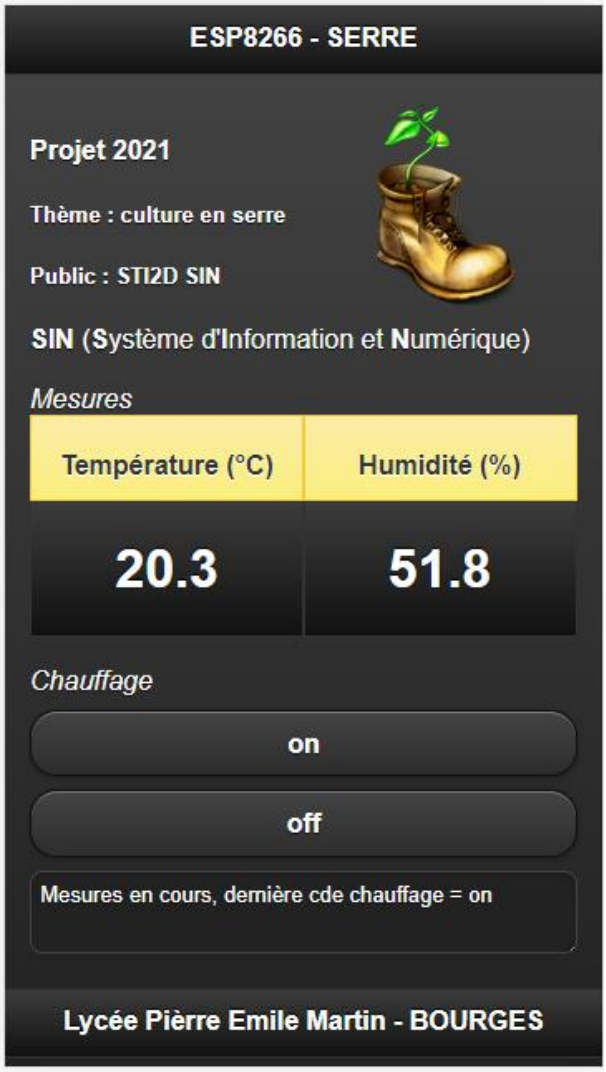


MINI SERRE

IoT et ESP8266

RÉSUMÉ

Dans ce document, l'affichage des grandeurs physiques et le contrôle d'un actionneur à partir d'une page web est assuré par un seul circuit ESP8266. Cette application illustre ce qui peut être fait dans la salle de classe avec les technologies suivantes : requêtes HTTP, JavaScript (JQuery, JQuery mobile), Ajax, Json, Arduino, LittleFS, I2C.

Systèmes d'Information et Numérique

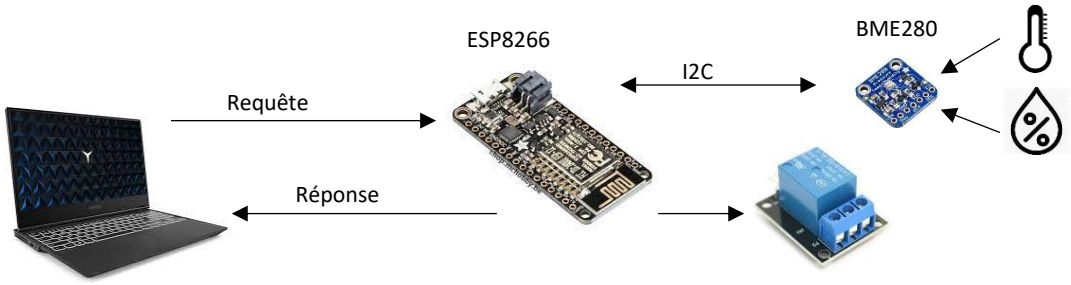
Ressources

- Le système de fichiers LittleFS (ESP) https://bit.ly/3uMv5dA
- JavaScript et JQuery – AJAX https://bit.ly/3hiGWMO

Objectif

Échanger des données entre une carte à µC et un navigateur.

1. Organisation



Client : navigateur

Le client demande périodiquement au serveur la **température** et l'**humidité** mesurées dans la mini serre, et assure l'envoi des commandes de chauffage à partir d'une action de l'utilisateur. Il affiche également les messages de contrôle de la communication.

