Projet – Sapin de Noël

1 Interactions Humain-Machine sur le web

Les Interactions Humain-Machine (IHM) (on trouve aussi parfois Interfaces Humain-Machine) sur le web définissent les moyens et les outils mis en oeuvre afin qu'une personne physique puisse agir (contrôler, communiquer) à distance, par l'intermédiaire d'Internet par exemple. Parmi ces moyens, on trouve le langage HTML.

1.1 Le langage HTML

Un document HTML est composé de deux parties séparées par des *balises* :

- son *en-tête*, entre les balises `<head>` et `</head>` ;
- son *corps*, entre les balises `<body>` et `</body>`.

Un document HTML ressemble à ceci :

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>Mon titre</title>
</head>

<body>

</body>

</html>
```

Activité 1

1. Télécharger le dossier « Projet sapin » puis ouvrir le fichier `index.html` dans Atom et dans Firefox.
2. Ajouter la ligne suivante dans l'en-tête du fichier HTML :

   ```
   <link rel="icon" href="christmas_tree_favicon.ico">
   ```

   Rafraîchir la page ouverte dans Firefox : à quoi cette ligne a-t-elle servi ?
3. Changer le titre de la page en « Mon beau sapin ».
4. Modifier le fichier `index.html` afin que la page HTML ressemble à celle décrite par le fichier `page_sans_CSS.pdf`.
   - les liens CSS, JavaScript et Python pointeront respectivement vers :

     https://www.w3schools.com/css/default.asp,
     https://www.w3schools.com/js/default.asp,
     https://www.w3schools.com/python/default.asp ;
   - le lien Raspberry Pi 3 modèle B pointera vers :

     https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

1.2 CSS

Le langage CSS (pour l'anglais *Cascading Style Sheets*) est un langage permettant de définir les propriétés graphiques des éléments HTML constituant une page web.

Activité 2

1. Rajouter la ligne suivante dans l'en-tête du fichier HTML :

```
<link rel="stylesheet" href="style.css" type="text/css">
```

   Ouvrir ensuite le fichier `style.css` dans Atom (on gardera le fichier `index.html` ouvert).

2. Le langage CSS définit une notion de *sélecteur* qui est un moyen d'identifier à quels éléments s'applique une propriété. Écrire les lignes suivantes dans le fichier `style.css` :

```
body {
    font-family: sans-serif;
}
```

   Quel effet ces lignes ont-elles produit ?
   Vocabulaire. – On dit que `body` *est un sélecteur.* `font-family` *est une propriété CSS et* `sans-serif` *est sa valeur.*

3. Changer la couleur des titres de niveau 1 à l'aide des lignes suivantes :

```
h1 {
    color: #B3000C;
}
```

   Remarque. – La couleur est donnée en utilisant la notation hexadécimale.

4. Changer également la couleur des titres de niveau 2 (couleur : `FF0012`) et celle du contenu des balises `strong` (couleur : `green`).

5. Afin que le mot « interfaces » ait une autre couleur, on utilise un identifiant. Remplacer la partie :

```
<strong>interfaces</strong>
```

   par :

```
<strong id="interfaces">interfaces</strong>
```

   puis ajouter, dans le fichier CSS :

```
#interfaces {
    color: #058A5F;
}
```

   Vocabulaire. – On dit que `interfaces` *est un identifiant.*

6. Modifier l'apparence et le comportement des liens hypertextes à l'aide de :

```
a {
    color: #B3000C;
}
```

   puis de :

```
a:hover {
    color: #D8D8D8;
}
```

7. Modifier l'apparence des listes en ajoutant les lignes suivantes :

```
ul {
    list-style-image: url(boule.ico);
}
```

8. Ajouter une classe à la première image en remplaçant la ligne :

```
<img src="raspberry_pi.jpg" alt="un Raspberry Pi">
```

en :

```
<img src="raspberry_pi.jpg" alt="un Raspberry Pi" class="images">
```

Ajouter alors la même classe à la deuxième image puis modifier l'apparence des deux images de la page à l'aide des lignes :

