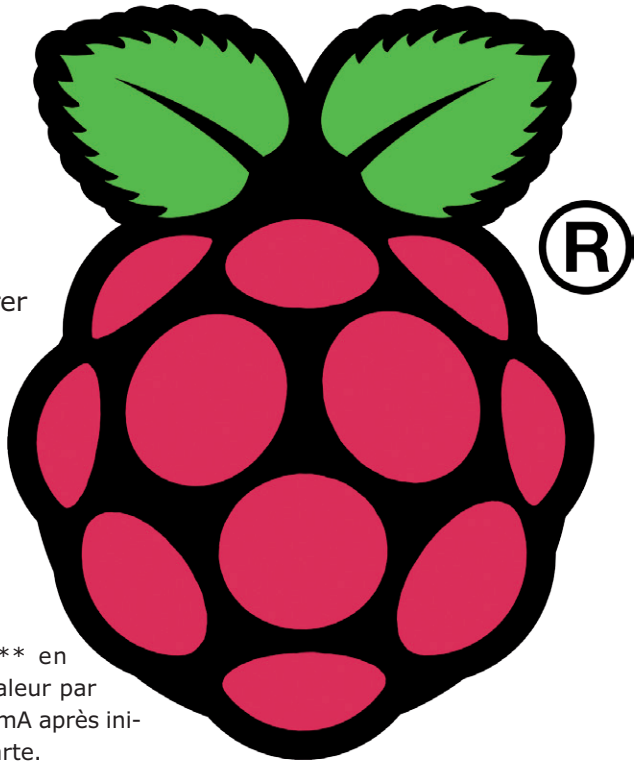
Raspberry Pi – part nº 2

à chacun sa part de tarte à la framboise : bien choisir sa patte



Dans le premier .POST consacré à Raspberry Pi, nous avions expliqué comment installer *Raspbian* et configurer cet ordinateur au nom framboisé. Nous allons voir ici comment programmer les broches GPIO de son connecteur d'extension, un connecteur qui vous sera déjà familier si vous avez lu l'article « carte de prototypage pour Raspberry Pi » du numéro de mars d'Elektor [1].

Tony Dixon
(Royaume-Uni)

Connecteur d'extension : donner de la consistance à la patte

Le côté bon marché de la carte mis à part, aux yeux d'un électronicien le connecteur d'extension représente certainement l'aspect matériel le plus excitant du Raspberry Pi. Placé à côté de la sortie vidéo composite, l'accès à ses 26 signaux est facilité par sa structure même : une barrette sécable double rangée au pas de 2,54 mm.
Ces 26 signaux se répartissent en trois catégories :

- Alimentation : 5 V CC, 3,3 V CC (*), ainsi que la masse.
  (*) Notez que le rail 3,3 V ne peut fournir que 50 mA.
- Entrée/sortie : les signaux des entrées/sorties à usage général (GPIO).
- Interfaces de communication : UART série, SPI et I²C

La plupart des 17 GPIO disponibles sur le connecteur d'extension peuvent être dotées d'autres fonctions, en particulier d'interfaces SPI, I²C et UART série (**tableau 1**).
Chaque GPIO peut fournir entre 2 et 16 mA suivant la configuration de l'étage de sortie. Cette configuration se fait par l'intermédiaire d'un registre (cf. la fiche technique du BCM2835 (lien*** en fin d'article), la valeur par défaut étant de 8 mA après initialisation de la carte.
La révision 2 du Raspberry est équipée d'un second connecteur d'extension, plus petit, nommé P5 (**tableau 2**). Il donne accès à quatre GPIO supplémentaires, ainsi qu'à l'interface audio PCM de la puce Broadcom 2835, un plus que les audiophiles apprécieront.
La révision 2 a également apporté quelques changements aux signaux disponibles sur le connecteur P1. L'interface *I2C0* a ainsi été remplacée par *I2C1*. La modification est mineure, mais souvenez-vous de ce détail si vous prévoyez d'utiliser des périphériques I²C.

Installation de la bibliothèque Python GPIO

Avant de mettre en œuvre un exemple classique, nous avons besoin d'une bibliothèque d'accès aux E/S matérielles du Pi. Il en existe plusieurs, nous nous servirons ici de la bibliothèque Python *RPi.GPIO*. Inutile de se préoccuper de Python lui-même, l'interpréteur est présent par défaut sur la distribution Raspbian. Commençons donc par installer les outils de développement Python depuis un terminal LX (**fig. 1**) :

Tableau 1. Brochage du connecteur d'extension

nom de la 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

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.

