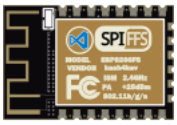


ESP8266FS

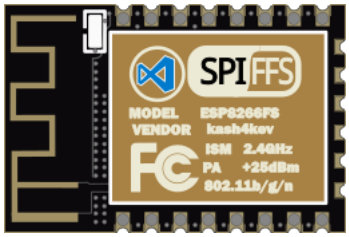
kash4kev | 27,300 installs | ★★★★★ (1) | Free

Visual Studio Code extension for ESP8266/ESP32 File System (SPIFFS)

Install Trouble Installing?

Visual Studio Code extension for ESP8266/ESP32 File System (SPIFFS)

> **Note:** This extension will be retired after this version and the code will be forked to "**vscode-espspiffs**" so as to better reflect the SPIFFS nature of this extension.



Welcome to the Visual Studio extension for the **ESP8266/ESP32 File System SPIFFS**.

This extension provides the same functionality for VSCode as the Arduino ESP8266 filesystem uploader and Arduino ESP32 filesystem uploader do for the Arduino IDE: it packages and uploads a BLOB to an ESP8266/ESP32 allowing the device to use a portion of it's Flash Memory as a Storage Device using SPIFFS (**SPI** **F**lash **F**ile **S**ystem).

Whereas the **Arduino IDE** versions adds menu items to the IDE (*Tools/ESP??? Sketch Data Updoad*), VSCode provides no such mechanism. Instead, this extension implements a VSCode command (`ESP8266FS: Upload SPIFFS`) to perform the same task.

This extension also adds commands to unpack, list, and "visualize" the contents of a **SPIFFS** image.

While this extension really doesn't need the **Arduino IDE** installed - it only needs the ESP8266 or ESP32 package and tools - it's best to have it installed anyway. This extension is meant to be a companion extension for the Arduino for Visual Studio Code plugin, which relies on the **Arduino IDE** to compile and upload code through their toolchain.

Features

- Works with or without the **Arduino for Visual Studio Code** plugin installed. Just needs **mkspiffs**, **esptool** or **esptool.py**, and **espota.py** (if using OTA updating).
- Implements VSCode commands to:
  - upload,
  - download,
  - pack,
  - unpack,
  - list,
  - and visualize a **SPIFFS** image.
- Uses settings from:
  - **.vscode/arduino.json**,
  - **settings.json**
  - or **...arduino.../preferences.txt**.
- Overrides available for all toolchain settings in the **settings.json** file.

> Tip: Add the "esp866fs.packSpiffs" and "esp866fs.uploadSpiffs" command to your **gulp/webpack** toolchain to turn your VSCode/ESP8266/ESP32 into a "one-button" dev cycle.

Categories

Other

Tags

Arduino ESP32 ESP8266 esptool SPIFFS

Resources

Issues
Repository
Homepage
License
Changelog
Download Extension

Project Details

kash4kev/vscode-esp8266fs
Last Commit: 3 years ago
2 Pull Requests
13 Open Issues

More Info

Version	1.1.0
Released on	2018/2/10下午10:53:17
Last updated	2018/5/23下午6:59:27
Publisher	kash4kev
Unique Identifier	kash4kev.vscode-esp8266fs
Report	Report Abuse

Requirements

The ESP8266 core for Arduino or ESP32 core for Ardiuno needs to be installed on your computer.

- For the ESP8266, it is best to use the Arduino IDE's Board Manager (*Tools/Board/Board Manager...*) or use the Arduino for Visual Studio Code's `Arduino: Board Manager` command.
- For the ESP32, follow the instructions in their **README.md** for the relevant OS.

If you manually install the package, you can still use this extension by setting the overrides.

ESP8266 vs. ESP32

Without getting in the differences between the two processors, their Arduino development environments are install in two different locations, through two different methods.

The toolchains for the two are similar, but slightly different. To upload files the **ESP8266** uses a compile tool called **esptool** and **ESP32** uses a python-based program called **esptool.py**, and the command-line arguments for the two are completely different.

Finally, the **boards.txt** found in both locations use different settings for the same item. The **EPS32** uses a "partitions" directory with CSV files to describe the memory layouts of the flash.

Chip	OS	Package Location
ESP8266	Windows	C:\Users\X\AppData\Local\Arduino15\esp8266\hardware\esp8266\2.4.1
	Mac	~/Library/Arduino15/packages/esp8266/hardware/esp8266/2.4.1
	Linux	~/.arduino15/packages/esp8266/hardware/esp8266/2.4.1
ESP32	Windows	C:\Users\X\Documents\Arduino\hardware\espressif\esp32
	Mac	~/Documents/Arduino/hardware/espressif/esp32
	Linux	~/Arduino/hardware/espressif/esp32

Installation

Open VS Code and press `F1` or `Ctrl+Shift+P` to open command palette, select **Install Extension** and type `vscode-esp8266fs`.