```
.images {
    width: 20%;
}
```

La déclaration `width: 20%;` est appliquée à tous les éléments ayant pour classe `images`.

Remarque importante. – Dans un fichier HTML, un identifiant est unique alors que plusieurs éléments peuvent avoir la même classe.

9. Modifier l'apparence des boutons grâce aux lignes :

```
input {
    background-color: green;
    color: white;
    border: solid;
    padding: 15px 32px;
}
```

10. Dans le fichier HTML, où faut-il rajouter la classe `boite-flex` pour modifier l'emplacement des boutons à l'aide des lignes suivantes ?

```
.boite-flex {
    display: flex;
    justify-content: center;
}
```

Le faire puis écrire les lignes précédentes dans le fichier CSS.

11. Les propriétés CSS `margin-left`, `margin-right`, `margin-top` et `margin-bottom` permettent d'ajouter des marges. Ajouter :
    - une marge à gauche de `30px` aux titres de niveau 1 ;
    - une marge à gauche de `60px` aux titres de niveau 2 ;
    - une marge à gauche et une marge à droite de `60px` à chaque paragraphe ;
    - une marge à gauche et une marge à droite de `80px` à chaque liste ;
    - une marge en haut et une marge en bas de `10px` à chaque image.

1.3 Le langage JavaScript

JavaScript est l'un des langages les plus importants du web. Il est principalement utilisé pour gérer l'interactivité dans un navigateur web (côté client) mais peut également être utilisé côté serveur via la plateforme NodeJS.

Activité 3

1. Dans le dossier `Projet_sapin` créer le dossier `js`, puis créer à l'intérieur de ce dossier le fichier `script.js`. L'arborescence du projet doit ressembler à ceci :

```
/Projet_sapin
├── index.html
├── page_sans_CSS.pdf
├── /css
│   └── style.css
├── /img
│   ├── boule.ico
│   ├── christmas_tree_favicon.ico
│   ├── raspberry_pi.jpg
│   └── relais.jpg
└── /js
    └── script.js
```

   Mettre à jour, dans `index.html`, les liens vers le fichier `style.css` et les images et icônes (`boule.ico`, `christmas_tree_favicon.ico`, `raspberry_pi.jpg` et `relais.jpg`).

2. Ouvrir le fichier `index.html` dans Firefox et le fichier `script.js` dans Atom.

   (a) Ajouter le code suivant dans le fichier `script.js` :

```
function Bouton_ascendant() {
     alert("Bouton ascendant cliqué ! ");
  }
```

   (b) Afin d'appeler la fonction `Bouton_ascendant()` dans le fichier `index.html`, ajouter dans le fichier `index.html`, avant la balise `</body>` la ligne suivante :

```
<script type="text/javascript" src="./js/script.js"></script>
```

   On prendra l'habitude de placer le code JavaScript en fin de page HTML plutôt qu'entre les balises `<head>`.

   Modifier la ligne :

```
<input type="button" value="Ascendant" />
```

   en :

```
<input type="button" value="Ascendant" onclick="Bouton_ascendant();" />
```

   (c) Rafraîchir la page `index.html`, puis cliquer sur le bouton « Ascendant ». Que se passe-t-il ?

   (d) Modifier le fichier `script.js` en remplaçant l'instruction :

```
alert("Bouton Ascendant cliqué ! ");
```

   en :

```
console.log("Bouton Ascendant cliqué ! ")
```

   Rafraîchir la page `index.html`, puis cliquer sur le button « Ascendant ». Que se passe-t-il ?

   (e) Dans Firefox, ouvrir la `Console Web` : `Outils > Développement Web > Console Web`. Qu'observe-t-on ?

3. Afin de pouvoir interagir avec le serveur, nous allons envoyer une requête `GET` au serveur pour qu'il exécute la commande associée au bouton cliqué.
   (a) Ajouter à la fonction `Bouton_ascendant()` du fichier `script.js` les lignes suivantes :