Technologies : HTML, JQuery (Ajax), jQuery Mobile

Serveur : ESP8266, BME280, relais

L'ESP8266 sert les composants du site, mesure la température et l'humidité avec un circuit BME280 et active le relais de chauffage à la demande du client.

Technologie : Arduino
Bibliothèques : LittleFS, ESP8266WiFI, ESP8266mDNS, EEPROM, ESP8266WebServer, SparkFunBME280

2. IDE

VSCode et platformIO

3. Fonctionnalités

- **Côté client**
  - **Zone de texte pour les messages**
  - **Acquisition des mesures et commande de l'actionneur (AJAX + fonctions de rappel)**
    - Utilisation élaborée d'AJAX (fonction globale $.ajax).
    - Affichage de messages indiquant la réception des mesures et la bonne réception des commandes de l'actionneur (fonctions de rappel).
    - Affichage de messages d'erreur indiquant la non-communication entre le client et le serveur (fonctions de rappel).
    - Indication de la perte de la connexion (NC) dans la zone d'affichage des mesures.
- **Côté serveur**
  - **Intégration du site dans un ESP8266**
    - Gestion de la zone mémoire pour les fichiers avec la bibliothèque *LittleFS*
    - Création d'un serveur HTTP avec la bibliothèque *ESP8266WebServer*

    PB : temps de chargement du CSS Solution : à masquer avec un loader (à faire)
  - **Transmission simultanée des mesures**
    - Codage en JSON
  - **Traitement de la perte de connexion avec le capteur I2C (à faire)**
  - **Coupure de l'alimentation du chauffage en cas de perte de la connexion avec le navigateur (à faire)**
  - **Mesure du courant dans le dispositif de chauffage (à faire)**
  - **Régulation TOR de la température dans la serre (à faire)**

4. Traitements à réaliser par le client et/ou le serveur

		États		Traitement spécifique au
Communication	Wifi	Connecté	Non connecté	Client
	I2C			Client et serveur
Actionneur	Envoi commande	On	Off	Client
	Fonctionnement	Actif	Inactif	Serveur

5. Le code

❖ Côté client

➢ **Affichage des messages** : displayMessage()

responseError	heater	Message	N°
0	0	Mesures en cours, dernière cde chauffage = off	0
0	1	Pas de réception, dernière cde chauffage = on	1
1	0	Mesures en cours, dernière cde chauffage = off	2
1	1	Pas de réception, dernière cde chauffage = on	3

Remarque : responseError et heater sont des variables globales initialisées à 0.

```
Algorithme displayMessage
Variable
state : entier

début
    state ← responseError*2¹ + heater*2⁰

    selon (state)
      0 : Afficher(message0) ; fin selon
      1 : Afficher(message1) ; fin selon
      2 : Afficher(message2) ; fin selon
      3 : Afficher(message3) ; fin selon
      Autrement : Afficher("Erreur d'affichage") ; fin selon
    fin selon
fin
```

➢ **Chargement des mesures** : loadMeasures() // Fonction appelée toutes les n secondes

```
Algorithme loadMeasures
Variables
 température ← " ", humidité ← " " : chaînes de caractères

début
    Envoyer une requête à l'ESP8266 pour obtenir la valeur de la température et de l'humidité
    La demande a abouti // SUCCES
    début
            température ← température reçue
            humidité ← humidité reçu
            si responseError = 1 alors
              Effacer message
              responseError ← 0
            fin si
            Afficher la température et l'humidité
            Appeler displayMessage
    fin

    La demande a échoué, après timeout (timeout<n) // ERROR
    début
            responseError ← 1
            Appeler displayMessage
    fin
fin
```

- **Envoi des commandes** : gestionnaire d'évènement déclenché par un **clic de souris**

Algorithme Gestionnaire
début
Envoyer une requête à l'ESP8266 contenant la commande (on ou off) du relais
La demande a abouti et on la reçoit dans la réponse // SUCCES
début
si réponse = 'on'
alors
heater ← 1
Appeler displayMessage
sinon si réponse = 'off'
heater ← 0
Appeler displayMessage
fin si
fin

