**ESP8266WiFi library**

ESP8266 - це все про Wi-Fi. Якщо ви прагнете підключити свій новий модуль ESP8266 до мережі Wi-Fi, щоб почати надсилати та отримувати дані, це гарне місце для початку. Якщо ви шукаєте більш детальні відомості про те, як програмувати конкретні функції мережі Wi-Fi, ви також знаходитесь у правильному місці.

**Introduction**

The [Wi-Fi library for ESP8266](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) has been developed basing on [ESP8266 SDK](http://bbs.espressif.com/viewtopic.php?f=51&t=1023), using naming convention and overall functionality philosophy of [Arduino WiFi library](https://www.arduino.cc/en/Reference/WiFi). Over time the wealth Wi-Fi features ported from ESP9266 SDK to [esp8266 / Adruino](https://github.com/esp8266/Arduino) outgrow [Arduino WiFi library](https://www.arduino.cc/en/Reference/WiFi) and it became apparent that we need to provide separate documentation on what is new and extra.

Бібліотека [Wi-Fi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) для ESP8266 була розроблена на базі ESP8266 SDK, використовуючи конвенцію щодо іменування та загальну філософію функціональності бібліотеки [Arduino WiFi](https://www.arduino.cc/en/Reference/WiFi). З часом багатовікові функції Wi-Fi перенесені з ESP9266 SDK на [esp8266/Adruino](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) переросли [Arduino WiFi](https://www.arduino.cc/en/Reference/WiFi) бібліотеку, і стало очевидно, що потрібно надати окрему документацію про те, що є новим та додатковим.

This documentation will walk you through several classes, methods and properties of [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) library. If you are new to C++ and Arduino, don’t worry. We will start from general concepts and then move to detailed description of members of each particular class including usage examples.

Ця документація пройде через кілька класів, методів та властивостей бібліотеки [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi). Якщо ви новачок в C ++ і Arduino, не хвилюйтеся. Почнемо з загальних концепцій, а потім перейдемо до детального опису членів кожного конкретного класу, включаючи приклади використання.

The scope of functionality offered by [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) library is quite extensive and therefore this description has been broken up into separate documents marked with :arrow\_right:.

Обсяг функціональних можливостей, що пропонуються бібліотекою [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi), досить великий, і тому опис було розбито на окремі документи, позначені :arrow\_right:.

**Quick Start**

Hopefully you are already familiar how to load [Blink.ino](https://github.com/esp8266/Arduino/blob/master/libraries/esp8266/examples/Blink/Blink.ino) sketch to ESP8266 module and get the LED blinking. If not, please check [this tutorial](https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/using-arduino-ide) by Adafruit or [another great tutorial](https://learn.sparkfun.com/tutorials/esp8266-thing-hookup-guide/introduction)developed by Sparkfun.

Щоб підключити модуль ESP до Wi-Fi (наприклад, підключення мобільного телефону до гарячої точки), потрібно лише кілька рядків коду:

*#include <ESP8266WiFi.h>*

void setup()

{

Serial**.**begin(115200);

Serial**.**println();

WiFi**.**begin("network-name", "pass-to-network");

Serial**.**print("Connecting");

**while** (WiFi**.**status() **!=** WL\_CONNECTED)

{

delay(500);

Serial**.**print(".");

}

Serial**.**println();

Serial**.**print("Connected, IP address: ");

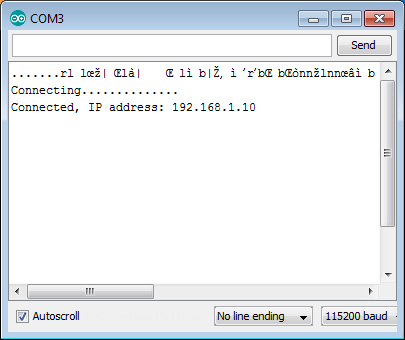
Serial**.**println(WiFi**.**localIP());

}

void loop() {}

In the line WiFi.begin("network-name", "pass-to-network") replace network-name and pass-to-network with name and password to the Wi-Fi network you like to connect. Then upload this sketch to ESP module and open serial monitor. You should see something like:

В рядку WiFi.begin ("network-name", "pass-to-network") замініть network-name та pass-to-network з ім'ям та паролем до мережі Wi-Fi, яку ви хочете підключити. Потім завантажте цей ескіз в модуль ESP та відкрийте Serial monitor. Ви повинні побачити щось на зразок:



*alt text*

How does it work? In the first line of sketch #include <ESP8266WiFi.h> we are including [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) library. This library provides ESP8266 specific Wi-Fi routines we are calling to connect to network.

