# TP – Le GPIO du Raspberry pi
# Lecture et écriture sur les ports standards

## Mots clés

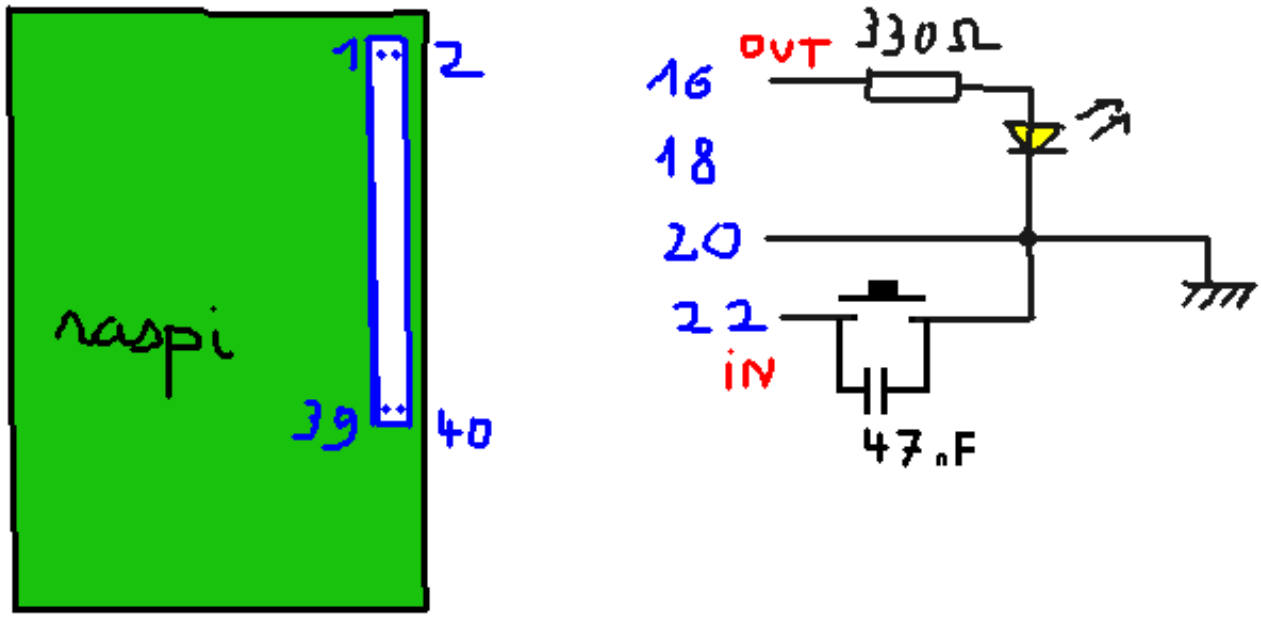
Raspberry pi, GPIO, langage Python, protocole HTTP, serveur web Apache, interface CGI, langage HTML, langage JavaScript, objet XMLHttpRequest, format JSON et AJAX.

## Matériel

1 Raspberry pi 3 (OS Raspbian Jessie)
1 résistance d'environ 330 Ω
1 LED
1 bouton poussoir normalement ouvert
1 condensateur de quelques dizaines de nanofarads

**Sommaire**

## 1. Schéma électrique




### In

De manière logiciel, il faudra définir la broche 22 (nom associé GPIO25) comme une entrée et activer la résistance de pull-up interne (50 kΩ – 65 kΩ).

Le bouton poussoir est normalement ouvert :

BP au repos (ouvert)  => niveau logique 1 (3,3 V) sur l'entrée 22
BP appuyé (fermé)  => niveau logique 0 (0 V) sur l'entrée 22

Le condensateur limite l'effet des rebonds mécaniques du bouton poussoir.

### Out

De manière logiciel, il faudra définir la broche 16 (nom associé GPIO23) comme une sortie.

LED allumée <=> sortie au niveau 1 (3,3 V)
LED éteinte  <=> sortie au niveau 0 (0 V)

***Question 1*** : Calculer le courant dans la LED quand celle-ci est allumée.
(Réponse page 19)

## 2. Lecture du niveau sur l'entrée associée au bouton poussoir

### 2.1. Programme Python en mode console

Dans toute la suite, nous utiliserons **Python dans sa version 2.7** (avec quelques petites modifications, cela fonctionne aussi avec Python 3).

#### 2.1.1. IDE Python

Nous faisons le choix d'utiliser Geany.
Pour l'installer, taper les commandes suivantes dans la console linux :

```
$ sudo apt-get update
```

```
$ sudo apt-get install geany
```

Ouvrir l'éditeur Geany
Fichier → Nouveau
Document → Définir l'encodage → Unicode (UTF-8)
Document → Définir le type de fichier → Fichier source Python

Enregistrer le fichier sur le bureau (/home/pi/Desktop) avec le nom
***lecture_entree_mode_console.py***
Pour exécuter le script :
Construire → Exécuter (ou touche F5)

### 2.1.2. Script à réaliser

Le script doit afficher en temps réel un changement de niveau sur la broche 22.
Par exemple :

```
En attente d'événements...
Touche Enter pour quitter.
ON
OFF
ON
OFF
ON
OFF
Bye !
```

