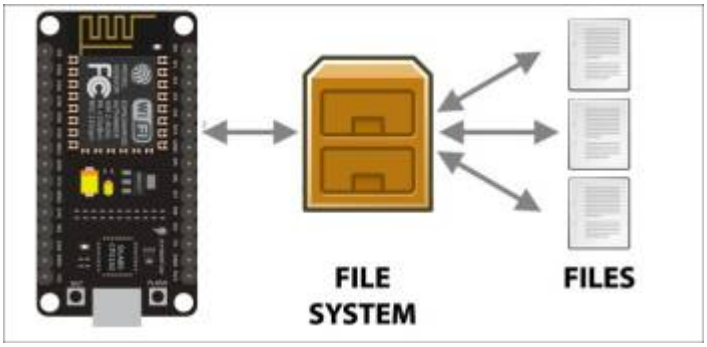
ESP - Le système de fichiers LittleFS

[Mise à jour le 18/8/2023]



- **Sources**
  - Le système de fichiers d'un ESP8266
- **Lectures connexes**
  - Wiki La carte ESP8266 Feather Huzzah
  - Utilisation de la mémoire SPIFFS d'un ESP8266
  - ESP8266. Lire, écrire, modifier des fichiers SPIFFS avec la librairie FS.h
  - ESP32. Débuter avec la librairie SPIFFS.h pour lire, écrire, modifier des fichiers.

1. Le système de fichier LittleFS (évolution de SPIFFS)

- **Sources**
  - Le système de fichiers d'un ESP8266

1.1 Généralités

Le système de fichiers **SPIFFS** (sous licence MIT) a été conçu par **Peter Andersson** pour les plateformes légères disposant de peu de mémoire vive. C'est un système de fichiers très simple, **ne supportant pas l'utilisation de répertoires**, mais proposant une interface de programmation assez similaire à celle de GNU/Linux (POSIX). Une astuce existe, car le caractère “/” est utilisable dans un nom de fichier.

On pourra donc écrire : **/img/logo.png** comme nom de fichier pour simuler un fichier logo.png situé dans un répertoire img.

> **Chemin de fichier**
>
> Le chemin de fichier doit toujours commencer par le caractère '/'.La **taille maximum** d'un nom de fichier est **31** (32 moins le caractère de fin de chaîne du C/C++ '\0'.

Le système de fichiers **LittleFS** (utilisé dans l'IDE PlatformIO) est une amélioration de **SPIFFS**. Les

différences existantes entre SPIFFS et LittleFS sont décrites dans la rubrique "Système de fichiers d'un ESP8266".

1.2 La flash de l'ESP8266

Sur un ESP8266, même si le système de fichiers est stocké sur la même puce flash que le programme, **la programmation d'un nouveau croquis ne modifiera pas le contenu du système de fichiers**. Le système de fichiers peut ainsi stocker des données, des fichiers de configuration ou du contenu pour le serveur Web.

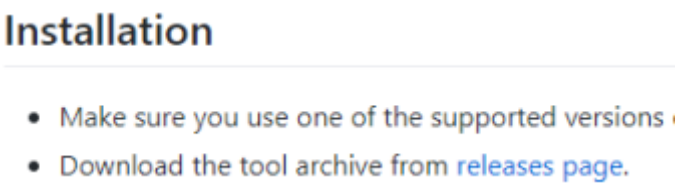
```
|--------------|-------|---------------|--|--|--|--|--|
^              ^       ^               ^     ^
Sketch    OTA update   File system   EEPROM  WiFi config (SDK)
```

2. Préparation des IDE

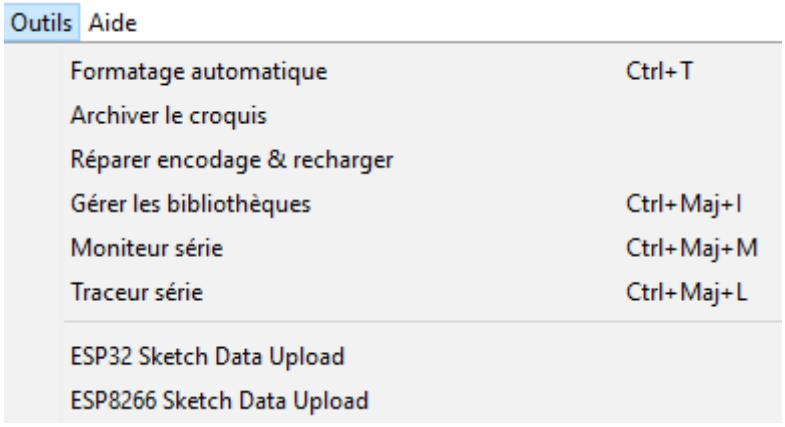
2.1 Arduino

> **Téléchargement**
> Le **téléchargement de fichiers** à partir du PC dans la zone mémoire du système de fichiers des ESP8266 et ESP32 nécessite l'installation de plugins dans l'éditeur Arduino.

- **Téléchargement**
  - Les plugins sont téléchargeables sous la forme de **fichiers .zip** via le lien *Releases* de la page Github : pour l'ESP8266, pour l'ESP32.



- **Installation**
  - **Dézipper** les fichiers dans le répertoire **../Arduino/tools**. A l'ouverture de l'IDE Arduino, les outils doivent apparaître dans le menu **Outils** comme ci-dessous.



2.2 PlatformIO dans VSCode

Le système de fichiers SPIFFS étant obsolète, PlatfomIO utilise littleFS.

- **Configuration** : ajouter le code ci-dessous dans **platformio.ini**.

```
board_build.filesystem = littlefs
monitor_speed = 115200
```

3. Contenu de la zone mémoire File System

Les bibliothèques **FS.h** et **LittleFS.h** fournissent les fonctionnalités nécessaires à la mise en oeuvre du système de fichiers. Le code du croquis **infos** ci-dessous affiche l'état du stockage de la zone mémoire du système de fichier d'un ESP8266 et les éventuels fichiers présents.

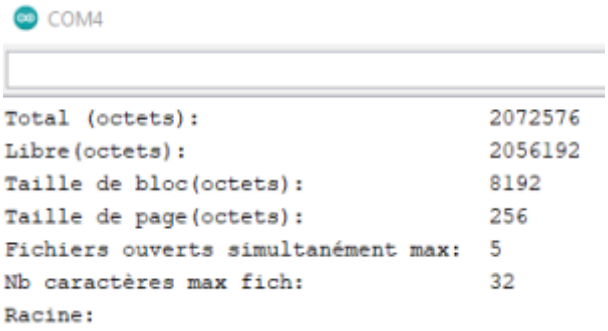
> **Zone mémoire**
> La taille de la zone mémoire réservée à SPIFFS est configurable sur un ESP8266, ESP32 en fonction de la carte sélectionnée. Voir ici.

infos.cpp

```
// IDE : Arduino
// Ressource "Le système de fichiers d'un ESP8266"
// https://arduino-esp8266.readthedocs.io/en/latest/filesystem.html
#include <FS.h>
#include <LittleFS.h>

void setup() {
  FSInfo fs_info;
```

```
  Serial.begin(115200);
  delay(500);

  if (!LittleFS.begin()) {
    Serial.println("Une erreur est apparue lors du montage de
LittleFS");
  }
  else {
    //LittleFS.format(); // RAZ zone SPI
    LittleFS.info(fs_info);
    Serial.println();
    Serial.print("Total (octets):                    ");
    Serial.println(fs_info.totalBytes);

    Serial.print("Libre(octets):                     ");
    Serial.println(fs_info.totalBytes - fs_info.usedBytes);

    Serial.print("Taille de bloc(octets):            ");
    Serial.println(fs_info.blockSize);

    Serial.print("Taille de page(octets):            ");
    Serial.println(fs_info.pageSize);

    Serial.print("Fichiers ouverts simultanément max:  ");
    Serial.println(fs_info.maxOpenFiles);

    Serial.print("Nb caractères max fich:            ");
    Serial.println(fs_info.maxPathLength);
  }

  Serial.println("Racine:");
  Dir dir = LittleFS.openDir("/");
  while (dir.next()) {
    Serial.print(dir.fileName());
    Serial.print("\t\t");
    Serial.println(dir.fileSize());
  }
}

void loop() {
  delay(100);
}
```

Exemple de résultat obtenu dans le moniteur série si la zone mémoire du système de fichiers est vide.
Remarque : ouvrir le moniteur série dans l'IDE Arduino avant de télécharger le fichier.



PlatformIO

Télécharger le projet PlatformIO pour VSCode. Le SSID et le mot de passe du réseau doivent avoir été préalablement chargés dans l'eeprom émulée avec ce croquis.