Як це працює? У першому рядку ескізу #include<ESP8266WiFi.h> ми включаємо [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) бібліотеку. Ця бібліотека надає спеціальні режими Wi-Fi ESP8266, які ми викликаємо для підключитися до мережі.

Actual connection to Wi-Fi is initialized by calling:

Фактично з'єднання Wi-Fi ініціалізується за викликом:

WiFi**.**begin("network-name", "pass-to-network");

Connection process can take couple of seconds and we are checking for this to complete in the following loop:

Процес з'єднання може тривати кілька секунд, і ми перевіряємо його завершення в наступному циклі:

**while** (WiFi**.**status() **!=** WL\_CONNECTED)

{

delay(500);

Serial**.**print(".");

}

The while() loop will keep looping while WiFi.status() is other than WL\_CONNECTED. The loop will exit only if the status changes to WL\_CONNECTED.

Цикл while() продовжуватиметься, поки WiFi.status() не є WL\_CONNECTED. Цикл закінчиться, лише якщо статус зміниться на WL\_CONNECTED. Інколи дописують лічильник щоб вийти з циклу і працювати без точки доступу, або перезавантажити модуль.

The last line will then print out IP address assigned to ESP module by [DHCP](http://whatismyipaddress.com/dhcp):

Останній рядок роздруковує IP-адресу, призначену модулю ESP [DHCP](https://uk.wikipedia.org/wiki/DHCP):

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

If you don’t see the last line but just more and more dots ........., then likely name or password to the Wi-Fi network in sketch is entered incorrectly. Verify name and password by connecting from scratch to this Wi-Fi a PC or a mobile phone.

Якщо в послідовному порту виводиться лише точки «.........», то імовірно ім'я або пароль до мережі Wi-Fi в програмі були введені неправильно. Перевірте ім'я та пароль, підключившись з початку до цього Wi-Fi на ПК або мобільному телефоні.

*Note:* if connection is established, and then lost for some reason, ESP will automatically reconnect to last used access point once it is again back on-line. This will be done automatically by Wi-Fi library, without any user intervention.

Примітка. Якщо з'єднання було встановлено, а потім, з якоїсь причини, втрачено, ESP автоматично підключиться до останньої використовуваної точки доступу, коли вона знову буде доступною. Це буде зроблено автоматично бібліотекою Wi-Fi, без втручання користувача.

That’s all you need to connect ESP8266 to Wi-Fi. In the following chapters we will explain what cool things can be done by ESP once connected.

Це все, що потрібно для підключення ESP8266 до Wi-Fi. У наступних уроках ми розкажемо про можливості ESP при підключені до мережі.

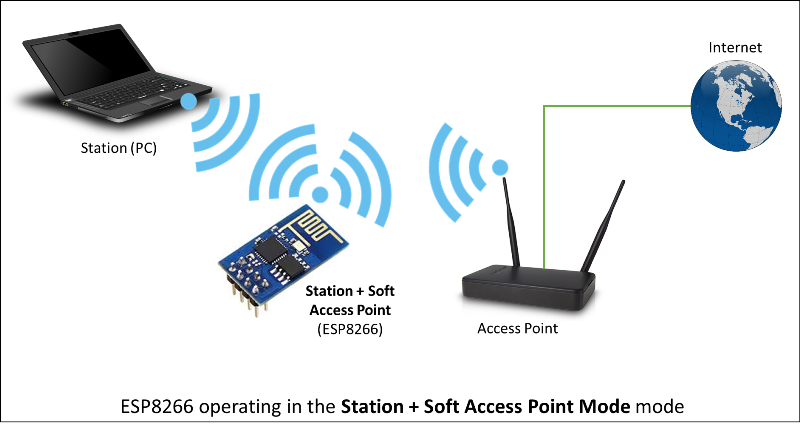
**Who is Who**

Devices that connect to Wi-Fi network are called stations (STA). Connection to Wi-Fi is provided by an access point (AP), that acts as a hub for one or more stations. The access point on the other end is connected to a wired network. An access point is usually integrated with a router to provide access from Wi-Fi network to the internet. Each access point is recognized by a SSID (**S**ervice **S**et **ID**entifier), that essentially is the name of network you select when connecting a device (station) to the Wi-Fi.

