

TP4 Raspberry PI

Réalisation d'un serveur WEB

L'ordinateur Raspberry Pi, connecté en réseau, peut devenir un serveur WEB permettant le contrôle distant de processus ou la transmission à distance de données à travers Internet.

Ce TP montre comment créer un serveur WEB affichant deux boutons ON et OFF sur lesquels l'utilisateur peut agir.

Il existe plusieurs bibliothèque de site WEB embarqué pour Raspberry Pi, l'ordinateur a cependant des ressources limitées, il est préférable d'utiliser un serveur peut gourmand en ressources processeur et éviter les médias trop volumineux qui satureraient la mémoire flash. La bibliothèque retenue dans cet exemple est **Webpy**.

Pour plus d'information rendez-vous sur <http://webpy.org/>

I. Installation du serveur WEB webpy

1. Télécharger le module en tapant la commande :

```
wget http://webpy.org/static/web.py-0.37.tar.gz
```

2. Installer le serveur en tapant les commandes suivantes :

```
tar -xzf web.py-0.37.tar.gz (décompression)
cd web.py-0.37
python setup.py install
```

II. Création des dossiers du serveur WEB

Taper les commandes suivantes :

```
cd / (racine)
sudo mkdir webpyserver (le dossier du serveur)
cd /webpyserver
sudo mkdir templates ( contiendra les pages html dont index.html)
sudo mkdir static (feuille de style CSS éventuelle)
Remarque : le port du serveur Web sera 8080
```

III. Réalisation du site WEB

1. Dans le dossier webpyserver, créer le fichier gpio4.py
Ce programme récupère les données des GPIO et gère l'interface homme-machine de la page Web.

```
#!/usr/bin/env python
import web
import RPi.GPIO as GPIO
from web import form
# definit GPIO4 en sortie
GPIO.setmode(GPIO.BCM)
GPIO.setup(4, GPIO.OUT)

# definit la page de nom index pour le site web
urls = ('/', 'index')
dossier_web = web.template.render('templates')
app = web.application(urls, globals())
# definit les boutons a afficher
ma_forme = form.Form(
    form.Button("btn", id = "btnon", value = "on", html = "On", class_ = "bouton_on"),
    form.Button("btn", id = "btnoff", value = "off", html = "Off", class_ = "bouton_off")
)
# definit l'action a effectuer quand la page index est appelée
class index:
```

```
# utilise quand la page est demandee
def GET(self):
    forme = ma_forme()
    return dossier_web.index(forme, "Raspberry Pi control GPIO4")

# utilise quand une forme web est soumise
def POST(self):
    userdata = web.input()
    if userdata.btn == "on":
        GPIO.output(4,True) # Allume la LED
    if userdata.btn == "off":
        GPIO.output(4,False) # Eteind la LED
# recharge la page web
raise web.seeother('/')
# programme
if __name__ == '__main__':
    app.run()
```

2. Dans le sous dossier **templates** créer le fichier **index.html**

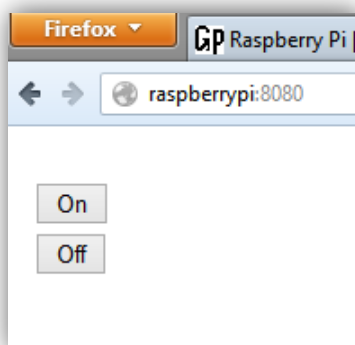
C'est la page html d'accueil du site Web, elle est ici simplifiée au maximum. Il est possible de l'enrichir avec un éditeur html comme NVU ou KOMPOSER (www.nvu.com)

```
$def with (form, title)
<!doctype html>
<html>
    <head>
        <title>$title</title>
    </head>

    <body>
        <br />
        <form class="form" method="post">
            $:form.render()
        </form>
    </body>
</html>
```