Or launch VS Code Quick Open (`Ctrl+P`), paste the following command, and press enter.

```
ext install vscode-esp8266fs
```

You can also install directly from the Marketplace within Visual Studio Code: search for `ESP8266FS`.

Getting Started

After installing this extension, you need to:

1. Create a new VSCode Project via the **Arduino for Visual Studio Code** extension (Command: `Arduino: Initialize`) or a new sketch with the **Arduino IDE** (`File/New`).
2. Add the URL **http://arduino.esp8266.com/stable/package_esp8266com_index.json** to the Additional Board URL settings.
3. Either:

   a. Install the **ESP8266** board from the Board Manager: (VSCode: `Arduino: Board Manager`, or IDE: `Tools/Board/Board Manager...`).

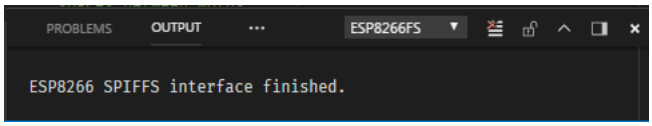
   b. Install the **ESP32** board using the instructions in their **README.md**.
4. Select an **ESP8266/ESP32** board as the target development board.
5. Select the desired **SPIFFS** program/storage split (Arduino: *Tools/Flash Size...*, VSCode: *arduino.changeBoardType*).

6. Create and populate a directory with the files to be uploaded to the target **ESP??? SPIFFS** partition - **`i.e. that will replace the current SPIFFS contents!`**
7. Set the `esp8266fs.dataFiles` setting to point the base directory of the files that will be uploaded. If the default directory **./data** is used, this setting can be ignored.
8. Optionally - Set the `esp8266fs.spiffsImage` setting to a filename that **mkspiffs** will create.
9. Done - you can now run the commands provided by this extension on the `esp8266fs.dataFiles` and `esp8266fs.spiffsImage`.

> Note: Maximum length of a file name in **SPIFFS** is 32 characters. Subdirectories are "simulated" in that a file name really contains the "/" of the file's folder. I.e. a file stored at "/abc/def/ghi.txt" has a name with 16 characters. Files are packed relative to the `path` setting and not of the base OS.

Commands

All of the commands send their spew to the ESP8266FS OUTPUT window. The amount of spew is dictated by the `logLevel`. Setting it to "debug" will send more spew back to the hosting debugger and has no effect on using the extension.



Name	Command id	Description
`ESP8266FS: Upload SPIFFS`	*esp8266fs.uploadSpiffs*	Upload a SPIFFS image.
`ESP8266FS: Download SPIFFS`	*esp8266fs.downloadSpiffs*	Download a SPIFFS image.
`ESP8266FS: Pack SPIFFS`	*esp8266fs.packSpiffs*	Creates the SPIFFS image.
`ESP8266FS: Unpack SPIFFS`	*esp8266fs.unpackSpiffs*	Unpacks the contents of a SPIFFS image.
`ESP8266FS: List SPIFFS`	*esp8266fs.listSpiffs*	List the contents of a SPIFFS image.
`ESP8266FS: Visualize SPIFFS`	*esp8266fs.visualizeSpiffs*	"Visualizes" the contents of a SPIFFS image.

Upload

`ESP8266FS: Upload SPIFFS` - This command sends the `esp8266fs.spiffsImage` to the **ESP8266** using the **esptool**, **epstool.py** or **espota.py** tool (depending on the output port or target chip).

Dowload

`ESP8266FS: Download SPIFFS` - This command fetchs the `esp8266fs.spiffsImage` from the **ESP8266** using the **epstool.py** (EPS8266 users can point `esp8266fs.esptool.executable` to a copy of **esptool.py**).

Pack

`ESP8266FS: Pack SPIFFS` - this command packs all of the files in the `esp8266fs.dataFiles` subdirectory using the **mkspiffs** tool into the `esp8266fs.spiffsImage` file.

Unpack

`ESP8266FS: Unpack SPIFFS` - this command will take the `esp8266fs.spiffsImage` and unpack all the contents into the `esp8266fs.dataFiles` folder using the **mkspiffs** tool.

List

`ESP8266FS: List SPIFFS` - this command will list the contents of the `esp8266fs.spiffsImage` using the **mkspiffs** tool.

