Raspberry Pi : Part no 3

Série, tu sens bon la framboise

Tony Dixon
(Royaume-Uni)

Vous vous souvenez du dernier .POST framboisé ? Nous y avions présenté le connecteur d'extension du *Raspberry Pi* ainsi que les signaux GPIO. C'est le tour de l'interface UART. Déjà présente sur les premiers ordinateurs et toujours utilisée, l'interface série est un peu la « mère des interfaces », un protocole de communication heureusement disponible depuis le connecteur d'extension du *Raspberry*.

Interfaces série

L'interface série UART (*Universal Asynchronous Receiver/Transmitter*) est l'une des trois interfaces série présente sur le connecteur d'extension du *Pi*, les deux autres étant les interfaces I²C et SPI.

Comme le montrent les **tableaux 1a** et **b**, l'interface UART est accessible depuis les broches 8 (*TxD*) et 10 (*RxD*). Malheureusement, un seul autre signal UART est présent sur le connecteur d'extension, à savoir RTS (*Request To Send*) sur la broche 11.
Difficile dans ce cas d'établir une communication entre deux machines.

Augmenter la tension

Nous avons d'abord besoin d'une interface RS-232 pour convertir le signal UART de 3,3 V en signaux de +/-12 V tels qu'utilisés par la norme RS-232. Sur le schéma de l'adaptateur présenté sur la **figure 1**, c'est un MAX3232 ou autre émetteur-récepteur RS-232 équivalent de 3,3 V qui nous fournit les niveaux du signal RS-232. La **figure 2** donne une idée de la configuration du circuit de l'auteur, avec un accessoire commercial pour l'interface RS232.

Désactiver la console série

Par défaut, la distribution *Raspbian* autorise l'accès à la console via l'interface UART du