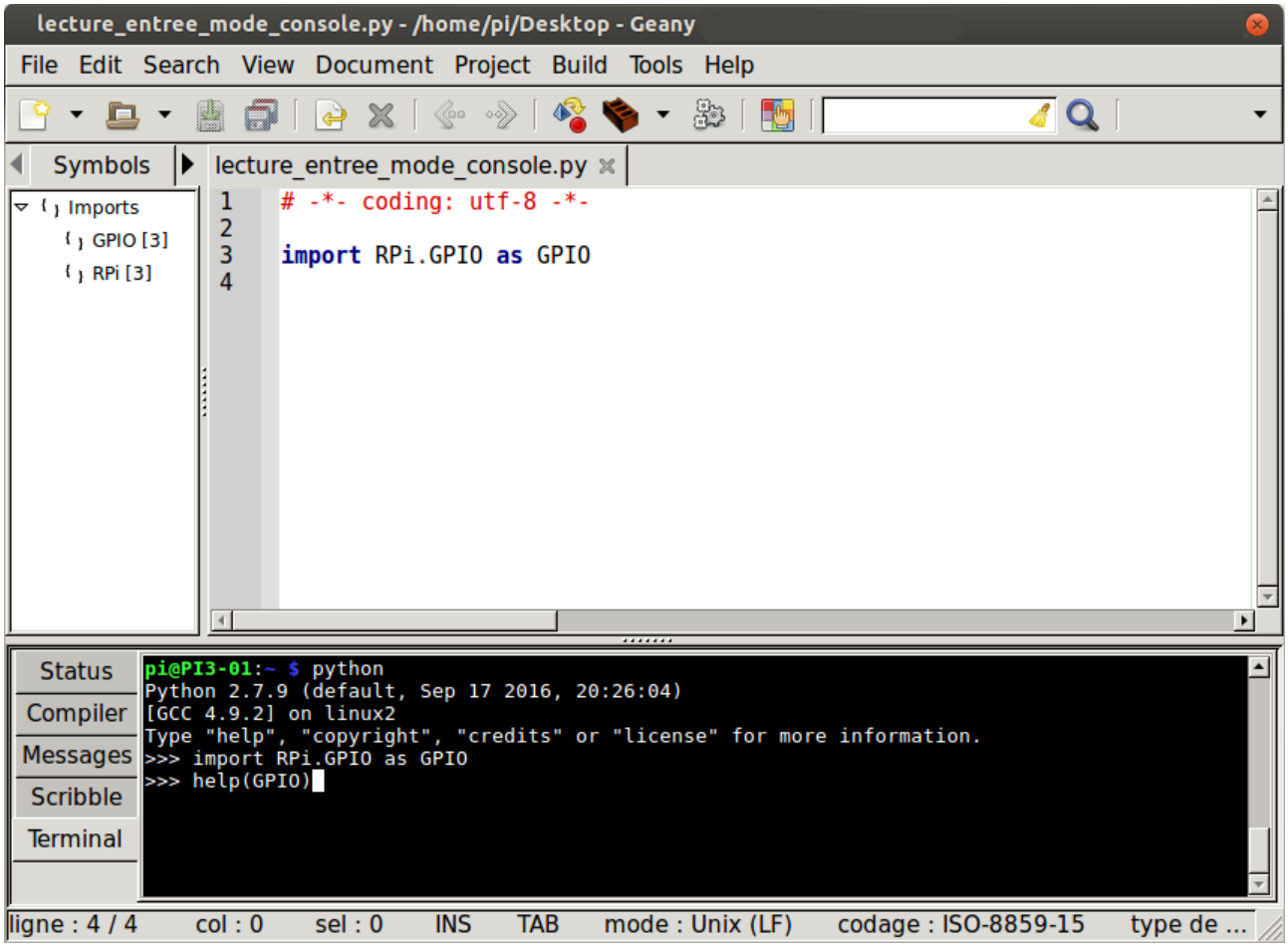
- Aide sur le module RPi.GPIO

Pour obtenir de l'aide sur le module RPi.GPIO, on pourra utiliser la console Python dans un terminal :

Geany → Affichage → Afficher la fenêtre de message
Dans le terminal linux :
$ python
Dans la console Python :

```
>>> import RPi.GPIO as GPIO
>>> help(GPIO)
>>> help(GPIO.add_event_detect)
```

(touche q pour quitter)
(CTRL + d pour quitter la console Python)




- Quelques indications :

Pour définir l'encodage du fichier (1ère ligne du script) :

```
# -*- coding: utf-8 -*-
```

Pour importer le module RPi.GPIO :

```
import RPi.GPIO as GPIO
```

Pour choisir la manière de nommer les broches du GPIO :

```
# board numbering system
GPIO.setmode(GPIO.BOARD)


# configuration en entrée avec résistance de pull-up
channel = 22
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_UP)


# edge detection & antirebonds
# la fonction affichageniveau() (à créer) est appelée à chaque changement du
niveau d'entrée
GPIO.add_event_detect(channel, GPIO.BOTH, callback=affichageniveau, bouncetime=10)

# fin du programme
raw_input("""En attente d'évenements...
```

```
Touche Enter pour quitter.
""")

print "Bye !"
```

## 2.2. Programme Python en mode graphique avec le module Tkinter

Le module Tkinter permet de réaliser des interfaces graphiques de manière relativement simple :


Nous allons nous contenter de compléter le script
***lecture_entree_mode_graghique_ACOMPLETER.py*** disponible dans les ressources.
Copier le script sur votre bureau (`/home/pi/Desktop`).

Exécuter le script.

Deux modifications sont à faire :

1) ligne 161 à compléter : actuellement, le bouton Valider n'a aucun effet (à comparer avec le bouton Quitter ligne 172)
2) ligne 115 (« ON » ne s'affiche pas)

## 2.3. Graphe déroulant en temps réel avec le module matplotlib de Python

### 2.3.1. Installation du module matplotlib

Le cas échéant, pour installer le module matplotlib pour Python 2 :

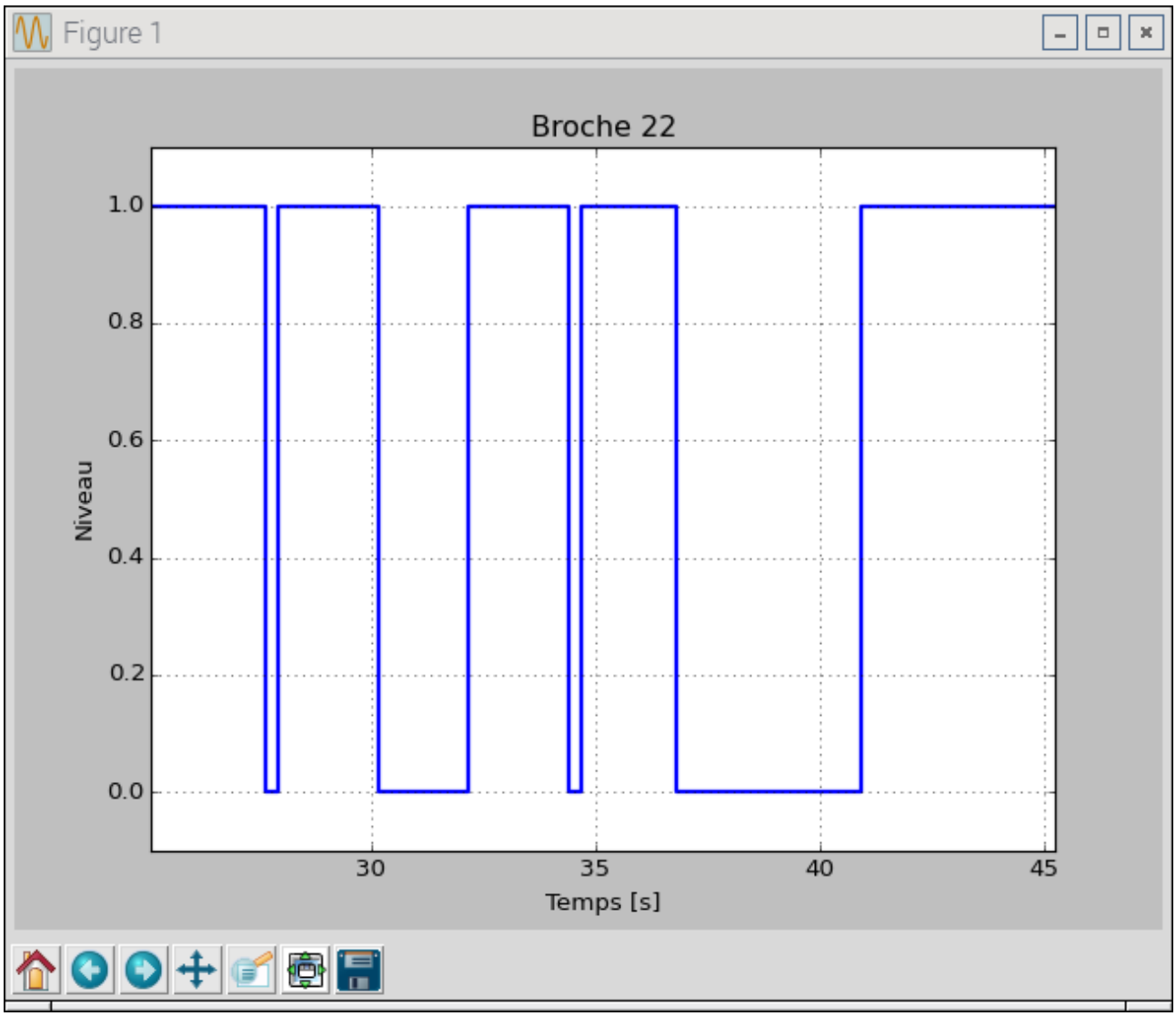
```
$ sudo apt-get install python-matplotlib
```