Visualize

`ESP8266FS: Visualize SPIFFS` - this command will "visualize" the contents of the `esp8266fs.spiffsImage` using the **mkspiffs** tool.

Options

The following Visual Studio Code settings are available for the **ESP8266FS** extension. These can be set in the global user preferences `Ctrl+,` or workspace settings (.vscode/settings.json). The later overrides the former. None of these settings are necessary as all have default values, or are deduced from the environment.

.vscode/settings.json

```
{
    ...

        //--- Python path - needed for espOTA.py

    "python.pythonPath": "C:/Python34/python.exe",

        //--- ESP8266FS for Visual Studio Code settings

    "esp8266fs.dataFiles": "./data",
    "esp8266fs.preferencesPath": "C:/Users/X/AppData/Local/Arduino15",
    "esp8266fs.arduinoUserPath": "C:/Users/X/Documents/Arduino",
    "esp8266fs.spiffsImage": "./temp/spiffs.image.bin",
    "esp8266fs.logLevel": "normal",

    "esp8266fs.mkspiffs.executable": "C:/Users/X/AppData/Local/Arduino15/packages/esp8266/tools/mkspiffs/(
    "esp8266fs.mkspiffs.debugLevel": "0",
    "esp8266fs.mkspiffs.allFiles": true,

    "esp8266fs.esptool.executable": "C:/Users/X/AppData/Local/Arduino15/packages/esp8266/tools/esptool/0.4
    "esp8266fs.esptool.verbosity": "vvv",

    "esp8266fs.esptool.py.before": "default_reset",
    "esp8266fs.esptool.py.after": "hard_reset",
    "esp8266fs.esptool.py.no_stub": "false",
    "esp8266fs.esptool.py.trace": "false",
    "esptool.py.spi_connection": "SPI",
    "esptool.py.compress": "true",
    "esptool.py.verify": "false",

    "esp8266fs.espota.py": "C:/Users/X/AppData/Local/Arduino15/packages/esp8266/hardware/esp8266/2.4.0/esp
    "esp8266fs.espota.esp.port": 8266,
    "esp8266fs.espota.host.ip": "0.0.0.0",
    "esp8266fs.espota.host.port": 12345,
    "esp8266fs.espota.debug": true,

    ...
}
```

vscode

- `python.pythonPath` - Path to the **python** executable; as defined by the VSCode Python Environment. Can be set via the `Python: Select Interpreter` command. If not specified, then "**python**" will be used. Only needed if using OTA (espota.py).

esp8266fs

- `esp8266fs.dataFiles` - Location of the base directory of the files to be uploaded to the ESP8266's SPIFFS. File names will be generated relative to this path. Default is "**./data**".
- `esp8266fs.preferencesPath` - Location of the **Arduino IDE**'s "preferences.txt" file and installed libraries. This value does not need to be set, but exists to override the default location.
- `esp8266fs.spiffsImage` - Name of the packed **SPIFFS** image. Default is "**./spiffs.image**".
- `esp8266fs.logLevel` - Changes the amount of spew produced. Set to either `normal`, `verbose`, `silent`, or `debug`. Default is **normal**.

mkspiffs

- `esp8266fs.mkspiffs.executable` - Path to **mkspiffs** executable. If not specified, then ESP8266FS will attempt to locate it through the Arduino settings.
- `esp8266fs.mkspiffs.debugLevel` - Debug spew level for **mkspiffs**. Default is **0**.
- `esp8266fs.mkspiffs.allFiles` - Tells **mkspiffs** to include ignored files (*.DS_Store* and *.git* directories). Default is **false**.

esptool

- `esp8266fs.esptool.executable` - Path to **esptool** executable. If not specified, then ESP8266FS will attempt to locate it through the Arduino settings.
- `esp8266fs.esptool.verbosity` - **esptool** verbosity. Add more **v**'s to be more verbose. Default is no **v**'s.

esptool.py

- `esptool.py.before` - "What to do before connecting to the chip".
- `esptool.py.after` - "What to do after esptool.py is finished".
- `esptool.py.no_stub` - "Disable launching the flasher stub, only talk to ROM".
- `esptool.py.trace` - "Enable trace-level output of esptool.py interactions".
- `esptool.py.spi_connection` - "Override default SPI Flash connection. Value can be SPI, HSPI or a comma-separated list of 5 I/O numbers to use for SPI flash (CLK,Q,D,HD,CS)".
- `esptool.py.compress` - "Compress data in transfer (default "true" unless "no-stub" is specified)".
- `esptool.py.verify` - "Verify just-written data on flash (mostly superfluous, data is read back during flashing)".