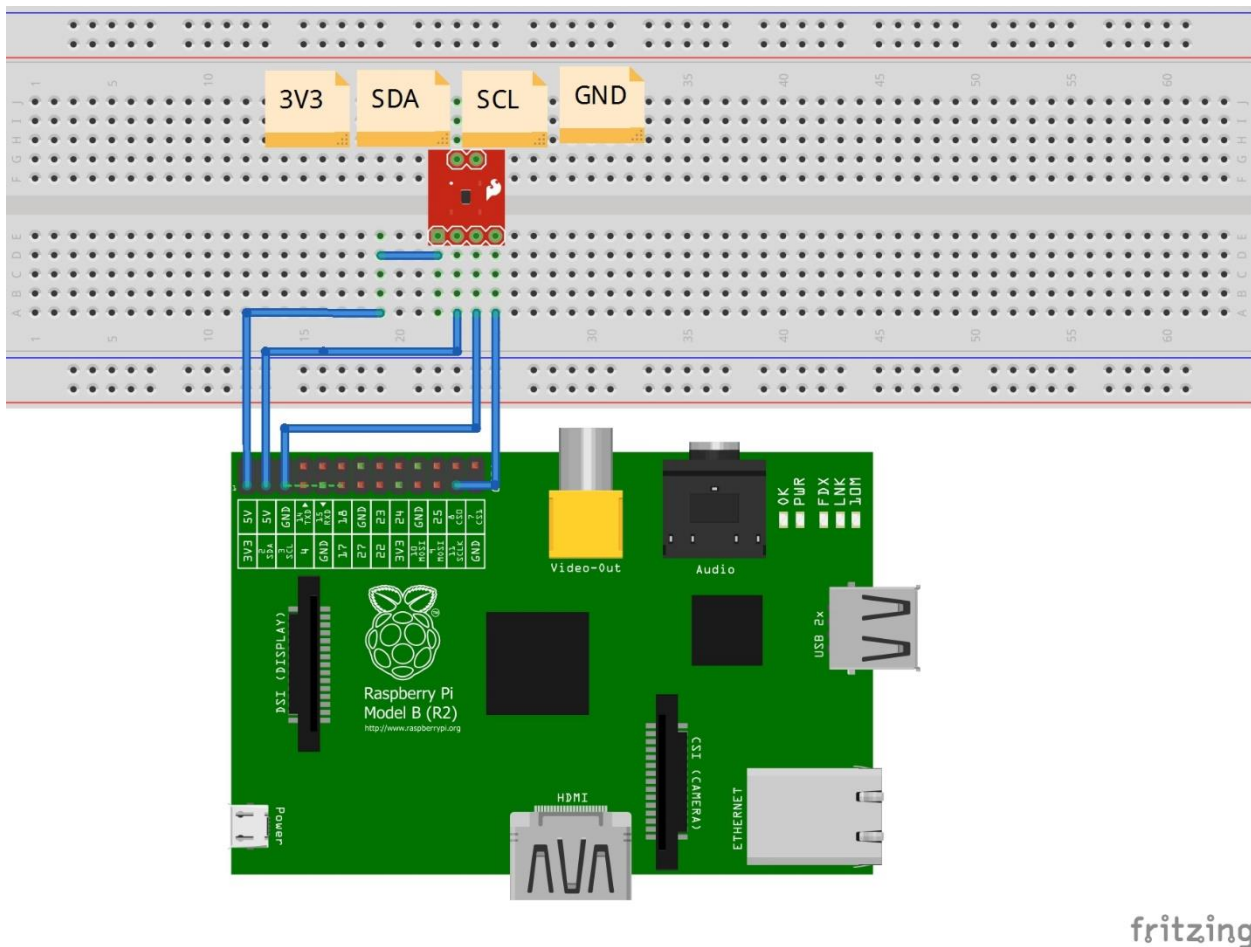
3. Lancer le programme gpio4.py et accéder au serveur de la Raspberry Pi depuis un PC en réseau à l'aide d'un navigateur à l'adresse : XXX.XXX.XXX.XXX :8080

Résultat attendu :



IV. Thermomètre WEB :

1. Réaliser le montage suivant



2. Dans webpyserver, créer le fichier **webTempGpio4.py** suivant :

```
#!/usr/bin/env python

import RPi.GPIO as GPIO # gestion E/S
import web, smbus # charge le serveur WEB et le gestionnaire I2C from web import form # form
est utilise

GPIO.setmode(GPIO.BCM) #numeration GPIO BCM2835, 4 pour GPIO4
GPIO.setup(4, GPIO.OUT) # definit GPIO4 en sortie

bus_pi = smbus.SMBus(1) # declare le busI2C SMBUS(1)

# adresse I2C du TMP102
addrTMP102 = 0x48

#fonction de mesure de la temperature avec un TMP102 (I2C)
def mesTemp():
    x = bus_pi.read_word_data(addrTMP102,0)
    #la lecture est pf puis PF , il faut les permuter
    msb=x>>8
    lsb=x&0x00FF
    wtemp=((lsb<<8)|msb)>>4
    temp=wtemp*0.0625
    round(temp,2) # arrondi au centieme
    # ecrit un echo du serveur sur la console
    print 'temperature : ',temp , 'degres C\n\r'
    return temp

# indique l emplacement des fichiers html
```

```
render = web.template.render('templates/')

# definit la page de nom index pour le site web
urls = ('/', 'index')

# definit les boutons a afficher
ma_forme = form.Form(
    form.Button("btn", id = "btnon", value = "on", html = "Allumee", class_ = "bouton_on"),
    form.Button("btn", id = "btnoff", value = "off", html = "Eteinte", class_ = "bouton_off")
)

# definit l action a effectuer quand la page index est appelee
class index:
    def GET(self):
        # utilise quand la page est demandee
        forme = ma_forme()
        t=mесTemp()
        return render.index(forme,t) #renvoie le retour de la page html

# utilise quand une forme web est soumise
def POST(self):
    userdata = web.input()
    if userdata.btn == 'on':
        GPIO.output(4,True)
        print 'LED GPIO4 allumee'
    if userdata.btn == 'off':
        GPIO.output(4,False)
        print 'LED GPIO4 eteinte'
    # recharge la page web
    raise web.seeother('/')

# le programme
if __name__ == '__main__':
    app = web.application(urls, globals())
    app.run()
```

3. Créer le fichier **templates/index.html**

```
$def with (form,temp)

<em>La temperature</em>

<!-- temp represente la temperature -->
$if temp:
est de <b> <FONT size=30> $temp </b> </FONT> degres C <br />
$if (temp>=25.0):
    <FONT color="red" size=15>
    il fait chaud
    </FONT>
    (>25 degres)
$else:
    <FONT color="blue" size=12>
    il fait froid
    </FONT>
    (<25 degres)
$else:
<em>Ca ne fonctionne pas</em>
<br />
Les boutons commandent la LED sur GPIO4 <br />
Une action sur les boutons recharge la page <br />
<form class="form" method="post">
$:form.render()
</form>
<br /><em><b>
Lycee C.Colomb </b>94370 Sucy en Brie
</em><br />
```

4. Lancer le programme gpio4.py et accéder au serveur de la Raspberry Pi depuis un PC en réseau à l'aide d'un navigateur à l'adresse : XXX.XXX.XXX.XXX :8080

Résultat attendu :

