

## TP4 :

### Utilisation des GPIO d'une raspberry pi

#### Objectifs :

Ces TP ont pour objectifs :


- Pratiquer les différentes manières d'utilisation des GPIO sur une raspberry pi 2.
- S'initier à la programmation python à travers les GPIO de la Rspi2.
- S'initier à la domotique.

#### 1. Introduction au GPIO de la Rspi2 :

La Rspi2 est considéré comme un Pocket PC (PC de poche) muni d'un connecteur J8 de 40 broches contenant :

- des broches d'entrée/sorties numériques (General Purpose Input/Output GPIO) .
- certaines GPIO ont des fonctions alternatives comme la communication via des protocoles standards comme UART, I2C, SPI ou la possibilité de fournir un signal PWM ou Clk.
- des broches d'alimentation (3.3V, 5V, GND).

**NB : Toutes les sorties GPIO à 1 sont au potentiel +3.3V et lorsqu'elles sont à 0 sont au potentiel 0V. De même les entrées GPIO doivent recevoir des potentiels +3.3V ou 0V. Une entrée attaquée par un niveau logique TTL de 5V peut détruire directement le Soc BCM2836 et rendre la Rspi2 inutilisable.**



Alternate Function	BCM Num	Physic Num	BCM Num	Alternate Function
	3.3V PWR	1	2	5V PWR
I2C1 SDA	GPIO 2	3	4	5V PWR
I2C1 SCL	GPIO 3	5	6	GND
	GPIO 4	7	8	UART0 TX
	GND	9	10	UART0 RX
	GPIO 17	11	12	GPIO 18
	GPIO 27	13	14	GND
	GPIO 22	15	16	GPIO 23
	3.3V PWR	17	18	GPIO 24
SPI0 MOSI	GPIO 10	19	20	GND
SPI0 MISO	GPIO 9	21	22	GPIO 25
SPI0 SCLK	GPIO 11	23	24	GPIO 8
	GND	25	26	GPIO 7
	Reserved	27	28	Reserved
	GPIO 5	29	30	GND
	GPIO 6	31	32	GPIO 12
	GPIO 13	33	34	GND
SPI1 MISO	GPIO 19	35	36	GPIO 16
	GPIO 26	37	38	GPIO 20
	GND	39	40	GPIO 21
				SPI0 CS0
				SPI0 CS1
				SPI1 CS0
				SPI1 MOSI
				SPI1 SCLK

Figure 2.1 Les GPIO de la Rspi2 avec leur numérotation BCM et physique

Les broches GPIO jouant un double rôle, sont par défaut configurées en GPIO numériques, à moins de les activer pour une autre fonction ; En somme, on a :

24x GPIO pins au max  
2x SPI bus  
2x 5V power pins

1x Serial UART  
1x I2C bus

2x 3.3V power pins  
8x Ground pins

En plus chaque broche peut être configurée pour être reliée par une résistance à +3.3V (Pull-Up) ou par une résistance à Gnd (Pull-Down) (voir La figure 2.2)

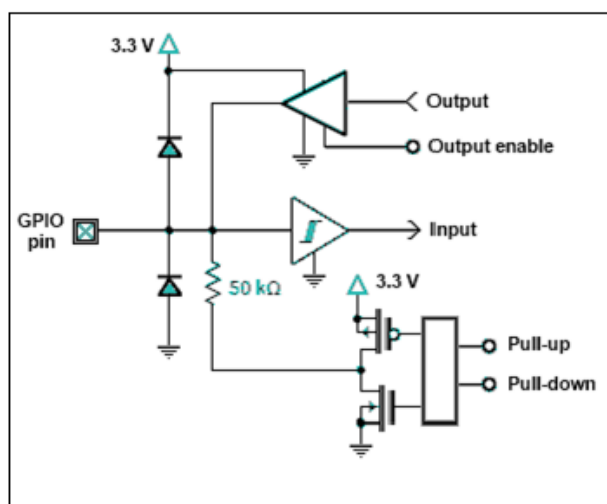


Figure 2.2 Schéma de branchement interne d'un GPIO

Le tableau 2.1 suivant indique la configuration par défaut des broches à la mise sous tension :

Tableau 2.1 : Branchement interne des GPIO à la mise sous tension

GPIO#	Power-on Pull	Alternate Functions	Header Pin
2	PullUp	I2C1 SDA	3
3	PullUp	I2C1 SCL	5
4	PullUp		7
5	PullUp		29
6	PullUp		31
7	PullUp	SPI0 CS1	26
8	PullUp	SPI0 CS0	24
9	PullDown	SPI0 MISO	21
10	PullDown	SPI0 MOSI	19
11	PullDown	SPI0 SCLK	23
12	PullDown		32
13	PullDown		33

GPIO#	Power-on Pull	Alternate Functions	Header Pin
16	PullDown	SPI1 CS0	36
17	PullDown		11
18	PullDown		12
19	PullDown	SPI1 MISO	35
20	PullDown	SPI1 MOSI	38
21	PullDown	SPI1 SCLK	40
22	PullDown		15
23	PullDown		16
24	PullDown		18
25	PullDown		22
26	PullDown		37
27	PullDown		13
35*	PullUp		Red Power LED
47*	PullUp		Green Activity LED

\* = Raspberry Pi 2 seulement GPIO 35 & 47 ne sont pas disponibles pour Raspberry Pi 3.

plus d'info sur les GPIO : <https://pinout.xyz/#>

## 2. Utilisation des GPIO de la Rspi2 en mode commande (Shell):

L'utilisation des GPIO d'une Rspi2 peut être fait:

- en mode commandes et scripts shell : directement à travers l'interface système /sys (système de fichier virtuel)
- en mode commande en utilisant des utilitaires binaires opérant en mode commande, comme **gpio** et **pigs**. basé la librairie **wiringpi** ou l'utilitaire **pigs** installé avec la librairie **pigpio**.
- via des programmes écrits en différents langages (C, Python, ...) basés sur des librairies, comme :
  - **wiringpi** (voir <http://wiringpi.com/>)
  - **pigpio** (voir <http://abyz.co.uk/rpi/pigpio/>)
  - **Rpi.GPIO** (<https://pypi.python.org/pypi/RPi.GPIO>)

L'utilisation des GPIO implique des configurations bas niveau du Soc BCM2836 ; elle peut donc nécessiter les droits du super-utilisateur (root).

### 2.1. Utilitaire binaire gpio (accompagnant wiringPi) sous Raspbian

La distribution **Raspbian** inclut par défaut une installation de la librairie **wiringpi** et par conséquent elle inclut aussi un utilitaire appelé **gpio** qui peut être utilisé pour configurer et programmer les GPIO de la Rspi2.

2.1.1. Connectez-vous à la Rspi2 à l'aide de Vnc Wierver et ouvrez un terminal virtuel (**Voir TP1**);

2.1.2. Vérifiez l'installation de la librairie **wiringpi** en utilisant la commande suivante qui doit afficher la version de la librairie actuellement installée :

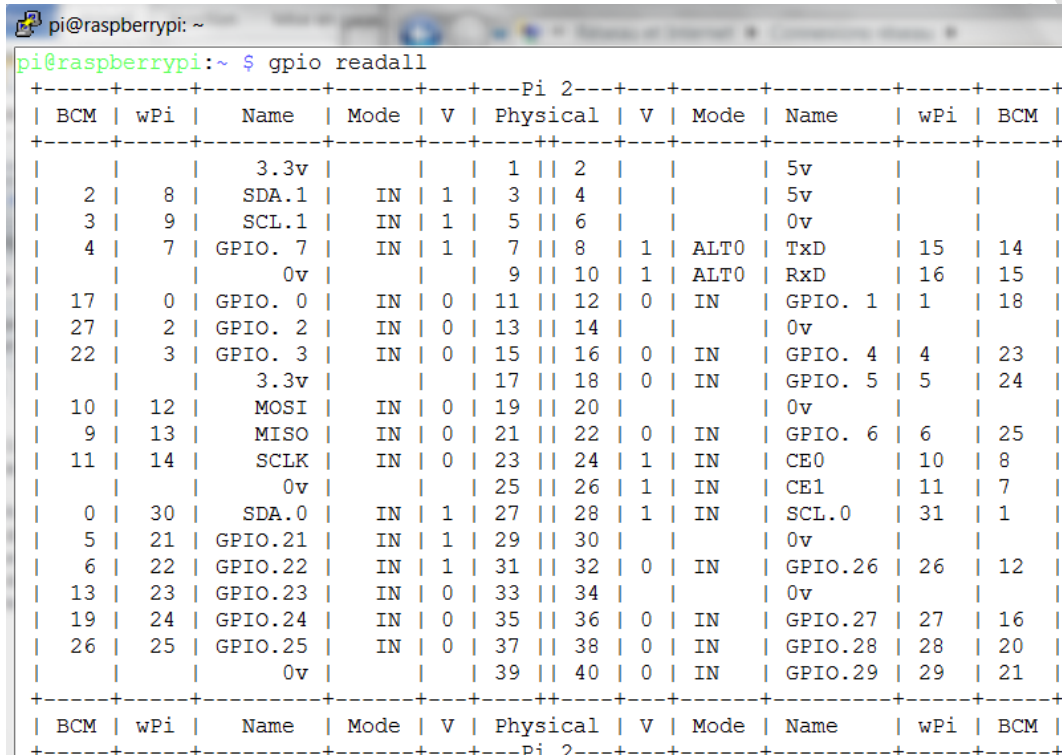
```
pi@raspberrypi ~$ gpio -v
```

2.1.3. La librairie **wiringpi** est sous licence GPL (donc free), malheureusement elle n'est plus mise à jour par son développeur. Son manuel d'utilisation peut être affiché par la commande

```
pi@raspberrypi ~$ man gpio
```

