Entrées/sorties GPIO sur Raspberry Pi

Cet article est un cours d'introduction aux entrées et sorties numériques sur carte Raspberry Pi, permettant de comprendre leur fonctionnement, leur connexion et de les tester depuis le shell, avant de voir leur programmation dans différents langages.

http://www.pobot.org/Entrees-sorties-GPIO-sur-Raspberry.html

jeudi 6 septembre 2012, par Julien H.

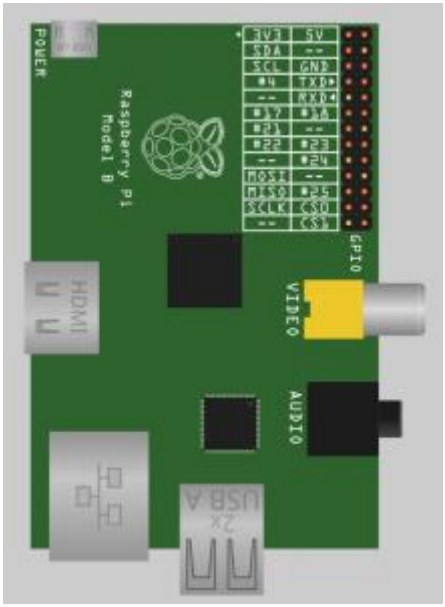
La carte Raspberry Pi donne accès à des **entrées et sorties numériques** appelées GPIO (en anglais "general purpose input & output") contrôlées par le processeur ARM.

Elles sont à usage multiple :
- en entrée numérique tout ou rien, pour détecter un interrupteur par exmeple
- en sortie numérique tout ou rien, pour activer un relais par exemple
- en sortie numérique PWM, pour contrôler la puissance moyenne d'une led par exemple
- en protocole I2C, d'échanger avec une ou plusieurs puces
- en protocole SPI, idem
- en protocole UART, d'échanger avec une seule puce (ou un PC)

D'autres usages sont possibles (audio PCM, vidéo sur les connecteurs DSI et CSI), nous y reviendrons si nécessaire, contactez-nous pour toute question d'ordre technique.

Plusieurs connecteurs donnent accès aux GPIO, mais le principal est un connecteur comportant 2 rangées de 13 picots mâles distants du pas standard de 2,54 mm.

Voici les signaux qu'on peut y trouver. Certaines pattes ont deux usages, mais pour plus de simplicité, ne tenez compte que de celui indiqué sur l'image ci-dessous pour réaliser vos premiers montages.



Position du connecteur GPIO de la RPi

Image réalisée depuis Fritzing avec le modèle d'Adafruit.

Bien entendu cela limite les pattes de connexion I/O simples à 8 seulement, mais si vous voulez avoir accès aux 53 GPIO, à vos risques et périls ! Si vous avez besoin de plus de 8 pattes, il est peut-être préférable d'utiliser un circuit d'extension, mais effectivement on peut en récupérer 5 autres si on n'a pas besoin des usages spécifiques (SPI par exemple) du connecteur principal. Et encore quelques autres en se connectant sur les autres connecteurs.

Surtout, notez que le numéro de GPIO n'est pas sa position sur le connecteur, mais son numéro dans les registres de la puce ARM BCM2835. C'est donc celui qui sera utilisé dans la plupart des bibliothèques d'accès.

Premier test

Avant d'utiliser un langage de programmation pour faire son propre logiciel, voyons comment tester les pattes d'entrées/sortie depuis le shell, en ligne de commande. Cet usage est vraiment pratique et permet de tirer parti de la présence d'un système d'exploitation Linux, avec toutes les fonctionnalités disponibles en ligne de commande ou en script bash (boucles, accès aux fichiers, etc...).

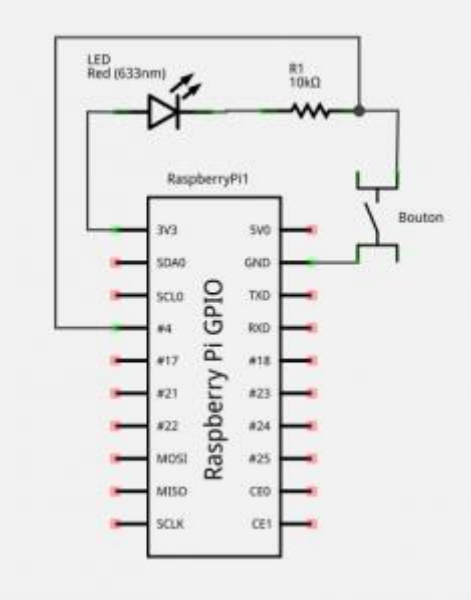
On va utiliser les éléments suivants pour lire une entrée reliée à un bouton et pour faire allumer une led :

- une carte Raspberry Pi
- une distribution Linux basée sur Debian (l'Occidentalis 1.0)
- une plaque d'essai Labdec ou breadboard
- un bouton (interrupteur)
- une résistance 10k
- une led
- des fils de connexion

C’est le minimum. Vous pouvez utiliser un clavier et un écran pour utiliser un terminal de la Raspberry Pi, ou y accéder depuis un PC distant grâce à une liaison réseau Ethernet.

