**ESP32 SPI flash memory files management**

**J. Lemaire**

Pierrefeu – November 2019

# Objectives

Working on a small project using an ESP32 (on an HELTEC Wifi Kit 32 board)[[1]](#footnote-2) in view to provide an assistance in trekking activity, I incountered the problem of managing files stored in an external SPI flash memory (4MB here). In fact, I wanted to upload .gpx files (GPS traces) in this memory.

There are many options to perform this :

* Use the **uploader file system plugin to the Arduino IDE**[[2]](#footnote-3), as explained for instance in the Random Nerd tutorial « Install ESP32 Filesystem Uploader in Arduino IDE »[[3]](#footnote-4). Here, just need to prepare a **data** subfolder in the sketch folder and to select the « **ESP32 Sketch Data Upload** » menu item. Very easy for a developper but not for a basic user and a little drawback : the previous content of the flash memory is completely replaced by the content of the **data** folder.
* In the same direction, there are direct tools from Espressif, but not equally for a basic user : **spiffsgen.py**, **spiffs\_creaye\_partition\_image**, **mkspiffs**, etc. [[4]](#footnote-5)
* Create a web server on the ESP32 and just use an internet browser on the client side. G6EJD presents a project[[5]](#footnote-6) [[6]](#footnote-7) in this direction using an SD Card and an ESP8266, but it’s relatively easy to adapt it to an ESP32[[7]](#footnote-8) and the SPI Flash Memory using the SPIFFS library[[8]](#footnote-9). « Easy » but need a lot of work nevertheless…
* Implement an FTP server on the ESP32 and use an FTP client program on the client side. Fortunately all the job has been already done here and there is an operational and beautiful (code very clear and compact) solution from David Paiva (nailbuster), working with an ESP32 and the SPI Flash memory[[9]](#footnote-10). It was in inspired by a Jean-Michel Gallego’s work for an Arduino Mega2560 or Due plus an Ethernet shield[[10]](#footnote-11) and adapted to both ESP8266 and ESP32, using only the **WiFi**, **WiFiClient**, **SPIFFS** and **FS** libraries for the ESP32[[11]](#footnote-12).

Initially I thought that using an FTP Client program may be difficult for a non specialist user, but finally I changed my opinion, specially using client programs such as **WinSCP**[[12]](#footnote-13) (Windows) or **AndFTP**[[13]](#footnote-14) (Android). So I retained the fourth solution with just very little improvments :

* Only code for ESP32
* Accept anonymous connections (no password then required)
* Access Point mode for the WiFi server

Here I would like to show how to use it.

# The SPIFFS library[[14]](#footnote-15)

Basic library added on installing the ESP43 boards support in Arduino IDE, it’s the key enabling the access to the SPI flash memory, not only to read a file stored in this memory as well explained in this Random Nerd tutorial « ESP32 Web Server using SPIFFS »[[15]](#footnote-16), but also to manage completely the set of these files (read, write, append, delete, etc.), because it defines an **SPIFFSFS** class which inherits frome the powerfull **FS** class.

This inheritance explains why accessing to an SPI SD card or accessing to the SPI flash memory are very similar, but there are differences. For instance, just an important restriction here : it produces only a flat structure with **no folders**. So need to understand that an accepted filename like « /myFolder/myFile.txt » is only a name for a file ! But these pseudo « paths » can be used in filters.

# The WiFi library[[16]](#footnote-17)

Basic libray again on installing ESP43 boards support in Arduino IDE, it defines many classes used here :

* **WiFiClass** and its **WiFi** instance to activate the Wifi system
* **WiFiAPClass** to set an access point with a DHCP, using the **softAP** method
* **WiFiServer** to enable a TCP server
* Etc.

Contrary to the ESP8266, I don’t have found systematic references on these classes. But the Random Nerd tutorials « ESP32 Web Server – Arduino IDE »[[17]](#footnote-18) and « How to Set an ESP32 Access Point for Web Server »[[18]](#footnote-19) are good introductions to understand the creation of a Station or an Access Point Web Server, using only the **WiFiServer** class.

