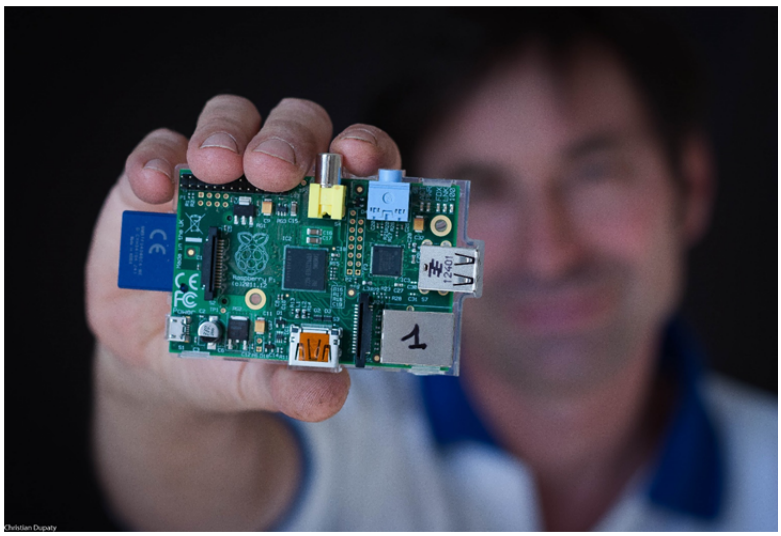
RASPBERRY PI

INSTALLATION-CONFIGURATION

INTERFACES DE COMMUNICATIONS



GPIO

Christian Dupaty

BTS Systèmes Numériques

Lycée Fourcade - Gardanne

Académie d'Aix-Marseille

1) TP : GPIO

17 GPIO (*General Purpose Input/Output*) sont disponibles sur le connecteur de la Raspberry Pi. La plupart des broches supportant les GPIO peuvent être réassignées à des périphériques de communication. En mode GPIO ces broches peuvent être configurées soit en entrée binaire soit en sortie binaire.

http://code.google.com/p/raspberry-gpio-python/wiki/Main
Le matériel : http://elinux.org/RPi_Low-level_peripherals
La bibliothèque : http://code.google.com/p/raspberry-gpio-python/

Installation :

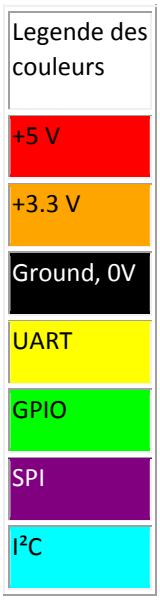
Installation de Rpi.GPIO, la bibliothèque de gestion Entrées/Sorties

1) récupérer la bibliothèque : `wget http://pypi.python.org/packages/source/R/RPi.GPIO/RPi.GPIO-0.3.1a.tar.gz`
2) Décompresser : `tar xvzf RPi.GPIO-0.3.1a.tar.gz` (remplacer le numéro de version si nécessaire)
3) Aller dans le répertoire de la bibliothèque : `cd RPi.GPIO-0.3.1a`
4) Installation : `sudo python setup.py install`
5) détruire l'archive et le dossier d'installation

```
cd ..
sudo rm –tf RPi.GPIO-0.3.1a/
rm RPi.GPIO-0.3.1a.tar.gz
```

6) OU avec PIP

```
apt-get remove python-rpi.gpio
pip install --upgrade RPi.GPIO
```

