

Domaine de Connaissances : 3.1.4 Traitement de l'Information
3.2.4 Transmission de l'Information

Réseau de Terrain avec nano PC: Réseau LAN Client/Serveur

Connaissances
abordées :

☒ Raspberry pi
☒ GPIO

☒ LINUX Raspian
☒ Serveur Apache/MySQL ☒ SSH
☒ Réseau LAN

-B Python : RPIO.GPIO

RPIO.GPIO est une bibliothèque PYTHON qui simplifie l'accès au GPIO du Raspberry Pi . Nous écrivons des programmes en python qui seront exécutés avec les commandes PHP EXEC() et SYSTEM().

Cette bibliothèque ne peut être utilisée qu'en super utilisateur ou avec la commande « sudo ».

Nous allons paramétrer le Raspberry pi pour que le serveur apache2(www-data) puisse exécuter des commande système linux.

Ouvrir le fichier suivant : `pi@raspberrypi ~ $ sudo nano /etc/sudoers`

Puis rajouter la ligne sous root : `www-data ALL=(ALL) NOPASSWD: ALL`

```
# User privilege specification
root    ALL=(ALL:ALL) ALL
www-data ALL=(ALL) NOPASSWD: ALL
```

Sauvegardez le fichier avec « **Ctrl+o** » puis valider en tapant sur « ENTREE » et fermez l'éditeur avec « **Ctrl+x** ».

Nous allons créer deux fichiers Python (voir des tutoriels) comme exemple :

Je veux allumer une led sur la broche 11 du connecteur GPIO et vérifier son état.

Côté php, j'envoie le numéro de broche que je veux mettre à « 1 » et j'affiche que l'état est bien à « 1 »

```
<?php
unset($status);//je détruits la variable
$pint=11;//broche n°11
exec("sudo test ".$pint,$status);//Exécution du programme sudo test et j'envoie la variable $pint contenant 11
echo($status[0]);//Réception de l'état de la variable coté python print(var)
?>
```

Côté python programme test:

```
1  #!/usr/bin/env python
2  import sys"""librairie système"""
3  import RPi.GPIO as GPIO""" librairie GPIO"""
4  GPIO.setmode(GPIO.BOARD)"""choisir le connecteur gpio pour les broches GPIO.setmode(GPIO.BCM)"""
5  pintest = sys.argv[1] """ permet de récupérer la variable venant de php"""
6  B = int(pintest)"""convertir en entier"""
7  GPIO.setup(B,GPIO.OUT)"""mettre la broche 11 en sortie"""
8  GPIO.output(B,GPIO.HIGH)""" mettre à 1 la sortie 11 du connecteur"""
9  A = GPIO.input(B)"""lire la sortie 11 du connecteur"""
10 print (A)"""afficher le contenu et l'envoyer $status
```

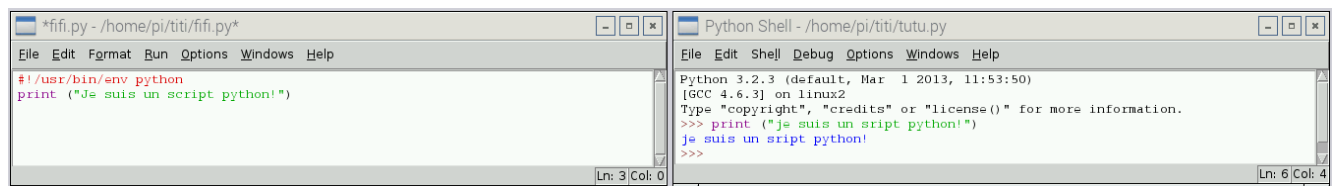
Nous allons le placer ou l'écrire sur notre raspberry ouvrir python 3



Ouvrir python Shell afin de tester vos lignes de commande, ouvrir aussi un fichier format *.py

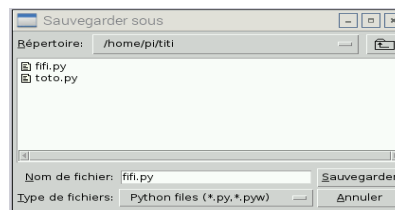
Sur ce fichier mettre la ligne suivante en entête pour qu'il puisse être interprété comme du python :

`#!/usr/bin/env python`



Vous tester vos lignes avec le Shell (fenêtre de droite) puis les copier dans fichier (fenêtre de gauche) si elles fonctionnent correctement.