2.1.4. Lisez l'état de toutes les GPIO à l'aide de la commande :

```
pi@raspberrypi ~$ gpio readall
```



BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
		3.3v			1	2		5v		
2	8	SDA.1	IN	1	3	4		5v		
3	9	SCL.1	IN	1	5	6		0v		
4	7	GPIO. 7	IN	1	7	8	1	ALT0	TxD	15
		0v			9	10	1	ALT0	RxD	16
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1
27	2	GPIO. 2	IN	0	13	14		0v		
22	3	GPIO. 3	IN	0	15	16	0	IN	GPIO. 4	4
		3.3v			17	18	0	IN	GPIO. 5	5
10	12	MOSI	IN	0	19	20		0v		
9	13	MISO	IN	0	21	22	0	IN	GPIO. 6	6
11	14	SCLK	IN	0	23	24	1	IN	CE0	10
		0v			25	26	1	IN	CE1	11
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31
5	21	GPIO.21	IN	1	29	30		0v		
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26
13	23	GPIO.23	IN	0	33	34		0v		
19	24	GPIO.24	IN	0	35	36	0	IN	GPIO.27	27
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28
		0v			39	40	0	IN	GPIO.29	29

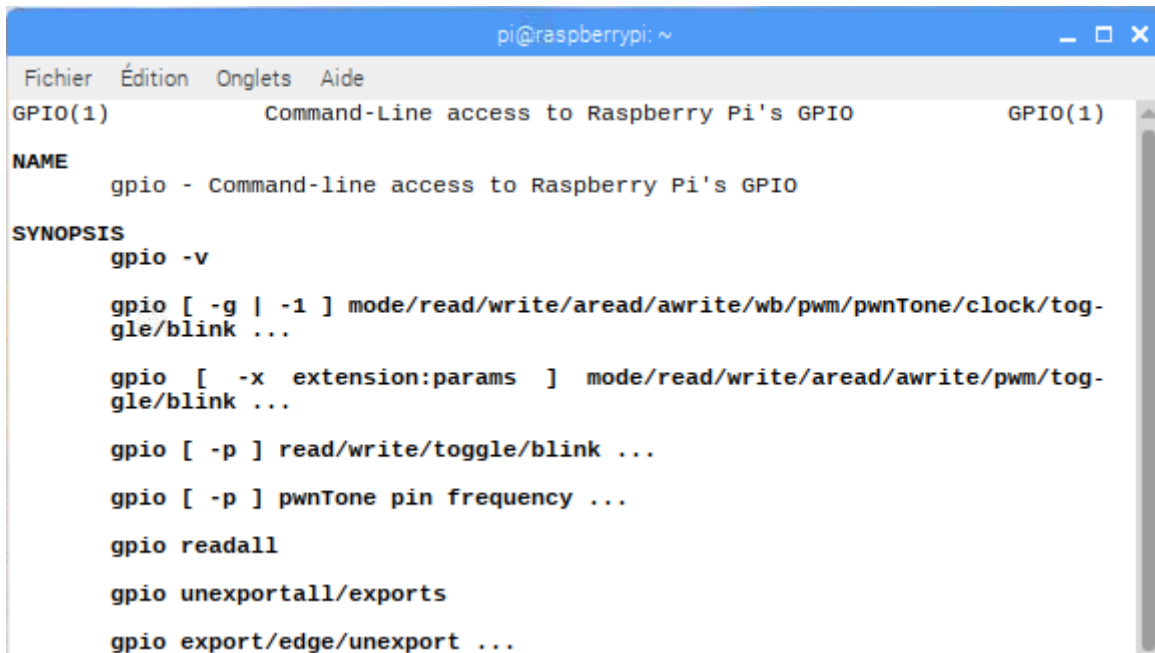
**NB: il y a trois numérotations des broches :**

- la numérotation physique : position de la broche dans le connecteur de 1 à 40
- la numérotation constructeur Broadcom : BCM
- la numérotation établie par l'éditeur de la bibliothèque wiringpi : wPi.

**Exemple :** la broche ayant le numéro physique 29, possède le numéro constructeur BCM 6 et le numéro wPi 22 ou le nom GPIO. 22.

2.1.5. Le tableau précédent montre la configuration de chaque GPIO pin (In or Out) et sa valeur (1 ou 0), à la mise sous tension ce tableau est compatible avec le tableau 2.1.

2.1.6. Ci-dessous la syntaxe générale de la commande **gpio** issue du manuel



```
pi@raspberrypi: ~
Fichier  Édition  Onglets  Aide
GPIO(1)  Command-Line access to Raspberry Pi's GPIO  GPIO(1)

NAME
  gpio - Command-line access to Raspberry Pi's GPIO

SYNOPSIS
  gpio -v

  gpio [ -g | -1 ] mode/read/write/aread/awrite/wb/pwm/pwnTone/clock/tog-
  gle/blink ...

  gpio [ -x extension:params ] mode/read/write/aread/awrite/pwm/tog-
  gle/blink ...

  gpio [ -p ] read/write/toggle/blink ...

  gpio [ -p ] pwnTone pin frequency ...

  gpio readall

  gpio unexportall/exports

  gpio export/edge/unexport ...
```

NB : les numéros de pin sont interprétés par défaut selon la numérotation wiringpi sauf si on ajoute à la commande l'option :

- -g : auquel cas ils le seront selon numérotation BCM
- -1 : auquel cas ils le seront selon numérotation physique

Et par la suite quelques utilisation de base de bases:

**gpio mode <pin> in|out |up|down|tri**

Elle configure la broche <pin> (<pin> **numéro wpi de la broche**) comme entrée/sortie. En plus elle permet de choisir une résistance interne en mode PullUp ou PullDown ou sans résistance (Tristate ou Open Drain).

**gpio write <pin> 0|1**

Met à 1 ou 0 la pin ayant le **numéro wPi <pin>**

**gpio -g write <pin> 0|1**

Met à 1 ou 0 la pin ayant le **numéro BCM <pin>**

**gpio read <pin>**

Lit la pin ayant le **numéro wPi <pin>** et affiche sa valeur

**gpio -g read <pin>**

Lit la pin ayant le **numéro BCM <pin>** et affiche sa valeur

**gpio toggle <pin>**

Bascule l'état de la pin ayant le numéro wpi <pin>

**gpio -g toggle <pin>**

Bascule l'état de la pin ayant le numéro BCM<pin>

Essayez la série des commandes suivantes, en utilisant la numérotation wPi :

```
pi@raspberrypi ~$ gpio mode 0 out      #configure pin GPIO.0 en sortie
pi@raspberrypi ~$ gpio write 0 1       #met à 1 pin GPIO.0
pi@raspberrypi ~$ gpio read 0          #Affiche l'état de pin GPIO.0
pi@raspberrypi ~$ gpio readall         #diagnostiquer tous
pi@raspberrypi ~$ gpio toggle 0 ; gpio read 0  #bascule et lit GPIO.0
pi@raspberrypi ~$ gpio toggle 0 && gpio read 0  #bascule et lit GPIO.0
```

### **gpio mode <pin> pwm**

Configure la broche <pin> (wPi) en mode PWM (Pulse Width Modulation)

### **gpio pwm <pin> <dc>**

Attribue à la broche <pin> le rapport cyclique (duty cycle) <dc> dont la valeur doit être entre 0 et 1023 incluses ; si <dc>= 0 → 0% la broche toujours à 0 si <dc>=1023 → 100% la broche est toujours à 1, si <dc> =512 → 50%

**NB : Le Soc BCM2835 possède seulement deux blocs PWM : PWM0 et PWM1 ; pas toutes les GPIO sont utilisables en mode PWM, seules BCM GPIO12, GPIO13, GPIO18, GPIO19, GPIO40, GPIO41, GPIO45 (voir BCM2835 datasheet :**

<https://www.raspberrypi.org/app/uploads/2012/02/BCM2835-ARM-Peripherals.pdf>

, page 102 et 103) ce qui correspond pour une Rspi2 aux broches numéros (wPi) 23,26, 24, 1 les seules présentes sur le connecteur à 40 broches.

```
pi@raspberrypi ~$ gpio readall          #notez le mode de GPIO.26 (Wpi)
pi@raspberrypi ~$ gpio mode 26 pwm      #configure pin GPIO.26 en pwm
pi@raspberrypi ~$ gpio readall          #notez à nouveau le mode de GPIO.26
pi@raspberrypi ~$ gpio pwm-ms           #mode mark-space
pi@raspberrypi ~$ gpio pwmc 4095        #fixe le diviseur à sa val max
pi@raspberrypi ~$ gpio pwmr 1024        #fixe range à sa valeur par défaut
pi@raspberrypi ~$ gpio pwm 26 0         #fixe le rapp cyclique à 0%
pi@raspberrypi ~$ gpio read 26          #si répété donne tjrs 0
pi@raspberrypi ~$ gpio pwm 26 1024      #fixe le rapp cyclique à 100%
pi@raspberrypi ~$ gpio read 26          #si répété donne tjrs 1
pi@raspberrypi ~$ gpio pwm 26 512       #fixe le rapp cyclique à 50%
pi@raspberrypi ~$ gpio read 26          #si répété donne 0 ou 1
pi@raspberrypi ~$ gpio mode 26 out      #configure pin GPIO.26 en out
pi@raspberrypi ~$ gpio readall          #notez à nouveau le mode de GPIO.26
```

On peut utiliser aussi un oscilloscope software pour visualiser le signal pwm.

