

The background of the slide features a silhouette of several communication towers against a sunset sky. The towers are of various heights and designs, some with multiple antenna arrays. Thin lines representing cables or guy wires connect the towers to the ground. The sky transitions from a deep orange near the horizon to a dark blue at the top. A semi-transparent white rectangle is centered over the image, containing the title text.

ESP-32 – Serveur web et système de fichiers

Objectifs

- Répondre à des requêtes HTTP avec l'ESP32
- Transférer une structure de fichiers sur l'ESP32
- Manipuler des fichiers avec LittleFS

Répondre à une requête HTTP

- La plateforme fournit la classe `WebServer` qui permet :
 - D'écouter sur un port
 - De lancer des fonctions à la réception d'un couple verbe / nom de ressource
- Pour cela :
 - Créez un objet de type `WebServer` (port TCP par défaut = 80)
 - Enregistrez des ressources statiques avec la méthode « `serveStatic` »
 - Enregistrez des ressources dynamiques et des verbes avec la méthode « `on` »
 - Enregistrez une fonction de réponse pour les pages non trouvées
 - Appelez la méthode « `begin` »

WebServer – Classe pour ranger le code

```
class ServeurWeb {  
public:  
    ServeurWeb();  
    void tick();  
  
private:  
    WebServer* m_webServer;  
  
    void afficherRacine();  
  
    void ajouterFichiersStatiques(String const& p_debutNomFichier);  
    void ajouterFichiersStatiques(String const& p_debutNomFichier,  
    | | | | | | | | | | | | | | | | | | File& p_fichier);  
    |  
    void ressourceNonTrouvee(String const& p_nomFichier);  
  
    void allumer(void);  
    void eteindre(void);  
};
```

WebServer – Classe pour ranger le code

```
ServeurWeb::ServeurWeb() {
    LittleFS.begin(true);

    pinMode(2, OUTPUT);

    this->m_webServer = new WebServer();

    this->ajouterFichiersStatiques("/");

    this->m_webServer->on("/", HTTPMethod::HTTP_GET,
| | | | | | | | | | | | [this]() { this->afficherRacine(); });
    this->m_webServer->on("/allumer", HTTPMethod::HTTP_GET,
| | | | | | | | | | | | [this]() { this->allumer(); });
    this->m_webServer->on("/eteindre", HTTPMethod::HTTP_GET,
| | | | | | | | | | | | [this]() { this->eteindre(); });
    this->m_webServer->onNotFound(
| | [this]() { this->ressourceNonTrouvee(this->m_webServer->uri()); });
    this->m_webServer->begin();
}
```

WebServer – Classe pour ranger le code

```
void ServeurWeb::afficherRacine() {  
    Serial.println("Réception requête");  
    Serial.println(this->m_webServer->uri());  
  
    this->m_webServer->sendHeader("Location", "index.html", true);  
    this->m_webServer->send(301, "text/plain", "");  
}
```

```
void ServeurWeb::allumer() {  
    digitalWrite(2, HIGH);  
    this->m_webServer->send(200, "text/plain", "DEL allumée");  
}
```

WebServer – Classe pour ranger le code

- Dès que possible lancer la méthode « handleClient », dans le cas de la classe, c'est à chaque appel de loop

```
void ServeurWeb::tick() { this->m_webServer->handleClient(); }
```

SPIFFS – Ancien système de fichier

- SPIFFS (SPI Flash File System) : système de fichiers léger adapté aux MCUs qui disposent d'une mémoire flash
- Ne gère pas les répertoires
- ~~Les répertoires sont simulés : on ajout un « / » dans le nom du fichier~~ Ne fonctionne plus dans les versions actuelles
- Le nom du fichier ne doit pas dépasser 31 caractères
- Le nom d'un fichier débute toujours par « / »
- Ne pas mettre d'espaces ou d'accents dans les noms de fichiers ou de « répertoire »

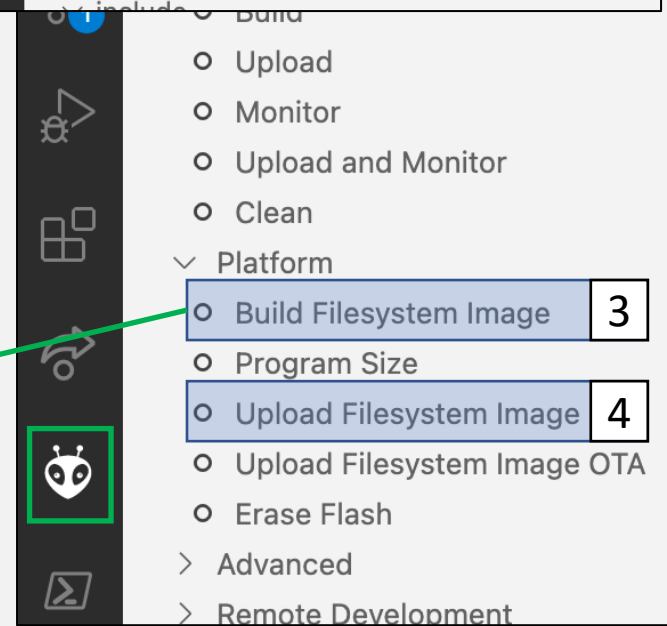
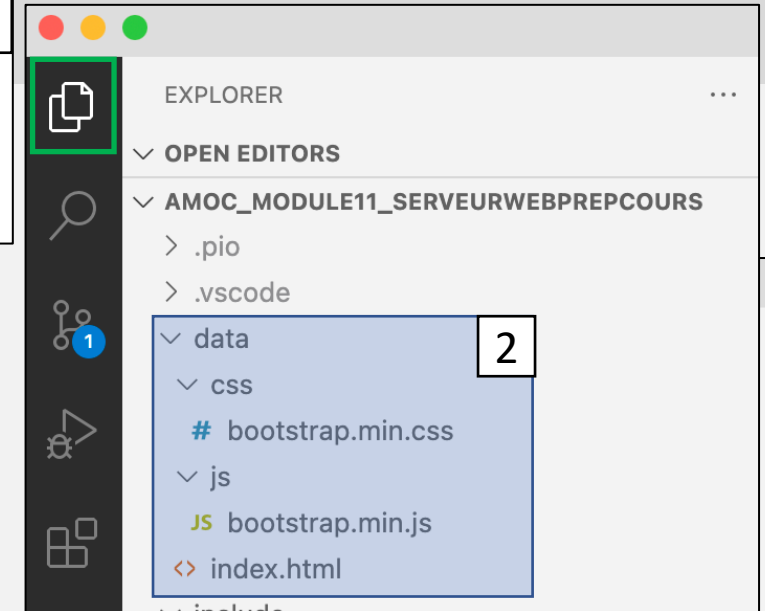
LittleFS – Système de fichiers conseillé

