

SERVIDOR WEB CON ARCHIVOS SEPARADOS

En esta práctica elaboraremos el mismo servidor web integrado en el anterior proyecto, pero con archivos HTML y CSS almacenados en el sistema de archivos. En lugar de tener que escribir HTML y CSS en el boceto de Arduino, crearemos archivos HTML y CSS separados.


TECNOLOGÍA SPIFFS (LITTLEFS EN ESP8266).

Los dispositivos ESP, disponen de una especie de disco de estado sólido en su interior, conocido por las siglas **SPIFFS** (del inglés *Serial Peripheral Interface File Flash System*), esto quiere decir que dentro del ESP se dispone de un espacio de memoria FLASH al que con ciertas condiciones podemos acceder para leer o escribir en él..

El ESP8266 es un dispositivo *SPIFF* denominado littleFS que fue diseñado para sistemas con *memoria SPI NOR flash* en dispositivos embebidos. El sistema de archivos tiene como objetivo principal, usar la menor memoria RAM posible.

INCLUSIÓN DE UN FAVICON

También almacenaremos un archivo de imagen para incluir un favicon en la página web. El favicon es el pequeño icono que aparece junto al título en la pestaña del navegador web.

Abrir Visual Code y esperar a que aparezca el ícono de casa . Pulsarlo y dará la bienvenida a Platformio.

Pulsar +New Project.

En Name: Asignar un nombre al nuevo proyecto, por ejemplo Platformio3.

En Board: Seleccionar WEMOS D1 R1.

Desactivar casilla Location y seleccionar la carpeta donde se guardará. Pulsar finish.

Esperar a que el proyecto se inicialice.

Aparecerá un menú vertical para Platformio3:

Seleccionar src y luego main.cpp en donde escribiremos el siguiente código:

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <LittleFS.h>

const char* ssid = "MEGACABLE-F823F";
const char* password = "v345FgMd";

// Crear objeto AsyncWebServer en el puerto 80
AsyncWebServer server(80);

void initFS() {
  if (!LittleFS.begin()) {
```

```

Serial.println("Un error ha ocurrido mientras se montaba LittleFS");
}
Serial.println("LittleFS montado exitosamente");
}

void initWiFi() { WiFi.mode(WIFI_STA); WiFi.begin(ssid, password);

Serial.print("Conectando a la red WiFi ..");
while (WiFi.status() != WL_CONNECTED) { Serial.print('.');
delay(1000);
} Serial.println(WiFi.localIP());
}

void setup() { Serial.begin(115200);
initWiFi();
initFS();

server.on("/", HTTP_GET, [](AsyncWebServerRequest *request)
{
request->send(LittleFS, "/index.html", "text/html");
});

/* Cuando recibe esa solicitud, envía el texto HTML guardado en el archivo index.html para construir
la página web. El primer argumento de la función send () es el sistema de archivos donde se guardan
los archivos, en este caso LittleFS para el ESP8266). El segundo argumento es la ruta donde se
encuentra el archivo. El tercer argumento se refiere al tipo de contenido (texto HTML).
Cuando el archivo HTML se carga en su navegador, solicitará el archivo CSS y el favicon. Estos son
archivos estáticos guardados en el mismo directorio (SPIFFS o LittleFS). */

server.serveStatic("/", LittleFS, "/");

server.begin();
}

void loop() {

}

```

Grabar con Files – Save o con Ctrl - S

El archivo Platformio.ini deberá contener lo siguiente (si algo falta, agregarlo):

```

[env:d1]
platform = espressif8266
board = d1
framework = arduino
monitor_speed = 115200
lib_deps = ESP Async WebServer
board_built.filesystem = littlefs

```

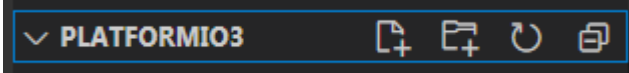
ARCHIVO HTML

Vamos a crear una carpeta con nombre data junto con las carpetas del proyecto:

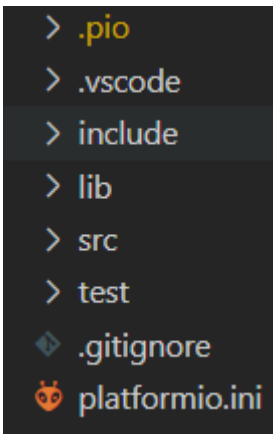


Selecciona el ícono de Explorer (generalmente está a la izquierda)

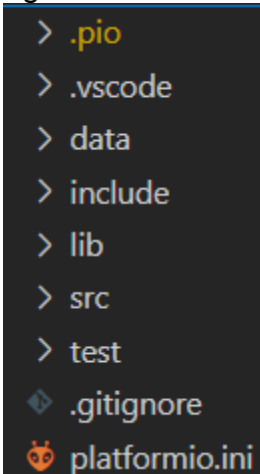
Selecciona Platformio3



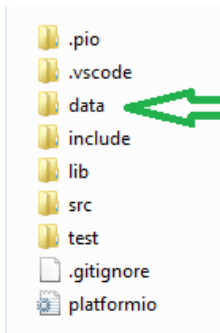
Asegurarse de que así estén las opciones (pulsa Esc si es necesario):

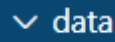


Luego selecciona New Folder asignar el nombre data y pulsar enter. Deben quedar de la siguiente manera:



Y en el explorador se observarán las subcarpetas así:



Seleccionar data  data y luego con clic derecho seleccionar New File


Teclear index.html Enter.

Teclear el siguiente código:

```
<!DOCTYPE html>
<html>
<head>
<title>Servidor Web en archivos</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" type="text/css" href="style.css">
<link rel="icon" type="image/png" href="favicon.png">
</head>
<body>
<h1>Servidor Web</h1>
<p>Curso de IoT<br>Este es el primer servidor Web con archivos separados.</p>
</body>
</html>
```

Guardar con File – Save o con Ctrl – S.

ARCHIVO CSS

Seleccionar data  data y luego con clic derecho seleccionar New File

Teclear style.css Enter.

Teclear el siguiente código:

```
html {
font-family: Arial;
text-align: center;
}
body {
max-width: 400px;
margin:0px auto;
}
```

CREACIÓN DEL FAVICON

En el siguiente sitio web, crear una imagen favicon.png en tamaño 16 por 16:

<https://www.ionos.mx/tools/crear-favicon>

Utilizando el explorador del sistema operativo, guardar la imagen favicon.png en la carpeta data.

Si compilamos con la opción  nos aparecerá el mensaje exitoso.

advanced Memory Usage is available via "PlatformIO Home > Project Inspect"

RAM: [====] 44.2% (used 36192 bytes from 81920 bytes)

Flash: [====] 41.9% (used 437192 bytes from 1044464 bytes)

==== [SUCCESS] Took 6.54 seconds
=====

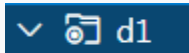
CARGA DE LA IMAGEN DEL SISTEMA DE ARCHIVOS

Finalmente, deben cargarse los archivos (index.html, style.css y favicon.png) al sistema de archivos:

1.- Haga clic en el icono PIO en la barra lateral izquierda. Las tareas del proyecto deberían abrirse.



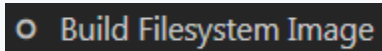
2. Seleccione la opción D1



3. Expanda el menú Plataforma.



4. Seleccione Construir imagen del sistema de archivos.



5. Luego, haga clic en Cargar imagen del sistema de archivos.



De esta forma se grabarán los archivos en la Wemos D1, apareciendo el mensaje exitoso:

Writing at 0x00300000... (100 %)

Wrote 1024000 bytes (2035 compressed) at 0x00300000 in 0.2 seconds (effective 42933.1 kbit/s)...

Hash of data verified.

Leaving...


Hard resetting via RTS pin...

```
===== [SUCCESS] Took 11.98 seconds
=====
```

Terminal will be reused by tasks, press any key to close it.

NOTA IMPORTANTE: Si aparece el siguiente error, la solución es desconectar y conectar de nuevo el cable USB que energiza y comunica a la Wemos D1.

```
open raise SerialException("could not open port {!r}: {!r}".format(self.portstr, ctypes.WinError()))
serial.serialutil.SerialException: could not open port 'COM3': PermissionError(13, 'Acceso denegado.',
None, 5)
*** [uploadfs] Error 1
```

6.- Por último grabar en el hardware  y esperar el mensaje exitoso

```
Writing at 0x00040000... (89 %)
Writing at 0x00044000... (94 %)
Writing at 0x00048000... (100 %)
Wrote 441344 bytes (298929 compressed) at 0x00000000 in 26.4 seconds (effective 133.6 kbit/s)...
Hash of data verified.
```

Leaving...

Hard resetting via RTS pin...

```
===== [SUCCESS] Took 35.17 seconds
=====
```

Terminal will be reused by tasks, press any key to close it.

Pulsar botón de Reset en Wemos D1 y abrir el ,monitor 

A continuación seleccionar la IP que deberá abrirse con el navegador, debe aparecer lo siguiente:

