C5-160551-INFO Objets Connectés : Prog. Microcontrôleur TEA2

©Bernard Besserer

année universitaire 2021-2022

Le but de ce TEA est de vous faire développer les pages HTML (essentiellement des interfaces) qui seront utilisé pour les TP à venir. Etant donné que vous être dans le parcours “Techno Web”, cela ne devrait pas vous poser de problèmes !

1 Considérations générales

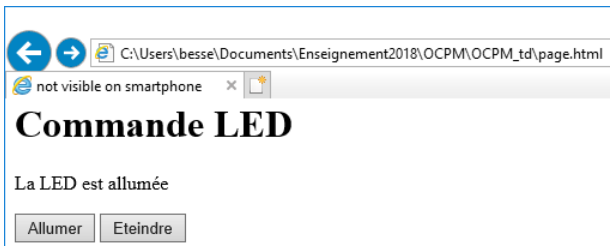
La page HTML (correspondant à une interface homme-machine) sera envoyé depuis le serveur HTTP qui sera actif sur le microcontrôleur ESP32. **On oublie donc les frameworks énormes et les pages web utilisant des dizaines de bibliothèques, et on n’utilise pas *bootstrap* dans le cadre de ces TP**. De plus, votre smartphone sera connecté au point d’accès Wifi (AP = Access Point = Hotspot) qui sera mis en service directement par le microcontrôleur, ce qui veux dire qu’il n’y aura pas d’accès internet en même temps. Alors même si, dans votre page HTML, vous faites appel à des bibliothèques en ligne via des URL (mode CDN *Content Delivery Network*), celles-ci ne seront pas accessibles (à moins d’être déjà dans cache du smartphone/tablette).

Vous pouvez tenter d’héberger ces bibliothèques. Vous pouvez utiliser du CSS3 (ce qui permet maintenant de beaux graphiques et des animations calculées directement par le navigateur, donc par le smartphone) et vous pouvez utiliser du javascript, mais sans excès.

Pour réaliser ce TEA, utilisez n’importe quel éditeur qui vous permette d’écrire facilement du HTML. Effectuez le rendu des pages HTML en les ouvrant dans Chrome et/ou Firefox (Chrome et/ou Safari si vous êtes sur plate-forme Apple). Ce n’est pas la peine d’écrire des pages *responsive*. Vous savez que vous allez utiliser votre smartphone pour afficher l’interface, donc adaptez vos pages (taille du texte, des objets, etc...). Vous pouvez par exemple utiliser cet attribut de style :

```
<style>
body {
zoom: 200%;
}
</style>
```

2 Le plus simple : Page permettant un contrôle on/off par navigation



Écrire une page HTML affichant un titre, une ligne décrivant l’état de la LED (par exemple “la LED est allumée”), et deux boutons avec les labels ON et OFF. Voir copie d’écran ci-contre pour la version minimaliste. N’oubliez pas qu’elle s’affichera sur smartphone, vous pouvez ajouter style et décoration.
Le texte de la page HTML transmise au client doit donc indiquer en toutes lettres l’état de la LED. On vous propose deux façon de réaliser cela :

2.1 Méthode 1

On crée deux page quasi identiques, la seule chose qui change, c’est le texte précisant l’état de la LED. Les boutons sont en fait des liens de navigation (balise <a>) affichés comme des boutons. Si la LED est allumée, le serveur répondra en transmettant une page `on.html`, sinon la page `off.html`. Chaque page contient les boutons, qui doivent provoquer une navigation vers ces pages.

2.2 Méthode 2

Il n'y a pas d'interpréteur PHP actif sur l'ESP32. On ne peut donc **pas** utiliser un code comme :

```
<?php
    echo 'La LED est : '+$LED_STATE +'suite du message');
?>
```

ou `LED_STATE` serait une variable PHP.