```
var request = new XMLHttpRequest(); //Initialisation
request.open("GET", "/Ascendant"); //Création de la requête GET
request.send(); //Envoi de la requête GET au serveur
```

   La requête GET ne va pas aboutir pour le moment puisque le serveur web n'a pas été mis en place.
   Quelle erreur obtient-on dans la `Console Web` lorsqu'on clique sur le bouton « Ascendant » ?
   (b) Écrire une fonction pour chacun des boutons de la page web :

   `Bouton_descendant()`, `Bouton_alterne()` et `Bouton_aleatoire()`.

   Ces fonctions, enverront respectivement les requêtes GET :

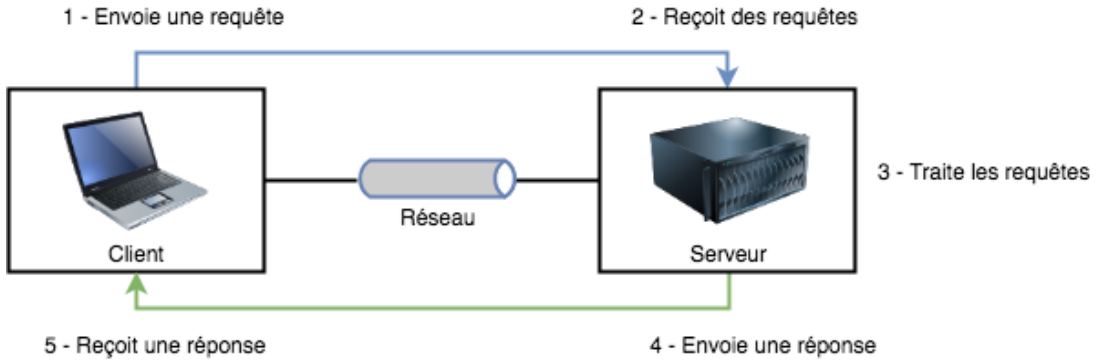
   `"/Descendant"`, `"/Alterné"` et `"/Aléatoire"`.

   Modifier le fichier `index.html` en conséquence. *On vérifiera que les boutons envoient bien un message, dans la* `Console Web`*, confirmant que le clic a bien été pris en compte.*
4. Écrire en commentaire une description de la fonction `Bouton_ascendant()`.
5. Rechercher la définition des requêtes HTTP de type `GET` et `POST`, ainsi que leur utilité.
6. *Facultatif. – On utilise ici 4 fonctions faisant presque la même action : envoyer une requête* `GET` *constituée d'une action à effectuer sur les guirlandes lumineuses. Il est possible d'écrire une seule fonction en récupérant la valeur du paramètre* `value` *du bouton avec l'instruction* `event.target.value`
   Effectuer les modifications nécessaires dans les fichiers `script.js` *et* `index.html` *pour « factoriser » votre code.*

2 Interaction client/serveur

— Le logiciel serveur offre un service sur le réseau. Le serveur accepte des requêtes, les traite et envoie le résultat au demandeur.

— Le logiciel client utilise le service offert par le logiciel serveur. Le client envoie des requêtes et reçoit des réponses.



Activité 4

1. Dans le dossier `Projet_sapin` créer les dossiers `templates` et `static` puis créer le fichier `app.py` afin de reproduire l'arborescence ci-dessous :

```
/Projet_sapin
├── app.py
├── page_sans_CSS.pdf
├── /static
│   ├── /css
│   │   └── style.css
│   ├── /img
│   │   ├── boule.ico
│   │   ├── christmas_tree_favicon.ico
│   │   ├── raspberry_pi.jpg
│   │   └── relais.jpg
│   └── /js
│       └── script.js
└── /templates
    └── index.html
```

2. Ouvrir le fichier `app.py` dans Atom, puis y ajouter les lignes suivantes :

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

from flask import Flask, render_template

# Initialisation de la méthode
app = Flask(__name__)

# Gestion du chemin
@app.route("/", methods=["GET"])
def affiche_index():
    return render_template("index.html")

@app.route("/Ascendant", methods=["GET"])
def action_ascendant():
    print("Requête GET /Ascendant bien reçue")
    return ("Succès",200)

# Lancement du serveur web
if __name__ == "__main__":
    app.run(host="127.0.0.1", port=5555, debug=True)
```