Puis sauvegarder votre fichier :



Donner les droits à votre fichier : `pi@raspberrypi ~ $ sudo chmod 777 /home/pi/titi/fifi.py`

Droits 777 : correspond à `rw-rw-rw-` (soit tous les droits) :

```
pi@raspberrypi ~/titi $ ls -l
total 8
-rwxrwxrwx 1 pi pi 58 août 30 20:11 fifi.py
-rwxrwxrwx 1 pi pi 38 août 29 15:04 toto.py
```

Tester votre programme ainsi si vous êtes dans son répertoire : `pi@raspberrypi ~/titi $./fifi.py`
Je suis un script python!

Si non il faut taper son chemin absolue: `/home/pi/titi/fifi.py`

Pour exécuter n'importe où votre programme, on va le copier dans un répertoire particulier, pour connaître l'ensemble des répertoires où votre programme peut-être stocké et exécuté sans donner son chemin taper :

```
pi@raspberrypi ~/titi $ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/usr/games
```

Chaque chemin proposé est séparé par « : », on va choisir `/usr/local/bin`, on copie le fichier sur ce répertoire

`pi@raspberrypi ~/titi $ sudo cp fifi.py /usr/local/bin` on va supprimer son extension .py en effectuant la commande suivante, ainsi le fichier est exécutable sans extension:

`pi@raspberrypi ~/titi $ sudo mv /usr/local/bin/fifi.py /usr/local/bin/fifi`

Taper le nom du fichier sous n'importe quel dossier : `pi@raspberrypi ~/titi $ fifi`
Je suis un script python!

Vous pouvez effectuer les mêmes opérations avec le programme test mais en super utilisateur.

En ligne de commande : Tapez : **sudo python** pour pouvoir exécuter les instructions python avec les droits du super-utilisateur (ROOT):

Version de la carte :

GPIO.RPI_REVISION

Version de la librairie RPi.GPIO.:

GPIO.VERSION

Importation de la librairie :

Import RPi.GPIO as GPIO

Choix de la désignation des broches BCM (N° de broche du microcontrôleur) ou Connecteur GPIO :

GPIO.setmode(GPIO.BCM) ou **GPIO.setmode(GPIO.BOARD)**

Il est plus simple de choisir le connecteur, exemple pour le GPIO 0 le numéro broche du connecteur est 11

GPIO#	NAME	GPIO#	NAME
1	3.3 VDC Power	17	5.0 VDC Power
2	GPIO 0	18	5.0 VDC Power
3	GPIO 1	19	Ground
4	GPIO 2	20	GPIO 15 Tx/D (UART)
5	GPIO 3	21	GPIO 16 Rx/D (UART)
6	GPIO 4	22	GPIO 17 PCM_CLK/PWM0
7	GPIO 5	23	Ground
8	GPIO 6	24	GPIO 4
9	GPIO 7	25	GPIO 5
10	GPIO 8	26	Ground
11	GPIO 9	27	GPIO 6
12	GPIO 10	28	GPIO 10
13	GPIO 11	29	GPIO 11
14	GPIO 12	30	GPIO 12
15	GPIO 13	31	GPIO 13
16	GPIO 14	32	GPIO 14
17	GPIO 15	33	GPIO 15
18	GPIO 16	34	GPIO 16
19	GPIO 17	35	GPIO 17
20	GPIO 18	36	GPIO 18
21	GPIO 19	37	GPIO 19
22	GPIO 20	38	GPIO 20
23	GPIO 21		
24	GPIO 22		
25	GPIO 23		
26	GPIO 24		
27	GPIO 25		
28	GPIO 26		
29	GPIO 27		
30	GPIO 28		
31	GPIO 29		
32	GPIO 30		
33	GPIO 31		
34	GPIO 32		
35	GPIO 33		
36	GPIO 34		
37	GPIO 35		
38	GPIO 36		

Connecteur : 0 GPIO 0

Attention! The GPIO pin numbering used in this diagram is intended for use with WiringPi / Pi4J. This pin numbering is not the raw Broadcom GPIO pin numbers.