Vérification dans une console Python :

```
>>> import matplotlib
```

### 2.3.2. Graphe déroulant du niveau d'entrée

Le script ***graphe_deroulant_interrupteur_gpio.py*** est disponible dans les ressources.




A tester !

Pour obtenir un autre mode de défilement, commenter la ligne 110 et décommenter la ligne 109.

## 3. Écriture en sortie : commande d'une led

### 3.1. Programme Python en mode console

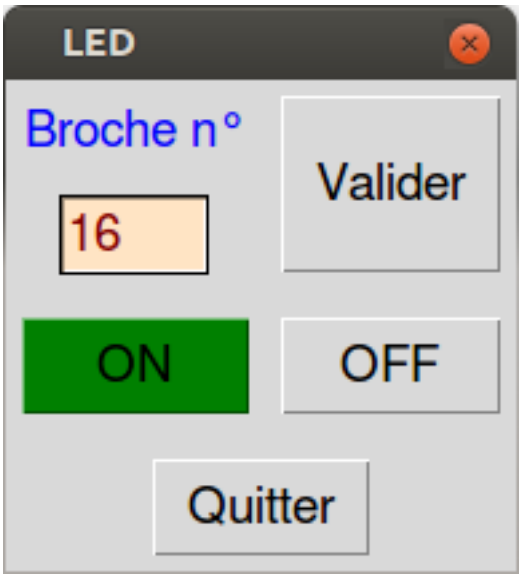
Écrire le script ***led_mode_console.py*** qui permet de piloter la led branchée sur la sortie 16.
Par exemple :

```
La LED est actuellement allumée (niveau 1)

Changer l'état de la LED (c) ou quitter (q) ? c
La LED est maintenant éteinte

Changer l'état de la LED (c) ou quitter (q) ? c
La LED est maintenant allumée

Changer l'état de la LED (c) ou quitter (q) ? q
Bye !
```

### 3.2. Programme Python en mode graphique avec le module Tkinter

Le script ***led_mode_graphique.py*** est disponible dans les ressources.

A tester !




## 4. Interface web avec un serveur Apache

On désire maintenant lire ou écrire sur le GPIO via une page web.
Cela permettra d'avoir accès au GPIO de votre Raspberry Pi depuis le monde entier à travers le réseau Internet.

### 4.1. Qu'est-ce qu'un serveur web ?

Un serveur web est une application qui communique avec le protocole HTTP d'Internet.
Un client web (généralement un navigateur web) envoie une requête HTTP au serveur web.

Exemple de requête HTTP :

```
GET https://fr.wikipedia.org
```

Le serveur web renvoie au client la ressource située à l'URL `https://fr.wikipedia.org`

Ici, la ressource est une page web, c'est-à-dire un document qui contient du code HTML.
Le type « MIME » est alors « `text/html` ».

Mais cela peut aussi être :

- une image (type MIME `image/jpeg`, `image/png`, etc.)
- des données, par exemple au format JSON : `application/json`
  `{"ville":"valence","temperature":6.9}`
- un document pdf (`application/pdf`)
- un fichier de code JavaScript (`application/javascript`)
- du son, une vidéo, un document LibreOffice, etc.

Noter bien que c'est toujours le client qui demande, et le serveur qui répond.

## 4.2. Installation d'un serveur web Apache sur votre Raspberry Pi

Dans la console linux :

```
$ sudo apt-get update
$ sudo apt-get install apache2
```

