**Filesystem**

**Flash layout (компонування пам’яті)**

Хоча файлова система зберігається на тому ж флеш-чіпі, що і програма, завантаження нового скетчу не впливає на вміст файлової системи. Це дозволяє використовувати файлову систему для зберігання даних, файлів налаштування або вмісту для веб-сервера.

Наступна діаграма ілюструє флеш-макет, що використовується в середовищі Arduino:

**|--------------|-------|---------------|--|--|--|--|--|**

**^** **^** **^** **^** **^**

Sketch OTA update File system EEPROM WiFi config (SDK)

Розмір файлової системи залежить від розміру флеш-чіпу. Залежно від плати, вибраної в IDE, у вас є такі параметри для розподілу флеш-пам’яті:

| **Board** | **Flash chip size, bytes** | **File system size, bytes** |
| --- | --- | --- |
| Generic module | 512k | 64k, 128k |
| Generic module | 1M | 64k, 128k, 256k, 512k |
| Generic module | 2M | 1M |
| Generic module | 4M | 3M |
| Adafruit HUZZAH | 4M | 1M, 3M |
| ESPresso Lite 1.0 | 4M | 1M, 3M |
| ESPresso Lite 2.0 | 4M | 1M, 3M |
| NodeMCU 0.9 | 4M | 1M, 3M |
| NodeMCU 1.0 | 4M | 1M, 3M |
| Olimex MOD-WIFI-ESP8266(-DEV) | 2M | 1M |
| SparkFun Thing | 512k | 64k |
| SweetPea ESP-210 | 4M | 1M, 3M |
| WeMos D1 & D1 mini | 4M | 1M, 3M |
| ESPDuino | 4M | 1M, 3M |

**Примітка**: для використання будь-яких функцій файлової системи в програмі додайте до скетчу наступний рядок:

*#include "FS.h"*

**File system limitations (Обмеження файлової системи)**

Реалізація файлової системи для ESP8266 повинна була враховувати обмеження мікросхеми, таких як обмежена оперативна пам'ять. Система [SPIFFS](https://github.com/pellepl/spiffs) була обрана, оскільки вона призначена для невеликих систем, такий вибір призвів до деяких спрощень та обмежень.

По-перше, позаду сцен, SPIFFS не підтримує каталоги, він просто зберігає "плоский" список файлів. Проте, всупереч традиційним файловим системам, в назвах файлів дозволено символ "/" у вигляді косої риски, тому функції, що стосуються списку каталогів (наприклад, openDir ("/ website")), в основному просто фільтрують імена файлів і зберігають ті, які починаються з запитуваного префікса (/website /). Практично кажучи, це мало як відрізняється.

По-друге, для назв файлів загалом є 32 символи. Символ '\ 0' зарезервований для завершення рядка C, тому нам лишається 31 символ, які можна використати.

Назви файлів рекомендується зберігати коротко і не використовувати багато вкладень в каталозі, оскільки повний шлях кожного файлу (включаючи каталоги, символи '/', базове ім'я, точка і розширення) має бути максимально 31 символом. Наприклад, назва файлу /website/images/bird\_thumbnail.jpg складається з 34 символів і це може призвести до деяких проблем. Наприклад, з командою – exists() або у випадку, якщо існує інший файл який починається з тих самих перших 31 символів.

**Warning**: ця межа легко досягається, і якщо її проігнорувати, проблеми можуть залишатися непоміченими, оскільки при складанні або в режимі виконання не з'являється повідомлення про помилку.

Щоб отримати додаткові відомості про внутрішні реалізації SPIFFS, перегляньте файл [readme SPIFFS](https://github.com/esp8266/Arduino/blob/master/cores/esp8266/spiffs/README.md).

**Uploading files to file system**

*ESP8266FS* is a tool which integrates into the Arduino IDE. It adds a menu item to *Tools* menu for uploading the contents of sketch data directory into ESP8266 flash file system.

ESP8266FS є інструментом, який інтегрується в IDE Arduino. Він додає елемент меню до меню Інструменти для завантаження файлів, що містяться в каталозі скетчу, до файлової системи ESP8266.