Installation de piscope:

```
pi@raspberrypi ~$ wget byz.me.uk/rpi/pigpio/piscope.tar
pi@raspberrypi ~$ tar xvf piscope.tar

pi@raspberrypi ~$ cd PISCOPE pi@raspberrypi ~$ make hf
pi@raspberrypi ~$ make install
```

Lancez le daemon **pigpiod**

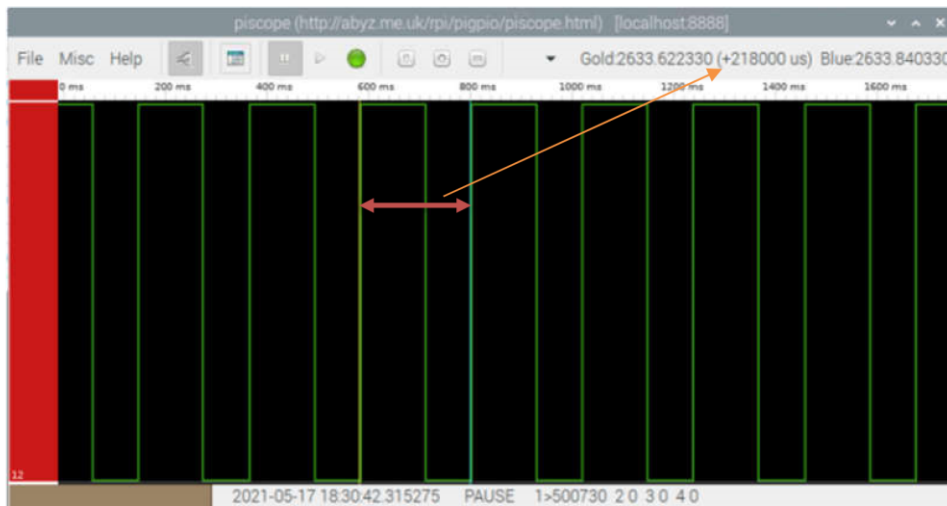
```
pi@raspberrypi ~$ sudo pigpiod
```

Lancez piscope en arrière plan :

```
pi@raspberrypi ~$ piscope &
```

Pour la configuration précédente on doit avoir un signal de période :

$$4095 * 1024 / 19.2 \text{MHz} = 218 \text{ms}$$



### **gpio wfi <pin> <edge>**

wfi : wait for interruption ; cette commande se bloque jusqu'à la détection sur <pin> d'un front de même type que celui indiqué par le paramètre edge, soit : falling, rising ou both.

```
pi@raspberrypi ~$ gpio mode 6 in      #configure GPIO 8 en entrée
pi@raspberrypi ~$ gpio mode 6 up      #configure Pullup
pi@raspberrypi ~$ gpio wfi 6 falling  #attendre un front descend sur GPIO 6
```

pour sortir de cette état de blocage tapez Ctrl+C.

Configurez GPIO 21 en sortie

```
pi@raspberrypi ~$ gpio mode 21 out    #configure GPIO 21 en sortie
pi@raspberrypi ~$ gpio write 21 0     # GPIO 9 à zéro
```

Reliez GPIO 21 et GPIO 6 à une plaque d'essai par des fils, mettre ces deux fils en contacts. Puis lancez la commande

```
pi@raspberrypi ~$ gpio wfi 6 falling ; echo "front descendant"
```

Ouvrez un **autre terminal virtuel** et lancez la commande suivante :

```
pi@raspberrypi ~$ gpio write 21 1 #mise à 1 de GPIO21 et donc de GPIO6
pi@raspberrypi ~$ gpio write 21 0 #mise à 0 de GPIO21 et donc de GPIO6
```

Cette dernière commande provoque un front descendant sur GPIO 21 et donc sur GPIO 6 ce qui débloque la commande précédente et permet l'affichage du message "front descendant".

On peut se passer des fils de liaisons entre GPIOs 6 et 21 ; le front descendant sur GPIO 6 peut se faire à l'aide de la configuration de la résistance de PULLUP/PULLDOWN. En effet, la suite de commande suivante permet de créer un front descendant sur GPIO 6:

```
pi@raspberrypi ~$ gpio mode 6 up ; gpio mode 6 down
```

et provoque le même effet que si on avait utilisé GPIO 21 pour commander GPIO 6 par une liaison physique.



2.1.7. Les commandes gpio peuvent superviser ou contrôler les GPIO de manière simple quand l'application n'a pas de contraintes temporelles sévères ; on peut les utiliser dans des scripts.

Exemple 1 : feu tri-couleurs (Rouge/orange/Vert) une voie

Editez un nouveau fichier feu3\_v1.sh à l'aide de nano :

```
pi@raspberrypi ~$ nano feu3_v1.sh
```

Saisissez le script suivant et enregistrez par Ctrl+X , O , Enter (tout ce qui suit un # est un commentaire et est optionnel sauf la première ligne).

```
#!/bin/bash
#cette premier ligne indique quel shell sera appelé pour exécuter ce script
#/bin/sh pointe sur dash et non pas bash
#certaines différences peuvent apparaitre dans la syntaxe des scripts
#shell selon le shell utilisé !!!!

#####feu tricolors 1 voie#####
# Tapez Ctr+C pour arrêter le programme
#Utilisation de variables
#Rouge --> GPIO. 0 #Orange--> GPIO. 1 #Vert ---> GPIO. 2
Rouge=0 ; Orange=1 ; Vert=2    #variables shell
#les délais en secondes
delai_rouge=30
delai_orange=5
delai_vert=30

#configuration en sortie des 3 GPIO et init à 0
#for i in \x27 seq 0 2 \x27    #\x27 pour simple quote `
for i in $Rouge, $Orange, $Vert  #$V contenu de la var V
do
    gpio mode $i out
    gpio write $i 0
done
#fonctionnement normal
while true;
do
    gpio write $Rouge 1; gpio write $Orange 0  #Rouge
    sleep $delai_rouge
    gpio write $Rouge 0 ; gpio write $Vert 1 #Vert
    sleep $delai_vert
    gpio write $Orange 1; gpio write $Vert 0  #Orange
    sleep $delai_orange
done
```

Rendez le script exécutable :

```
pi@raspberrypi ~$ chmod +x feu3_v1.sh
```



2.1.8. Pour visualiser le fonctionnement du feu tricolours on utilisera un autre script. Editez un nouveau fichier visu.sh à l'aide de nano :

```
pi@raspberrypi ~$ nano visu.sh
```

Saisissez le script suivant et enregistrez par Ctrl+X , O , Enter

```
#!/bin/bash
#visualisation des états des feux
while true;
do
    date "+%H:%M:%S"      #afficher l'heure
    gpio read 0 ; gpio read 1 ; gpio read 2;
    sleep 1;              #raffaichir l'affich après 1s
    clear;                #effacer l'écran
done
```

Rendez le script exécutable :

```
pi@raspberrypi ~$ chmod +x visu.sh
```

2.1.9. Lancez le script feu3\_v1.sh :

```
pi@raspberrypi ~$ ./feu3_v1.sh
```

2.1.10. Alors que le script précédent est en train de tourner **ouvrez un autre terminal virtuel**, et lancez le script de visualisation d'état des feux ;

```
pi@raspberrypi ~$ ./visu.sh
```

2.1.11. vérifiez le bon fonctionnement alors du script feu tri-couleurs.

2.1.12. L'outil **gpio** utilisé jusqu'à maintenant est un programme qui peut s'exécuter par n'importe quel utilisateur avec les droits du super-utilisateur root, son bit setuid est positionné à 1 comme on peut le voir à l'aide de cette commande :

```
pi@raspberrypi ~$ ls -l /usr/local/bin
total 36
-rwsr-xr-x 1 root root 33336 mai  6 11:29 gpio
```

|\_\_> rws (r lecture/ w écriture/s execution pour les autres utilisateurs avec les mêmes droits que root).

**Exercice d'application : (Travail à faire en trinôme et à rendre).**

**Modifier les scripts précédents pour permettre de contrôler et de visualiser un feu de signalisation tricolours à deux voies perpendiculaires ; le vert d'une voies doit être allumé après un temps de 30sec après que le rouge de l'autre est allumé.**

**2.2. Commande des GPIO à l'aide de l'interface sysfs (système):**

**2.3.** Pour pouvoir utiliser une GPIO, dans les processus utilisateurs sans besoin d'utiliser l'utilitaire **gpio**, et sans avoir les droits de root, il faut exporter les GPIO à utiliser par tous les autres utilisateurs (s'il y en a).