La demande a échoué, après timeout (timeout<n) // ERROR
début
responseError ← 1
Appeler displayMessage
fin
fin

❖ Côté serveur

Algorithme
// Initialisation
Initialiser les E/S de la carte
Se connecter au Wifi
Démarrer le gestionnaire de fichiers
Créer les gestionnaires de requête avec les paramètres (chemin, fonction de gestion)
Démarrer le serveur HTTP

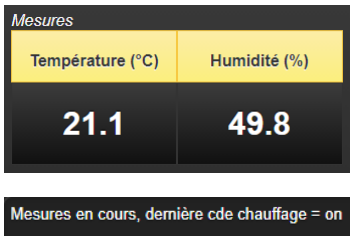
Répéter (toujours)
début
Attendre et gérer les requêtes
fin

Fonction gestion_Requête1()
début
…
fin

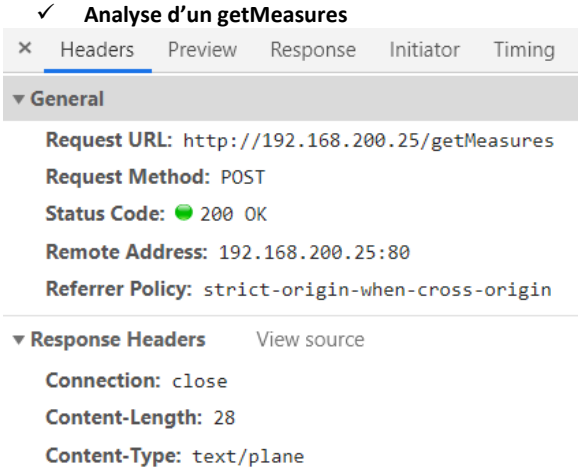
6. Tests avec les outils du navigateur

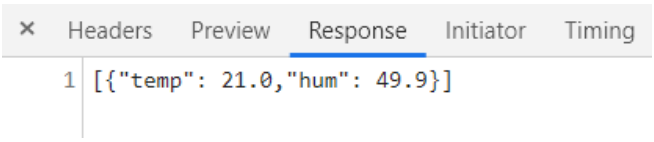
❖ Connexion à la carte ESP8266

getMeasures	200	xhr	VM17:1	112 B	161 ms
setHeater	200	xhr	VM17:1	85 B	103 ms
getMeasures	200	xhr	VM17:1	112 B	2.12 s
getMeasures	200	xhr	VM17:1	112 B	170 ms

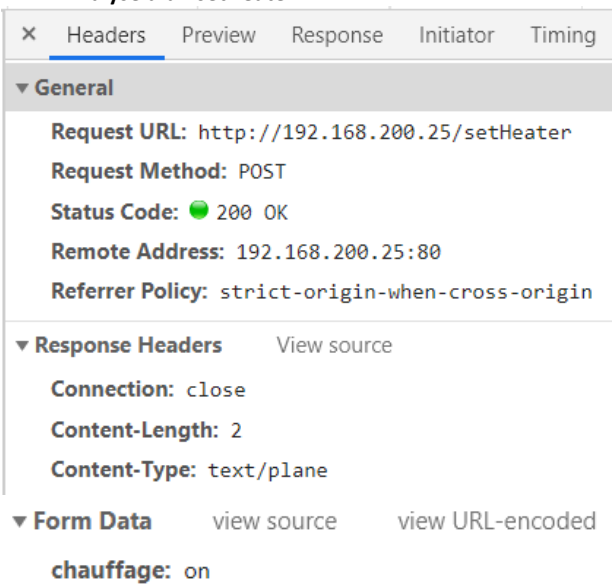


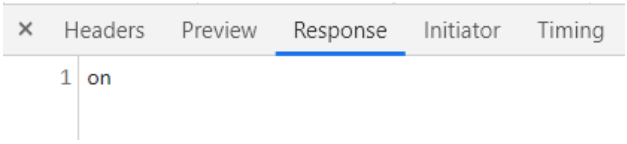
✓ **Analyse d'un getMeasures**





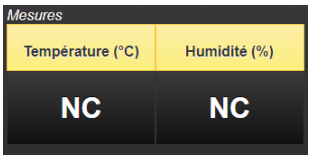
✓ **Analyse d'un setHeater**





❖ Perte de la connexion

Le navigateur (Chrome) respecte les 4s réglées dans le code avant d'annuler (canceled) la requête (1). Après quatre tentatives il considère la requête perdue (failed) après 1s (2).