* Завантажте інструмент: <https://github.com/esp8266/arduino-esp8266fs-plugin/releases/download/0.3.0/ESP8266FS-0.3.0.zip>.
* У каталозі Sketchbook Arduino створіть директорію tools (якщо вона ще не існує)
* Розпакуйте інструмент у каталозі tools (шлях виглядатиме як <home\_dir>/Arduino/tools/ESP8266FS/tool/esp8266fs.jar)
* Перезавантаєте Arduino IDE
* Відкрийте скетч (або створіть новий і збережіть його)
* Перейдіть до каталогу програми (Ctrl+K)
* Створіть там каталог з назвою data і додайте в нього будь-які файли, які вам треба помістити в файлову систему
* Переконайтеся, що ви обрали плату, порт і закрили Serial Monitor
* Виберіть *Інструменти> ESP8266 Sketch Data Upload*. Почнеться завантаження файлів у flash файлової системи ESP8266. Після завершення, інформаційна панель IDE повідомить про завантаження даних в SPIFFS.

**File system object (SPIFFS)**

**begin**

SPIFFS**.**begin()

Цей метод монтує файлову систему SPIFFS. Цю команду потрібно викликати до використання будь-яких інших API-команд. Повертає *true*, якщо файлову систему успішно встановлено, інакше - *false*.

**end**

SPIFFS**.**end()

Цей метод розмонтовує файлову систему SPIFFS. Використовуйте цей метод перед оновленням SPIFFS, використовуючи OTA.

**format**

SPIFFS**.**format()

Formats the file system. May be called either before or after calling begin. Returns *true* if formatting was successful.

Форматування файлової системи. Може бути викликаний як до, так і після виклику функції – begin(). Повертає *true*, якщо форматування було успішним.

**open**

SPIFFS**.**open(path, mode)

Opens a file. path should be an absolute path starting with a slash (e.g. /dir/filename.txt). mode is a string specifying access mode. It can be one of “r”, “w”, “a”, “r+”, “w+”, “a+”. Meaning of these modes is the same as for fopen C function.

Відкриває файл. Змінна – path повинен бути абсолютним шляхом, починаючи з косої риски (наприклад, /dir/filename.txt). mode - це рядок, що визначає режим доступу, який може бути: "r", "w", "a", "r +", "w +", "a +". Значення цих режимів таке ж, як для функції fopen C.

r Open text file **for** reading**.** The stream **is** positioned at the

beginning of the file**.**

r Відкрити текстовий файл для читання. Курсор розміщений на

       початок файлу

r**+** Open **for** reading **and** writing**.** The stream **is** positioned at the

beginning of the file**.**

r+ Відкрити для читання та письма. Курсор розміщений на

       початок файлу

w Truncate file to zero length **or** create text file **for** writing**.**

The stream **is** positioned at the beginning of the file**.**

w Обрізати файл до нульової довжини або створити текстовий файл для написання.

       Курсор розміщений на початку файлу.

w**+** Open **for** reading **and** writing**.** The file **is** created **if** it does

**not** exist, otherwise it **is** truncated**.** The stream **is**

positioned at the beginning of the file**.**

w+ Відкритий для читання та письма. Файл створюється, якщо він є

       не існує, інакше він зрізається. Курсор розміщений на

       початку файлу.

a Відкрити для додавання (написання в кінці файлу). Файл

       створюється, якщо його не існує. Курсор розміщений на

       кінець файлу.

a Open **for** appending (writing at end of file)**.** The file **is**

created **if** it does **not** exist**.** The stream **is** positioned at the

end of the file**.**

a+ Відкрито для читання та додавання (написання в кінці файлу).

       Файл створюється, якщо його не існує. Курсор для читання

       знаходиться на початку файлу, а для письма на кінець файлу.

a**+** Open **for** reading **and** appending (writing at end of file)**.** The

file **is** created **if** it does **not** exist**.** The initial file

position **for** reading **is** at the beginning of the file, but

output **is** always appended to the end of the file**.**

Повертає *File*  об'єк. Щоб перевірити, чи файл успішно відкрито, використовуйте логічний оператор.