**Test de fonctionnement du serveur**

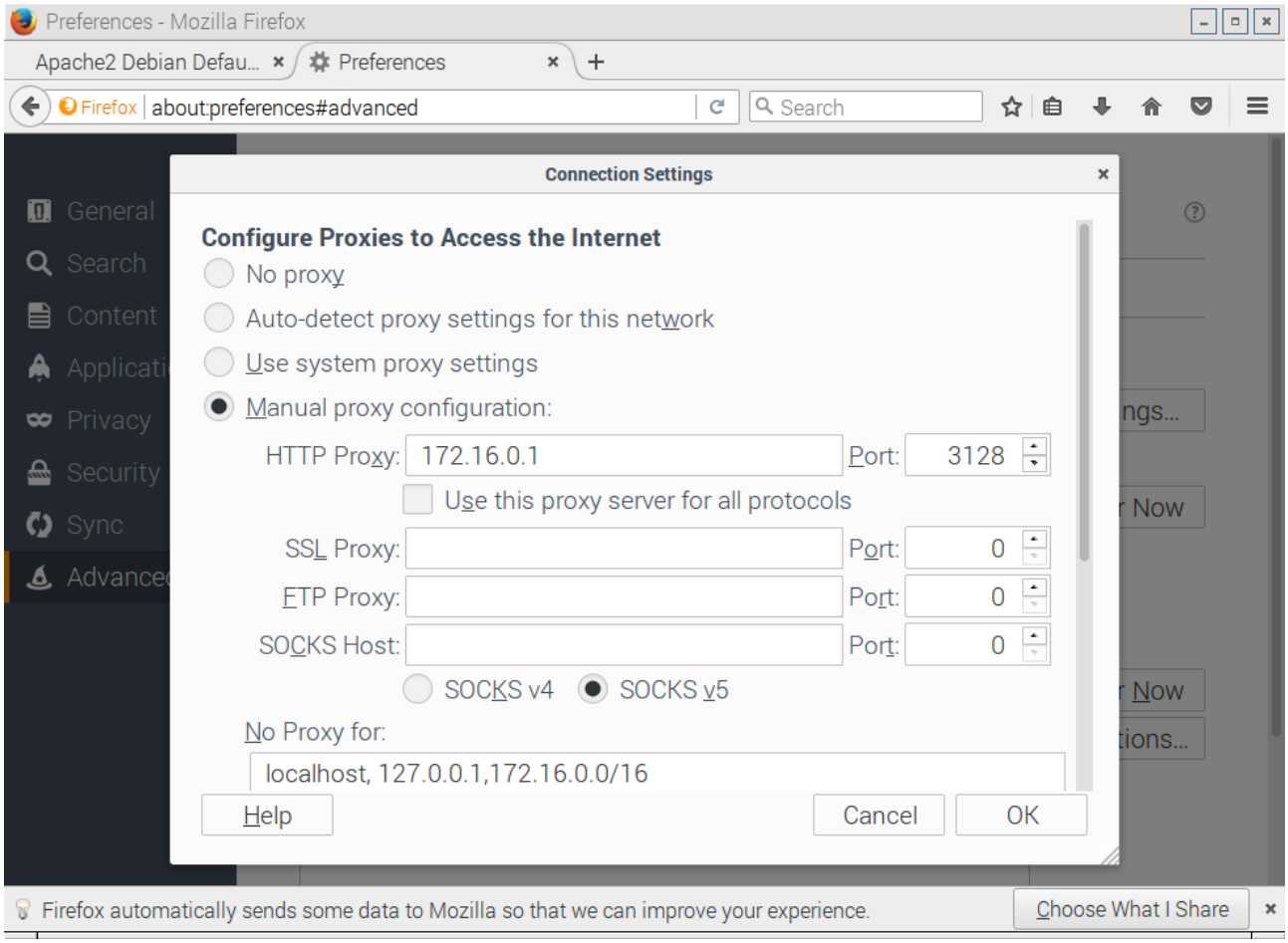
Le navigateur par défaut (Epiphany) posant quelques problèmes de configuration, nous allons installer Firefox :

```
$ sudo apt-get update
$ sudo apt-get install firefox-esr
```

**Paramétrage du proxy dans Firefox (si nécessaire)**

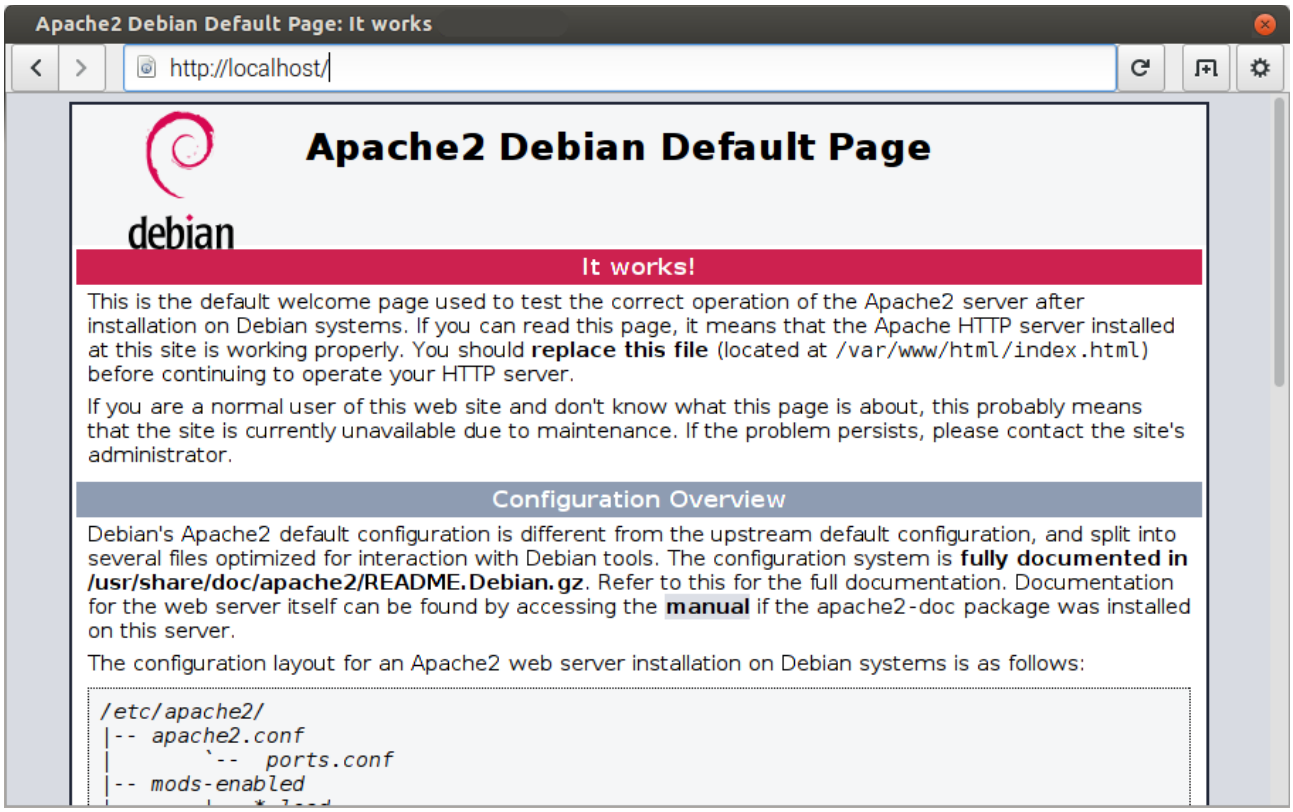
Menu → Preferences → Advanced → Network → Settings

Par exemple :




Taper `http://localhost` dans la barre d'adresse.

Votre navigateur doit afficher la page d'accueil par défaut de votre serveur web :




Votre serveur web est également accessible depuis le réseau local (et depuis Internet si les routeurs intermédiaires sont bien configurés).

Récupérer votre adresse IP avec la commande système **ifconfig** (par exemple : 172.16.115.201). Depuis le Raspberry pi ou le PC de votre voisin, taper l'URL `http://172.16.115.201` dans un navigateur et vérifier que vous avez bien accès à votre serveur web !

**Racine du site web (Document root)**

`/var/www/html`

C'est dans ce répertoire que sont stockées les ressources du serveur web.

Par exemple, `http://localhost/rep/index.html` pointe vers le fichier situé à l'emplacement : `/var/www/html/rep/index.html`

Si le fichier n'existe pas, vous aurez droit à la fameuse erreur 404 !

## 4.3. Langages de programmation côté serveur

Avec Apache, on utilise habituellement le langage PHP (dont la syntaxe est très proche du langage C).

