Raspberry Pi 2 Liaison Série

1 Présentation



UART (Universal Asynchronous Receiver Transmitter) assurent

l'émission et la réception asynchrone.

Sur le Raspberry Pi, les niveaux sont

 $0v,\,3.3v$

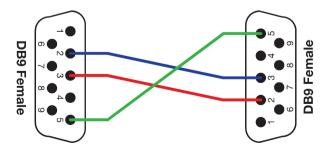
TXD : Port GPIO14 (broche 8) RXD : Port GPIO15 (broche 10)

Une carte hat qui comprend les circuits nécessaires à la conversion de niveaux, doit être ajoutée pour obtenir une liaison série standard RS232.

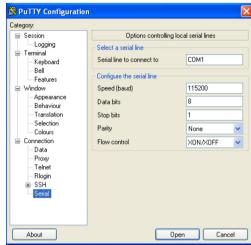


https://www.abelectronics.co.uk/p/51/Serial-Pi-Plus

Connecter la carte sur le port com du PC avec un câble DB9 femelle/femelle. La broche 2 correspond à RX (data receive). La broche 3 à TX (data transmit). La broche 5 est la masse.



ouvrir PuTTY et configuer la liaison de la façon suivante:



Mettre sous tension le Rapberry Pi

```
Putty
Uncompressing Linux... done, booting the kernel.
    0.000000] Booting Linux on physical CPU 0xf00
    0.000000] Initializing cgroup subsys cpuset
    0.000000] Initializing cgroup subsys cpu
    0.000000] Initializing cgroup subsys cpuacct
    0.000000] Linux version 4.1.7-v7+ (dc4@dc4-XPS13-9333) (gcc version 4.8.3 2
0140303 (prerelease) (crosstool-NG linaro-1.13.1+bzr2650 - Linaro GCC 2014.03)
#817 SMP PREEMPT Sat Sep 19 15:32:00 BST 2015
    0.000000] CPU: ARMv7 Processor [410fc075] revision 5 (ARMv7), cr=10c5387d
    O.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instructio
    0.000000] Machine model: Raspberry Pi 2 Model B Rev 1.1
    0.000000] cma: Reserved 8 MiB at 0x3a800000
    0.000000] Memory policy: Data cache writealloc
    0.000000] [bcm2709_smp_init_cpus] enter (9420->f3003010)
    0.000000] [bcm2709 smp init cpus] ncores=4
    0.000000] PERCPU: Embedded 13 pages/cpu @b9f64000 s20608 r8192 d24448 u5324
```

Sous Rasperry PI2, l'UART est connu sous le nom de ttyAMA0. On le trouve donc ici : /dev/ttyAMA0.

Par défaut l'UART du Raspberry Pi sert de port de débug pour Linux.

2 Libérer l'UART du mode debug

Empêcher l'émission de messages du Kernel et l'activation du mode debugging sur l'UART

```
pi@raspberrypi /dev $ sudo nano /boot/cmdline.txt

dwc_otg.lpm_enable=0 console=ttyAMA0,115200

kgdboc=ttyAMA0,115200- console=tty1 root=/dev/mmcblk0p2
rootfstype=ext4 elevator=deadline rootwait
```

devient

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

Pour désactiver le terminal sur l'UART, il faut éditer le fichier /etc/inittab et commenter la ligne faisant référence à l'UART :

```
#Spawn a getty on Raspberry Pi serial line
#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Après le redémarrage, on peut vérifier qu'aucun processus n'utilise UART /dev/ttyAMA0.

```
pi@raspberrypi ~ $ ps aux | grep ttyAMA0
```

3 Configuration

Stty permet de connaître la configuration actuelle de la liaison

```
pi@myraspberry ~/C $ stty -F /dev/ttyAMA0
speed 9600 baud; line = 0;
-brkint -imaxbel
```

sur l'écran ci-dessus on peut voir que la vitesse de transmission est 9600 bauds

4 Utilisation en ligne de commande

Sous Linux, "tout est fichier". L'envoi ou la réception de données sur le RPi se fait en lisant ou en écrivant dans /dev/ttyAMA0.

Par exemple pour envoyer le contenu du fichier achat.c sur la sortie série, la commande pourrait être la suivante:

```
pi@myraspberry ~/C $ cat achat.c > /dev/ttyAMA0
```

Autre exemple : avec la commande echo

```
pi@myraspberry ~/C $ echo 'bonjour le monde' > /dev/ttyAMA0
```

5 Programmation en C

En C, nous utiliserons par exemple **open read write close** pour en savoir plus : man 2 open man 2 read etc etc

```
#include <stdio.h> /* Standard input/output definitions */
#include <string.h> /* String function definitions */
#include <fcntl.h> /* File control definitions */
#include <errno.h>

int main(int argc, char **argv) {
   int retour,fd;
   char recu[256];
   const char *message1="Bienvenue sur Raspberry Pi\n";
   const char message2[]="Message envoyé: ";

memset(recu, 0, 256); // efface le buffer
   fd = open("/dev/ttyAMAO", O_RDWR | O_NOCTTY);
```

```
write(fd, message1, 27); // envoi du message 1
read(fd, recu, 255); // reception d'un message
write(fd, message2, 18); // envoi du message 2
write(fd, recu, 255); // envoi du message reçu

printf("Message recu:\n%s\n", recu);
retour=close(fd);
if ( retour == -1 ) {
    printf("pb fermeture: %s\n", strerror(errno));
    return(errno);
}
return 0;
}
```