### 2.3.1. Commencez par analyser le contenu du dossier /sys/class/gpio

```
pi@raspberrypi ~$ cd /sys/class/gpio/
pi@raspberrypi /sys/class/gpio/$ ls -l #voir contenu avec détails droits
total 0
-rwxrwx--- 1 root gpio 4096 mai 14 12:07 export
lrwxrwxrwx 1 root gpio 0 mai 14 11:46 gpiochip0 -> ../../devices/platform/soc/3f200000.gpio/gpio/gpiochip0
-rwxrwx--- 1 root gpio 4096 mai 14 12:17 unexport
```

Notez bien que seuls root et les membres du groupe gpio ont le droit de modification des fichiers export et unexport ; l'utilisateur pi appartient au groupe gpio donc il peut aussi modifier ces deux fichiers ; tout autre utilisateur n'appartenant pas à ce groupe ne pourra pas modifier ces deux fichiers. En effet affichez l'identité de l'utilisateur pi pour vérifier les groupes auxquels il appartient:

```
pi@raspberrypi /sys/class/gpio/$ id pi #identité de user pi
uid=1000(pi)gid=1000(pi) groupes=1000(pi), 4(adm), 20(dialout), 24(cdrom),
27(sudo), 29(audio), 44(video), 46(plugdev), 60(games), 100(users), 101(input), 108(netdev), 999(spi), 998(i2c), 997(gpio)
```

### 2.3.2. Exportez GPIO 26 (Numérotation BCM soit broche physique numéro 37) vers le domaine utilisateur; ceci se fait par une écriture du numéro 26 vers le fichier "virtuel" /sys/class/gpio/export

```
pi@raspberrypi /sys/class/gpio/$ echo 26 > export
```

Rappel: le symbole > redirige la sortie de la commande echo 26 vers le fichier /sys/class/gpio/export ; n'oubliez pas de mettre des espaces avant et après ce symbole. N'importe quel utilisateur peut maintenant utiliser BCM 26 comme entrée ou sortie numérique.

### 2.3.3. Affichez à nouveau le contenu du répertoire /sys/class/gpio et notez la création d'un nouveau dossier **gpio26** (à vrai dir un lien vers un répertoire)

```
pi@raspberrypi /sys/class/gpio/$ ls -l
total 0
0 -rwxrwx--- 1 root gpio 4096 mai 14 13:11 export
0 lrwxrwxrwx 1 root gpio 0 mai 14 13:11 gpio26 -> ../../devices/platform/soc/3f200000.gpio/gpiochip0/gpio/gpio26
0 lrwxrwxrwx 1 root gpio 0 mai 14 11:46 gpiochip0 -> ../../devices/platform/soc/3f200000.gpio/gpio/gpiochip0
0 -rwxrwx--- 1 root gpio 4096 mai 14 12:17 unexport
```

### 2.3.4. Explorez le contenu de ce répertoire qui a été créé automatiquement, et notez bien la présence de deux fichiers **direction** et **value** ainsi que leurs droits d'accès:

```
pi@raspberrypi /sys/class/gpio/$ ls -l gpio26/
total 0
-rwxrwx--- 1 root gpio 4096 mai 14 13:11 active_low
lrwxrwxrwx 1 root gpio 0 mai 14 13:11 device -> ../../../../gpiochip0
-rwxrwx--- 1 root gpio 4096 mai 14 13:11 direction
-rwxrwx--- 1 root gpio 4096 mai 14 13:11 edge
```

```
drwxrwx--- 2 root gpio 0 mai 14 13:11 power
lrwxrwxrwx 1 root gpio 0 mai 14 13:11 subsystem ->
../../../../../../../../class/gpio
-rwxrwx--- 1 root gpio 4096 mai 14 13:11 uevent
-rwxrwx--- 1 root gpio 4096 mai 14 13:11 value
```

Pour donner aux autres utilisateurs la possibilité de commander GPIO 26 (BCM), il faut modifier les droits du fichier value ; comme suit :

```
pi@raspberrypi /sys/class/gpio/$ sudo chmod o+rw gpio26/value
pi@raspberrypi /sys/class/gpio/$ ls -l gpio26/value
-rwxrwxrwx- 1 root gpio 4096 mai 14 17:13 gpio26/value
```

2.3.5. Configurez la broche gpio 26 (BCM) en sortie et y la mettre à 1 ; puis testez sa valeur pour vérifier ; ensuite mettez-la à 0 et lisez sa valeur pour vérifier :

```
pi@raspberrypi /sys/class/gpio/$ echo out > gpio26/direction
pi@raspberrypi /sys/class/gpio/$ echo 1 > gpio26/value
pi@raspberrypi /sys/class/gpio/$ cat gpio26/value
1
#valeur lue
pi@raspberrypi /sys/class/gpio/$ echo 0 > gpio26/value
pi@raspberrypi /sys/class/gpio/$ cat gpio26/value
0
#valeur lue
```

La mise à 1 ou à 0 d'une sortie se ramène donc à une écriture dans un fichier. Le test d'une valeur d'une entrée se ramène à une simple lecture d'un fichier. Ce qui rejoint le principe fondamental de Linux/unix tout est un fichier !

2.3.6. Configurez la broche gpio 26 (BCM) en entrée et lisez sa valeur actuelle 1 :

```
pi@raspberrypi /sys/class/gpio/$ echo in > gpio26/direction
pi@raspberrypi /sys/class/gpio/$ cat gpio26/value
0
#valeur lue
pi@raspberrypi /sys/class/gpio/$ gpio readall #verifi avec l'outil gpio
```

**2.3.7. Les valeurs valides pour le fichier de direction sont :**

**out : sortie à 0 ; low : sortie à l'état bas, high : sortie à l'état haut ; in : entrée ;**

**le fichier active\_low à lecture/écriture permet d'inverser la logique de la gpio correspondante : si on y écrit 0 (valeur par défaut) écrire un 1 dans le fichier value implique un état haut sur la broche, et écrire la valeur 0 dans le fichier value implique un état bas sur la broche ; écrit un 1 dans active\_low inverse cette logique.**

2.3.8. Pour que les utilisateurs ne puissent plus utiliser cette broche 26 on doit arrêter son export, en écrivant la valeur 26 au fichier **/sys/class/gpio/unexport**, ceci aura comme conséquence de supprimer le dossier (à vrai dire le lien) **/sys/class/gpio/gpio26**

```
pi@raspberrypi /sys/class/gpio/$ echo 26 > unexport
pi@raspberrypi /sys/class/gpio/$ ls
export gpiochip0 unexport
```

### Un script shell exemple utilisant l'interface système sysfs:

Editer le script suivant à l'aide de nano, puis le rendre exécutable puis le lancez.

```
#!/bin/bash
#script gpio_sysf Gpio_Num
#permet de faire bascule une GPIO avec une période de 20s
if [ -z $1 ] #si le premier argument du script est vide (oublié!)
then
    echo -e " syntaxe:\ngpio_sysfs Gpio_Num" ; exit 1
fi
GPIO="$1" #premier arg donné au script=Num gpio
SYS=/sys/class/gpio #chemin de base
DEV=/sys/class/gpio/gpio$GPIO #chemin au rep de la gpio

if [ ! -d $DEV ] ; then #si directory n'existe pas
# il faut exporter d'abord la gpio
echo $GPIO >$SYS/export
fi

# la configurer comme sortie
echo out >$DEV/direction
#definition d'une fonction put
function put() {
# met à 1 ou à 0 la gpio
echo $1 >$DEV/value # $1 premier param de la fonction
}
# boucle principale:
while true ; do
put 1 #mettre la gpio à 1
echo "GPIO $GPIO: on $(date)" # affiche date complète
sleep 10 #attendre 10s
put 0 #mettre la gpio à 0
echo "GPIO $GPIO: off $(date +%H:%M:%S)" #affiche date formatée
sleep 10 #attendre 10s
done
```

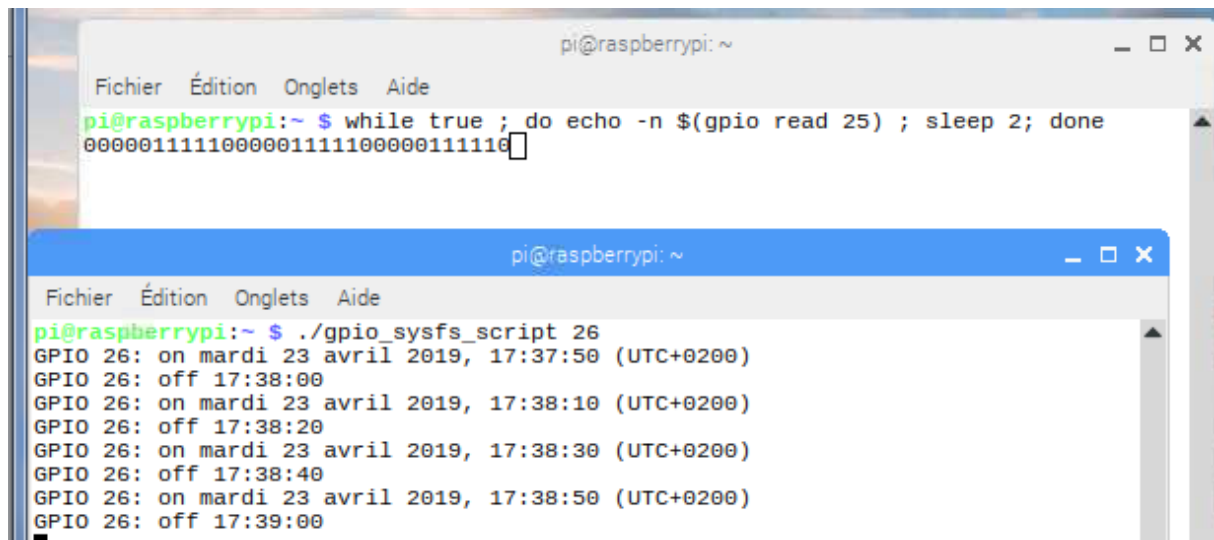
Ouvrez un autre terminal virtuel et lancer la boucle de lecture de la GPIO, en remplaçant **GPIO\_Num** avec le le numéro de la Gpio que vous avez utilisé lors du lancement de script .