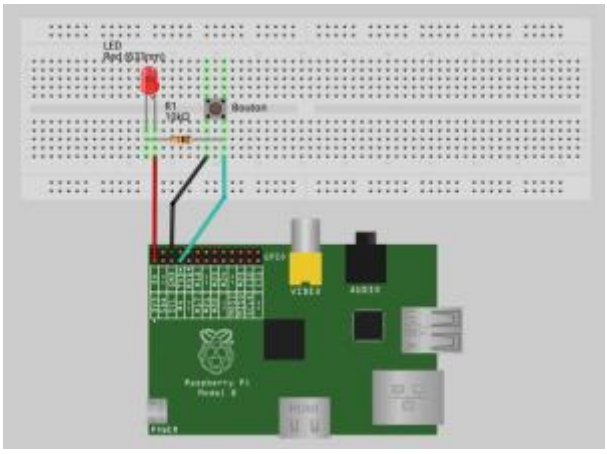
Le montage

Voici le schéma du circuit ainsi réalisé. Notez que la Raspberry Pi est stylisée, réduite à son connecteur d’entrées/sorties principal.



Montage simple autour du GPIO RPi

Pour les débutants, voici une vue des composants tels qu’on les implantera sur une plaque d’essai :



Implantation des composants

L’installation logicielle

Comme nous aimons la simplicité, et que nous avons choisi la Raspberry Pi pour la communauté d’experts Linux capables d’écrire de bons programmes fiables, nous n’allons pas utiliser de codes complexes mais des outils pratiques.

Pour accéder aux GPIO depuis le shell en ligne de commande, nous utiliserons donc **WiringPi**, une bibliothèque C et des outils de compilation partagés par Gordon sur son site.

L'installation est un peu différente de celle qu'il décrit sur son site, alors je me permets de recopier ce que j'ai fait (sur une Raspbian / Occidentalis) :

```
atelier@ubuntu:~$ ssh pi@10.42.0.77
pi@10.42.0.77's password: raspberry
Linux raspberrypi 3.2.27+ #102 PREEMPT Sat Sep 1 01:00:50 BST 2012 armv6l
pi@raspberrypi ~ $ sudo apt-get install git-core
...
pi@raspberrypi ~ $ mkdir gpio
pi@raspberrypi ~ $ cd gpio/
pi@raspberrypi ~/gpio $ git clone git://git.drogon.net/wiringPi
...
pi@raspberrypi ~/gpio $ cd wiringPi/
pi@raspberrypi ~/gpio/wiringPi $ git pull origin
remote: Counting objects: 11, done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 8 (delta 6), reused 0 (delta 0)
Unpacking objects: 100% (8/8), done.
From git://git.drogon.net/wiringPi
  99095e3..30d79da  master     -> origin/master
Updating 99095e3..30d79da
Fast-forward
wiringPi/wiringPi.c |  108 ++++++++++++++++++++++++++++++++++++++++++++++--
1 file changed, 104 insertions(+), 4 deletions(-)
pi@raspberrypi ~/gpio/wiringPi $ cd wiringPi/
pi@raspberrypi ~/gpio/wiringPi/wiringPi $ sudo make install
[CC] wiringPi.c
[AR] wiringPi.o wiringPiFace.o wiringSerial.o wiringShift.o gertboard.o
piNes.o lcd.o piHiPri.o piThread.o softPwm.o wiringPiSPI.o
  text           data            bss            dec            hex
filename
  5992            516            304           6812           1a9c
wiringPi.o (ex libwiringPi.a)
  1908              8              8           1924            784
wiringPiFace.o (ex libwiringPi.a)
   828              0              0            828            33c
wiringSerial.o (ex libwiringPi.a)
  1696              0              0           1696            6a0
wiringShift.o (ex libwiringPi.a)
   208              0              0            208             d0
gertboard.o (ex libwiringPi.a)
   828              0            100            928            3a0
piNes.o (ex libwiringPi.a)
  2452              0              4           2456            998
lcd.o (ex libwiringPi.a)
    76              0              0             76             4c
piHiPri.o (ex libwiringPi.a)
    76              0             96            172             ac
piThread.o (ex libwiringPi.a)
   372              4            512            888            378
```

```
softPwm.o (ex libwiringPi.a)
   420              1             20            441            1b9
wiringPiSPI.o (ex libwiringPi.a)
[install]
install -m 0755 -d /usr/local/lib
install -m 0755 -d /usr/local/include
install -m 0644 wiringPi.h        /usr/local/include
install -m 0644 wiringSerial.h        /usr/local/include
install -m 0644 wiringShift.h        /usr/local/include
install -m 0644 gertboard.h        /usr/local/include
install -m 0644 piNes.h               /usr/local/include
install -m 0644 softPwm.h        /usr/local/include
install -m 0644 lcd.h                /usr/local/include
install -m 0644 wiringPiSPI.h        /usr/local/include
install -m 0644 libwiringPi.a        /usr/local/lib
pi@raspberrypi ~/gpio/wiringPi $ cd ../gpio/
pi@raspberrypi ~/gpio/wiringPi/gpio $ sudo make install
cp gpio /usr/local/bin
chown root.root /usr/local/bin/gpio
chmod 4755 /usr/local/bin/gpio
mkdir -p /usr/local/man/man1
cp gpio.1 /usr/local/man/man1
pi@raspberrypi ~/gpio/wiringPi/gpio $
```

Les dernières commandes sont très importantes : elles copient le programme binaire compilé dans un répertoire permettant l'accès à la commande depuis n'importe où sur votre Linux. Elles installent aussi le manuel au bon endroit pour être appelé.