Tableau 2. Brochage du connecteur P5

nom de la broche	fonction	autre fonction
P5-01	5,0 V	
P5-02	3,3 V	
P5-03	GPIO28	PCM_CLK
P5-04	GPIO29	PCM_FS
P5-05	GPIO30	PCM_DIN
P5-06	GPIO31	PCM_DOUT
P5-07	GND	
P5-08	GND	

Figure 1.
Le terminal LX.

```
sudo apt-get install python-dev
```

Servons-nous ensuite de l'utilitaire *wget* pour télécharger le paquet contenant les sources de *RPi.GPIO* [2] :

```
wget http://pypi.python.org/
packages/source/R/RPi.GPIO/RPi.GPIO-
0.5.0a.tar.gz
```

Une fois l'archive téléchargée, extrayez-la avec :

```
tar -zxf RPi.GPIO-0.5.0a.tar.gz
```

La commande a créé un nouveau dossier contenant les fichiers de l'archive. Placez-vous dans ce dossier :

```
cd RPi.GPIO-0.5.0a
```

Lancez l'installation avec :

```
sudo python setup.py install
```

La bibliothèque Python *RPi.GPIO* devrait maintenant être installée.

Hello la LED qui clignote : blinky.py

Nous avons tout ce qu'il faut côté logiciel pour programmer le clignotement d'une LED. La **figure 2** montre notre montage. Nous avons utilisé une petite carte de prototypage MiniPiio [3] avec notre Pi, et nous avons câblé une LED et une résistance de 680 Ω entre *GPIO017* (broche 11) et la masse.
Une fois votre montage prêt, double-cliquez sur l'icône IDLE du bureau pour lancer l'EDI et la console Python (**fig. 3**).
Pour créer un nouveau programme dans

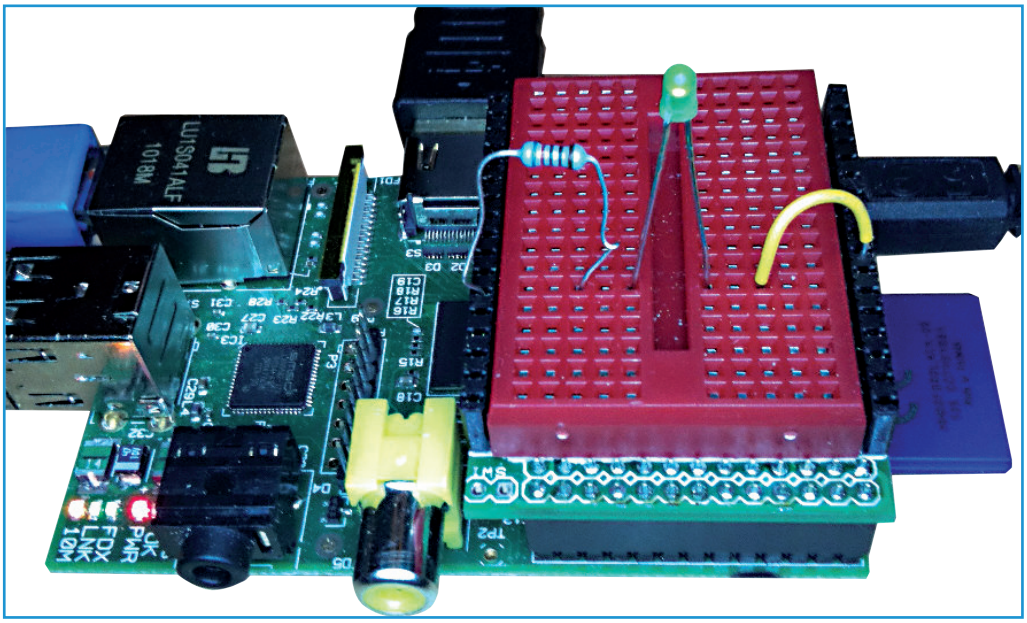


Figure 2.
Pi et la carte MiniPiio.

l'éditeur, choisissez *New Window* dans le menu *File*.
Dans l'éditeur (**fig. 4**), copiez le code du **Listage 1**.

Sauvegardez-le sous le nom *blinky.py*, puis dans un terminal LX rendez-le exécutable :

```
chmod +x blinky.py
```

Pour le lancer, entrez :

```
sudo ./blinky.py
```

Astuce : si vous démarrez IDLE depuis un terminal en faisant précéder la commande par **sudo** (soit **sudo idle**), vous disposerez *de facto* des droits corrects pour exécuter vos programmes « RPi.GPIO » depuis IDLE.

Mode d'adressage des broches avec RPi.GPIO

La bibliothèque *RPi.GPIO* permet de désigner les broches GPIO de deux façons. Un programme qui importe *RPi.GPIO* doit donc toujours définir le mode d'adressage utilisé avec ou *GPIO.setmode(GPIO.BOARD)*, ou *GPIO.setmode(GPIO.BCM)*.