# The ESP32FtpServer library

As previously announced, it’s just the David Paiva esp8266FTPServer[[19]](#footnote-20) library in which the ESP8266 specific code has been erased (to simplify) and with a little modification in the **FtpServer::userPassword()** method for accepting **anonymous** connections (in this case, the password does’nt matter) :

**else if ( strcmp( parameters, \_FTP\_PASS.c\_str() ))**

replaced by :

**else if ( strcmp( "anonymous", \_FTP\_USER.c\_str() ) && strcmp( parameters, \_FTP\_PASS.c\_str() ))**

Renamed **ESP32FtpServer**, this library defines an **FtpServer** class with only 2 public methods :

* **void** **begin(String uname, String pword);**
* **void** **handleFTP();**

In plus of a very clear code, the beauty of the David Paiva’s work consists in the real simplicity to activate an FTP server connected to the SPI flash memory. For instance, to create such a server accepting anonymous connections under a Wifi server running in Access Point mode, just need to use the following Arduino sketch :

// FTPServerSample.ino

#include "ESP32FtpServer.h"

// For Wifi Access Point

const char**\*** ssid **=** "ftpESP32"**;**

const char**\*** passwordWifi **=** "123456789"**;**

// For Ftp server

const char**\*** username **=** "anonymous"**;**

const char**\*** passwordFtp **=** ""**;** // Doesn't matter when "anonymous"

FtpServer ftpSrv**;**

void setup**(**void**)** **{**

Serial**.**begin**(**115200**);**

WiFi**.**softAP**(**ssid**,** passwordWifi**);**

Serial**.**println**(**WiFi**.**softAPIP**());**

**if** **(**SPIFFS**.**begin**(true))** **{**

Serial**.**println**(**"SPIFFS opened!"**);**

ftpSrv**.**begin**(**username**,** passwordFtp**);**

**}**

**}**

void loop**(**void**)** **{**

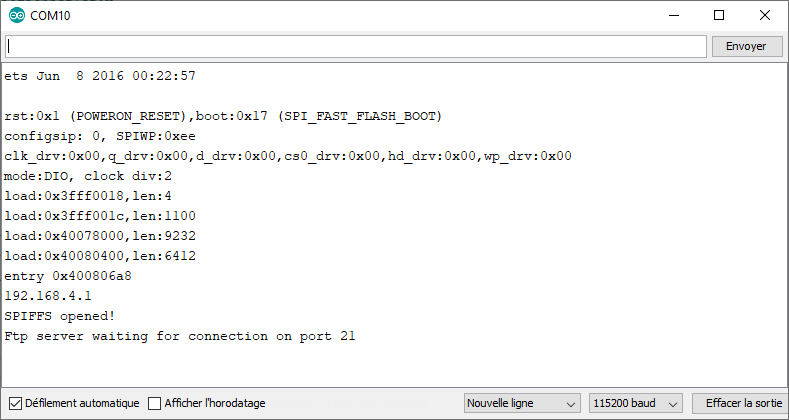
ftpSrv**.**handleFTP**();**

**}**

So, for the programmer, using this library is a very simple option !

**Remarks**

* Note the IP address of the server displayed when starting : **192.168.4.1** here



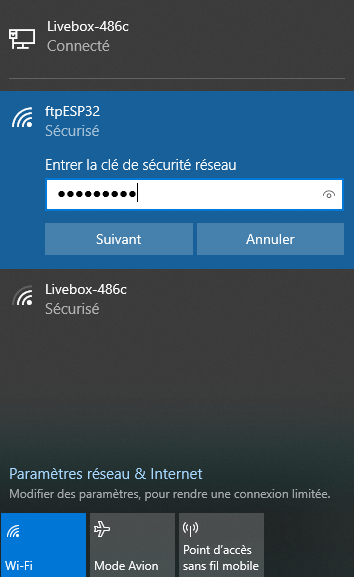
* Obviously the David Pavia’s code enables the creation of a **simple** FTP server (but here completey sufficient), opening 2 channels, one for the commands (port 21) and the other for the data (port 50009). But with these limitations : no security, no encryption, only one connection at a time, passive mode for the client. Moreover, because there is no DNS entry for this server, the client has to precise its IP address (192.168.4.1 then)[[20]](#footnote-21)