<http://www.pi4j.com>

Choix du type(sens) de la broche :

Sortie : **GPIO.setup(11, GPIO.OUT)** ou entrée : **GPIO.setup(11, GPIO.IN)**

Résistance de tirage sur une entrée (pull-up ou pull-down) :

GPIO.setup(11, GPIO.IN, pull_up_down=GPIO.PUD_UP) ou

GPIO.setup(11, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)



Ecriture de l'état sur une sortie :

Sortie à « 1 » : `GPIO.output(11, GPIO.HIGH)` ou Sortie à « 0 » : `GPIO.output(11, GPIO.LOW)`

Lecture d'une entrée :

`GPIO.input(11) -> Variable = GPIO.input(11)`

Réinitialisation en entrée de toutes les broches :

`GPIO.cleanup()`

Détection d'un changement d'état d'une entrée :

On choisit préalablement l'évènement front montant ou descendant que nous voulons détecter ou un changement d'état.

Montant : **`GPIO.add_event_detect(11, GPIO.RISING)`**

Descendant : **`GPIO.add_event_detect(11, GPIO.FALLING)`**

Etat : **`GPIO.add_event_detect(11, GPIO.BOTH)`**

On peut aussi arrêter ces évènements sur la broche sélectionnée

`GPIO.remove_event_detect(11)`

L'évènement est mémorisé puis utilisé lors d'un test (attention on peut perdre des évènements si d'autres évènements identique arrivent sur la broche avant d'avoir effectué le test)

If `GPIO.event_detect(11)` :

Bloque d'instructions exécutées si un évènement a été enregistré, on peut attendre l'évènement suivant car on vide la mémoire d'évènements en le testant.

Il est préférable d'utiliser la méthode suivante qui nous rapproche de tous nos langages javascript,vb2015 :

`GPIO.add_event_detect(n_broche, GPIO.FALLING, callback=test, bouncetime=100)`

Le déclenchement d'un évènement va appeler la fonction test, pour éviter les rebonds (appuie de bouton,...)et appeler plusieurs fois la fonction test on rajoute une temporisation **100 ms**. « bouncetime » est en ms.

```
def test(n_broche):  
    print('front descendant détecté!')  
    print('broche numéro : %s'%n_broche)
```

Nous voulons transmettre des informations venant du serveur apache à un script python qui va lui renvoyer une réponse, exemple piloter une sortie du GPIO et lire l'état d'une de ses entrées afin d'informer un client web.

Attention, l'exécution du script python doit se faire rapidement, aucune boucle infinie n'est tolérée car la page PHP attend la fin du script python pour exécuter la suite de ses propres instructions.

Remarque importante: pour des processus complexes, la méthode préconisée est la suivante :

1- Ecrire un programme python indépendant (exécuté par exemple au démarrage du raspberry pi) qui gère le processus comme sur un microcontrôleur de façon autonome. La différence fondamentale avec une gestion classique d'un processus est de contrôler (lire) aussi l'état des sorties du GPIO car elles peuvent être modifiées par d'autres programmes (voir 2) en parallèles.

2-Ecrire des petits scripts en python autre que le programme principale pour lire ou écrire sur les entrées/sorties du GPIO qui auront pour rôle d'informer un client WEB et de lui permettre d'agir sur une partie du processus. Attention de conserver le choix de la cartographie du programme principale, de ne pas modifier une sortie en entrée et vis et versa.

Recevoir dans un programme python le contenu d'une variable venant d'un script PHP et lui renvoyer une réponse :

Importation de la librairie système :

```
Import sys
```

Récupération variables systèmes envoyées par script PHP avec l'instruction EXEC:

Exemple :Je veux exécuter un programme python « *essai.py* » en super utilisateur qui est sous le dossier(/usr/local/bin) et lui transmettre deux variables venant de mon script PHP :REMARQUE le format des variables est transmis uniquement sous forme des chaînes de caractères.

```
<?php
$variable_1=11;//un nombre entier 11
$variable_2= "toto";// un texte toto
exec("sudo essai.py $variable_1 $variable_2",$status);
?> // j'indique à mon script php d'exécuter essay.py, pas besoin de préciser le chemin car il est placé
sous le dossier « /usr/local/bin » et je lui envoie le contenu de deux variable « 11 » et « toto ».
```