La bibliothèque externe `Flask` *permet de créer un serveur web en langage Python. Pour en savoir plus :* *https://flask-doc.readthedocs.io/.*

3. Ouvrir les fichiers `index.html` et `style.css` dans Atom, puis modifier les chemins des ressources. Par exemple :

```
<link rel="stylesheet" href="./css/style.css" type="text/css" />
```

devient :

```
<link rel="stylesheet" href="./static/css/style.css" type="text/css" />
```

4. Activer l'environnement virtuel en écrivant la commande `env_python` dans le terminal, puis lancer le script `app.py` avec la commande `python app.py`.
5. Ouvrir un navigateur web et écrire l'adresse http://localhost:5555.
   (a) Dans Firefox, ouvrir la `Console Web` puis cliquer sur le bouton « Ascendant ». Que constate-t-on ?
   (b) Dans le terminal, d'où le fichier `app.py` a été lancé, que constate-t-on ?
6. (a) Reprendre la question 5. avec l'adresse http://127.0.0.1:5555. Quelle est la différence entre les deux adresses utilisées ?
   (b) À quoi sert le nombre `5555` ajouté à la fin de l'adresse ?
7. Ajouter les routes correspondant aux différentes requêtes `GET` construites dans l'activité 3 puis vérifier que les actions sont bien prises en compte côté client et côté serveur.
8. *Les 4 routes construites, correspondant aux actions à exécuter sur les guirlandes lumineuses, sont très similaires. Il est possible de faire passer en variable la commande de la requête en écrivant* `@app.route('/<command>', meth` *et d'insérer la variable* `command` *comme paramètre dans la fonction associée à la route.*
   Effectuer les modifications nécessaires dans le fichier `app.py` *pour « factoriser » votre code.*

2.1 Le matériel utilisé

Le projet nécessite le matériel suivant :

— un Raspberry Pi 3 modèle B ;

— un module à relais ;

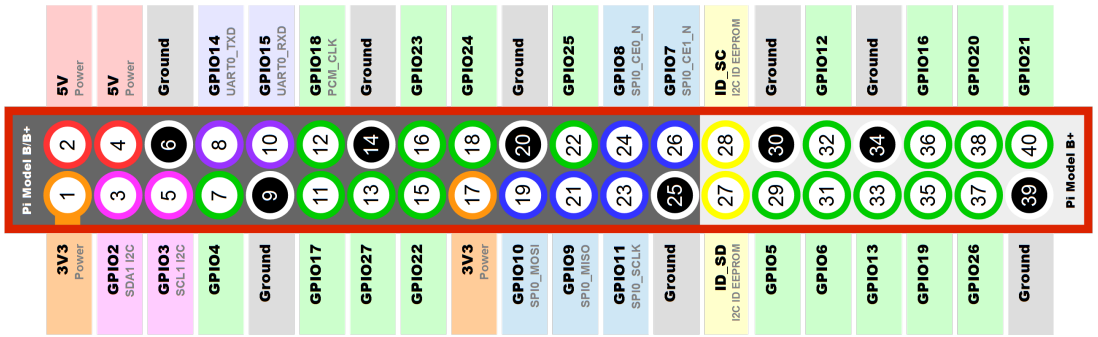
— un sapin et des guirlandes !

Le Raspberry Pi est un « nano-ordinateur » créé à l'origine pour favoriser l'apprentissage du langage informatique. Pour ce projet, on utilisera le Raspberry Pi 3 B avec la distribution GNU/Linux Raspbian (déjà installée).