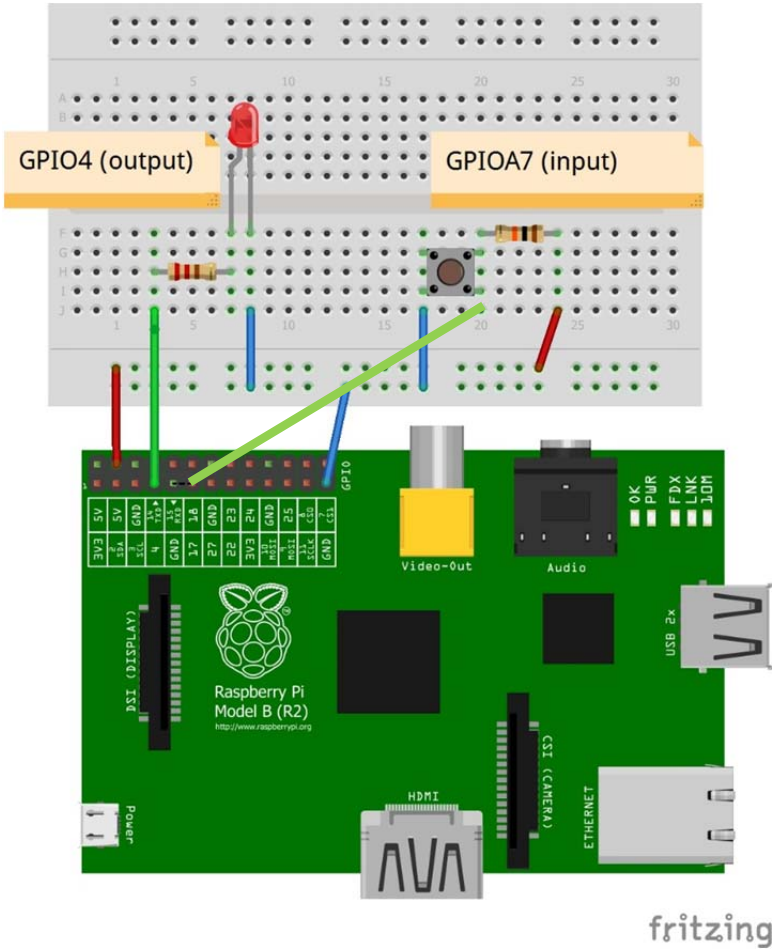


Utilisation dans un code Python

```
import RPi.GPIO as GPIO      # charge la bibliotheque
GPIO.setmode(GPIO.BOARD)  # numérotation du connecteur Rasperry, ex 7 pour GPIO4
GPIO.setmode(GPIO.BCM)    # numérotation du circuit BCM2835, ex 4 pour GPIO4

GPIO.setup(11, GPIO.OUT)   # place la broche (ou le GPIO) en sortie
GPIO.setup(7, GPIO.OUT, initial=GPIO.HIGH)  # GPIO en sortie , initialisé à 1
GPIO.setup(17, GPIO.IN)   # place la broche (ou le GPIO) en entree
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_UP)   # active une resistance
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)


GPIO.output(11, True)     # met le GPIO à l'état 1  (True ou 1)
GPIO.output(11, false)     # met le GPIO à l'état 0 (False ou 0)
GPIO.input(7)  # retourne l'état du GPIO
       # ex a= GPIO.input(7)  ou if (GPIO.input(7)) :

GPIO.cleanup()  # replace tous les GPIO en entrées
GPIO.RPI_REVISION  # version de la carte
GPIO.VERSION   # version de la bibliothèque
```

TESTS :

Placer une LED rouge en série avec une résistance de 150 ohms entre la broche 7 du connecteur (GPIO4) et le 0v et un bouton poussoir avec une résistance de pull-up (10K) sur la broche 11 (GPIO17)



Le programme suivant fait clignoter la LED avec une période de 1S.

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.OUT)

while True:
    print("Allume")
    GPIO.output(4,1) # Allume la LED
    time.sleep(0.5)
    print("Eteind")
    GPIO.output(4,0) # eteind la LED
    time.sleep(0.5)
```

Pour tester un front montant sur un GPIO:

```
while GPIO.input(channel) == GPIO.LOW:
    time.sleep(0.01)  # laisser 10mS au processeur pour faire autre chose
```

La function wait for edge fait la même chose mais sans time.sleep

```
GPIO.wait_for_edge(channel, GPIO.RISING)  # RISING, FALLING, BOTH
```

Détection d'évènements (Interruptions) :

Une interruption est un processus répondant à une sollicitation matériel asynchrone et déclenchant l'exécution d'un sous-programme, ex :

- L'appui sur un bouton
- Le débordement d'un TIMER
- L'arrivé d'un caractère dans une UART

Lorsque l'évènement est détecté par le matériel et suivant les niveaux de priorité accordés à l'évènement, le programme en cours est interrompu et un sous-programme d'interruption est exécuté.
L'utilisation des interruptions évite la scrutation et donc libère du temps pour d'autres tâches. Un évènement ne peut être manqué, l'évènement est mémorisé, si la tâche associée n'est pas prioritaire il peut être traité plus tard.

Commandes python d'activation des interruptions sur GPIO :

```
GPIO.add_event_detect(channel, GPIO.FALLING)  # active detection
faire_un_truc()
if GPIO.event_detected(channel):
    print('bouton appuye')
```

Annulation de la détection:

```
GPIO.remove_event_detect(channel)
```

Création d'une fonction qui sera appelée automatiquement lors de la détection de l'évenement.

```
def fonctionIT(channel):
    print('le bouton a été enfoncé!')
    print('sur le GPIO %s'%channel)

GPIO.add_event_detect(channel, GPIO.FALLING, callback=fonctionIT)
```

Pour éviter les rebonds:
La fonction add_event_detect recheche un front actif (FALLING ou RISING) et temporise (bouncetime) pour attendre la fin des rebonds

```
GPIO.add_event_detect(channel, GPIO.FALLING, callback=fonctionIT, bouncetime=200
)
```