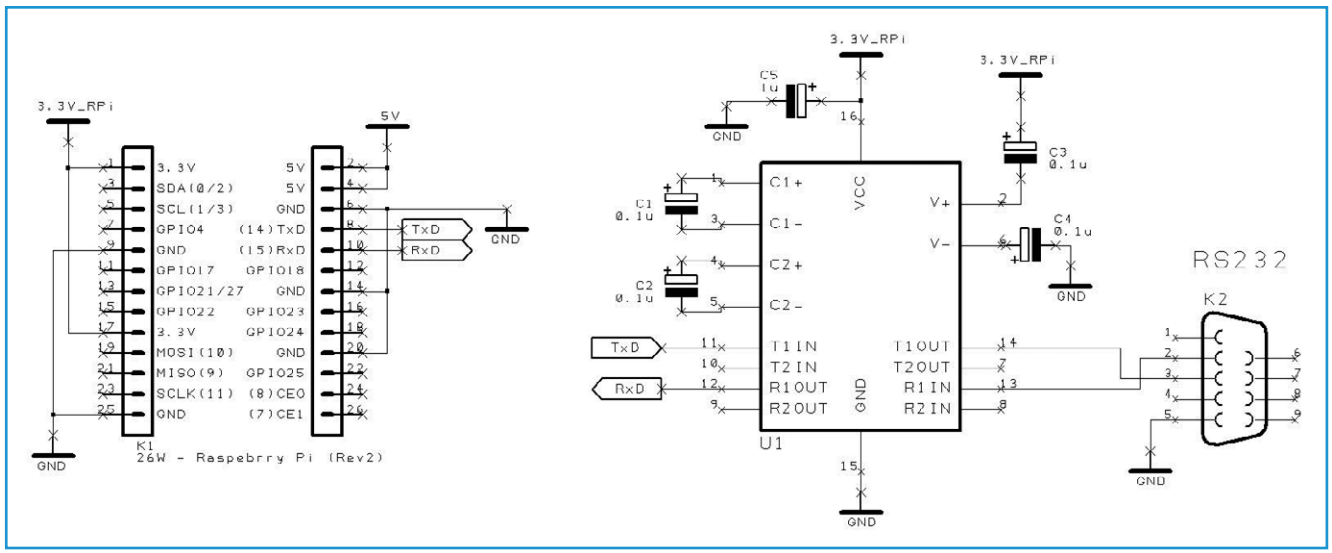

Figure 1.
Schéma de l'adaptateur RS-232 pour *Raspberry Pi.*

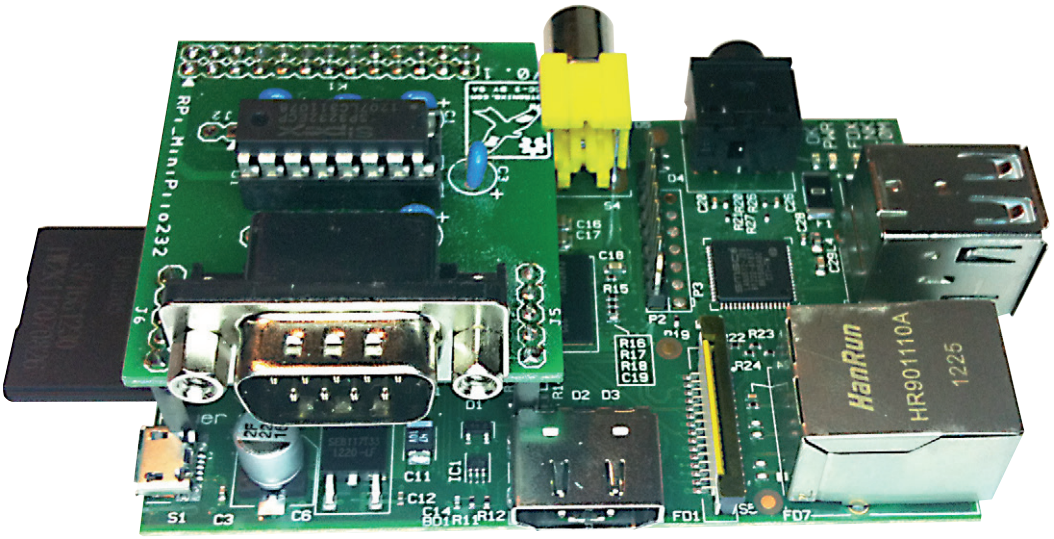

Figure 2.
Pi et la carte d'extension RS232.

Raspberry Pi. Cette fonction est très utile pour celui qui n'a ni écran ni clavier pour dialoguer avec le mini-ordinateur, mais peut se transformer en problème lorsqu'un programme utilisateur a besoin d'accéder au port série.

Nous allons donc commencer par désactiver cette option par défaut. Pour cela nous n'aurons pas à modifier plus de trois lignes des fichiers *cmdline.txt* et *inittab.*

Avant toute chose, effectuons une copie de sauvegarde de ces deux fichiers :

```
sudo cp /boot/cmdline.txt /boot/cmdline.
bak
sudo cp /etc/inittab /etc/inittab.bak
```

Utilisons ensuite *Leafpad* pour modifier *cmdline.txt* :

```
sudo leafpad /boot/cmdline.txt
```

Nous devons supprimer deux paramètres de configuration : *console=ttyAMA0,115200* et *kgdboc=ttyAMA0,115200*.
Votre ligne de configuration par défaut se présente probablement ainsi :

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

Après avoir supprimé *console=ttyAMA0,115200* et *kgdboc=ttyAMA0,115200*, la ligne doit être :

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

Enregistrez le fichier et fermez l'éditeur.

Désactivons enfin l'utilisation du port série *ttyAMA0* définie dans le fichier */etc/inittab :*

```
sudo leafpad /etc/inittab
```

En fin de fichier se trouve une ligne de configuration qui contient l'adresse du port *ttyAMA0.*

Commentez-la, autrement dit insérez le caractère # au début de cette ligne.

Enregistrez le fichier */etc/inittab*, fermez *Leafpad*, et redémarrez *Pi*.

Pour accéder au port série dans nos programmes, nous pouvons désormais utiliser l'adresse de périphérique *ttyAMA0*.

Installer la bibliothèque série *Python*

L'interpréteur *Python* est installé par défaut dans la distribution *Raspbian*. Pour accéder à l'interface UART du *Pi*, il ne nous reste donc plus qu'à installer la bibliothèque matérielle qui facilitera notre travail de programmation.

Tableau 1. Brochage du port série RS-232				
broche	**signal**	**description**	**ETTD**	**ETCD**
1	DCD	détection de porteuse	IN	OUT
2	RD (ou RxD)	réception de données	IN	OUT
3	TD (ou TxD)	transmission de données	OUT	IN
4	DTR	équipement prêt	OUT	IN
5	GND	masse du signal	GND	GND
6	DSR	prêt à recevoir	IN	OUT
7	RTS	demande de transmission	OUT	IN
8	CTS	autorisation d'émettre	IN	OUT
9	RI	détection de sonnerie	IN	OUT

Tableau 1a.			
nom de broche	**fonction**	**autre fonction**	**RPi.GPIO**
P1-02	5,0 V	-	-
P1-04	5,0 V	-	-
P1-06	GND	-	-
P1-08	GPIO14	UART0_TXD	RPi.GPIO8
P1-10	GPIO15	UART0_RXD	RPi.GPIO10
P1-12	GPIO18	PWM0	RPi.GPIO12
P1-14	GND	-	-
P1-16	GPIO23		RPi.GPIO16
P1-18	GPIO24		RPi.GPIO18
P1-20	GND	-	-
P1-22	GPIO25		RPi.GPIO22
P1-24	GPIO8	SPI0_CE0_N	RPi.GPIO24
P1-26	GPIO7	SPI0_CE1_N	RPi.GPIO26