# The client side

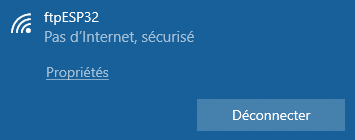
Is it really simple for a basic hiker (non ESP32 specialist !) to load GPS traces (.gps files) in the SPI flash memory ? I think so if he uses a simple FTP client program : just need to understand what is necessary to perform and especially the previous restrictions.

Below, an example under Windows 10 on a PC, using the **WinSCP**[[21]](#footnote-22) program, but the job is the same on a smartphone, for instance with the **AndFTP**[[22]](#footnote-23) application under Android.

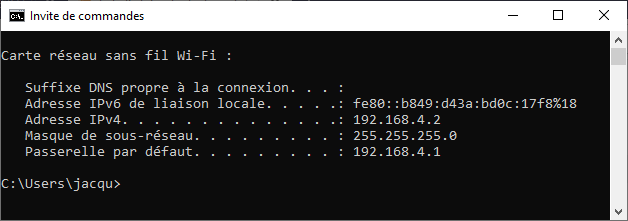
1. Add a Wifi connection to the **ESP32** Wifi Access Point, with **« ftpESP32 »** as key and « **123456789** » as password :



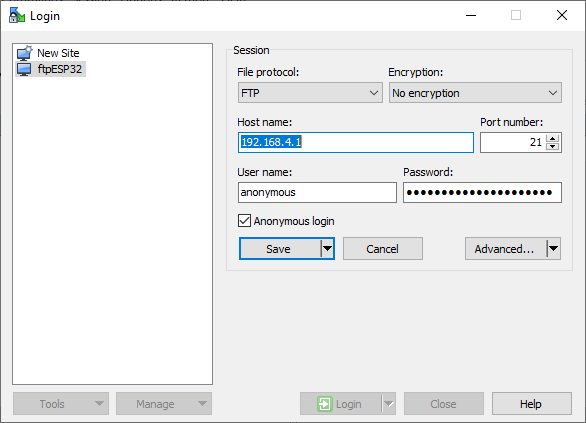
When accepted by the ftpESP32 server, a connection is established with the server :



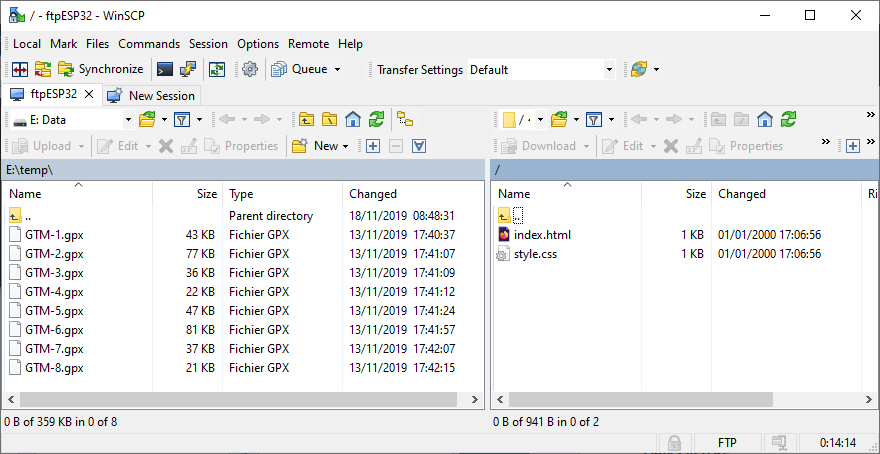
Using the **IPConfig** command, it can be observed that we are effectively a station in a new network :