Пристрої, які підключаються до мережі Wi-Fi, називаються станціями (STA). Підключення до Wi-Fi забезпечується точкою доступу (AP), яка діє як концентратор для однієї або декількох станцій. Точка доступу на іншому кінці підключена до дротової мережі. Точка доступу, як правило, інтегрована з маршрутизатором, щоб забезпечити доступ до Інтернету через мережу Wi-Fi. Кожна точка доступу визнається ідентифікатором SSID (**S**ervice **S**et **ID**entifier), що по суті є ім'ям мережі, яку ви вибираєте під час підключення пристрою (станції) до Wi-Fi.

ESP8266 module can operate as a station, so we can connect it to the Wi-Fi network. It can also operate as a soft access point (soft-AP), to establish its own Wi-Fi network. Therefore we can connect other stations to such ESP module. ESP8266 is also able to operate both in station and soft access point mode. This provides possibility of building e.g. [mesh networks](https://en.wikipedia.org/wiki/Mesh_networking).

Модуль ESP8266 може працювати як станція, тому ми можемо підключити його до мережі Wi-Fi. Він також може працювати як обмежена точка доступу (soft-AP), щоб створити власну мережу Wi-Fi. Таким чином ми можемо з'єднати інші станції з таким налаштуванням модуля ESP. ESP8266 також може працювати одночасно як станція і як обмежена точка доступу. Це дає можливість побудови, наприклад, [mesh мережі](https://uk.wikipedia.org/wiki/Mesh-%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D1%96).



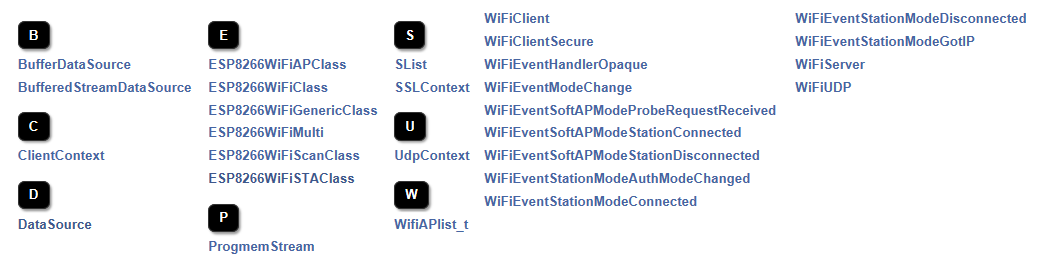
The [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) library provides wide collection of C++[methods](https://en.wikipedia.org/wiki/Method_(computer_programming)) (functions) and [properties](https://en.wikipedia.org/wiki/Property_(programming)) to configure and operate an ESP8266 module in station and / or soft access point mode. They are described in the following chapters.

Бібліотека [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) забезпечує широкий набір [методів](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) (функцій) та [властивостей](https://uk.wikipedia.org/wiki/%D0%92%D0%BB%D0%B0%D1%81%D1%82%D0%B8%D0%B2%D1%96%D1%81%D1%82%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) C ++ для налаштування та керування модулем ESP8266 в режимі станції та / або обмеженої точки доступу. Вони описані в наступних розділах.

## Class Description (опис)

The [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) library is broken up into several classes. In most of cases, when writing the code, user is not concerned with this classification. We are using it to break up description of this library into more manageable pieces.

Бібліотека [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) розбита на кілька класів. У більшості випадків при написанні коду користувач не має відношення до цієї класифікації. Ми використовуємо їх лише для опису цієї бібліотеки на більш примітивних частинах.



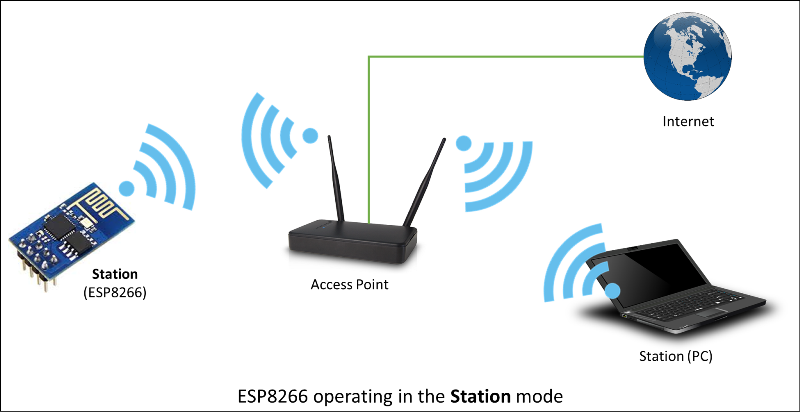
Chapters below describe all function calls ([methods](https://en.wikipedia.org/wiki/Method_(computer_programming)) and [properties](https://en.wikipedia.org/wiki/Property_(programming)) in C++ terms) listed in particular classes of [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi). Description is illustrated with application examples and code snippets to show how to use functions in practice. Most of this information is broken up into separate documents. Please follow to access them.

У наведених нижче розділах описано всі виклики функцій (методи та властивості у термінах C ++), перелічені в окремих класах [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi). Опис ілюстрований прикладами програм та фрагментами коду, щоб показати, як використовувати функції на практиці. Більша частина цієї інформації розбита на окремі документи. Будь ласка, дотримуйтесь, щоб отримати доступ до них.

**Station**

Station (STA) mode is used to get ESP module connected to a Wi-Fi network established by an access point.

Режим станції (STA) використовується для підключення модуля ESP до мережі Wi-Fi, встановленої точки доступу.



Station class has several features to facilitate management of Wi-Fi connection. In case the connection is lost, ESP8266 will automatically reconnect to the last used access point, once it is again available. The same happens on module reboot. This is possible since ESP is saving credentials to last used access point in flash (non-volatile) memory. Using the saved data ESP will also reconnect if sketch has been changed but code does not alter the Wi-Fi mode or credentials.

Клас станції має кілька функцій для полегшеного керування Wi-Fi-з'єднанням. Якщо з'єднання буде втрачено, ESP8266 автоматично підключиться до останньої використаної точки доступу, коли вона знову стане доступною. Те ж саме відбувається при перезавантаженні модуля. Це можливо, оскільки ESP зберігає облікові дані до останньої використаної точки доступу у флеш-пам'яті (енергонезалежній). Використовуючи збережені дані з флеш, ESP перепід'єднатися, навіть якщо скетч був змінений, але режим Wi-Fi не був змінений.

[Station Class documentation](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/station-class.html)

Check out separate section with [examples](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/station-examples.html).

Ознайомтеся з окремим розділом з прикладами.

**Soft Access Point**

An [access point (AP)](https://en.wikipedia.org/wiki/Wireless_access_point) is a device that provides access to Wi-Fi network to other devices (stations) and connects them further to a wired network. ESP8266 can provide similar functionality except it does not have interface to a wired network. Such mode of operation is called soft access point (soft-AP). The maximum number of stations connected to the soft-AP is five.

[Точка доступу](https://uk.wikipedia.org/wiki/%D0%91%D0%B5%D0%B7%D0%B4%D1%80%D0%BE%D1%82%D0%BE%D0%B2%D0%B0_%D1%82%D0%BE%D1%87%D0%BA%D0%B0_%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D1%83) (AP) - це пристрій, який забезпечує доступ до мережі Wi-Fi іншим пристроям (станціям) і з'єднує їх далі з дротовою мережею. ESP8266 може забезпечити подібну функціональність, за винятком відсутності інтерфейсу до дротової мережі. Такий режим роботи називається обмеженою точкою доступу (soft-AP). Максимальна кількість станцій, підключених до soft-AP, становить п'ять.



The soft-AP mode is often used and an intermediate step before connecting ESP to a Wi-Fi in a station mode. This is when SSID and password to such network is not known upfront. ESP first boots in soft-AP mode, so we can connect to it using a laptop or a mobile phone. Then we are able to provide credentials to the target network. Once done ESP is switched to the station mode and can connect to the target Wi-Fi.

Режим soft-AP часто використовується як проміжний етап перед підключенням ESP до Wi-Fi у режимі станції. Так роблять коли SSID та пароль до мережі не відомі на стадії програмування. ESP спершу завантажується в режимі soft-AP, і ми можемо підключитися до неї за допомогою ноутбука або мобільного телефону. Так ми зможемо надати облікові дані для підключення до цільової мережі. Після завершення роботи ESP переключається в режим станції і намагатиметься підключатися до цільової точки Wi-Fi, якщо підключення не буде можливим то ESP знову перейде в режим точки доступу.

Another handy application of soft-AP mode is to set up [mesh networks](https://en.wikipedia.org/wiki/Mesh_networking). ESP can operate in both soft-AP and Station mode so it can act as a node of a mesh network.