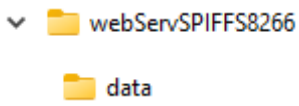
4. Serveur Web statique

Le serveur se compose :

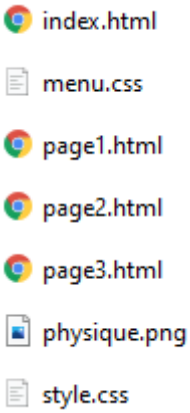
1. Des fichiers du site (HTML, CSS, JavaScript)
2. Du code C++ (croquis Arduino ou main.cpp dans platformIO)

4.1 Fichiers du site

Pour être téléchargés dans la zone mémoire du système de fichiers, les fichiers doivent être placés dans un sous-répertoire **data** du répertoire du croquis Arduino ou du projet PlatformIO.



- **Téléchargement à partir du croquis Arduino**
  Cliquer sur ESP8266 Sketch Data Upload dans le menu **Outils** de l'IDE Arduino pour que les fichiers situés dans *data* soient transférés dans la mémoire du système de fichiers.



Exemple

```
SPIFFS Image Uploaded
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Compressed 2072576 bytes to 43761...
Wrote 2072576 bytes (43761 compressed) at 0x00200000 in 3.9 seconds
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

- Le contenu de la mémoire peut être vérifié en utilisant le croquis **infos**
  Exemple

```
Racine:
/index.html             1203
/menu.css               278
/page1.html             1213
/page2.html             1207
/page3.html             1211
/physique.png           36178
/style.css              1736
```

- **Téléchargement à partir du projet platformIO (VSCode)**
  Cliquer sur **Upload Filesystem Image** pour que les fichiers situés dans *data* soient transférés dans la mémoire du système de fichiers.

4.2 Algorithme

```
Algorithme serveurHTTP
// Initialisations
```

```
Se connecter au point d'accès wifi
Démarrer le système de fichiers
Créer les gestionnaires de requête avec les paramètres : chemin et nom de la
fonction de gestion

Répéter (toujours)
   début
      Attendre une requête
   fin
```

4.3 Code

- **Croquis Arduino**

serveurHTTP.cpp

```
// IDE Arduino
//
https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WebServ
er
// Bibliothèques
#include <FS.h>
#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>
// ----------------------------------------------------------------------
-------------
// 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);
```

```
// Connexion au réseau 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);
    }
}

// Initialisation
void setup() {
   // Configuration du moniteur série
    Serial.begin(115200); // pour les tests
    delay(500);

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

    // Connexion au Wifi
    setup_wifi();
```

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

  // Gestionnaires de requête
  // (Les requêtes sont automatiquement gérées par la bibliothèque
ESP8266WebServer)
  server.serveStatic("/", LittleFS, "/index.html");
  server.serveStatic("/index.html", LittleFS, "/index.html");
  server.serveStatic("/page1.html", LittleFS, "/page1.html");
  server.serveStatic("/page2.html", LittleFS, "/page2.html");
  server.serveStatic("/page3.html", LittleFS, "/page3.html");
  server.serveStatic("/physique.png", LittleFSS, "/physique.png");
  server.serveStatic("/menu.css", LittleFS, "/menu.css");
  server.serveStatic("/style.css", LittleFS, "/style.css");
  server.begin();
}

void loop() {
  // Attente et gestion des requêtes
  server.handleClient();
}
```

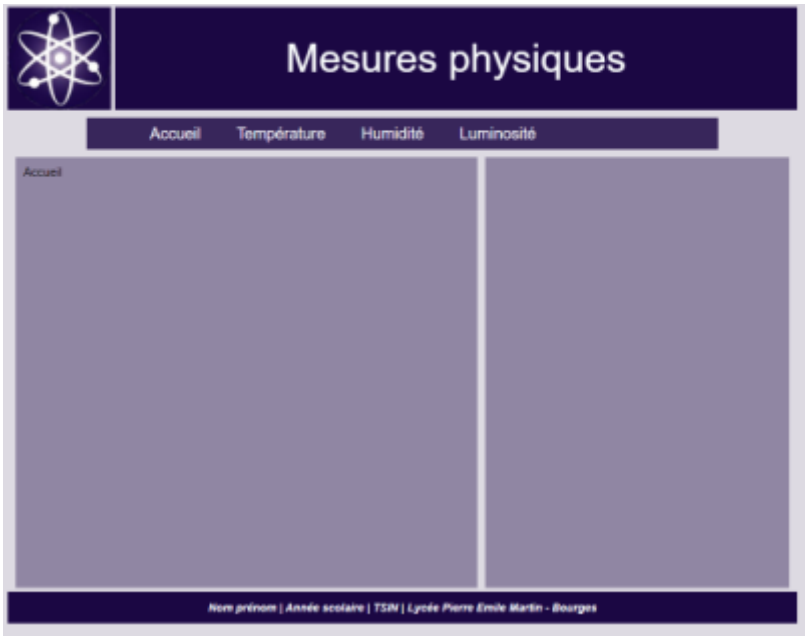
- **Projet PlatformIO**

PlatformIO

Télécharger le projet PlatformIO pour VSCode. Le téléchargement des fichiers du site, situés dans data, se fait avec **Upload Filesystem Image**. Le SSID et le mot de passe du réseau doivent avoir été préalablement chargés dans l'eeprom émulée avec ce croquis.