**Attention à la différence de numérotation entre l'interface sysfs (BCM) et la commande gpio (wiringpi)**

```
pi@raspberrypi ~$ while true ; do gpio read GPIO_Num ; done
```

ou bien si vous voulez avoir les 0 et 1 dans la même ligne

```
pi@raspberrypi ~$ while true ; do echo -n $(gpio read GPIO_Num ) ; done
```



```
pi@raspberrypi: ~  
Fichier Édition Onglets Aide  
pi@raspberrypi:~ $ while true ; do echo -n $(gpio read 25) ; sleep 2; done  
0000011111000001111100000111110  
  
pi@raspberrypi: ~  
Fichier Édition Onglets Aide  
pi@raspberrypi:~ $ ./gpio_sysfs_script 26  
GPIO 26: on mardi 23 avril 2019, 17:37:50 (UTC+0200)  
GPIO 26: off 17:38:00  
GPIO 26: on mardi 23 avril 2019, 17:38:10 (UTC+0200)  
GPIO 26: off 17:38:20  
GPIO 26: on mardi 23 avril 2019, 17:38:30 (UTC+0200)  
GPIO 26: off 17:38:40  
GPIO 26: on mardi 23 avril 2019, 17:38:50 (UTC+0200)  
GPIO 26: off 17:39:00
```

## 2.4. Programmation différées des I/O sous Raspbian à l'aide de la commande at

La domotique implique, des fois, de faire des tâches de manière automatique, dans des heures et/ou jours, bien précis. On peut alors, utiliser les commandes gpio combinées avec les commandes shell **at** ou l'outil de planification des tâches dans le temps **cron**.

### 2.4.1. Vérifiez d'abord la date de votre Rspi2.

```
pi@raspberrypi ~$ date
```

### 2.4.2. Pour régler la date, utilisez la commande suivante :

```
pi@raspberrypi ~$ sudo date MMDDhhmmYYYY
```

Où : MM : mois ;DD : jour ;hh : heure ;mm : minutes ;YYYY : année.

2.4.3. La commande **at** permet d'exécuter des commandes à une date/heure précise, installez la commande **at** d'abord:

```
pi@raspberrypi ~$ sudo apt-get install at
```

### 2.4.4. Testez les commandes suivantes:

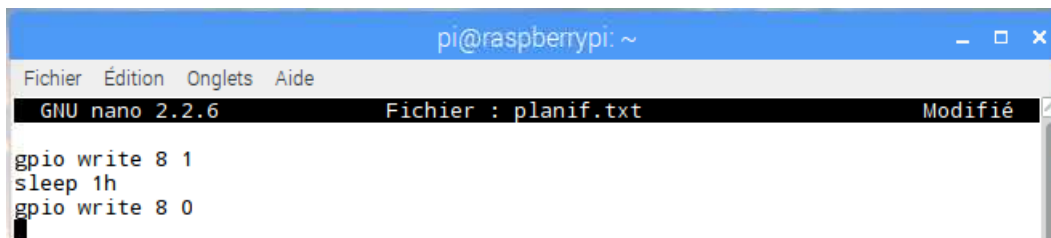
```
pi@raspberrypi ~$ gpio mode 8 out          #config GPIO 8 en sortie  
pi@raspberrypi:~$ date                      #vérifier date  
jeudi 11 mai 2017, 09:26:12 (UTC+0100)  
pi@raspberrypi:~$ at 9:30                   #planifier jobs pour 9:30  
warning: commands will be executed using /bin/sh  
at> gpio write 8 1                          #Mettre à 1 GPIO 8  
at> sleep 2m                                #attendre 2 min  
at> gpio write 8 0                          #Remettre à 0 GPIO  
at> <EOT>                                   #Tapez Ctrl+D pour terminer la saisie  
job 11 at Thu May 11 09:30:00 2017
```

une fois validée la saisie de la commande at 9:30, vous êtes invités à saisir les commandes à exécuter à la date/heure précisée dans la commande ; cette saisie se termine quand vous tapez Ctrl+D. Le job reçoit alors un numéro (dans l'exemple précédent il a le numéro 11).

Les commandes peuvent être réunies dans un fichier ; Editez un fichier texte avec **nano**

```
pi@raspberrypi:~$ nano planif.txt
```

et y saisissez les commandes suivantes



```
pi@raspberrypi: ~
Fichier  Édition  Onglets  Aide
GNU nano 2.2.6      Fichier : planif.txt      Modifié
gpio write 8 1
sleep 1h
gpio write 8 0
```

Sauvegardez le fichier planif.txt (Ctrl+X , O , enter).

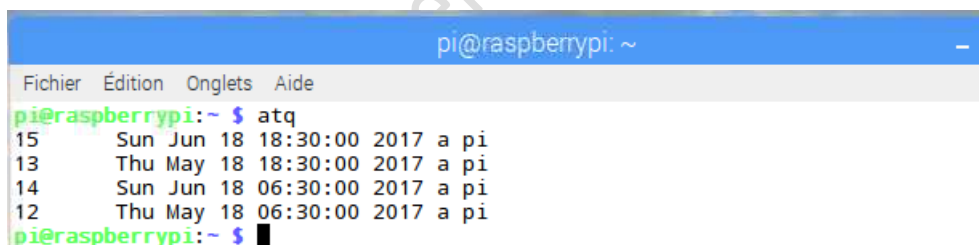
Exécutez les commandes suivantes qui planifient 4 jobs :

```
pi@raspberrypi:~$ at -f planif.txt 6:30 5/18/17
pi@raspberrypi:~$ at -f planif.txt 18:30 5/18/17
pi@raspberrypi:~$ at -f planif.txt 6:30 6/18/17
pi@raspberrypi:~$ at -f planif.txt 18:30 6/18/17
```

**Le 18 Mai et 18 Juin à 6h30 et 18h30** la sortie GPIO 8 passe à l'état 1 et le reste pendant 1h seulement.

NB : la date est format MM/JJ/AA **le mois avant le jour !**

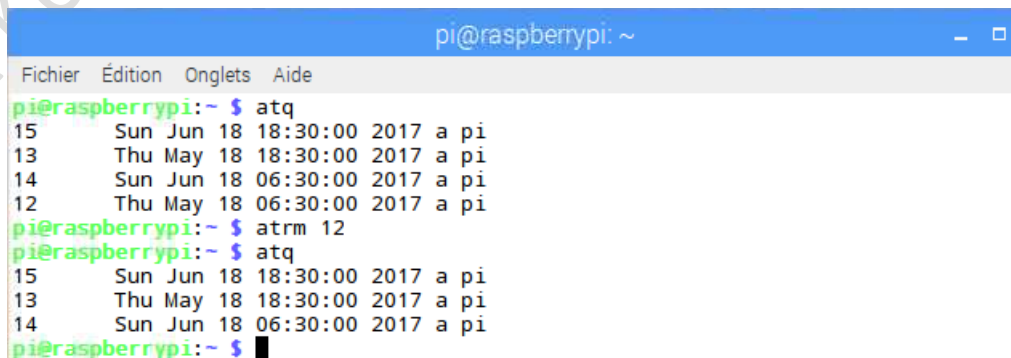
Pour voir le contenu de la file des jobs, utilisez la commande **atq**



```
pi@raspberrypi: ~
Fichier  Édition  Onglets  Aide
pi@raspberrypi:~ $ atq
15      Sun Jun 18 18:30:00 2017 a pi
13      Thu May 18 18:30:00 2017 a pi
14      Sun Jun 18 06:30:00 2017 a pi
12      Thu May 18 06:30:00 2017 a pi
pi@raspberrypi:~ $
```

Pour supprimer le job 12 par exemple tapez la commande :

```
pi@raspberrypi:~$ atrm 12      #supprime le job 12
pi@raspberrypi:~$ atq         #afficher les jobs planifiés pour vérif
```



```
pi@raspberrypi: ~
Fichier  Édition  Onglets  Aide
pi@raspberrypi:~ $ atq
15      Sun Jun 18 18:30:00 2017 a pi
13      Thu May 18 18:30:00 2017 a pi
14      Sun Jun 18 06:30:00 2017 a pi
pi@raspberrypi:~ $ atrm 12
pi@raspberrypi:~ $ atq
15      Sun Jun 18 18:30:00 2017 a pi
13      Thu May 18 18:30:00 2017 a pi
14      Sun Jun 18 06:30:00 2017 a pi
pi@raspberrypi:~ $
```

On peut utiliser une planification relativement à l'instant présent (**now**) en utilisant la forme suivante :

```
pi@raspberrypi:~$ at -f planif.txt now +Numb <unit>
```

où Numb est un entier et <unit> peut être soit minutes ; hours ; days; weeks; months ; years

Exemples :

```
pi@raspberrypi:~$ at -f planif.txt now +5
```

```
#cmdes ds planif.txt seront exécutées après 5 min
```

```
pi@raspberrypi:~$ at now +2 d
```

```
warning: commands will be executed using /bin/sh
```

```
at> gpio write 8 1 #Mettre à 1 GPIO 8
```

```
at> <EOT> #Tapez Ctrl+D pour terminer la saisie
```

```
#la mise à 1 de GPIO 8 sera faites d'ici 2 jours
```

La commande sleep, possède la syntaxe suivante : sleep NUMBER[SUFFIX] ; où [SUFFIX] peut être s pour secondes (par défaut), m pour minutes, h pour heures, d pour jours.

