TP Entrées-sorties (1)
Bus I2C & Raspberry
module F.A.S. - IG3 - Polytech Montpellier

Les dernières diapos de cours précisent un certain nombre de détail sur la façon dont les communications se déroulent sur le bus I2C. Nous allons utiliser le package *i2c-tools* qui nous évitera de descendre au niveau de l'information bit à bit.

Connectez-vous à un raspberry sur lequel vous disposez d'un compte, sinon créez-vous un compte sur un des raspberry de la salle (voir TP précédent).

1) permettre au raspberry de communiquer par le bus i2c :

- lancez l'utilitaire `raspi-config` (en `sudo`) et grâce à lui **activez i2c** puis **reboot** et /dev/i2c-1 devrait apparaître comme un fichier, vérifiez. Tiens au fait, si on le fait avec l'option -l, que signifie le 1er caractère en début de ligne ?
- ça devrait aussi mettre i2c-dev dans /etc/modules s'il n'y ait pas déjà, vérifiez
- vérifiez aussi que si le fichier `/etc/modprobe.d/raspi-blacklist.conf` existe, alors la ligne `blacklist i2c-bcm2708` (si elle apparait) est commentée (commence par un #).
- Que fait la commande `dmesg` ? Filtrez sa sortie pour ne voir que les lignes qui contiennent le mot i2c. Pouvez-vous en déduire si le noyau Unix a chargé ce module lors du dernier redémarrage ?

2) installer les outils shell pour interroger le bus i2c :

- Cherchez sur internet quel est le gestionnaire de package le plus efficace entre `aptitude` et `apt-get`.
- Avec celui choisi lancez l'installation du module i2c-tools (par exemple `sudo aptitude install i2c-tools`).
- `i2cdetect -y 1` permet de voir si quelque chose est connecté sur le port i2c. Alors ?
- Insérez maintenant le shield grove sur les dernières broches du borner noir du raspberry, côté carte microSD (attention : voir sur internet une photo pour être sûr de l'insérer dans le bon sens et au bon endroit afin de ne pas tout griller tout de suite).
- On teste à nouveau `i2cdetect -y 1` et on devrait voir le shield a une certaine adresse, quelle est cette adresse ?
- Sur un **port i2c** (attention, pas un port digital ou analog) du shield, connectez maintenant le petit écran LCD (« RGB Backlight »)

- Relancez `i2cdetect` pour voir si vous devinez sur quelles adresses peut être accéder l'écran LCD ? Comment expliquer une adresse avec des lettres dedans ?

- 3) Affichages sur l'écran LCD depuis la CLI
- L'écran LCD se commande en envoyant à une adresse ou une autre suivant que l'on s'adresse au fond d'écran (0x62 pour RGB) ou aux caractères affichés (0x3e)
- Demandez le manuel sur la commande **`i2cset`** qui permet de positionner des valeurs dans des registres (petites mémoires) connectés au bus i2c

Il faut d'abord initialiser l'écran en envoyant la valeur nulle dans les registres 0x00 et 0x01 :

```
i2cset -y 1 0x62 0x00 0x00   # mode 1 init, normal mode
i2cset -y 1 0x62 0x01 0x00   # mode 2 init
```

Ensuite on peut donner une couleur au fond de l'écran LCD en envoyant une valeur indiquant la force de chaque composante de couleur (Red = 0x04, Green = 0x03, Blue = 0x02). Cette force peut être composée de valeurs entre 0x00 et OxFF. Par exemple :

si on tape les commandes suivantes, de quelle couleur sera le fond d'écran ?

```
i2cset -y 1 0x62 0x02 0x00
i2cset -y 1 0x62 0x03 0xFF
i2cset -y 1 0x62 0x04 0x00
i2cset -y 1 0x62 0x08 0xAA   (cette dernière commande mystérieuse concerne l'état de sortie des LED (permet d'activer les commands précédentes ;)
```

Ensuite pour afficher des caractères, il faut d'abord initialiser l'écran en envoyant les commandes (registre 0x80) suivantes :

```
i2cset -y 1 0x3e 0x80 0x01  # clear display
i2cset -y 1 0x3e 0x80 0x0F  # display on, block cursor
i2cset -y 1 0x3e 0x80 0x38  # 2 lines
```

Puis on peut afficher des caractères (registre 0x40) en envoyant leurs codes ASCII respectifs sur l'adresse 0x3e.