4.4 Exemple de résultat obtenu

Serveur et site intégrés à un ESP8266.



5. Pour aller plus loin

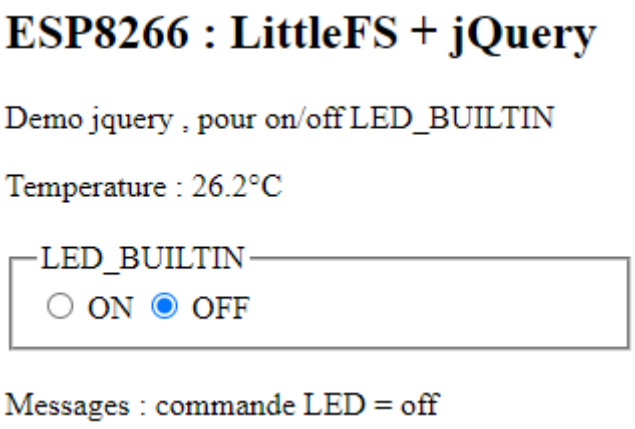
- **Ressources**
  - Utilisation de la mémoire SPIFFS d'un ESP8266

5.1 Afficher le contenu d'un fichier

5.2 Écrire dans un fichier

5.3 Interface Web dynamique (Mini Serre)

- **Exemple 1**
  - Matériels : ESP8266, BME280
  - Langages : HTML, JavaScript (JQuery), Arduino
  - Description : les pages du site sont placées dans la zone mémoire LittleFS, le serveur HTTP envoie la valeur de la température, mesurée par le BME280, en réponses aux requêtes du navigateur et commande la Led de la carte lors d'un clic sur les boutons radio.

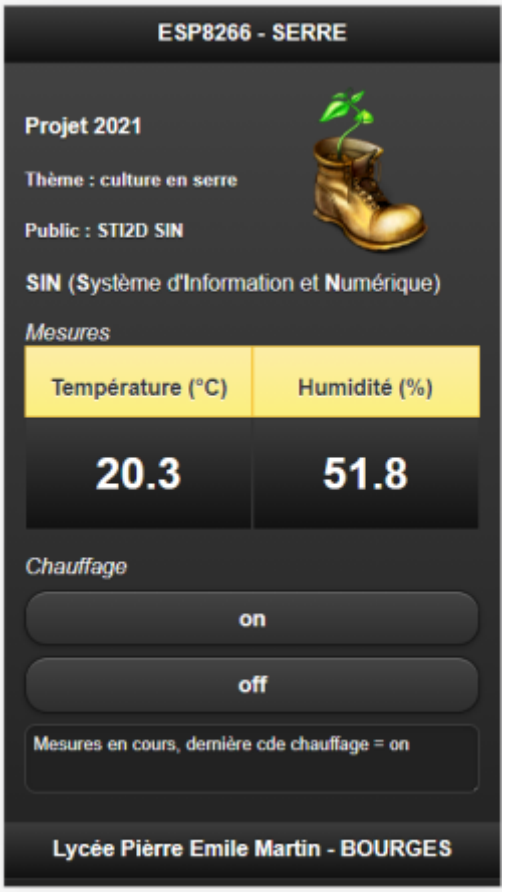


PlatformIO

Télécharger le projet PlatformIO pour VSCode. Le téléchargement des fichiers du site, situés dans data, se fait avec **Upload Filesystem Image**. Le SSID et le mot de passe du réseau doivent avoir été préalablement chargés dans l'eeprom émulée avec ce croquis.

- **Exemple 2** : Mini Serre

- Matériels : ESP8266, BME280, relais
- Langages : HTML, JavaScript (JQuery, JQuery Mobile), Arduino
- Description : à télécharger



PlatformIO

Télécharger le projet PlatformIO pour VSCode. Le téléchargement des fichiers du site, situés dans data, se fait avec **Upload Filesystem Image**. Le SSID et le mot de passe du réseau doivent avoir été préalablement chargés dans l'eeprom émulée avec ce croquis.

5.4 Serveur FTP