Plus d'infos, consultez le manuel : **man at** et **man sleep**

## 2.5. Programmation des tâches périodiques des I/O sous Raspbian à l'aide du service cron

La commande **at** n'est pas très adaptée à la planification des tâches répétitives dans le temps ; on utilise pour ce type de tâche le service **cron**.

2.5.1. La planification des tâches répétitives se fait en modifiant un fichier crontab, via la commande crontab -e, il faut choisir par la suite l'éditeur de texte à utiliser pour cette modification (choisissez nano en validant par Enter).

```
pi@raspberrypi:~$ crontab -e
no crontab for pi - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/ed
 2. /bin/nano <---- easiest
 3. /usr/bin/vim.tiny

Choose 1-3 [2]:
```

2.5.2. Les tâches à planifiées sont à ajouter dans ce fichier après la ligne commentée (#) :

### # m h dom mon dow command

Cette ligne vous rappelle la syntaxe à utiliser pour les lignes à ajouter dans ce fichier :

m : minutes (0 - 59) ; h : heures (0 - 23) ; dom (day of month) : jour du mois (1 - 31) ; mon (month) : mois (1 - 12) ; dow (day of week) : jour de la semaine (0-6) 0 pour dimanche ; command : c'est la commande à exécuter.

Chacun des 5 premiers champs peut contenir :

- un nombre valide par exemple 6 : la commande est exécutée quand ce champ prend la valeur 6 ;
- \* : la commande est exécutée quel que soit la valeur du champ
- suite de valeurs séparées par des virgules, exemple 3,5,10 : la commande est exécutée quand le champ prend les valeurs 3, 5 ou 10. Ne pas mettre d'espace après la virgule ;
- une plage de valeurs comme 3-7 : commande exécutée pour les valeurs 3 à 7 ;
- \*/3 : commande exécuté pour toutes les valeurs multiples de 3 (par exemple à 0 h, 3 h, 6 h, 9 h...)



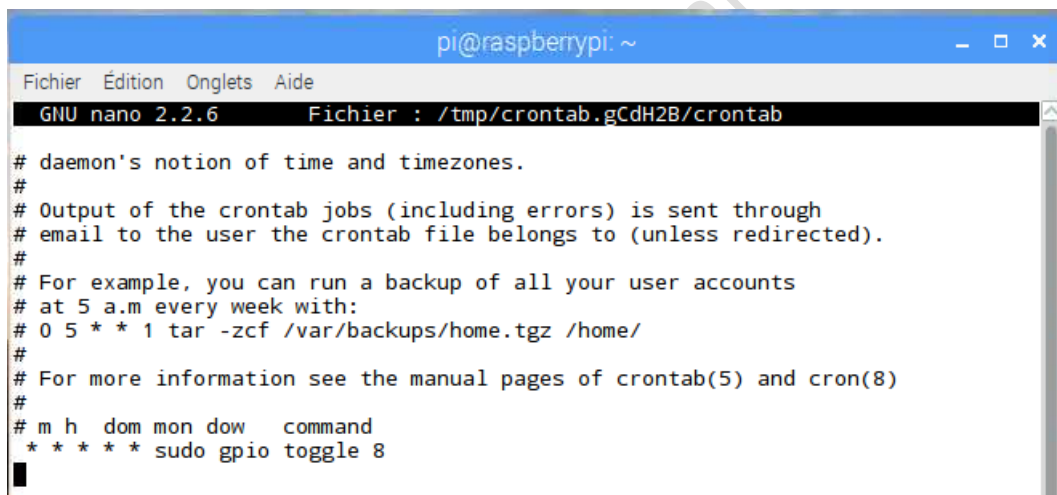
Quelques exemples à analyser :

**# m h dom mon dow command**

45 * * * * commande	Toutes les heures à 45 minutes exactement. Donc à 00 h 45, 01 h 45, 02 h 45, etc.
00 * * 0 commande	Tous les dimanches à minuit (dans la nuit de samedi à dimanche).
30 4 15 * * commande	Tous les 15 <sup>èmes</sup> jours du mois à 4 h30 du matin.
0 7 * 8 * commande	Tous les jours du mois d'Aout à 7h du matin.
0 * 4 12 * commande	Toutes les heures du 4 décembre.
* * * * * commande	Toutes les minutes
0 7 * * 1-5 commande	Tous les jours du Lun au vend à 7h (Un réveil par exple)

2.5.3. Ajoutez au fichier crontab.txt déjà ouvert avec nano la ligne suivante :

**\* \* \* \* \* sudo gpio toggle 8**



```
pi@raspberrypi: ~
Fichier  Édition  Onglets  Aide
GNU nano 2.2.6  Fichier : /tmp/crontab.gCdH2B/crontab

# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
* * * * * sudo gpio toggle 8
```

Cette commande va faire basculer GPIO 8 chaque minute et tous les jours !

2.5.4. Après sauvegarde du fichier (Ctrl+X, O, Enter), la commande **crontab** confirm l'installation de la table **crontab** et en informe le service **cron** des changements.

```
pi@raspberrypi:~$ crontab -e
no crontab for pi - using an empty one
crontab: installing new crontab
```

2.5.5. Vérifiez le résultat de cette tâche en utilisant la commande suivante

```
pi@raspberrypi:~$ while true; do gpio read 8 ; sleep 20; done
```

2.5.6. Analysez et Programmez la tâche suivante ; pour vérifier avancer l'heure du système.

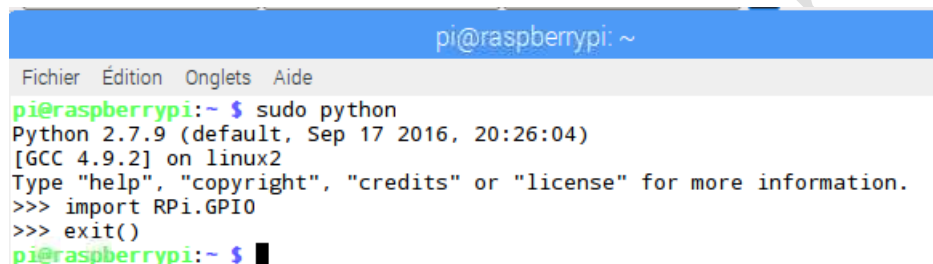
**0 20 \* \* \* sudo gpio mode 9 out ; sudo gpio write 9 1 ; sleep 2m ; sudo gpio write 9 0**

### 3. Programmation des GPIO de la Rspi2 en Python:

Python est un langage qui a vite fait sa place parmi les langages les plus utilisés. Il a l'avantage d'être simple à utiliser, et dispose d'une riche librairie sous forme de modules qui le rendent un peu passe partout ; **et surtout il a été adopté par les développeurs pour raspberryPi.**

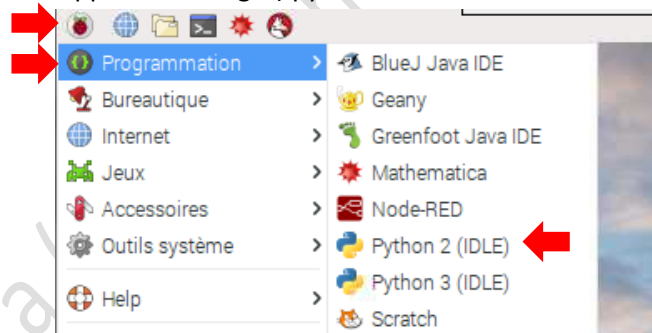
Python est un langage interprété et non pas compilé. Actuellement il y a deux versions majeures qui ne sont pas toujours compatibles **Python 3** et **python 2**. Les deux versions de python sont installées par défaut sur raspbian. Ici on fait juste une introduction à l'utilisation de python pour commander les GPIO.

3.1.1. Pour pouvoir utiliser les GPIO sous python il faut que le module RPi.GPIO soit installé. Vérifiez s'il est déjà installé : Ouvrez un terminal et lancez python en mode interactif ; puis tentez d'importer le module RPi.GPIO ; si aucune erreur n'est signalée alors il est bien installé ; quittez alors python par exit().



```
pi@raspberrypi: ~
Fichier  Édition  Onglets  Aide
pi@raspberrypi:~$ sudo python
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import RPi.GPIO
>>> exit()
pi@raspberrypi:~$
```

3.1.2. A partir du menu des applications de la Rspi2, rubrique programmation lancez l'IDE (Environnement de Développement Intégré) python 2.