- LittleFS : comme SPIFFS, système de fichiers léger adapté aux MCUs qui disposent d'une mémoire flash
- **Contient des répertoires**
- Le nom d'un fichier débute toujours par « / »
- Optimise l'utilisation de la Flash (nombre d'écriture par adresse)

LittleFS – Téléversement d'une image

- Pour créer une image de disque et la téléverser sur l'ESP32 :
 - Ajoutez « board_build.filesystem = littlefs » dans la section de configuration de votre carte dans le fichier « platformio.ini » (1)
 - Copiez vos fichiers dans le répertoire « data » de votre projet (le créer s'il n'existe pas !) (2)
 - Créez l'image (3)
 - Téléverser l'image dans votre ESP32 (attention cela écrase ce qui est pré-existant) (4)
 - En cas de problèmes, vous pouvez utiliser la fonction « Erase Flash » et recommencer le processus

```
[env:ESP32:LittleFS-WebServer]
platform = espressif32
board = esp32doit-devkit-v1
framework = arduino
monitor_speed = 115200
board_build.filesystem = littlefs
```



```
Building FS image from 'data' directory to .pio/build/ESP32:LittleFS-WebServer/littlefs.bin
/index.html
/test/subdir/README.md
/test/subdir/README.txt
/css/bootstrap.min.css
/js/bootstrap.min.js
```

LittleFS

- Pour utiliser LittleFS :

- Inclure le fichier d'entête

```
#include "LittleFS.h"
```

- Appeler la méthode « begin »

```
LittleFS.begin([true]);
```

- La méthode « begin » prends un booléen en paramètres. Par défaut, il est à faux. Il sert à indiquer si vous voulez formater la partition en cas de problème.

- Principales méthodes de LittleFS :

```
File open(const char* path, const char* mode = FILE_READ);  
File open(const String& path, const char* mode = FILE_READ);  
  
bool exists(const char* path);  
bool exists(const String& path);
```

LittleFS - File

- Principales méthodes de File :

```
size_t write(uint8_t) override;  
size_t write(const uint8_t *buf, size_t size) override;  
void flush() override;
```

```
int read() override;  
size_t read(uint8_t* buf, size_t size);  
size_t readBytes(char *buffer, size_t length);
```

```
bool seek(uint32_t pos);  
size_t position() const;  
size_t size() const;  
void close();  
const char* name() const;
```

```
boolean isDirectory(void);  
File openNextFile(const char* mode = FILE_READ);
```


LittleFS – Lister les fichiers et les ajouter en ressources statiques

```
void ServeurWeb::ajouterFichiersStatiques(String const& p_debutNomFichier) {  
    File racine = LittleFS.open("/");  
    ajouterFichiersStatiques(p_debutNomFichier, "", racine);  
}
```

```
void ServeurWeb::ajouterFichiersStatiques(String const& p_debutNomFichier,  
                                         String const& p_repertoireCourant,  
                                         File& p_repertoire) {  
  
    if (!p_repertoire) return;  
  
    Serial.println(String("Traitement du répertoire : ") + p_repertoire.name());  
  
    File fichier = p_repertoire.openNextFile();  
    while (fichier) {  
        String nomFichier = p_repertoireCourant + "/" + String(fichier.name());  
        if (fichier.isDirectory()) {  
            ajouterFichiersStatiques(p_debutNomFichier, p_repertoireCourant + "/" + fichier.name(), fichier);  
        } else {  
            if (nomFichier.startsWith(p_debutNomFichier)) {  
                Serial.println(String("Ajout du fichier statique : ") + nomFichier + " pour l'URI " + nomFichier);  
                this->m_webServer->serveStatic(nomFichier.c_str(), LittleFS,  
                                              nomFichier.c_str());  
            }  
        }  
        fichier.close();  
        fichier = p_repertoire.openNextFile();  
    }  
  
    p_repertoire.close();  
}
```

Références

- <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/spiffs.html>
- <https://github.com/espressif/arduino-esp32/tree/master/libraries/WebServer>