On utilisera donc la méthode suivante : Lors de la transmission vers le client, la page HTML sera découpé en deux. Le code HTML avant la "partie variable" sera une chaîne de caractère (type String) nommé `str_avant` et le code HTML après la "partie variable", qui sera une chaîne de caractère nommé `str_apres`. Le serveur pourra donc répondre en effectuant une concaténation de chaînes de caractère :
`client.send(str_avant + "La LED est allumée" + str_apres);`

Selon la methode 1, écrivez deux pages HTML, nommés on.html et off.html, et deposez-les sur moodle (ou au moins une, selon les modalités du dépot)

3 Page de contrôle d'une LED RGB

Votre page doit afficher un interrupteur on/off, et dans un formulaire (form) un champ de saisie `input type=color`. Il y aura donc aussi un bouton pour soumettre (*submit*) le formulaire.
Il faut que la chaîne de caractère correspondant à la couleur soit transmise lors de la requete GET, par exemple :
`GET /color.html?value=#FF5500` pour la couleur #FF5500 (FF(hex)=255 pour le rouge, 55(hex)=85 pour le vert, 00 pour le bleu).

3.1 Version simple

Utiliser les objets existant du HTML5, sans librairie Javascript ou code additionnel. Informations ici :
`https://developer.mozilla.org/fr/docs/Web/HTML/Element/Input/color`
Nommez votre page "rgb_simple.html" et déposez-la sur moodle

3.2 Version pleine page avec roue de couleur

Trouvez une bibliothèque très légère ou un code compact JS+CSS permettant l'affichage d'une roue des couleurs ("color wheel"). et affichez cet élément en pleine page.
Mettre en place la transmission d'une requête de type GET, soit à chaque action sur la roue de couleur (donc forcement avec du javascript et des *listener* sur des evenements), ou alors ajoutez un bouton (avec un label comme "OK" ou "SET", etc...) qui récupère la dernière couleur sélectionnée et génère la requête. La requête devra transmettre la couleur comme précédemment (par exemple un GET à l'URL `color.html?value=#FF5500`
Nommez votre page "rgb_wheel.html" et déposez-la sur moodle

4 Page avec requêtes asynchrone : AJAX

Ajax (Asynchronous JavaScript and XML) autorise la réalisation de sites web dynamiques sur le poste client en se servant de différentes technologies ajoutées aux navigateurs web entre 1995 et 2005. Ajax combine JavaScript, les requêtes de type XMLHttpRequest et les manipulations du DOM.

Partiellement repris de Wikipedia : Dans une application Web, la méthode classique de dialogue entre un navigateur et un serveur est la suivante : lors de chaque manipulation faite par l'utilisateur, le navigateur envoie une requête (par exemple GET) contenant une référence à une page Web, puis le serveur Web envoie la réponse sous forme d'une page Web à destination du navigateur. Celui-ci affichera alors la page qu'il vient de recevoir. **Chaque manipulation entraîne une nouvelle requête et une nouvelle réponse**.
Pour notre utilisations, Ajax est surtout utile pour rafraîchir un contenu, par exemple afficher une mesure effectué par l'ESP32 qui, rappelons-le, joue le rôle de serveur HTTP, de façon continue **sans** rafraîchir et ré-afficher toute la page Web.

En utilisant 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 (la demande de faire la mesure), 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**.
Les appels asynchrone Ajax seront généralement associés à la méthode Javascript SetInterval **qui permet d'appeler une fonction de manière répétée, avec un certain délai fixé entre chaque appel**. C'est une méthode de la classe Window. Par exemple :

```
setInterval(function(){ alert("Hello"); }, 3000);
```

affichera le pop-up toutes les 3 secondes. On remplacera ensuite le corps de la fonction par un code s'inspirant de celui-ci :

```
setInterval(ajaxCall, 1000); //1 sec

function ajaxCall() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function()
{
    if (this.readyState == 4 && this.status == 200)  // ready.State égal 4 signifie ‘‘DONE’’, requête traitée
{
// on recupere la reponse qui est transmise par le serveur HTTP sous forme de chaine de caractère
// avec this.responseText;
      // ... et on fait quelque chose avec cette valeur, en général manipulation du DOM pour affichage
    }
  };
  xhttp.open("GET", "XXXXXXXX", true);          // XXXXXXXX la requête que le serveur doit executer
  xhttp.send();
}
```

Le serveur détectera une requete GET avec comme cible XXXXXXXX. Dans le code en langage C du coté du serveur, il faudra, en réaction à cette requête, effectuer par exemple une mesure analogique et transmettra la valeur vers le client.

