http://oversimple.fr/utiliser-le-port-serie-du-raspberry-pi/

Utiliser le port série du Raspberry Pi

Le port série est un standard qui est très utilisé dans le monde de l'industrie. De nombreux informaticiens utilisent leur raspberry via le port ethernet pour ensuite y connecter une antenne wifi. Dans cet article, nous verrons comment utiliser ce port série avec une librairie POSIX afin de dialoguer au travers d'un programme C.

Désactiver le terminal sur le port série du raspberry pi

Nous souhaitons maintenant utiliser le port série du raspberry non plus pour ouvrir un terminal dessus mais pour émettre et recevoir des trames.
Nous allons pour cela configurer le linux embarqué pour qu'il ne redirige plus vers un terminal mais vers le port série.
Avant de faire cela, il vous faut un autre moyen de vous connecter à votre raspberry pi. Vous pouvez soit utiliser un câble ethernet et ouvrir une session ssh ou bien configurer un hotspot wifi comme décrit dans cet article.
Cependant, si vous disposez d'un clavier et d'un écran hdmi, ceci peut tout aussi bien faire l'affaire.

Pour désactiver le bind du terminal sur le port série il vous faut éditer deux fichiers.
Le premier et le plus important est : /etc/inittab

Ce fichier contient la commande qui active le bind du terminal (type vt100) sur le port série à une vitesse de 115200 baud.
Allez à la fin du fichier et commentez la ligne suivante en ajoutant un '#' devant :

```
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

La simple modification de ce fichier peut être suffisante. Cependant, le raspberry pi redirige automatiquement les informations du boot sur le port série. Si vous avez un système branché sur le port série lors de son démarrage, il est possible que vous ne souhaitiez pas qu'il puisse recevoir les informations du boot.

Pour cela, il vous faut également éditer le fichier boot/cmdline.txt

Le fichier doit ressembler à ceci :

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200
console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
```

Retirez toutes les références à ttyAMA0 qui est le nom du port série. Votre fichier doit maintenant ressembler à ceci :

```
dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4
elevator=deadline rootwait
```

Pour appliquer les changements réalisés il vous est nécessaire de rebooter le raspberry pi.
Vous pouvez vérifier le bon fonctionnement en utilisant le programme minicom. Par exemple, vous pourriez utiliser votre raspberry pi pour ouvrir une session sur un autre raspberry pi via le port série.
Comment ça mes idées sont parfois tordues?

Programmer en C

Il existe de nombreuses choses déjà toutes faites pour contrôler les ports séries. Cependant, rien ne vaut une bonne librairie POSIX qui a l'avantage d'être compatible avec tous les systèmes linux embarqués ou non.

Nous utilisons la librairie POSIX pour communiquer avec le Raspberry Pi. Celle ci est donc compatible sur l'ensemble des systèmes embarqués Linux.

Voici un exemple permettant de renvoyer le message reçu :

```
#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 */

typedef int serialPort;
serialPort SerialLib_open(const char * portPath);

int main(int argc, char **argv) {
   serialPort p = SerialLib_open("/dev/ttyAMA0"); /* The serial port of the
raspberry pi */
   char buffer[16];

   read(p, buffer ,15);
   write(p, buffer ,15);
   buffer[15] = "";

   printf("Message recu: %s n",buffer);
   return EXIT_SUCCESS;
}

/**
 * Open the specified serial port
 * and return the associated file descriptor
 *
 */
serialPort SerialLib_open(const char * serialPortPath) {
   int fd; /* File descriptor for the port */

   fd = open(serialPortPath, O_RDWR | O_NOCTTY);
   if (fd == -1) {
```

```
        /* Error opening the serial port */
        printf("Could not open the serial port : %s - ", serialPortPath);
    }else {
        fcntl(fd, F_SETFL, 0);
    }
    return (serialPort)fd;
}
```

Conclusion

Vous pouvez tester le code précédent avec un second rapberry pi par exemple. Ceci peut s'avérer très utile lors de l'utilisation de devices externes. Nous l'avons par exemple utilisé pour connecter notre rapsberry pi à un microcontrôleur 8bits.

It's oversimple isn't it?