Exemple :

`http://fsincere.free.fr/isn/langages_web/doc/calcul_carre_get.php?Nombre=4`

Quand il reçoit la requête HTTP :

```
GET http://fsincere.free.fr/isn/langages_web/doc/calcul_carre_get.php?Nombre=4
```

le serveur web Apache constate que la ressource demandée est un script PHP, et il passe la main à l'interpréteur PHP (module mod_php d'Apache).
Celui-ci génère du code HTML (type MIME `text/html`) qui sera envoyé au client :

```
<!DOCTYPE html>
<html>
        <head>
                <meta charset="UTF-8">
                <title>Calcul du carré</title>
        </head>
        <body>
                <h2>Résultat</h2>
                <p>Le carré de <strong>4</strong> est <strong>16</strong></p>
        </body>
</html>
```

## 4.4. Module CGI (Common Gateway Interface) d'Apache

PHP c'est très bien pour les applications web standards, mais ce n'est pas l'idéal pour accéder au port GPIO du Raspberry pi.
Il paraît judicieux d'utiliser ici Python.

Pour faire le lien entre Python (mais aussi Perl, un programme en langage C, etc.) et le serveur web, il faut activer le module CGI d'Apache.

### Activation

```
$ sudo a2enmod cgi
```

### Test de fonctionnement

L'emplacement du répertoire par défaut des scripts CGI est :

```
/usr/lib/cgi-bin
```

Un script ***test_cgi_python.py*** est disponible dans les ressources.

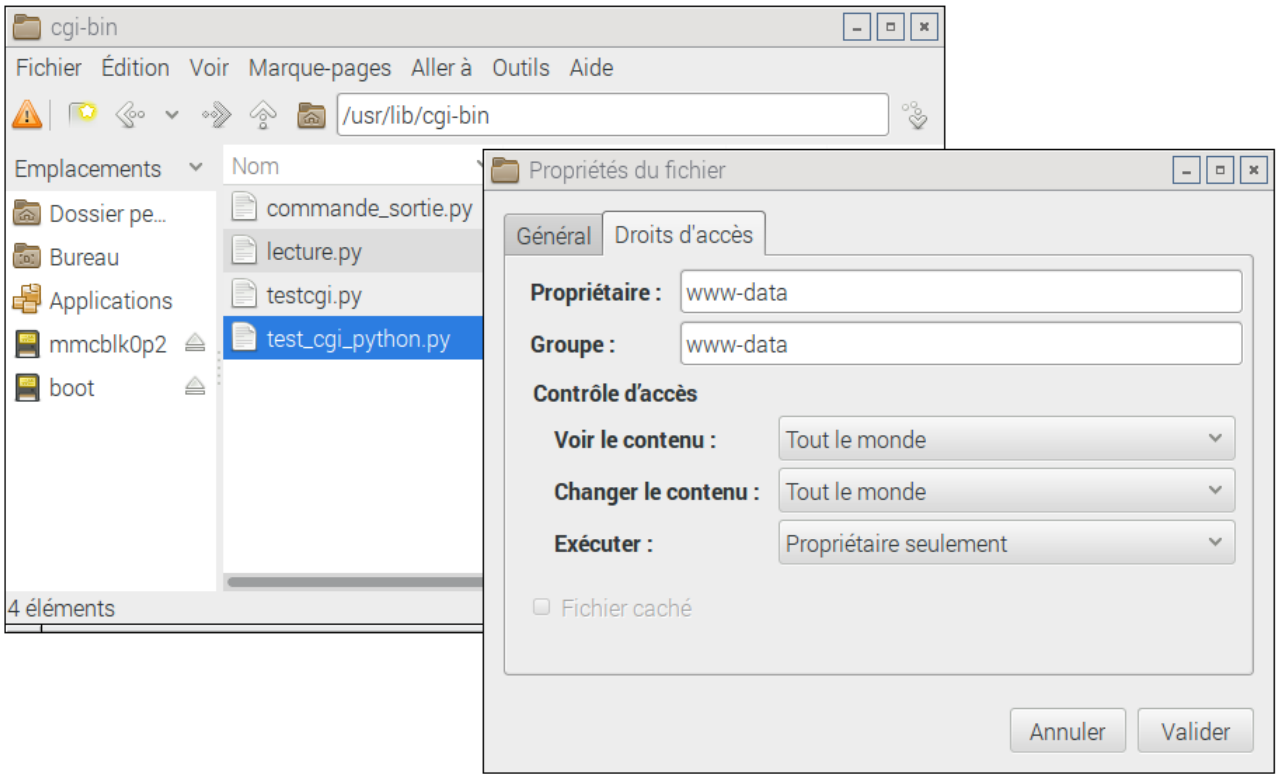
Il contient le code Python suivant :

```
#! /usr/bin/python
# -*- coding: utf-8 -*-

import time

print "Content-Type: text/html\n"
print """
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Une page Web</title>
    </head>

    <body>
        <h1>Une page Web avec CGI-Python</h1>
        <p>Heure courante {}</p>
    </body>
</html>
""".format(time.strftime('%H:%M:%S'))
```

Copier le script sur votre bureau.
Puis le déplacer dans le répertoire `/usr/lib/cgi-bin` à l'aide du gestionnaire de fichiers en mode root :

Menu → Autre → X-Terminal as root
`$ pcmanfm`

Copier / coller
puis changer les droits (notamment le droit en exécution) :
Droits d'accès → Propriétés




**Remarque**
La même chose en ligne de commande :

```
$ sudo cp /home/pi/Desktop/test_cgi_python.py /usr/lib/cgi-
bin/test_cgi_python.py
$ sudo chown www-data:www-data /usr/lib/cgi-bin/test_cgi_python.py
$ sudo chmod 766 /usr/lib/cgi-bin/test_cgi_python.py
```

Dans un navigateur :

`http://localhost/cgi-bin/test_cgi_python.py` doit afficher la page suivante :


## 5. Écriture sur le port GPIO via une page web

Nous allons piloter notre LED via une page web.

