C5-160551-INFO
Objets Connectés :
Programmation Microcontrôleurs - TP7

©Bernard Besserer

année universitaire 2021-2022

1 Cablage

Mettre en oeuvre le capteur de luminosité et le capteur de température.

1. Pour le capteur de luminosité : le composant CNY70, une résistance d'une centaine de KOhm (3eme bague = jaune). Voir indication section 1.1 du TP4. La valeur analogique sera envoyé sur l'entrée A3
2. Pour le capteur de température : le composant AD22103. Voir indications de mise en oeuvre section 1.2 du TP4. la valeur est envoyé sur l'entrée A4

Reprendre le petit programme de test (section 1.2 du TP4) pour s'assurer du bon fonctionnement des capteurs.
Gardez sous la main la LED RGB, qui servira pour la fin de ce TP.

2 Serveur Web et page HTML pour l'affichage d'information

Lors du TP précédent, nous avons utilisé un navigateur pour afficher une interface de commande. Ici, il s'agit d'utiliser votre smartphone pour afficher en "live" une information. Il s'agit toujours de l'affichage d'une page Web dans un navigateur, mais information affichée sera rafraîchie sans recharger la page, d'où l'utilisation de Ajax
Donc, dans votre code HTML, vous aurez au moins :
- L'appel de la methode setInterval() (méthode de la classe Window ou Worker) mettant en place une fonction callback qui sera appelée automatiquement à intervalle régulier,
- L'écriture de cette fonction callback, qui va émettre une requête asynchrone via l'objet XMLHttpRequest, et en cas de succès de cette requête, va modifier le DOM pour afficher la valeur renvoyé par le serveur HTTP

```
<script>
setInterval(function() {
        getData();                                          // Appel repetitif d'une fonction toute les 1000ms
}, 1000);

function getData() {
  var xhttp = new XMLHttpRequest();                         // instance d'une requete asynchrone
  xhttp.onreadystatechange = function() {                   // declaration inline de la fonction lorsque reponse
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("valeur").innerHTML =         // il faut qu'un objet HTML avec cet ID existe
      this.responseText;
    }
  };
  xhttp.open("GET", "readValeur", true);                     // on effectue la requete : navigation GET vers rea
  xhttp.send();
}
</script>
```

Et du coté de l'ESP32, il faut bien sur effectuer quelques actions lors de la requête GET vers readValeur, requête qui sera émise régulièrement.
Vous pouvez reprendre le code de base du TP6 (controle ON/OFF de la LED). Il faut mettre en place un callback en reponse à une requete GET sur ReadValeur, et écrire la fonction correspondante à ce callback, voir ci-après. A remarquer, la réponse sera en texte pur (type MIME text/plain et non pas text/html, pour que le navigateur du coté client ne recharge pas la page)

```
server.on("/readValeur", readValeur);          // callback pour la requete AJAX

...
...
...

void readValeur() {
    Serial.println("Requete asynchrone");      // pour information. Pourra être supprimé
                                               // ici effectuer la mesure, et formater la mesure comme une cha
    server.send(200, "text/plain", s);     // renvoie chaine de caratère s comme texte pur
}
```

1. **Affichage en mode texte :**
   Vous devez affranchir votre mesure toute les deux secondes environ. Pour réduire le nombre de requête et le trafic, en réponse à votre requête asynchrone, vous transmettez les deux mesures en simultané (vous formez une chaîne de caractère avec un séparateur entre les mesures, un symbole genre & ou _... ensuite, du coté web avec JS, vous séparez les deux valeurs avant d'injecter chaque valeur dans l' dans le DOM
2. **Affichage graphique, objets HTML5 natifs :**
   Conservez votre affichage textuel (qui permettra de controler si les valeurs vous parviennent), mais doublez l'affichage avec un affichage exploitant des objets HTML5 natif `<input type=range>`. Ces objets sont prévus pour saisir une valeur, mais vous pouvez aussi les utiliser uniquement pour afficher une information :

   Si vous donnez un ID à votre élément HTML de type `<input type=range>`, vous pouvez écrire le code JS suivant :

   ```
   var input = document.getElementById("ID_de_mon_objet_de_type_range");
   input.value = XXXX;
   ```

   On vous laisse déterminer si XXXX doit être numérique (type int) ou une chaîne de caractère...