espota

- `esp8266fs.espota.py` - Path to the **espota** python script. If not specified, then ESP8266FS will attempt to locate it through the Arduino settings.
- `esp8266fs.espota.esp.port` - IP port for the target **ESP8266**. Default is **8266**.
- `esp8266fs.espota.host.ip` - IP address for the host. Default is "**0.0.0.0**".
- `esp8266fs.espota.host.port` - IP port for the host. Default is a random port: 10000-60000.
- `esp8266fs.espota.auth` - Authentication password for the **espota** python script. Default is not set.
- `esp8266fs.espota.debug` - Enables debug output from the **espota** python script. Default is **false**.

.vscode/arduino.json

The following settings are per sketch settings (*defined by the **Arduino for Visual Studio Code** plugin). You can find them in `.vscode/arduino.json` in the workspace. The `.vscode/arduino.json` file has "per sketch" settings.

```
{
    "port": "COM6",
    "board": "esp8266:esp8266:generic",
    "configuration": "...FlashSize=4M3M,...ResetMethod=ck,..."
}
```

- `port` - Name of the serial port connected to the device. Can be set by the `Arduino: Select Serial Port` command. Alternatively, if you specify an IP address (x.x.x.x), then the `espota.py` script will be executed to communicate with the ESP8266 (**OTA** = **O**ver **T**he **A**ir).
- `board` - Current selected Arduino board alias. Can be set by the `Arduino: Change Board Type` command. Also, you can find the board list there.
- `configuration` - (*Undocumented*) A comma-delimited string of the configuration settings selected for all board "menu" items. **ESP8266FS** relies on four key/value pairs in the string: `FlashSize`, `FlashMode`, `FlashFreq` and `ResetMethod`.

Alternatively, if the `.vscode/arduino.json` file doesn't exist, or a particular setting is not defined, then the settings in the **Arduino IDE**'s `preferences.txt` file will be used instead. This file is generated by the **Arduino IDE** and is set globally for ALL sketches.

preferences.txt

```
board=generic
target_package=esp8266
target_platform=esp8266

serial.port=COM6

custom_FlashSize=generic_4M3M
custom_ResetMethod=generic_ck
```

Support

You can find the full list of issues at Issue Tracker, and you can submit a bug or feature suggestion.

Development

Installing Prerequisites:

- Git
- Node.js (>= 6.5.0)
- Npm (>= 3.10.3)

To *run and develop*, do the following:

1. `git clone https://github.com/kash4kev/vscode-esp8266fs`.
2. `cd vscode-esp8266fs`.
3. Open in Visual Studio Code (`code .`).
4. Install the dependent NPM packages (`Tasks/Run Task.../Install NPM packages`).
5. Press `F5` to debug.

Change Log

See the Change log for the details of changes for each version.

Known Issues

None - that I know of. Please submit a bug or feature suggestion if you find something amiss.

Release Notes

[1.1.0] 2018-5-23

- Added support for the ESP32 library package (espressif/esp32).
- Added support for "esptool.py" found in the ESP32 library.
- FlashSize now supports "generic" settings.
- Changed package.json "keyword" from "iot" -> "ESP32".
- "esp8266fs.upload" now only uploads - pack is done separately.
- Added "espspiffs.download" for user of "esptool.py".
- Added "esp8266fs.arduinoUserPath" setting.
- Added "esp8266fs.esptool.py..." settings.
- Removed unnecessary code.

[1.0.1] 2018-4-7

- Fixed errata and ran **markdownlint** on all .md files.
- Fixed all missing overrides. Expanded OTA overrides.
- Added `esp8266.packSpiffs` command.
- Made showErrorMessage modal.
- Added "#region" tags to source code.
- Refactored "upload" to minimize external tool requirements.
- Removed `esp8266fs.python`, defering to `python.pythonPath`.

[1.0.0] 2018-2-13

- Fixed errata and expanded documentation.
- Located espota.py correctly.
- Changed `esp8266fs.uploadData` -> `esp8266fs.uploadSpiffs`.
- Added `esp8266fs.unpackSpiffs`, `esp8266fs.listSpiffs`, and `esp8266fs.visualizeSpiffs`.
- Added a variety of settings.json values to allow full control of the mkspiffs *rocess.
- Fixed various bugs and cleaned up code heirarchy.
- Updated Github infrastructure.
- Tested on Windows, OSX, and Linux (Ubuntu).

[0.9.0] - 2018-02-11

- Initial release - out for review.

License