### 5.1. Droit d'accès du serveur Apache au port GPIO

Il faut commencer par ajouter l'utilisateur « www-data » au groupe « gpio » :
```
$ sudo adduser www-data gpio
```

Un redémarrage est nécessaire :
```
$ sudo reboot
```

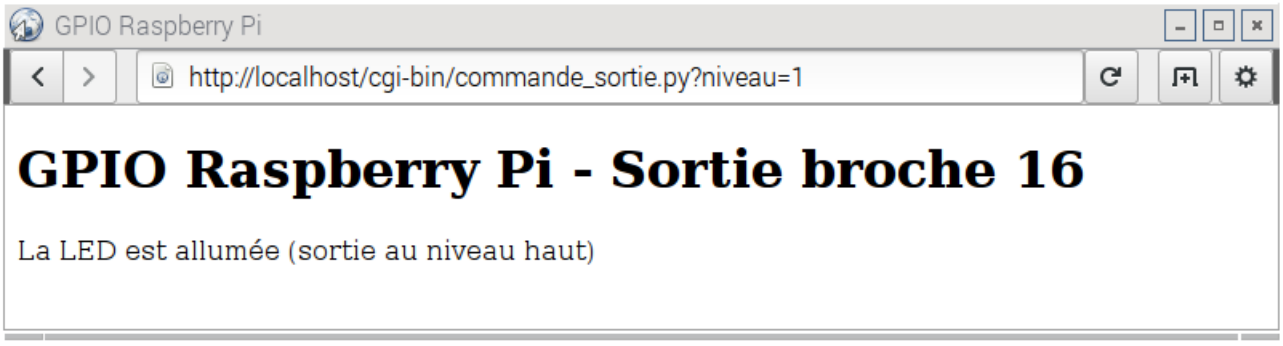
### 5.2. Script CGI en Python

Le script CGI ***commande_sortie_ACOMPLETER.py*** est disponible dans les ressources.

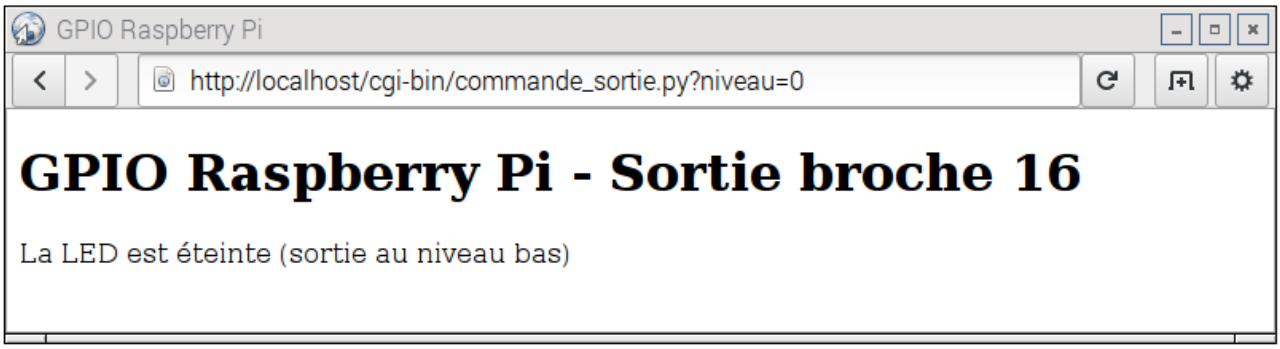
Copier le script dans le répertoire `/usr/lib/cgi-bin` (à renommer ***commande_sortie.py)***

Compléter le code (avec Geany) pour avoir le fonctionnement attendu suivant :

- Mise à 1 de la sortie :
`http://localhost/cgi-bin/commande_sortie.py?niveau=1`




- Mise à 0 de la sortie :
`http://localhost/cgi-bin/commande_sortie.py?niveau=0`




### 5.3. Formulaire HTML

Nous allons améliorer l'interface avec un « formulaire » HTML qui contient deux boutons :

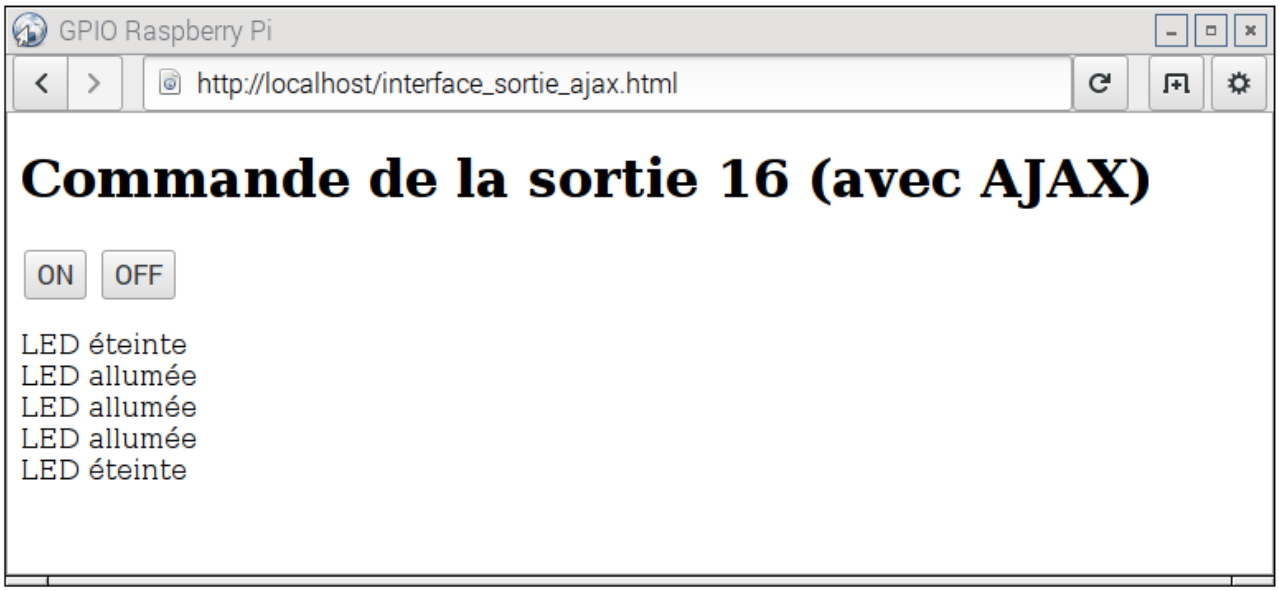

La page HTML ***interface_sortie_ACOMPLETER.html*** est disponible dans les ressources.

Copier la page dans le répertoire `/var/www/html` (à renommer ***interface_sortie.html)***

Compléter le code HTML :
- créer un bouton OFF
- créer le lien avec cgi-bin/commande_sortie.py?niveau=0

### 5.4. Formulaire dynamique avec AJAX

Améliorons encore l'interface avec la technique « AJAX » :




La page HTML ***interface_sortie_ajax.html*** est disponible dans les ressources.

Copier la page dans le répertoire `/var/www/html`

Tester le bon fonctionnement !

### 5.4.1. Qu'est-ce qu'AJAX (Asynchronous JavaScript and XML) ?

L'architecture informatique AJAX permet de construire des applications Web et des sites web dynamiques interactifs sur le poste client en se servant de différentes technologies ajoutées aux navigateurs web entre 1995 et 2005.