getMeasures	200	xhr	VM17:1	112 B	119 ms
getMeasures	200	xhr	VM17:1	112 B	519 ms
getMeasures	(canceled)	xhr	VM17:1	0 B	4.01 s
getMeasures	(canceled)	xhr	VM17:1	0 B	4.02 s
getMeasures	(canceled)	xhr	VM17:1	0 B	4.01 s
getMeasures	(canceled)	xhr	VM17:1	0 B	4.01 s
getMeasures	(failed)	xhr	VM17:1	0 B	1.04 s
getMeasures	(failed)	xhr	VM17:1	0 B	1.04 s
getMeasures	(failed)	xhr	VM17:1	0 B	1.05 s

(1)

(2)

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>ESP8266 - Démo Mini Serre</title>
  <link rel="stylesheet" href="jquery.mobile-1.2.0.min.css">
</head>

<body>
  <div data-role="page" data-theme="a">
    <div data-role="header">
      <h1>ESP8266 - SERRE</h1>
    </div>

    <div data-role="content">
      <div class="ui-grid-a">
        <div class="ui-block-a" style="text-align:left">
          <h4>Projet 2021</h4>
          <h5>Thème : culture en serre</h5>
          <h5>Public : STI2D SIN</h5>
        </div>
        <div class="ui-block-b" style="text-align:center;">
          <img src="plante.png">
        </div>
      </div>
      <div><strong>SIN </strong>(<strong>S</strong>ystème d'<strong>I</strong>nformation et
        <strong>N</strong>umérique)
      </div>
      <br>
      <label><em>Mesures</em></label>
      <div class="ui-grid-a" style="text-align:center;">
        <div class="ui-block-a">
          <div class="ui-bar-e" style="height:50px;">
            <h4>Température (°C)</h4>(°C)
          </div>
        </div>
        <div class="ui-block-b">
          <div class="ui-bar-e" style="height:50px;">
            <h4>Humidité (%)</h4>
          </div>
        </div>
        <div class="ui-block-a">
          <div class="ui-bar-a" style="height:80px;">
            <h1 id="temperature">NC</h1>
          </div>
        </div>
        <div class="ui-block-b">
          <div class="ui-bar-a" style="height:80px;">
            <h1 id="humidite">NC</h1>
          </div>
        </div>
      </div>
      <br>
      <label><em>Chauffage</em></label>
      <form action="/CdeChauffage" method="POST">
        <input type="button" name="chauffage" value="on" id="on">
        <input type="button" name="chauffage" value="off" id="off">
      </form>
      <textarea id="messages"></textarea>
    </div>

    <div data-role="footer" data-theme="a">
      <h4>Lycée Pierre Emile Martin - BOURGES</h4>
    </div>
  </div>
```

```
<script type="text/javascript" src="jquery-1.8.2.min.js"></script>
<script type="text/javascript" src="jquery.mobile-1.2.0.min.js"></script>
<script type="text/javascript">
  var responseError = false;
  var heater = false;
  // var heaterError = false;//TODO: avec vérification com I2C
  var msgComError = "Pas de réception";
  var msgMeasures = "Mesures en cours";
  // var msgMeasuresError = ""; //TODO: avec vérification com I2C
  var msgHeaterOn = ", dernière cde chauffage = on";
  var msgHeaterOff = ", dernière cde chauffage = off";
  // var msgHeaterError = ",erreur chauffage"; // TODO: avec mesure courant Chauffage

  $('#messages').empty();

  $('input').click(function () {
    $.ajax({
      url: '/setHeater',
      type: 'post',
      data: $(this).attr('name') + '=' + $(this).attr('value'),
      success: function (data) {
        if (data == 'on') {
          heater = true;
          displayMessage();
        } else if (data == 'off') {
          heater = false;
          displayMessage();
        } else {
          //heaterError = true;
          //TODO: avec mesure courant Chauffage
        }
      },
      timeout: 4000,
      error: function () {
        responseError = true;
        displayMessage();
      }
    });
    return false;
  });

  function loadMeasures() {
    var temperature = '', humidite = '';

