les entrées I2C depuis le périphérique à l'adresse 0x23 (Listing 9).