Et du coté client (navigateur), pour l'affichage, on utilisera les fonctions de manipulation du DOM, comme par exemple :

```
document.getElementById("putTheIdOfHTMLObjectHere").innerHTML = ....
```

4.1 Affichage d'une grandeur analogique, mode simple

Votre page doit permettre d'afficher une grandeur analogique en mode texte mais aussi sous forme d'une jauge plus ou moins remplie. On commença par utiliser l'objet HTML `<input type="range">`.
S'il s'agit d'une grandeur analogique transmise par l'ESP32, la valeur minimale sera 0 et la valeur max sera 4095
On utilise la technologie AJAX pour rafraîchir régulièrement l'information. Il faudra écrire du coté serveur une fonction GetMeasure qui sera également la requête pour récupérer la mesure (ce point sera traité en TP).// Du coté navigateur, il faudra modifier l'affichage de l'objet de type range pour correspondre à la valeur reçue.

```
<input type="range" name="rangeInput" min="0" max="4095" id="myRange">
<input type="text" id="mtText" value="">
```

Autres informations ici :
`randomnerdtutorials.com/esp8266-dht11dht22-temperature-and-humidity-web-server-with-arduino-ide/`
(aller à la section “Building the Web Page”)

4.2 Affichage d'une grandeur analogique avec indicateur à aiguille

Comme pour le 3.2, trouvez une bibliothèque très légère ou un code compact JS+CSS permettant l'affichage d'une valeur analogique sous la forme d'un affichage à aiguille (vu-metre)

Comme par exemple ici : `//plnkr.co/edit/Ij9BaSVeyArsqfcCCr0G` mais la bibliothèque d3js est déjà consequente, il faudrait trouver plus leger...
Les plus doués pourront essayer de le faire avec uniquement du CSS3.

Nommez votre page “analog.html” et deposez-la (ou une archive zip contenant les fichiers nécéssaires) sur moodle.

5 Interface de contrôle pour la gestion climatique d’une serre

La page doit afficher :

- La température (sous forme numérique et sous forme graphique), requetes AJAX
- Les valeurs journalières minimum et maximum de la température (c’est le l’ESP32 qui mémorise, mais la page web affiche les valeurs, requetes AJAX pour avoir le min et le max)
- L’ensoleillement, sous forme graphique (suite de pictogrammes. De petite taille, genre 48x48 pixels, au format GIF par exemple... on vous laisse vous debrouiller)
- La serre est équipée de volets pour atténuer la lumière, réglables de façon manuelle ou automatique. Un curseur ou bouton radio permet de choisir entre ces deux mode (A/M).
  Si manuel, l’utilisateur doit pouvoir choisir le réglage qui autorise 5 niveaux différents. Vous vous debrouillez pour que l’utilisateur puisse choisir entre ces 5 règlages. Vous pouvez réutiliser les pictogrammes "‘ensoleillement" pour le controle manuel.

Nommez votre page “serre.html” et deposez-la sur moodle

Annexe : Encapsuler une image dans une page WEB

Information utile si vous n’avez pas de systeme de fichier à disposition : vous pouvez encapsuler le codage d’une image dans un fichier texte (donc une page HTML !).

You can base64 encode the image data, so you would end up with something like this

```
<img src="
wAAACwAAAAADwAPAAACIISPeQHsrZ5ModrLlN48CXF8m2iQ3YmmKqVlRtW4ML
wWACH+H09wdGltaXplZCBieSBVbGVhZCBTbWFydFNhdmVyIQAAOw=="
alt="Base64 encoded image" width="150" height="150"/>
```

Here are some resources :
Base64 image converter : `www.bigfastblog.com/embed-base64-encoded-images-inline-in-html`