Listage : blinky.py

```
# !/usr/bin/python
import time
import RPi.GPIO as GPIO

# Configure Pi's GPIO pins
GPIO.setmode (BCM)
GPIO.setup (17,GPIO.OUT)

# Program loop
while True :
  GPIO.output (17,True)
  time.sleep (1)
  GPIO.output (17,True)
  time.sleep (1)
```

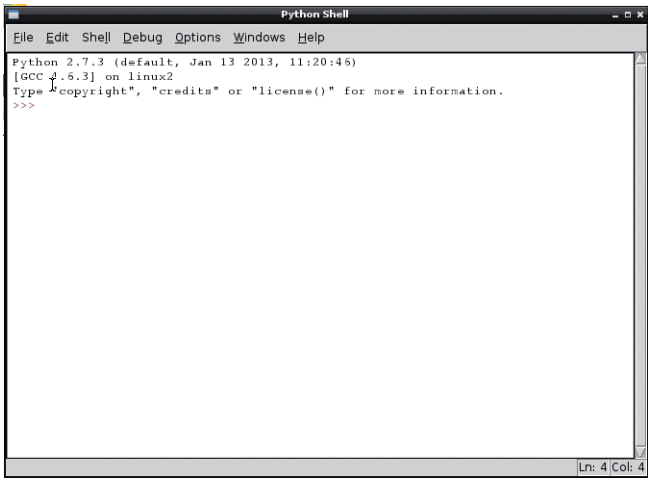


Figure 3.
La console Python d'IDLE.

Avec la méthode spécifiée par *GPIO.setmode(GPIO.BOARD)*, c'est le numéro de broche du connecteur d'extension P1 de la carte Raspberry Pi qui est utilisé. Avantage de cette méthode, le code n'aura pas à être corrigé si un jour les signaux du connecteur d'extension sont modifiés.
Avec *GPIO.setmode(GPIO.BCM)*, c'est la numérotation des GPIO telle que définie par le fabricant de la puce BCM2835 qui est utilisée. Pour vous aider à y voir plus clair entre ces différentes conventions d'adressage, le **tableau 3** établit la correspondance entre un numéro de broche, le nom du signal GPIO, et sa numérotation selon le mode d'adressage spécifié.
*** http://fr.scribd.com/doc/101830961/GPIO-Pads-Control2

(130110 – version française : Hervé Moreau)

Références

[1] www.elektor.com/120483
[2] https://pypi.python.org/pypi/RPi.GPIO
[3] www.dtronixs.com

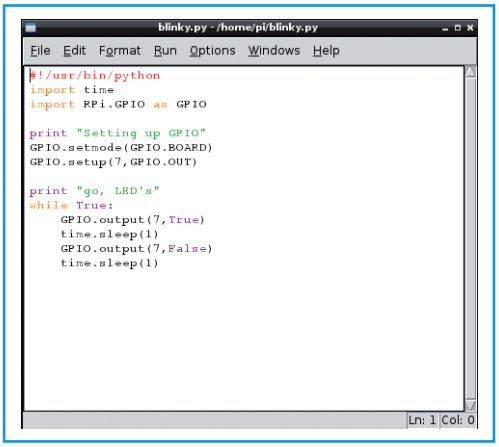


Figure 4.
L'éditeur d'IDLE.

Tableau 3. GPIO.setmode(GPIO.BCM) et GPIO.setmode(GPIO.BOARD)

nom de la broche	fonction	GPIO.setmode	
		GPIO.BCM	GPIO.BOARD
P1-01	3,3 V	-	-
P1-02	5,0 V	-	-
P1-03	GPIO0/2*	0/2	3
P1-04	5.0V	-	-
P1-05	GPIO1/3*	1/3	5
P1-06	GND	-	-
P1-07	GPIO4	4	7
P1-08	GPIO14	14	8
P1-09	GND	-	-
P1-10	GPIO15	15	10
P1-11	GPIO17	17	11
P1-12	GPIO18	18	12
P1-13	GPIO21/27*	21	13
P1-14	GND	-	-
P1-15	GPIO22	22	15
P1-16	GPIO23	23	16
P1-17	3,3 V	-	-
P1-18	GPIO24	24	18
P1-19	GPIO10	10	19
P1-20	GND	-	-
P1-21	GPIO9	9	21
P1-22	GPIO25	25	22
P1-23	GPIO11	11	23
P1-24	GPIO8	8	24
P1-25	0 V	-	-
P1-26	GPIO7	7	26

Note : * changements apportés par la révision 2.