Mon programme python :

```
Import sys """ Importation de la librairie système """
pintest = sys.argv[1] """ permet de récupérer « 11 »venant de php"""
textest = sys.argv[2] """ permet de récupérer « toto »venant de php"""
nombre= int(pintest)"""reconvertir la chaîne en entier"""
```

Je récupère les variables dans l'ordre chronologique d'envoi du script PHP dans un tableau `sys.argv` qui contient la liste des arguments que vous avez passée en ligne de commande, au moment de lancer votre programme. Exemple `python test.py arg1 arg2 arg3` , j'envoie à mon programme `test.py` trois

arguments récupérables par mon script python. Remarque **sys.argv[0]** contient le nom du programme « test.py ».

Pour renvoyer le contenu d'une variable au script php le procédé est différent. Nous allons recevoir le résultat du programme dans un tableau déclaré préalablement en php ici \$status (dans ma page php).

Dans le programme python lorsque celui-ci s'exécute la première commande « print() » rencontrée, son contenu interprétable est stockée dans \$status[0], chaque donnée affichée avec « print() » en python est stockée dans ce tableau qui s'incrémente automatiquement au « print() » suivant, ici aussi l'ordre chronologique est primordial pour la lecture du tableau en PHP.

Python :

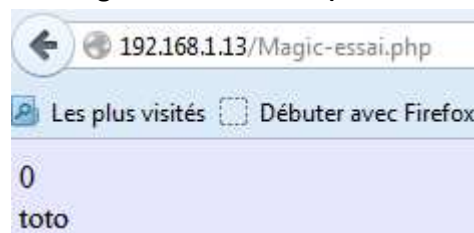
```
1  #!/usr/bin/env python
2  import sys""librairie système""
3  import RPi.GPIO as GPIO"" librairie GPIO""
4  GPIO.setmode(GPIO.BOARD)""choisir le connecteur GPIO""
5  pintest = sys.argv[1] "" permet de récupérer la variable venant de php""
6  B = int(pintest)""convertir cette variable en entier""
7  GPIO.setup(B,GPIO.OUT)""mettre la broche 11 en sortie""
8  GPIO.output(B,GPIO.LOW)"" mettre à 0 la sortie 11 du connecteur""
9  A = GPIO.input(B)""lire la sortie 11 du connecteur""
10 print (A)""afficher le contenu et l'envoyer dans $status[0]""
11 print ('toto')""afficher toto et l'envoyer dans $status[1]""
```

à chaque « print() » on remplit le tableau dans l'ordre chronologique

PHP :

```
<?php
unset($status);//je détruits la variable
$pint=11;//broche n°11
exec("sudo test ".$pint,$status);//Exécution du programme sudo test et j'envoie la variable $pint contenant "11"
echo($status[0]);//Réception de l'état de la variable coté python print(A) et l'afficher
echo '<br>';//mettre à la ligne
echo($status[1]);//Réception de l'état de la variable coté python print('toto') et l'afficher
?>
```

Affichage coté client web (l'état broche 11 à 0)



Les autres fonctions utiles pour le Raspberry pi

Temporisation :

```
import time"" Importation de la librairie temporisation ""
time.sleep(1)"" délais d'une seconde ""
time.sleep(.1)"" délais de 100ms ""
time.sleep(.01)"" délais de 10ms ""
```



Raspberry Pi



PWM: issue de la librairie RPi.GPIO, attention il faut lire attentivement le brochage sur le connecteur vous avez le droit uniquement aux broches : 12,32,33,35 (pwm0 et pwm1) pour le B+

```
GPIO.setmode(GPIO.BOARD) """" choix connecteur """"  
GPIO.setup(12,GPIO.OUT) """" broche 12 pwm0 en sortie """"  
M=GPIO.PWM(12,50) """" broche 12 pwm0 fréquence = 50 Hz """"  
M.start(50) """" rapport cyclique de départ 0% à 100 % ici 50 % """"  
M.stop() """" arrêt de l'horloge""""  
M.ChangeDutyCycle(25) """" Change le rapport cyclique de 0% à 100 % ici 25 % """"  
M.ChangeFrequency(0.7) """" Change la fréquence à 0,7 Hz """"
```