Ще одне зручне застосування режиму soft-AP полягає у створенні мережевих сіток. ESP може одночасно працювати як у режимі soft-AP, так і в – станції, щоб мати можливість функціонувати як [mesh мережа](https://uk.wikipedia.org/wiki/Mesh-%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D1%96).

[Soft Access Point Class documentation](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/soft-access-point-class.html)

Check out separate section with [examples](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/soft-access-point-examples.html).

**Scan**

To connect a mobile phone to a hot spot, you typically open Wi-Fi settings app, list available networks and pick the hot spot you need. Then enter a password (or not) and you are in. You can do the same with ESP. Functionality of scanning for, and listing of available networks in range is implemented by the Scan Class.

Щоб підключити мобільний телефон до точки, ви запускаєте налаштування Wi-Fi, перелистуєте доступні мережі та вибираєте необхідну точку, яка вам потрібна. Потім вводите пароль (якщо він є), і підключаєтесь. Ви можете зробити теж саме і з ESP. Функція сканування та переліку доступних мереж у діапазоні реалізується класом *Scan*.

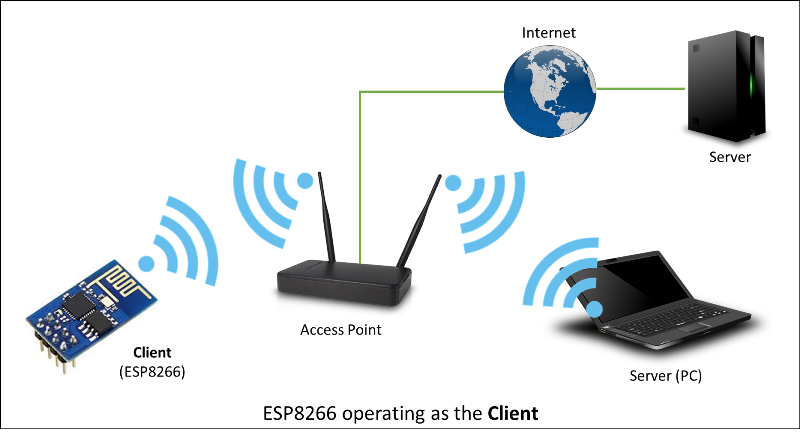
[Scan Class documentation](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/scan-class.html).

Check out separate section with [examples](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/scan-examples.html).

**Client**

The Client class creates [clients](https://en.wikipedia.org/wiki/Client_(computing)) that can access services provided by [servers](https://en.wikipedia.org/wiki/Server_(computing)) in order to send, receive and process data.

Клас клієнта створює [клієнтів](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), які можуть отримати доступ до послуг, що надаються [серверами](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80), для надсилання, отримання та обробки даних.



Check out separate section with [examples](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/client-examples.html) / [list of functions](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/client-class.html)

Ознайомтеся з окремим розділом з прикладами / переліком функцій

**Client Secure**

The Client Secure is an extension of [Client Class](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html#client) where connection and data exchange with servers is done using a [secure protocol](https://en.wikipedia.org/wiki/Transport_Layer_Security). It supports [TLS 1.1](https://en.wikipedia.org/wiki/Transport_Layer_Security#TLS_1.1). The [TLS 1.2](https://en.wikipedia.org/wiki/Transport_Layer_Security#TLS_1.2) is not supported.

«Клієнтський захист» - це розширення класу клієнта, при якому з'єднання та обмін даними з серверами здійснюється за допомогою [захищеного протоколу](https://uk.wikipedia.org/wiki/Transport_Layer_Security). Він підтримує алгоритм [TLS 1.1](https://en.wikipedia.org/wiki/Transport_Layer_Security#TLS_1.1), а [TLS 1.2](https://en.wikipedia.org/wiki/Transport_Layer_Security#TLS_1.2) вже не підтримується.



Secure applications have additional memory (and processing) overhead due to the need to run cryptography algorithms. The stronger the certificate’s key, the more overhead is needed. In practice it is not possible to run more than a single secure client at a time. The problem concerns RAM memory we can not add, the flash memory size is usually not the issue. If you like to learn how [client secure library](https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/WiFiClientSecure.h) has been developed, access to what servers have been tested, and how memory limitations have been overcame, read fascinating issue report [#43](https://github.com/esp8266/Arduino/issues/43).

