instancier un serveur. Un paramètre est obligatoire: le port auquel le serveur va se connecter. Vous pouvez également modifier leslogin et mot de passe par défaut. Le serveur va démarrer et fonctionner dans son propre fil d'exécution (thread) jusqu'à la fin du script. Il faut ajouter une boucle pour maintenir le serveur en fonctionnement. Nous utilisons la fonction webiopi.runLoop() pour cela. Elle met en sommeil le fil d'exécution principal jusqu'à un CTRL-C. Nous pouvons également passer une fonction à la boucle.

```
import webiopi
# Instancier le serveur WebIOPi
# Il démarre immédiatement
server=webiopi.Server(
          port = 8000,
          login="cambot",
          password="Cambot")

# Exécuter la boucle par défaut
webiopi.runLoop()

# Arrêter proprement le serveur
serveur.stop ()
```

Le script précédent démarre simplement le serveur WebIOPi. Nous pouvons utiliser l'application Web par défaut pour interagir avec le GPIO, l'API REST ou la bibliothèque Javascript. Nous pourrions contrôler directement les entrées du pont en H en Javascript, mais nous allons ajouter les macros REST go_forward et stop pour diminuer la latence.

Pour continuer, nous avons besoin d'une bibliothèque GPIO. Nous pouvons utiliser RPi.GPIO ou la bibliothèque intégrée GPIO, qui est un dérivé de RPi.GPIO. Une bibliothèque intégrée permet d'éliminer de nombreuses vérifications pour l'accès depuis le serveur et offre plus de fonctionnalités.

Juste après la section d'importation:

```
# Librairie GPIO intégrée
GPIO=webiopi.GPIO
```

Nous ajoutons des variables pour faciliter le contrôle du pont en H :

```
# GPIOs moteur gauche
L1=9  # L293 IN1 sur GPIO 9
L2=10 # L293 IN2 sur GPIO 10
LS=11 # L293 EN1 sur GPIO 11

# GPIOs moteur droit
R1=23 # L293 IN3 sur GPIO 23
R2=24 # L293 IN4 sur GPIO 24
RS=25 # L293 EN2 sur GPIO 25
```

Avant l'appel du serveur, nous écrivons les fonctions pour les moteurs gauche et droit, puis nous les insérons dans les macros go_forward et stop :

```
# Fonctions du moteur gauche
def left_stop():
    GPIO.output(L1, GPIO.LOW)
    GPIO.output(L2, GPIO.LOW)

def left_forward():
    GPIO.output(L1, GPIO.HIGH)
    GPIO.output(L2, GPIO.LOW)

# Fonctions du moteur droit
def right_stop():
    GPIO.output(R1, GPIO.LOW)
    GPIO.output(R2, GPIO.LOW)

def right_forward():
    GPIO.output(R1, GPIO.HIGH)
    GPIO.output(R2, GPIO.LOW)

# Réglage de la vitesse du moteur
def set_speed(speed):
    GPIO.pulseRatio(LS, speed)
    GPIO.pulseRatio(RS, speed)

# Fonctions de mouvement
def go_forward():
    left_forward()
    right_forward()

def stop():
    left_stop()
    right_stop()
```

Ensuite et toujours avant l'appel au serveur, nous initialisons le GPIO :

```
# Initialisation des broches GPIOs
GPIO.setFunction(LS, GPIO.PWM)
GPIO.setFunction(L1, GPIO.OUT)
GPIO.setFunction(L2, GPIO.OUT)

GPIO.setFunction(RS, GPIO.PWM)
GPIO.setFunction(R1, GPIO.OUT)
GPIO.setFunction(R2, GPIO.OUT)

set_speed(0.5)
stop()
```

Finalement, nous devons enregistrer les macros sur le serveur pour les ajouter à l'API REST. Ceci permettra de les appeler depuis l'appli web :

```
server.addMacro(go_forward)
server.addMacro(stop)
```

Article d'Eric PTAK

Nous verrons la suite dans le prochain numéro, mais si vous voulez prendre une longueur d'avance, visitez http://files.trouch.com/webiopi/cambot.zip