File f = SPIFFS.open("/f.txt", "w");

if (!f) {

Serial.println("file open failed");

}

**exists**

SPIFFS**.**exists(path)

Повертає *true*, якщо існує файл за даним шляхом, *false* - інакше.

**openDir**

SPIFFS**.**openDir(path)

Відкриває каталог з урахуванням його абсолютного шляху. Повертає об'єкт Dir.

**remove**

SPIFFS**.**remove(path)

Видаляє файл за його абсолютним шляхом. Повертає *true*, якщо файл був успішно видалений.

**rename**

SPIFFS**.**rename(pathFrom, pathTo)

Renames file from pathFrom to pathTo. Paths must be absolute. Returns *true* if file was renamed successfully.

Перейменовує файл з pathFrom до pathTo. Шляхи повинні бути абсолютними. Повертає true, якщо файл було успішно перейменовано.

**info**

FSInfo fs\_info;

SPIFFS**.**info(fs\_info);

Заповнює структуру *FSInfo (опис нижче)* інформацією про файлову систему. Повертає *true*, успішно, *false* в іншому випадку.

**Filesystem information structure**

struct FSInfo {

size\_t totalBytes;

size\_t usedBytes;

size\_t blockSize;

size\_t pageSize;

size\_t maxOpenFiles;

size\_t maxPathLength;

};

Це структура, яку можна заповнити методом *FS::info()*. totalBytes – загальний розмір корисних даних у файловій системі; usedBytes – кількість байтів, що використовуються файлами; blockSize – розмір блоку SPIFFS; pageSize – розмір логічної сторінки SPIFFS; maxOpenFiles – максимальна кількість файлів, які можуть бути відкриті одночасно; maxPathLength - максимальна довжина імені файлу (включаючи один байт для нульового закінчення)

**Directory object (Dir)**

The purpose of *Dir* object is to iterate over files inside a directory. It provides three methods: next(), fileName(), and openFile(mode).

Призначення об'єкта Dir - перебір файлів всередині каталогу. Для цього можна скористатися трьома способами: next(), fileName(), і openFile(mode).

The following example shows how it should be used:

Наступний приклад показує, як його слід використовувати:

Dir dir **=** SPIFFS**.**openDir("/data");

**while** (dir**.**next()) {

Serial**.**print(dir**.**fileName());

File f **=** dir**.**openFile("r");

Serial**.**println(f**.**size());

}

dir.next() повертає істинне значення, якщо у каталозі є файли для перебору. Це потрібно викликати перед викликом функцій fileName і openFile.

Метод openFile приймає аргумент *mode*, який має те ж значення, що і для функції SPIFFS.open.

**File object**

Функції SPIFFS.open та dir.openFile повертають об'єкт *File*. Цей об'єкт підтримує всі функції *Stream*, тому ви можете використовувати readBytes, findUntil, parseInt, println та всі інші методи *Stream* (потоку).

Існують також деякі функції, специфічні для об'єкта *File*.

**seek**

file**.**seek(offset, mode)

Ця функція поводиться як функція fseek C. Залежно від значення mode, він переміщує поточну позицію курсору у файлі таким чином:

* if mode is SeekSet, position is set to offset bytes from the beginning.
* if mode is SeekCur, current position is moved by offsetbytes.
* if mode is SeekEnd, position is set to offset bytes from the end of the file.
* якщо mode – SeekSet, позиція встановлюється на offset байтів з початку.
* якщо mode – SeekCur, поточне положення переміщається на offset байтів.
* якщо mode – SeekEnd, позиція встановлюється на offset байтів з кінця.

Повертає *true*, якщо позиція була успішно встановлена.

**position**

file**.**position()

Повертає поточну позицію всередині файлу в байтах.

**size**

file**.**size()

Повертає розмір файлу в байтах.

**name**

String name **=** file**.**name();

Повертає ім'я файлу, як const char\*. Перетворюючи його в *String* для зберігання.

**close**

file**.**close()

Закрийте файл. Жодна інша операція не повинна виконуватися в *Fileobject* після виклику функції close.

Список посилань: http://arduino-esp8266.readthedocs.io/en/latest/filesystem.html