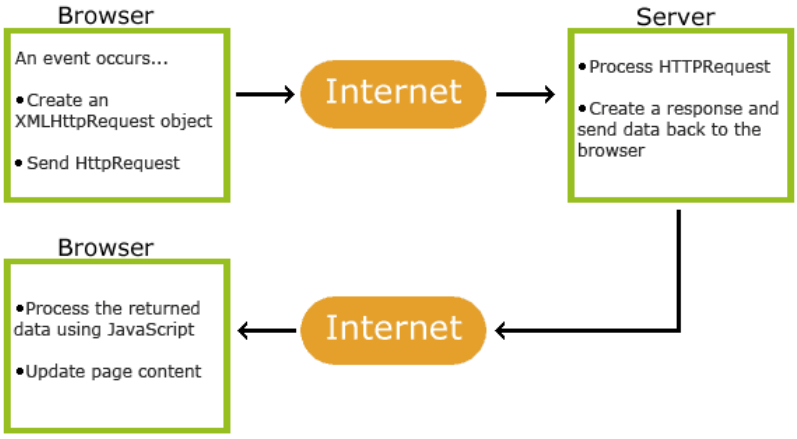
AJAX combine JavaScript, CSS, JSON, XML, le DOM et surtout l'objet **XMLHttpRequest** afin d'améliorer maniabilité et confort d'utilisation des applications internet riches (RIA) :

- le DOM (Document Object Model) et JavaScript permettent de modifier l'information présentée dans le navigateur en respectant sa structure ;
- l'objet XMLHttpRequest sert au dialogue asynchrone avec le serveur Web ;
- XML ou JSON structure les informations transmises entre serveur Web et navigateur.

Les applications AJAX fonctionnent sur tous les navigateurs Web courants (Google Chrome, Safari, Mozilla Firefox, Internet Explorer, Opera, etc.).

#### Principe

Avec AJAX, le dialogue entre le navigateur et le serveur se déroule la plupart du temps de la manière suivante : un programme écrit en langage de programmation JavaScript, incorporé dans une page web, est exécuté par le navigateur.
Celui-ci envoie en arrière-plan des demandes au serveur Web, puis modifie le contenu de la page actuellement affichée par le navigateur Web en fonction du résultat reçu du serveur, évitant ainsi la transmission et l'affichage d'une nouvelle page complète.




### XMLHttpRequest

Le XMLHttpRequest (souvent abrégé XHR) est un objet de programmation, utilisé dans les programmes en langage JavaScript pour assurer la communication entre le navigateur et un serveur Web.
Il est utilisé pour la communication asynchrone : envoyer les requêtes HTTP vers le serveur et déclencher des opérations lors de la réception de réponses de celui-ci.

### JSON

JavaScript Object Notation (JSON) est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML par exemple.
Bien qu'utilisant une notation JavaScript, JSON est indépendant du langage de programmation.
Il sert à faire communiquer des applications dans un environnement hétérogène.
Il est notamment utilisé comme langage de transport de données par AJAX et les services Web.
Le type MIME *application/json* est utilisé pour le transmettre par le protocole HTTP.

[Sources : Wikipedia & www.w3schools.com]

## 5.4.2. Explication du code de la page *interface_sortie_ajax.html*

Ouvrir le contenu du fichier.

La page HTML contient du code JavaScript dans la balise `<script>`.

Une fois la page reçue du serveur, le code JavaScript est exécuté par le navigateur (donc côté client).

Ligne 12 :
```
<button type="button" onclick="gpio(1)">ON</button>
```

Quand on clique sur le bouton ON, la fonction gpio() de JavaScript est exécuté, avec passage du paramètre 1 (qui représente le niveau souhaité).

Cette fonction fait deux choses :

a) Une requête HTTP :
`GET cgi-bin/commande_sortie.py?niveau=1`
est lancée en arrière fond (ligne 38) : elle provoque la mise à 1 de la sortie.

b) Le texte « LED allumée » est ajouté dynamiquement dans la page web (paragraphe de la ligne 15).

### 5.4.3. L'outil réseau de Firefox

Pour comprendre le mécanisme sous-jacent, nous allons utiliser un outil de Firefox :
Firefox → Menu → Développement Web → Réseau : XHR (XMLHttpRequest)

Observer les échanges entre le client et le serveur quand vous cliquez sur les boutons.

## 6. Lecture du niveau d'entrée sur le port GPIO via une page web

### 6.1. Données au format JSON

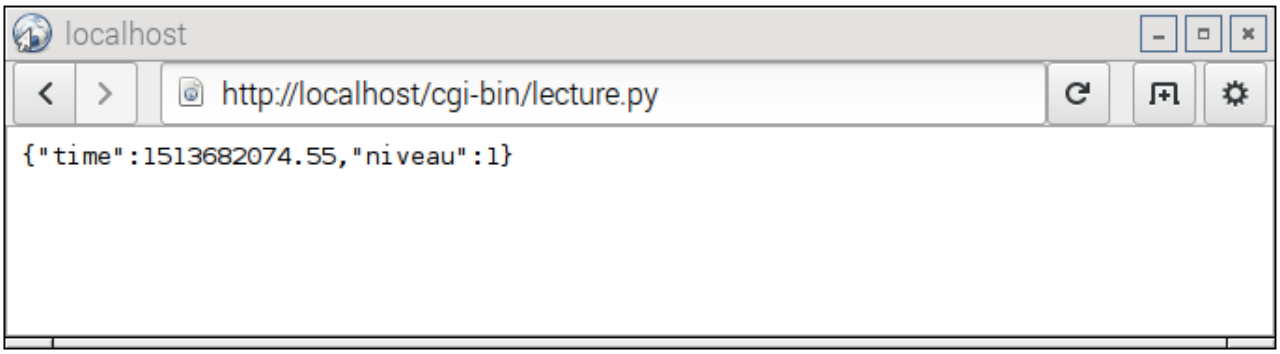
Nous allons créer une interface qui retourne le niveau à l'instant t au format JSON :
`{"time":1507291158.3383,"niveau":1}`

Le champ "time" donne le timestamp (c'est à dire le nombre de secondes écoulées depuis le 01/01/1970 à 00:00:00).

Le script CGI ***lecture.py*** est disponible dans les ressources.

Copier le script dans le répertoire `/usr/lib/cgi-bin`

A tester !