   Information sur le paramétrage des objets HTML de type range : `www.w3schools.com/tags/att_input_type_range.asp`

3. **Affichage graphique évolué :**
   Gardez toujours votre affichage textuel (en guise de contrôle) mais ajoutez un affichage plus sympathique à l'aide :
   - d'afficheur à aiguille (*needle gauge*, *analog gauge*), travail qui a dû être réalisé dans le cadre du TEA
   - (facultatif) Avec des pictogrammes (par exemple)
   Il faudra être vigilant sur la taille des données WEB (HTML+CSS+JS) que vous embarquez sur l'ESP32 : - Rappelez-vous, dans ce mode de fonctionnement, votre smartphone n'aura pas accès à internet donc pas accès aux bibliothèques en ligne CDN.
   - Utilisez la syntaxe RAW STRING pour votre contenu WEB, si possible dans un fichier distinct.
   - Déclarer vos pages avec `const char[] le_nom_de_votre_page[] PROGMEM = ....` les modificateurs `const` et `PROGMEM` sont important, la page (données statiques) est alors rangée dans la mémoire programme (mémoire flash), 4 MB dispo pour l'ESP32. Sinon, les données seront aussi recopiés dans la RAM (0.5 MB seulement, risque de plantage s'il n'y a plus la place pour la pile, par exemple)
   Si vos fichiers deviennent trop complexes pour être manipulés comme des chaînes de caractères (même en mode RAW String) ou si vous utilisez des données encodés non pas en ASCII mais en binaire comme des images (GIF ou PNG), vous devez passer à l'utilisation d'un file system comme le SPIFFS.

2.1 Utilisation de SPIFFS

- Voir PDF concernant SPIFFS, placé sous Moodle dans dans la partie cours.
- Vous pouvez répondre à une requête en transmettant directement un fichier prélevé de SPIFFS, du moins avec la méthode send de l'objet AsyncWebServer (si vous utilisez cette bibliothèque)
Par exemple, pour le callback répondant à une requête sur la "racine")

```
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send(SPIFFS, "/index.html", "text/html" );
});
```

mais attention, si dans votre fichier index.html, vous avez déclaré un lien vers un fichier CSS en écrivant :

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

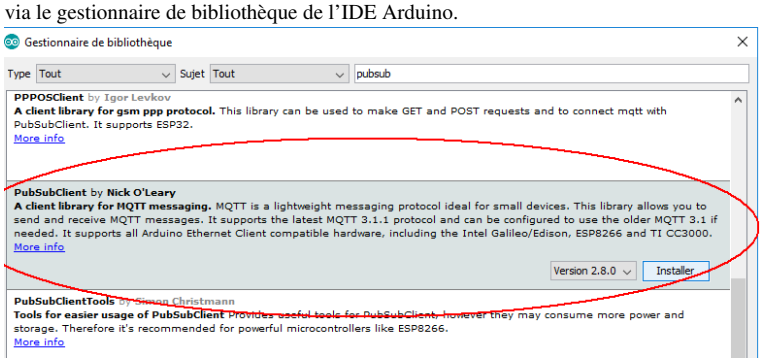
le navigateur de votre smartphone va automatiquement émettre une requête pour réclamer la "page" `style.css`, et du coté ESP32, vous devez donc aussi écrire le callback qui réponde à cette requête et qui transmet cette page.
Montrez votre affichage. Idéalement, laissez l'affichage texte pour les deux grandeurs mesurées (luminosité et température), utilisez l'objet `<input type=range>` pour la température (entre 0 et 40 degres, si vous pouvez), et un afficheur de votre choix ou de votre construction pour la luminosité (*needle gauge* ou pictogrammes)

3 utilisation de MQTT

A savoir qu'en général, l'accès au broker est protégé par un mot de passe. On peux également restreindre l'accès à un TOPIC. Pour des raisons de facilité, dans le cadre de ce TP, tous les mots de passe sont désactivés. Par contre, seule l'ecriture et la lecture vers un topic commençant par meteo est autorisé (attention a ne pas mettre de caractères accentués dans vos messages)
On suppose que le broker MQTT correspond à une station météo qui recueille des données de divers capteurs (luminosité et température dans le cadre de ce TP) et qui, en fonction des prévisions, publie des bulletins météos, avec un niveau de vigilance associé (niveau de vigilance vert, jaune ou rouge). Le niveau de vigilance rouge signifiant par exemple un avis de tempête et qu'il faut donc prendre des disposition !