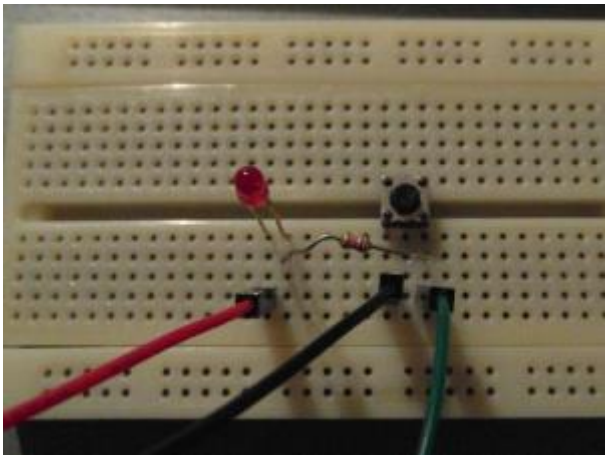
Si quelqu'un vous répond "RTFM" quand vous posez une question sur un programme, cela signifie "read the famous man" (en édulcoré), et il faut alors exécuter la commande man suivie du nom de l'exécutable concerné, depuis n'importe quel répertoire. Tapez "q" pour sortir.

```
pi@raspberrypi ~ $ man gpio
GPIO(14 June 2012)                                          GPIO(14 June
2012)

NAME
      gpio - Command-line access to Raspberry Pi and PiFace GPIO

SYNOPSIS
      gpio -v
      gpio [ -g ] read/write/pwm/mode ...
      gpio [ -p ] read/write/mode ...
      gpio unexportall/exports
      gpio export/edge/unexport ...
      gpio drive group value
      gpio pwm-bal/pwm-ms
      gpio pwmr range
      gpio load  i2c/spi
...
```

Il y a plusieurs pages. Vous pouvez les lire ou suivre nos tests.

Les tests

On a connecté la Raspberry au montage présenté plus haut, avec un signal d'entrée/sortie (interrupteur ou led) sur la patte 7 du connecteur, correspondant à la GPIO 4 de la Raspberry Pi.



Montage simple led + bouton

Cela ressemble au schéma, non ?

Selon l'explication donnée dans le manuel (man gpio), on pourra utiliser de manière équivalente les instructions suivantes :

```
pi@raspberrypi ~ $ gpio -g mode 4 out
pi@raspberrypi ~ $ gpio mode 7 out
```

On préférera la première écriture, qui sensibilise à l'usage général des GPIO et permettra d'aller vérifier les autres modes disponibles sur la patte 4 dans la doc du ARM BCM2835.

Voici pour éteindre la led puis la rallumer :

```
pi@raspberrypi ~ $ gpio -g mode 4 out
pi@raspberrypi ~ $ gpio -g write 4 1
pi@raspberrypi ~ $ gpio -g write 4 0
```

Pour lire l'état (il change en appuyant sur le bouton) :

```
pi@raspberrypi ~ $ gpio -g mode 4 in
pi@raspberrypi ~ $ gpio -g read 4
1
pi@raspberrypi ~ $ gpio -g read 4
0
pi@raspberrypi ~ $
```

On peut maintenant écrire un script bash :

```
pi@raspberrypi ~/gpio $ nano test.sh
#!/bin/bash

setup ()
{
 echo Setup
 gpio -g mode 4 in
}

waitButton ()
{
 echo -n "Waiting for button ... "
 while [ `gpio -g read 4` = 1 ]; do
   sleep 0.1
 done
 echo "Got it"
}

setup
while true; do
 waitButton
done
```

Un petit CTRL+O pour enregistrer et CTRL+X pour sortir et vous pouvez exécuter le programme :

```
pi@raspberrypi ~/gpio $ chmod 755 test.sh
pi@raspberrypi ~/gpio $ ./test.sh
Setup
Waiting for button ... Got it
Waiting for button ... Got it
Waiting for button ... Got it
Waiting for button ... Got it
```

Et là vous aurez atteint l'échelon numéro 1 en électronique ludique : vous avez découvert le problème de la détection de l'appui sur un bouton, avec la nécessité de détecter le passage sur un front (nom donné à un changement d'état) plutôt que sur l'état lui-même.

Vous pouvez maintenant rejoindre un club de robotique qui vous apprendra des solutions logicielles ou matérielles à ce problème... en attendant l'échelon 2 (nom de code : anti-rebond).

Vos commentaires