    $.ajax({
      url: '/getMeasures',
      type: 'post',
      dataType: 'json',
      success: function (data) {
        $.each(data, function (entryIndex, entry) {
          temperature = entry.temp;
          humidite = entry.hum;
        });
        if (responseError) { // Utilisé lors d'une reconnexion
          $('#messages').empty();
          responseError = false;
        }
        $("#temperature").html(temperature);
        $("#humidite").html(humidite);
        displayMessage();
      },
      timeout: 4000,
      error: function () {
        responseError = true;
        displayMessage();
      }
    });
  };
```

```
    function displayMessage() {
      let state = responseError ? 2 : 0;
      state = heater ? state + 1 : state;

      switch (state) {
        case 0:
          $('#messages').html(msgMeasures + msgHeaterOff).css('font-size','small');
          break;
        case 1:
          $('#messages').html(msgMeasures + msgHeaterOn).css('font-size','small');
          break;
        case 2:
          $('#messages').html(msgComError + msgHeaterOff).css('font-size','small');
          $("#temperature").html('NC');
          $("#humidite").html('NC');
          break;
        case 3:
          $('#messages').html(msgComError + msgHeaterOn).css('font-size','small');
          $("#temperature").html('NC');
          $("#humidite").html('NC');
          break;
        default:
          $('#messages').html("Erreur affichage").css('font-size','small');
          break;
      }
      return;
    }

    loadMeasures(); // premiere execution

    setInterval(function () {
      loadMeasures()
    }, 5000);

  </script>
</body>

</html>
```

```
/*-------------------------------------------------------------------------
Titre : ESP8266 : LittleFS + jQuery
// le : 13/5/2021
// IDE : VSCode + PlatformIO
// Fichier : main.cpp
// Contrôle de la température dans une serre et mesure de l'humidité
-------------------------------------------------------------------------*/
// Placé par PlatformIO
#include <Arduino.h>
// Système de fichiers
#include <LittleFS.h>
// Connexion au wifi
#include <ESP8266WiFi.h>
// mDNS pour la résolution des noms des hôtes
#include <ESP8266mDNS.h>
//  EEPROM : émule une EEPROM dans l'ESP8266
#include <EEPROM.h>
// Serveur HTTP
#include <ESP8266WebServer.h>
// Capteur BME280 SparkFun : température, humidité, pression
#include <SparkFunBME280.h>

#define Led LED_BUILTIN
#define RelaisChauffage 16 // D5 de la carte Particle

// Variables globales
int Temperature = 0;
int Humidite = 0;
// -------------------------------------------------------------------------
// Structure pour la configuration de la connexion au réseau wifi
struct EEconf
{                 // Les champs sont remplis avec les données préalablement placées en EEPROM émulée
                  // en utilisant le croquis infoClientMQTT_ESP8266.ino
  char ssid[32];      // SSID du réseau. Exemple : SynBoxLAN,
  char password[64];   // Mot de passe du réseau. Exemple : 12345678
  char myhostname[32]; // Nom donné au client MQTT. Exemple : ESP8266_1
} readconf;

// Objet pour la connexion au réseau wifi
WiFiClient espClient;
// Création d'un serveur HTTP
ESP8266WebServer server(80);
// Objet capteur BME280
BME280 sparkfunBME280;
// =============================================================================
// Fonction de connexion au Wifi
// -------------------------------------------------------------------------
void setup_wifi()
{
  // Mode station
  WiFi.mode(WIFI_STA);
  Serial.println();
  Serial.print("Tentative de connexion à ");
  Serial.println(readconf.ssid);
  // Connexion au Wifi
  WiFi.begin(readconf.ssid, readconf.password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(5000);
    Serial.print(".");
  }
  // Affichage
  Serial.println("");
  Serial.println("Connexion au Wifi ok");
  Serial.print("MAC: ");
  Serial.println(WiFi.macAddress());
  Serial.print("Adresse IP : ");
  Serial.println(WiFi.localIP());
```