3.1.3. A Partir du menu Fichier Créez un nouveau fichier et y saisissez le code suivant (les lignes commençant par # sont des commentaires peuvent ne pas être saisis) :

```

import RPi.GPIO as GPIO
#importer le module RPi.GPIO sous le nom GPIO; ceci permet de précéder les objet
#du module RPi.GPIO par GPIO au lieu de RPi.GPIO
import time #pour pouvoir utiliser sleep(x) = blocage pendant x secondes
GPIO.setmode(GPIO.BCM)
#la numérotation utilisée sera celle de BCM; autre possibilité GPIO.BOARD
GPIO.setup(26, GPIO.OUT)
#configurer GPIO 26 (BCM) soit 37 physique, en sortie
while True:
    # boucle infinie
    #indentation pour marquer le bloc
    GPIO.output(26, GPIO.HIGH)
    #Mettre à 1 GPIO 26
    time.sleep(2)
    #blocage pendant 2 s
    GPIO.output(26, GPIO.LOW)
    #Mettre GPIO 26 à 0
    time.sleep(2)
    #blocage pendant 2 s

```

3.1.4. A partir du menu **File/Save as...** Sauvegardez sous le nom **clignoter26** (pas la peine d'ajouter l'extension py, elle sera automatiquement ajoutée).

3.1.5. Exécutez ce programme à partir du menu **Run/Run Module** ou en utilisant le raccourcis clavier **F5**.

3.1.6. Une nouvelle fenêtre **Python Shell** s'ouvre ; si elle n'affiche aucun message d'erreur c'est que le programme est en exécution. Pour le vérifier, ouvrez un terminal et tapez-y la suite de commandes :

```

pi@raspberrypi: ~
Fichier  Édition  Onglets  Aide
pi@raspberrypi:~ $ while true; do gpio -g read 26; sleep 1; done
0
1
1
0
0
1
1
0
0
1
1
1
^C
pi@raspberrypi:~ $ while true; do gpio read 26; sleep 1; done
0
0
0
0
0
0
^C
pi@raspberrypi:~ $

```

NB l'utilisation de l'option **-g** car GPIO utilise la numérotation wPi si on ne spécifie pas **-g**

On voit bien que la sortie 26 (BCM) , soit 25 (wPi) clignote avec une période de 4 sec (2s On, 2s Off)

Erreur à ne pas commettre : **gpio read 26** donne l'état de 26 (wPi) soit 1 (BCM) qui est toujours à 0

3.1.7. Arrêtez le programme en fermant la fenêtre de Python shell qui a été ouverte (confirmez le kill du programme en exécution).

3.1.8. Modifiez le programme précédent pour que GPIO 26 clignote tant que GPIO 0 est à 1 et le programme se termine si GPIO 0 passe à 0, après avoir remis GPIO 26 et GPIO 0 à leur configuration par défaut ( IN et valeur=0).

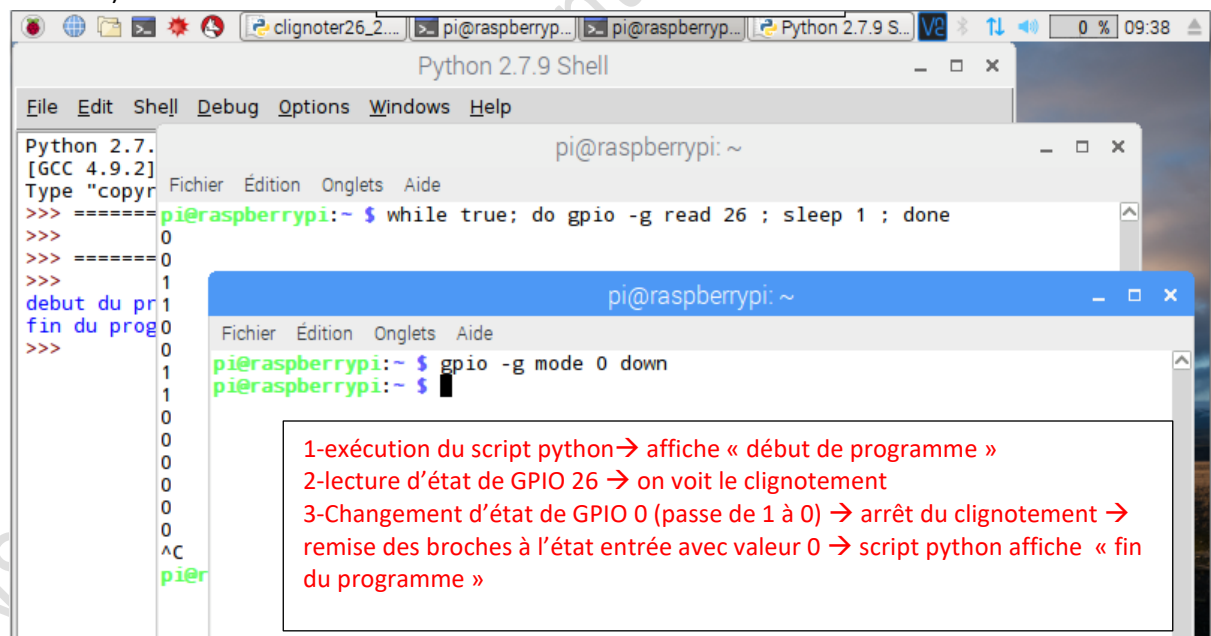
```

*clignoter26_2.py - /home/pi/clignoter26_2.py (2.7.9)*
File Edit Format Run Options Windows Help

import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(26, GPIO.OUT)
GPIO.setup(0, GPIO.IN)
#config GPIO 0 comme entrée
GPIO.setup(0, GPIO.IN, pull_up_down=GPIO.PUD_UP)
#pull_up_down= GPIO.PUD_UP ou
#config GPIO 0 avec une résistance relié à 3.3V --> valeur de GPIO 0 =1
print('debut du programme')
while GPIO.input(0): #tant que GPIO 0 est à 1 faire clignoter GPIO 26
    GPIO.output(26, GPIO.HIGH)
    time.sleep(2)
    GPIO.output(26, GPIO.LOW)
    time.sleep(2)
GPIO.cleanup()
#réinitialiser la config de toutes les GPIO à l'état de démarrage
#possibilité de réinitialiser une seul GPIO cleanup(Num)
print('fin du programme')

```

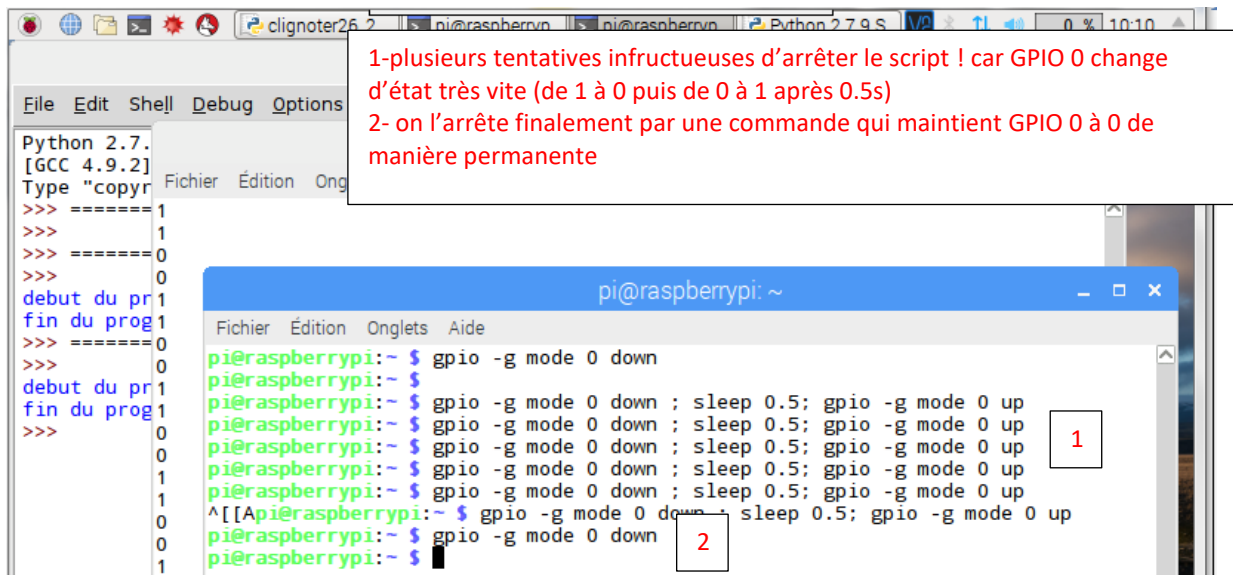
3.1.9. Sauvegardez le programme sous le nom **clignoter26\_2.py** et lancez son exécution (F5) ; pour vérifier le bon fonctionnement ouvrez deux terminaux : dans le premier préparer la série de commande qui permet de lire et visualiser l'état de GPIO 26 et dans l'autre préparer la commande qui permet de changer l'état de GPIO 0 en modifiant la configuration de la résistance de pullUp/Down pour faire passer l'entrée de 1 à 0. Lancez d'abord les commandes qui montrent le clignotement de GPIO 26 puis lancer la commande de changement d'état de GPIO 0 pour arrêter ce clignotement (et tout le programme en conséquence) : notez les messages dans la fenêtre Python shell. (Voir figure suivante).