Le code Python est le suivant :

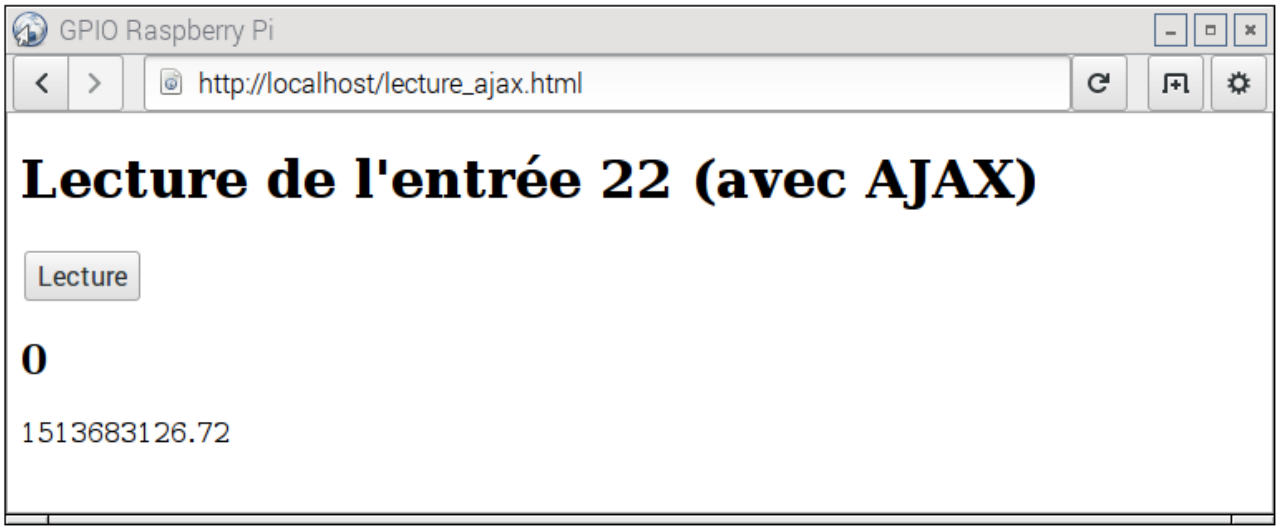
```
import time
import RPi.GPIO as GPIO

# choix de la broche
GPIO.setmode(GPIO.BOARD) # board numbering system

channel = 22
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_UP) # configuration en entrée

niveau = GPIO.input(channel)

# timestamp
timestamp = time.time()

print "Content-Type:application/json; charset=UTF-8\n"

print "{\"time\":" + str(timestamp) + ",\"niveau\":" + str(niveau)+"}"
```

### 6.2. Lecture « statique » avec AJAX

La page HTML ***lecture_ajax_ACOMPLETER.html*** est disponible dans les ressources.

Copier la page dans le répertoire `/var/www/html` (à renommer ***lecture_ajax.html)***

Compléter le code JavaScript pour afficher le niveau lu :




### 6.3. Lecture « dynamique » avec AJAX

La page HTML ***lecture_ajax2.html*** est disponible dans les ressources.

Copier la page dans le répertoire `/var/www/html`

A tester !

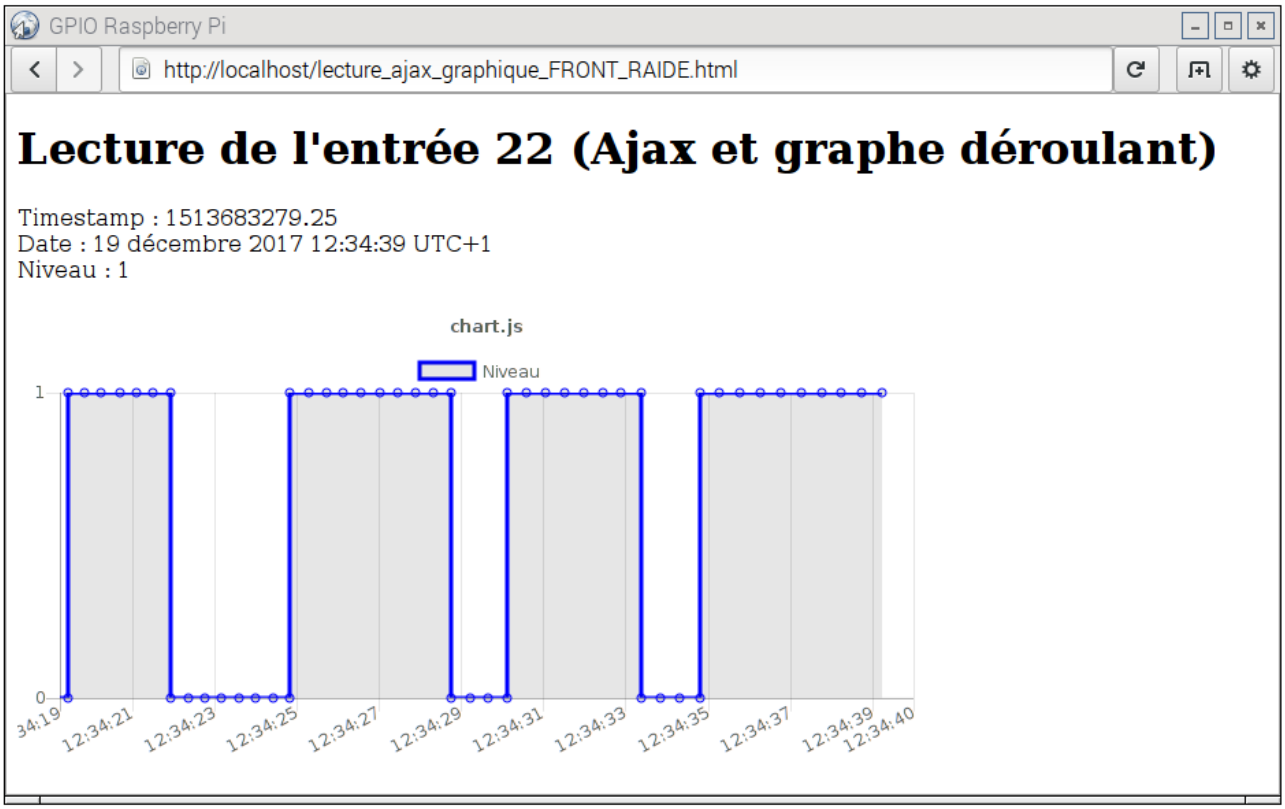
Modifier le paramètre de la fonction JavaScript *setTimeout()*.

## 6.4. Graphe déroulant avec la librairie graphique JavaScript chart.js

La page HTML ***lecture_ajax_graphique_ACOMPLETER.html*** et la librairie JavaScript « chart.js » ***Chart_bundle_min_250.js*** sont disponibles dans les ressources.

Copier la page dans le répertoire `/var/www/html` (à renommer ***lecture_ajax_graphique.html)***
Copier la librairie ***Chart_bundle_min_250.js*** dans le sous-répertoire `/var/www/html/javascript`

Compléter le code JavaScript pour avoir des fronts raides :




Site web de la librairie chart.js :
http://www.chartjs.org/

---

***Réponse à la question 1*** : Calculer le courant dans la LED quand celle-ci est allumée.

La tension aux bornes d'une LED allumée vaut environ 2 volts.
La tension aux bornes de la résistance est alors d'environ 3,3 – 2 = 1,3 V
Loi d'Ohm : 1,3 / 330 = 4 mA