3.1 Installation de la bibliothèque MQTT

Parmi les nombreuses bibliothèques MQTT, choisir la bibliothèque PubSubClient (bibliothèque minimaliste qui fait très bien l'affaire) qui s'installe via le gestionnaire de bibliothèque de l'IDE Arduino.



3.2 Code initial

Le code initial est basé sur l'exemple ... de la bibliothèque. Pour ce TP, le broker MQTT est installé sur le serveur qui est en fond de salle, il faut donc se connecter sur le reseau Wifi que celui-ci emet. Le SSID est OCPM, et il n'y pas de mot de passe. Votre ESP32 doit donc passe en mode Wifi Client (cf. TP4).
Concernant les paramètres du broker, l'adresse IP de la machine hébergeant celui-ci sera communiqué en séance (par defaut 192.168.199.188, mais cela peut changer). Le port utilisé par le broker est le port 1338.
Vous pouvez faire vos essais avec le topic "meteo/test". Si vous vous abonnez (subscribe) ce topic, et si vous y publiez (publish) un message (une courte chaîne de caractère), vous recevrez cette même chaine en retour.

3.3 Abonnement et publication vers le topic meteo

L'arborescence souhaitée est la suivante

```
meteo
    test
    vigilance
    tp1
        dupont
            temperature
            luminosity
        martin
            temperature
            luminosity
   tp2
        durand
            temperature
            luminosity
        moreau
            temperature
            luminosity
```

Évidemment : dupont, durand, martin et moreau sont à remplacer par le nom d'un des étudiants du binôme (pas de caractères accentués, pas d'espaces, en minuscule)
Votre programme doit :

1. Effectuer une mesure toute les 5 secondes et transmettre les deux valeurs (température, en ° Celcius et luminosité, valeur entre 0 et 4095) dans le topic correspondant (par exemple, le topic serait meteo/tp1/martin/temperature et le message 22.5

   Topic et message sont des **chaînes de caractères**. Limitez la valeur de la température à deux décimales.
   Vérification sur l'écran du serveur informatique hébergeant le broker MQTT d

2. Vous abonner au topic meteo/vigilance. En cas de changement de valeur, un callback se déclenche. Utilisez votre LED RGB qui doit s'allumer de la couleur du niveau de vigilance (initialisation à vert après un reset). Pour cela, branchez le canal ROUGE et le canal VERT de votre LED RGB sur deux GPIO et utilisez le mode "tout ou rien" avec digitalWrite, cf. TP6 section 2.1.2)
   Vérification du fonctionnement lors de lémission d'une alerte vigilance rouge

4 Monitoring du broker MQTT avec votre smartphone

Il existe, sur plusieurs plateforme (Windows, Linux, Android, iOS) des logiciels qui "s'interfacent" avec un broker MQTT (sous reserve d'avoir les droit d'accès) et qui en permette l'administration ou qui peuvent agir en tant que client MQTT. Les plus intéressants de ces logiciels permettent de mettre en place un "*dashboard*" ou "*panel*" : il s'agit d'une surface sur laquelle on peut agencer des afficheurs graphiques et des interfaces de contrôles, qui vont permettre de visualiser des informations (en s'abonnant à un topic) ou d'envoyer des informations (en publiant vers un topic).
L'application *IoT MQTT Panel* sous Android a été testé et fonctionne pour le broker tel que configuré pour ce TP. Après installation de l'application, vous devez configurer l'accès au broker (adresse IP, port), puis créer un Panel et ajouter sur ce panel des éléments de visualisation, puis vous abonner à un topic.

Vous pouvez ainsi visualiser graphiquement (comme avec l'objet HTML <input type=range> de la section 2) la valeur de température et de luminosité, en vous abonnant au topic meteo/dupont/temperature et meteo/dupont/luminosity (en remplaçant bien sur "dupont"...)
Attention, vous devez connecter votre smartphone au réseau OCPM pour cela.!!!
Un équivalent sur iOS serait *Home Assistant*, mais non testé !

IoT MQTT Panel
Rahul Kundu Outils
Tout public
Contient des annonces
Cette application est disponible pour votre appareil

Home Assistant 4+
Nabu Casa, Inc
4,4 • 9 notes
Gratuit

☑

Si vous avez réussi à constituer un *dashboard* ou *panel* montrant vos valeur de luminosité / température, montrer à l'intervenant.