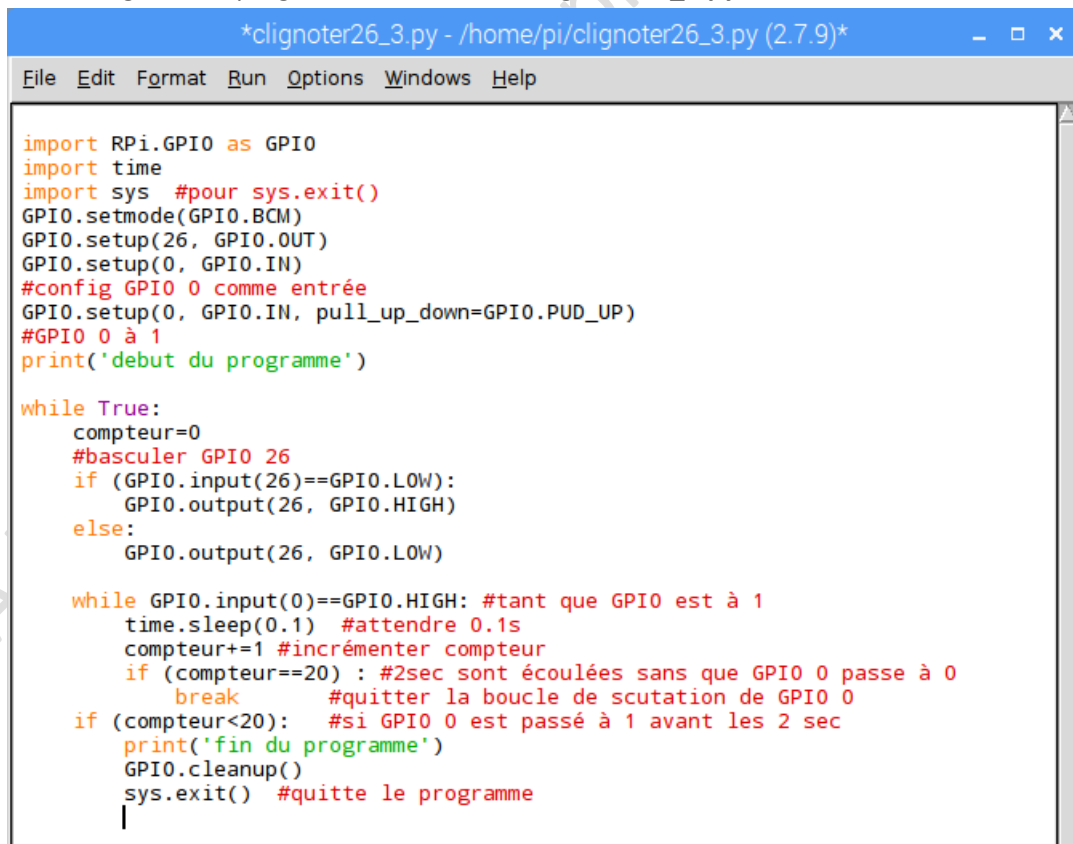
3.1.10. Dans le script précédent le test de l'état de GPIO 0 se fait une fois toutes les 4 secondes, donc au pire des cas le décalage entre l'ordre d'arrêter le clignotement est l'arrêt effectif peut atteindre 4 secondes ! Si GPIO 0 passe de 1 à 0 puis repasse à 1 dans un intervalle de temps de durée inférieur à 4

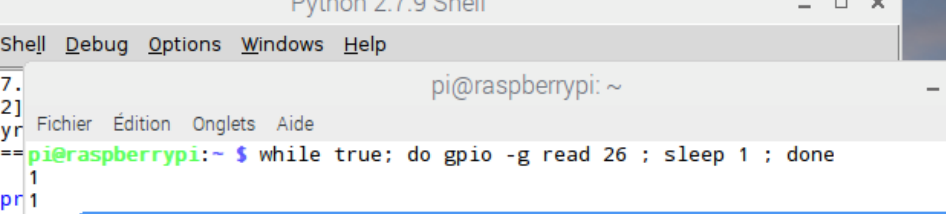
secondes, il y a de fortes chances que le clignotement ne s'arrête pas ! le programme rate alors la commande issue de GPIO 0. Essayez d'arrêter le programme avec la séquence de commande suivante :

```
pi@raspberrypi:~$ gpio -g mode 0 down ; sleep 0.5 ; gpio -g mode 0 up
```



3.1.11. Pour résoudre ce problème (en partie) modifiez le script précédent pour qu'il scrute (polling) l'état de GPIO 0 plus fréquemment à l'intérieur de la boucle et quitte cette boucle quand il détecte un état bas. Sauvegardez le programme sous le nom **clignoter26\_3.py**





The screenshot shows a terminal window titled "Python 2.7.9 Shell" with a menu bar (File, Edit, Shell, Debug, Options, Windows, Help). The terminal displays a Python script for controlling a motor using a sensor. The script is as follows:

```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help

Python 2.7.
[GCC 4.9.2]
Type "copyr
>>> =====
>>> debut du pr
fin du prog
>>>

pi@raspberrypi:~ $ while true; do gpio -g read 26 ; sleep 1 ; done

pi@raspberrypi:~ $ gpio -g mode 0 down ; sleep 0.5; gpio -g mode 0 up
pi@raspberrypi:~ $
```

A red box highlights the first line of the script, and a red number "1" is next to it.

1-Une seule tentative et le clignotement s'arrête et aussi le script

```
pi@raspberrypi:~$ while true; do gpio read 8 ; done
```

cette commande lit et affiche en permanence l'état de GPIO 8 ; on devrait voir GPIO 8 qui devrait changer d'état périodiquement.

4.1.5. Modifiez le programme précédent pour lui permettre d'afficher l'état de GPIO 8

```
int main (void)
{
    ...
    {    digitalWrite (PIN, HIGH) ; printf("1");fflush(stdout);
        delay (500) ;
        digitalWrite (PIN, LOW) ; printf("0");fflush(stdout);
    }
    ...
}
```

4.1.6. Recompilez le programme et observez le changement lors de l'exécution. (fflush(stdout) force le vidage du buffer de la sortie standard stdout pour afficher les 0 et 1 envoyés par les printf).

4.1.7. Saisissez le programme pwm1.c suivant :

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
int main (void)
{
    int rapp_cyc ; //rapport cyclique
    printf("PWM Test\n");
    wiringPiSetup(); //numérotation wPi
    pwmSetMode(PWM_MODE_MS); //Mode Mark space __|--|__|--|__|--|__
    pinMode (1, PWM_OUTPUT); //GPIO 1 en sortie PWM
    for (;;) { //fad In
        for (rapp_cyc = 0 ; rapp_cyc < 1024 ; rapp_cyc++) {
            pwmWrite (1, rapp_cyc) ; //config du rapport cyclique
            delay (1) ; //1ms
        }
        //fad Out
        for (rapp_cyc= 1023 ; rapp_cyc >= 0 ; rapp_cyc--) {
            pwmWrite (1, rapp_cyc) ;
            delay (1) ;
        }
    }
    return 0 ;
}
```

4.1.8. Compilez le programme et lancez-le en sudo :

```
pi@raspberrypi:~$ gcc -o pwm1 pwm1.c -lwiringPi
pi@raspberrypi:~$ sudo ./pwm1
```

4.1.9. Le programme précédent fait varier le rapport cyclique de 0 à 1023 puis de 1023 à 0, pour voir son effet ; dans un autre terminal tapez la série de commandes suivantes :

```
pi@raspberrypi:~$ while true; do gpio read 1 ; done
```



on observe alors une variation du nombre de 1 par rapport à celui de 0 (ce test n'est pas très précis mais ceci donne une idée sur ce qui se passe; un test avec un oscilloscope ou une led permet de voir la variation du rapport cyclique.

4.1.10. Saisissez le programme int0.c suivant :

```
#include <wiringPi.h>
#include <stdio.h>

void affiche(){ //fonction appelée si Interruption
printf("une interruption !! \n");
if(digitalRead(2)==1) digitalWrite(2,0); else digitalWrite(2,1);
}

int main(){
wiringPiSetup(); //numérotation wPi
pinMode(0,INPUT); //GPIO 0 entrée
pinMode(2,OUTPUT); GPIO 2 sortie pour indiquer interrup
wiringPiISR(0,INT_EDGE_FALLING,&affiche); //interrup sur front descendant
//Interrupt sur front descend sur GPIO 0
while(1); //pour maintenir le programme en vie
return 0;
}
```

4.1.11. Compilez le programme et lancez-le:

```
pi@raspberrypi:~$ gcc -o int0 int0.c -lwiringPi
pi@raspberrypi:~$ ./int0
```

4.1.12. Le programme attend un front descendant sur GPIO 0, et s'il se produit il exécute la fonction affiche, pour voir son effet ; dans un autre terminal tapez la série de commandes suivantes qui provoquent un front descendant, puis après une seconde lit la valeur de GPIO 2 :

```
pi@raspberrypi:~$ gpio mode 0 up ; gpio mode 0 down ; sleep 1; gpio read 2
```



```
pi@raspberrypi: ~
Fichier  Édition  Onglets  Aide
pi@raspberrypi:~ $ ./int0
une interruption !!
une interruption !!
une interruption !!
une interruption !!
□

pi@raspberrypi: ~
Fichier  Édition  Onglets  Aide
pi@raspberrypi:~ $ gpio mode 0 up; gpio mode 0 down ; sleep 1; gpio read 2
1
pi@raspberrypi:~ $ gpio mode 0 up; gpio mode 0 down ; sleep 1; gpio read 2
0
pi@raspberrypi:~ $ gpio mode 0 up; gpio mode 0 down ; sleep 1; gpio read 2
1
pi@raspberrypi:~ $ gpio mode 0 up; gpio mode 0 down ; sleep 1; gpio read 2
0
pi@raspberrypi:~ $ █
```

**NB:** la librairie wiringPI offer d'autres fonctions à consulter sur le lien : [wiringpi.com/Reference](http://wiringpi.com/Reference)