```
// Configuration de mDNS

  WiFi.hostname(readconf.myhostname);
  if (!MDNS.begin(readconf.myhostname))
  {
    Serial.println("Erreur de configuration mDNS !");
  }
  else
  {
    Serial.println("Répondeur mDNS démarré");
    Serial.println(readconf.myhostname);
  }
}
// ================================================================================
// Fonctions déclenchées en réponse aux requêtes HTTP
// --------------------------------------------------------------------------------
void handle404Error()
{
  String message = "ERREUR 404 \n Fichier inconnu de LittleFS \n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += " \n";
  server.send(404, "text/plain,message", message);
  Serial.println(message);
}
// Gestion des I/O
// --------------------------------------------------------------------------------
void handleHeater()
{
  String Commande = server.arg("chauffage"); // On recupére la valeur de l'argument
  Serial.println(Commande);
  if (Commande == "on")
  {
    digitalWrite(Led, LOW);
    digitalWrite(RelaisChauffage, HIGH);
  }
  else
  {
    digitalWrite(Led, HIGH);
    digitalWrite(RelaisChauffage, LOW);
  }
  server.send(200, "text/plane", Commande);
}

void handleMeasures()
{
  char msgTemp[16], msgHum[16];
  float temperature = sparkfunBME280.readTempC();
  dtostrf(temperature, 5, 1, msgTemp);
  float humidite = sparkfunBME280.readFloatHumidity();
  dtostrf(humidite, 5, 1, msgHum);
  String msginfos = "[{\"temp\":";
  msginfos += msgTemp;
  msginfos += ",";
  msginfos += "\"hum\":";
  msginfos += msgHum;
  msginfos += "}]";
  Serial.println(msginfos);
  server.send(200, "text/plane", msginfos);
}
```

```
// ================================================================================
void setup()
{
  // Configuation des I/O
  pinMode(Led, OUTPUT);
  digitalWrite(Led, HIGH); // Actif à l'état bas
  pinMode(RelaisChauffage, OUTPUT);
  digitalWrite(RelaisChauffage, LOW); // Actif à l'état haut

  // Configuration du moniteur série
  Serial.begin(115200);
  delay(500);

  // Lecture des paramètres sauvegardés par ARD_ESP_SauveInfosClientMqtt.ino
  EEPROM.begin(sizeof(readconf));
  EEPROM.get(0, readconf);

  // Connexion au Wifi
  setup_wifi();

  // Initialisation du bus I2C (la bibliothèque BME280 sparkfun ne le fait pas !)
  Wire.begin();
  // Test de la communication avec le capteur BME280
  if (sparkfunBME280.beginI2C() == false) // Début de la communication sur l'I2C
  {
    Serial.println("Le capteur ne répond pas");
    while (1)
      ; //TODO: A remplacer par chargement de la page et test des capteurs
  }

  // Démarrage du système de fichiers
  if (!LittleFS.begin())
  {
    Serial.println("Erreur initialisation LittleFS");
  }

  // Gestion des requêtes et démarrage du serveur HTTP
  server.serveStatic("/", LittleFS, "/index.html");
  server.serveStatic("/images/ajax-loader.gif",LittleFS,"/images/ajax-loader.gif");
  server.serveStatic("/index.html", LittleFS, "/index.html");
  server.serveStatic("/jquery.mobile-1.2.0.min.css", LittleFS, "/jquery.mobile-1.2.0.min.css");
  server.serveStatic("/jquery-1.8.2.min.js", LittleFS, "/jquery-1.8.2.min.js");
  server.serveStatic("/jquery.mobile-1.2.0.min.js", LittleFS, "/jquery.mobile-1.2.0.min.js");
  server.serveStatic("/plante.png", LittleFS, "/plante.png");
  server.on("/setHeater", handleHeater);    // gère la LED_BUILTIN de la carte
  server.on("/getMeasures", handleMeasures); // gère la mesure et l'envoi des grandeurs physiques
  server.onNotFound(handle404Error);        // page non trouvée
  server.begin();
}

void loop()
{
  // Gestion serveur HTTP
  server.handleClient();
}
```