images BMP ou GIF, de déplacer des zones graphiques, d'afficher du texte, y compris les fonts *TrueType* (Figure 3) utilisées habituellement sous Windows ou Linux.

Utilisation de l'écran tactile

Nous utiliserons un écran tactile résistif, c'est-à-dire que l'axe des X et celui des Y peut être symbolisé par 2 résistances. Lorsque votre doigt se déplace sur le film tactile, la valeur de la résistance change.

Un contrôleur adapté permet de savoir lorsqu'un appui ou un relâchement a lieu, et à quel endroit exactement.

Le processeur S3C2410 possède son propre contrôleur tactile et l'interface utilisée sera */dev/input/ts0*.

Vérifications préliminaires

La première chose à faire est sans doute de vérifier le fonctionnement du hardware et du bus I2C.

Nous connectons pour les besoins de l'application une carte d'extension I2C sur les 4 signaux suivants :

- VCC5V pour l'alimentation,
- GND (masse),
- SDA (data I2C),
- SCL (horloge I2C).

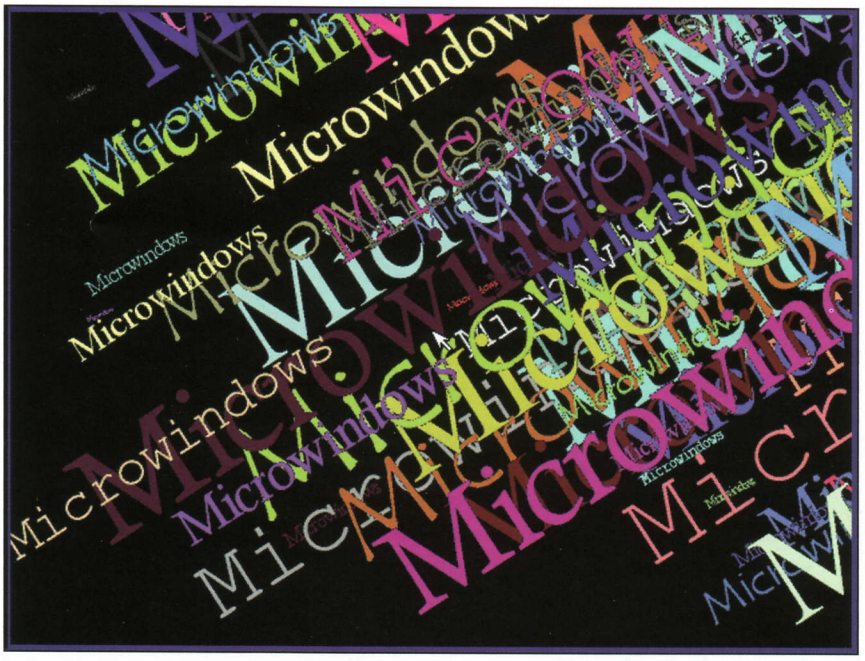

Figure 3. Nano-X avec gestion des fonts TrueType

Le bus I2C est un bus série synchrone, c'est-à-dire que l'horloge de synchronisation des données est transmise en même temps que les données. Pour être plus explicite, disons que les données sont transférées sur une ligne de données, et que pour chaque donnée transmises, un basculement du signal d'horloge à lieu afin de signaler qu'une nouvelle donnée est transmise sur le bus.

Il s'agit d'un bus série dans ce sens où les données sont transmises en série, bit après bit sur la ligne SDA. Chaque pulse sur la ligne SCL valide chaque bit transmis, permettant ainsi au périphérique I2C d'intercepter sans erreur chacun des bits envoyés.

Mieux encore, le bus I2C peut être utilisé dans un mode `mono-master` et `multi-slave`. Cela signifie que vous pouvez connecter divers périphériques I2C sur un même bus, chacun disposant de sa propre adresse sur le bus (d'une longueur de 7 bits).

Afin d'interroger un périphérique sur le bus, le *master* (le CPU) envoie en premier l'adresse du périphérique à interroger, avec en plus un bit indiquant si le *master* souhaite lire des données ou en écrire dans le périphérique.

Histoire de mettre tout le monde d'accord sur les débuts et fin de trame, l'échange est normalisé comme suit : le *master* débute l'échange en basculant les signaux SCL et SDA d'une certaine façon, et le termine en basculant les 2 signaux dans le sens inverse.

De nombreux périphériques sont disponibles depuis longtemps en accès I2C, ce qui permet de réaliser facilement des extensions de cartes : capteur de température, EEPROM, horloge temps réel */calendrier*, potentiomètre numérique, extenseur de bus, ...

Côté Linux, l'accès aux périphériques I2C est facilité grâce à l'interface */sys*. Contrairement à l'interface */proc* dans laquelle on

Listing 5. Connexion au serveur Nano-X

```
static GR_WINDOW_ID  MainWindow;
static GR_GC_ID gc;
static GR_FONT_ID font;
static int  fd_TS;
static int  fd_I2C;
static screenSpec thescreen;
static widget Relay[3];
static widget Led[3];
int main(int argc, char **argv) {
   struct ts_event ts;
   int ret;
   int X, Y;
   char input;
   // Init client port
   if (GrOpen() < 0) exit(1);
   MainWindow = GrNewWindowEx(GR_WM_PROPS_APPWINDOW, "demo_nanox",
         GR_ROOT_WINDOW_ID, 0, 0, SCREEN_WIDTH,
          SCREEN_HEIGHT, BGCOLOR);
   GrSelectEvents(MainWindow, GR_EVENT_MASK_EXPOSURE |
         GR_EVENT_MASK_CLOSE_REQ| GR_EVENT_MASK_KEY_DOWN |
         GR_EVENT_MASK_KEY_UP); GrMapWindow(MainWindow);
   gc = GrNewGC();
   GrSetGCUseBackground(gc, GR_FALSE);
   GrSetGCForeground(gc, FGCOLOR);
   GrSetGCBackground(gc, BGCOLOR);
   printf("Nano-X lib successfully connected !\n");
```