1. Run **WinSCP**, create a new site with these parameters and save it.

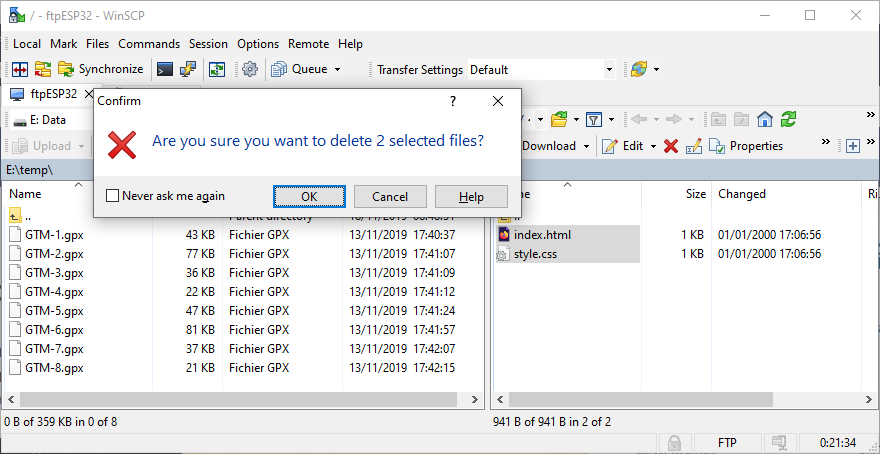


Then, clicking on Login, a connection with the ftp server is established :

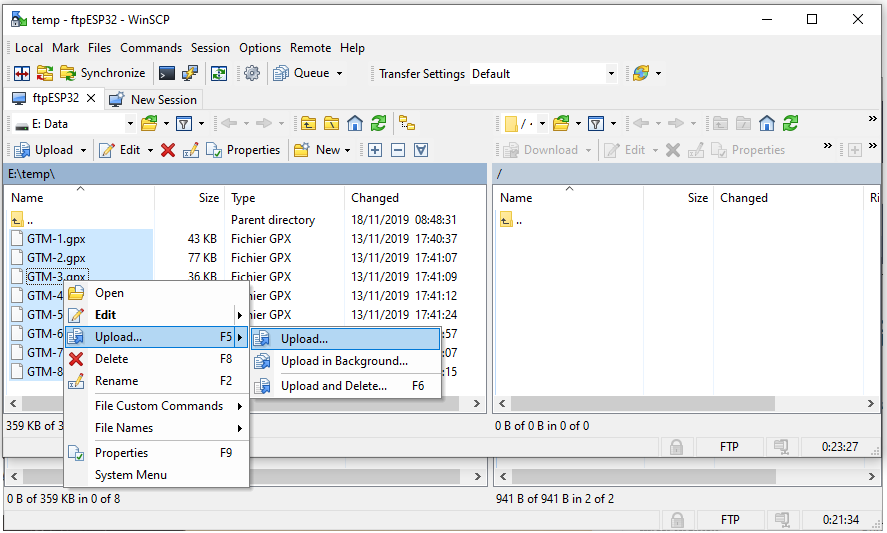


As usual with these programs, one can see both a local folder on the left (with **.gpx** files here) and the distant folder on the right, which is exactly the content of the SPI flash memory in the Random Nerd tutorial « ESP32 Web Server using SPIFFS »[[23]](#footnote-24).