Table 1b.				
nom de broche	**révision 1 de la carte**		**révision 2 de la carte**	
	fonction	**autre fonction**	**fonction**	**autre fonction**
P1-01	3,3 V	-	3,3 V	-
P1-03	GPIO0	I2C0_SDA	GPIO2	I2C1_SDA
P1-05	GPIO1	I2C0_SCL	GPIO3	I2C1_SCL
P1-07	GPIO4	GPCLK0	GPIO4	GPCLK0
P1-09	GND	-	GND	-
P1-11	GPIO17	RTS0	GPIO17	RTS0
P1-13	GPIO21		GPIO27	
P1-15	GPIO22		GPIO22	
P1-17	3,3 V	-	3,3 V	-
P1-19	GPIO10	SPI0_MOSI	GPIO10	SPI0_MOSI
P1-21	GPIO9	SPI0_MISO	GPIO9	SPI0_MISO
P1-23	GPIO11	SPI0_SCLK	GPIO11	SPI0_SCLK
P1-25	GND	-	GND	-
Note : I2C0_SDA et I2C0_SCL (GPIO0 & GPIO1), ainsi que I2C1_SDA et I2C1_SCL (GPIO2 & GPIO3) sont dotées de résistances de rappel de 1,8 kΩ au 3,3 V.				

Elle s'appelle *pySerial* et n'est pas fournie par défaut avec *Raspbian*. Pour la télécharger et l'installer, entrez dans une console *Lxterminal* :

```
sudo apt-get install python-serial
```

Comme sur la **figure 3**.

Programme d'exemple : serial.py

Maintenant que la bibliothèque *pySerial* est installée, testons-la à l'aide d'un court programme qui envoie des caractères à l'émulateur de terminal d'un PC.

Double-cliquez sur l'icône *IDLE* du bureau pour lancer l'*EDI* et la console *Python* (**fig. 4**).

Dans le menu *File*, sélectionnez *New Window* pour créer un nouveau programme dans l'éditeur.

Copiez le programme du **listage 1** dans l'éditeur (**fig. 5**).

Sauvegardez le programme saisi. Vous devez le rendre exécutable. Pour cela, retournez dans la console *Lxterminal*, et entrez la commande suivante :

```
chmod +x serial.py
```

Et pour exécuter ce programme :

```
sudo ./serial.py
```

Et voilà, avec un émulateur de programme ouvert côté PC et le port RS-232 de ce même PC connecté au *Pi*, l'émulateur de terminal devrait montrer le message « Hello Elektor ».

(130151 – version française Hervé Moreau)

Listage 1.

```
#!/usr/bin/python

import serial

ser = serial.Serial ('/dev/ttyAMA0', 115200, timeout=1)
ser.write ("Hello Elektor")
ser.close ()
```

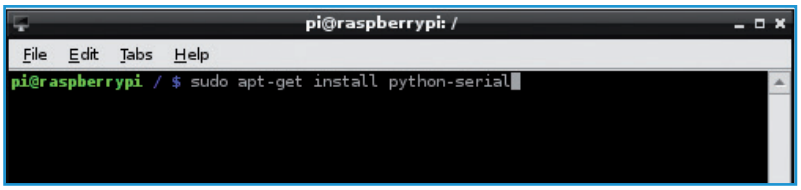

Figure 3. *LXTerminal.*

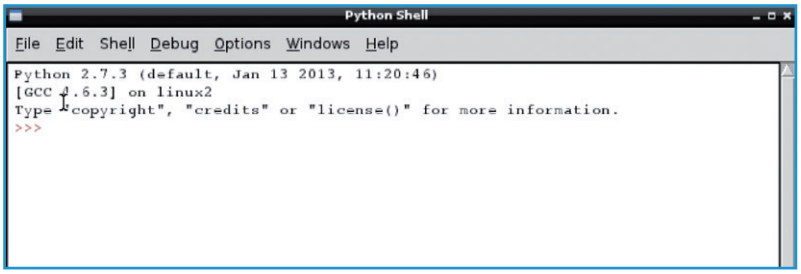

Figure 4. IDLE Python Shell.

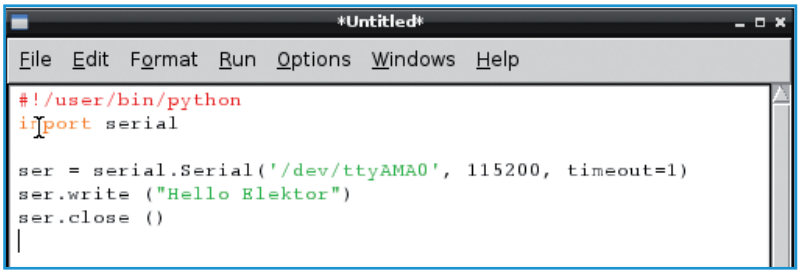

Figure 5. L'éditeur *IDLE*.

Références :

Raspberry Pi :
www.raspberrypi.org

Bibliothèque *pySerial* :
https://pypi.python.org/pypi/pyserial

Carte d'extension MiniPiio RS232 :
www.dtronixs.com