Par exemple,

```
i2cset -y 1 0x3e 0x40 80
```

affiche la lettre P majuscule. … Au passage, notez que la valeur à stocker dans le registre a été donnée en chiffres décimaux cette fois et plus en hexadécimal (mais c'est comme vous préférez ;)

Pour afficher tout un mot, envoyer simplement les lettres les unes après les autres. Allez-y essayer d'écrire votre prénom (les **codes ASCII** sont trouvables facilement sur internet)

4) Concevons un driver pour l'I2C en Python

Comme vous le constatez, il est pénible de devoir retenir ces adresses auxquelles on doit envoyer de l'information, et d'envoyer caractère à caractère ces informations. Ce qui nous manque ici c'est un *driver*, un programme qui va nous abstraire nous, programmeur, de tous ces détails techniques en proposant des fonctions de plus haut niveau.

- Avec le gestionnaire de packages vérifiez que le package `python-smbus` et/ou `python3-smbus` est installé :
  - lancez l'interpréteur python (v2 ou v3) et essayer d'importer le module nommé `smbus` . Si pas de message d'erreur alors il l'a trouvé, donc le module est installé
  - Sinon vous pouvez aussi vérifier son état (`i` pour installé) avec `aptitude search smbus` (ou `python-smbus`).
- Si ce module n'est pas installé, installez-le avec `aptitude` (vous trouverez de l'aide sur internet si sa commande `install` ne vous suffit pas). Après installation vérifiez qu'il est maintenant possible d'importer le module `smbus` dans un programme python
- En vous plaçant à la racine de votre arborscence personnelle, téléchargez le squelette de module python `driverI2C.py` ainsi que le programme `main.py` qui l'utilise (ce dernier nous servira de test pour vérifier que le driver fonctionne correctement).
- Créez un répertoire `TPdriver` et travaillez pour la suite dans ce dossier : placez-y une copie des deux fichiers pythons mentionnés ci-dessus.
- Ouvrez un nouvel onglet dans votre terminal et connectez-vous à nouveau à votre raspberry sur le même compte : dans ce terminal vous éditerez en permanence un de ces fichiers pythons avec nano, tandis que l'autre terminal vous servira pour tapez les commandes lançant le script / vérifiant des détails sur l'I2C par les commandes CLI vues précédemment.
- Regardez le contenu du module `driver.py` :
  - comment est organisé son contenu ?
  - quelle est l'instruction qui va correspondre à i2cset ?
- Complétez d'abord la fonction `setRGB` et testez-là en exécutant le programme `main.py`

- Complétez ensuite la fonction `setText` :
    - cherchez dans votre cours d'algo et/ou sur internet comment parcourir l'ensemble des caractères d'une chaine de caractère. Mettez en place la boucle correspondante permettant d'examiner tout à tour chaque lettre du paramètre reçu par `setText`
    - cherchez comment obtenir le code ASCII d'un caractère en python
    - faites un premier test en exécutant le programme `main.py`
    - Complétez maintenant votre fonction pour qu'elle passe à l'affichage sur la 2ème ligne quand elle rencontre un caractère «\n » ou quand elle a déjà affiché 16 caractères sur la première ligne
    - testez à nouveau en exécutant le programme `main.py`

Voila, vous avez bien méritez une pause (éducative !) : vous pouvez jeter un coup d'oeil à un projet assez inattendu d'utilisation de l'écran LCD : http://www.instructables.com/id/Toothbrushing-Instructor/

Dans cet exemple, ils proposent d'utiliser un driver écrit en C et non en Python.

➡ Quelle conséquence majeure cela a si vous utilisez leur driver ?

➡ Que faut-il faire pour réaliser cette application si vous voulez utiliser votre driver (écrit en Python) ?