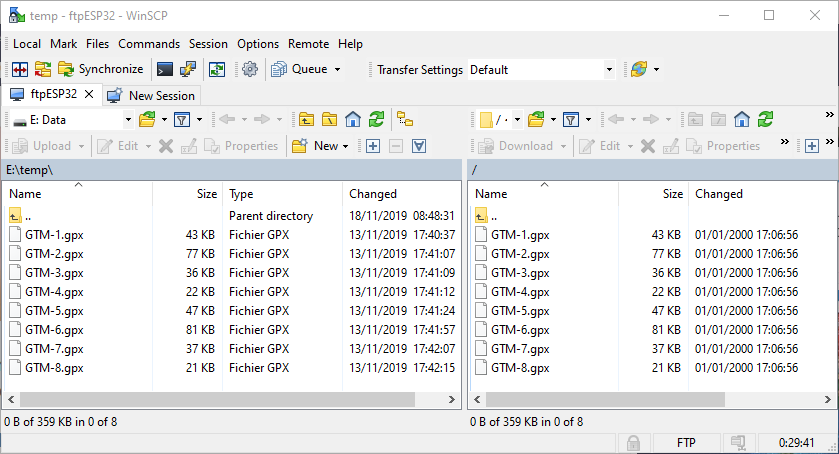
To delete the **index.htm** and **style.css** files on the server, just need to select them an to click on **Suppr** :



And then, to copy the **.gpx** files in the SPI flash memory, just need to select them again and to chose the **Upload** command :



The desired result in fine :



1. Disconnection

Just need to close WinSCP !

# Conclusion

This method appears very usefull and very simple to use both for the programmer and the user : a big thanks to Jean-Michel Gallego and David Paiva !

1. <https://heltec.org/project/wifi-kit-32/> [↑](#footnote-ref-2)
2. <https://github.com/me-no-dev/arduino-esp32fs-plugin/releases/tag/1.0> [↑](#footnote-ref-3)
3. <https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/> [↑](#footnote-ref-4)
4. <https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/storage/spiffs.html> [↑](#footnote-ref-5)
5. <https://github.com/G6EJD/ESP32-ESP8266-File-Download-Upload-Delete-Stream-and-Directory> [↑](#footnote-ref-6)
6. <https://www.youtube.com/watch?v=u6vmUIPU1i4> [↑](#footnote-ref-7)
7. Using the included **WebServer** library in place of the **ESP8266WebServer** library. [↑](#footnote-ref-8)
8. <https://github.com/G6EJD/SPIFFS-Examples/blob/master/SPIFFS.pdf> [↑](#footnote-ref-9)
9. <https://github.com/nailbuster/esp8266FTPServer> [↑](#footnote-ref-10)
10. <https://github.com/gallegojm/Arduino-Ftp-Server/tree/master/FtpServer> [↑](#footnote-ref-11)
11. They are all included in the Arduino IDE when installing the Espressif ESP32 support. [↑](#footnote-ref-12)
12. <https://winscp.net/eng/index.php> [↑](#footnote-ref-13)
13. <http://www.lysesoft.com/products/andftp/> [↑](#footnote-ref-14)
14. <https://github.com/espressif/arduino-esp32/tree/master/libraries/SPIFFS> [↑](#footnote-ref-15)
15. <https://randomnerdtutorials.com/esp32-web-server-spiffs-spi-flash-file-system/> [↑](#footnote-ref-16)
16. <https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFi/> [↑](#footnote-ref-17)
17. <https://randomnerdtutorials.com/esp32-web-server-arduino-ide/> [↑](#footnote-ref-18)
18. <https://randomnerdtutorials.com/esp32-access-point-ap-web-server/> [↑](#footnote-ref-19)
19. <https://github.com/nailbuster/esp8266FTPServer> [↑](#footnote-ref-20)
20. With the the Arduino Core for ESP32, a **DNSServer** library is provided : it can be used to add here a DNS server if prefered [↑](#footnote-ref-21)
21. <https://winscp.net/eng/index.php> [↑](#footnote-ref-22)
22. <http://www.lysesoft.com/products/andftp/> [↑](#footnote-ref-23)
23. https://randomnerdtutorials.com/esp32-web-server-spiffs-spi-flash-file-system/ [↑](#footnote-ref-24)