Activité 5

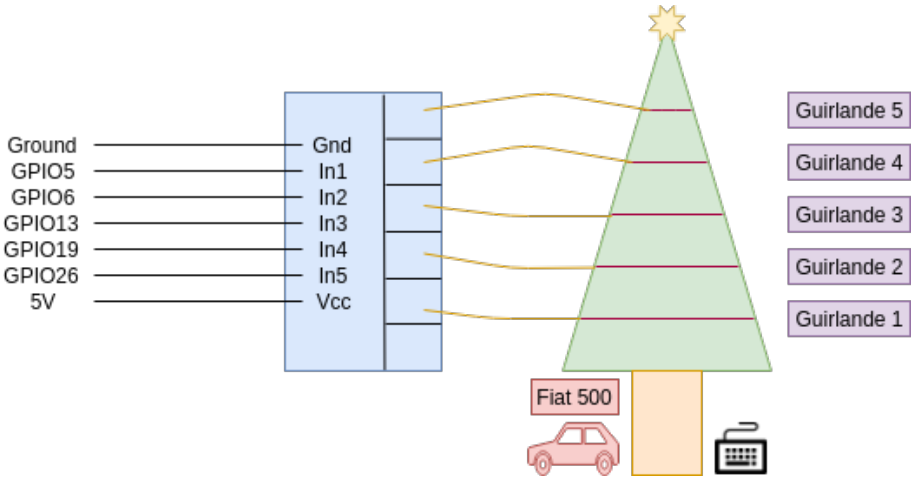
Retrouver les principales caractéristiques du Raspberry Pi 3 B et identifier les principaux composants de la carte.

Le Raspberry Pi 3 B dispose en particulier de 40 broches (ou ports) GPIO (de l'anglais *General Purpose Input/Output*) dont la plupart (voir le schéma ci-dessous) peuvent être utilisées aussi bien en entrée qu'en sortie. Pour ce projet, on utilisera le module Python `gpiozero` permettant de contrôler ces broches.



Le projet nécessite également l'utilisation d'un module à relais comme celui présenté ci-dessous :

1. Réaliser le montage ci-dessous :



2. Dans le dossier `Projet_sapin`, créer un fichier `fonctions_guirlandes.py` dans lequel on écrira les lignes suivantes :

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

from gpiozero import OutputDevice
from time import sleep

guirlande_1 = OutputDevice(26, active_high=False, initial_value=True)

guirlande_1.on()
sleep(2)
guirlande_1.off()
sleep(2)
```

   L'arborescence du projet ressemble donc à ceci :

```
/Projet_sapin
├── app.py
├── fonctions_guirlandes.py
├── page_sans_CSS.pdf
├── /static
│   ├── /css
│   │   └── style.css
│   ├── /img
│   │   ├── boule.ico
│   │   ├── christmas_tree_favicon.ico
│   │   ├── raspberry_pi.jpg
│   │   └── relais.jpg
│   └── /js
│       └── script.js
└── /templates
    └── index.html
```

3. Copier le dossier `Projet_sapin` sur le Raspberry Pi (le nom de l'utilisateur par défaut est `pi` et son mot de passe est `raspberry`), puis essayer le script précédent.
4. Effacer les lignes précédentes, puis écrire (toujours dans le fichier `fonctions_guirlandes.py`) les fonctions suivantes :
   - `allume_ascendant` qui allume les guirlandes une par une, du bas vers le haut, quatre fois ;
   - `allume_descendant` qui allume les guirlandes une par une, du haut vers le bas, quatre fois ;
   - `allume_alterne` qui allume alternativement les guirlandes de numéro impair et les guirlandes de numéro pair, quatre fois ;
   - `allume_aleatoirement` qui allume deux guirlandes au hasard, les éteint, puis recommence 19 fois.

   Chacune des fonctions commencera par éteindre toutes les guirlandes.
5. Essayer chacune des fonctions précédentes.
6. Apporter les modifications nécessaires pour terminer ce projet.