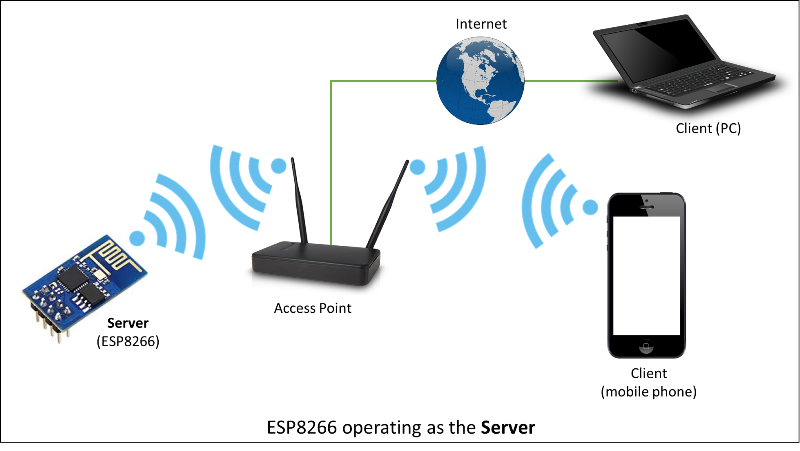
Алгоритми захисту займають додаткову пам'ять і час обробки через необхідність запуску алгоритмів шифрування. Чим складніший сертифікат ключа, тим більше ресурсів необхідно витратити. На практиці неможливо запустити більше одного захищеного клієнта за один раз. Проблема полягає в недостатньому об’єму оперативної пам'яті, яку ми не можемо додати, а флеш-пам'яті зазвичай вистачає. Ви можете самостійно ознайомитися з [бібліотекою захисту](https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/src/WiFiClientSecure.h) і дослідити деякі хитрощі економії оперативної пам’яті. Якщо ви хочете дізнатись, як влаштована бібліотека захисту клієнтом, доступ до того, які сервери були перевірені, та про те, як було пом'якшено обмеження пам'яті, див. У захоплюючому звіті про випуск № 43.

Check out separate section with [examples](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/client-secure-examples.html) / [list of functions](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/client-secure-class.html)

**Server**

The Server Class creates [servers](https://en.wikipedia.org/wiki/Server_(computing)) that provide functionality to other programs or devices, called [clients](https://en.wikipedia.org/wiki/Client_(computing)).

Клас *Server* створює [сервери](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80), які забезпечують функціональність для інших програм або пристроїв, які називаються [клієнтами](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)).



Clients connect to sever to send and receive data and access provided functionality.

Клієнти підключаються до сервера для надсилання та отримання даних, а також доступу до наданого функціоналу.

Check out separate section with [examples](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/server-examples.html) / [list of functions](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/server-class.html).

### UDP

The UDP Class enables the [User Datagram Protocol (UDP)](https://en.wikipedia.org/wiki/User_Datagram_Protocol)messages to be sent and received. The UDP uses a simple “fire and forget” transmission model with no guarantee of delivery, ordering, or duplicate protection. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram.

Клас UDP дозволяє надсилати та отримувати [User Datagram Protocol (UDP)](https://uk.wikipedia.org/wiki/UDP) повідомлення. UDP використовує просту модель "перестрілки та забуття", яка не гарантує доставку, замовлення чи дублювання захисту. UDP надає контрольні суми для цілісності даних та номери портів для вирішення різних функцій у джерелі та місці призначення датаграми.

Check out separate section with [examples](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/udp-examples.html) / [list of functions](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/udp-class.html).

### Generic

There are several functions offered by ESP8266’s [SDK](http://bbs.espressif.com/viewtopic.php?f=51&t=1023) and not present in [Arduino WiFi library](https://www.arduino.cc/en/Reference/WiFi). If such function does not fit into one of classes discussed above, it will likely be in Generic Class. Among them is handler to manage Wi-Fi events like connection, disconnection or obtaining an IP, Wi-Fi mode changes, functions to manage module sleep mode, hostname to an IP address resolution, etc.

Є кілька функцій, пропонованих SDK ESP8266, і не присутні в бібліотеці [Arduino WiFi](https://www.arduino.cc/en/Reference/WiFi). Якщо така функція не входить до одного з описаних вище класів, воно, імовірно, буде в *Generic Class*. Серед таких - обробник для керування подій Wi-Fi, таких як підключення, відключення або отримання IP, зміни режиму Wi-Fi, функції управління модулем сплячого режиму, ім'я хоста з дозволу IP-адрес тощо.

Check out separate section with [examples](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/generic-examples.html) / [list of functions](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/generic-class.html).

## Diagnostics

There are several techniques available to diagnose and troubleshoot issues with getting connected to Wi-Fi and keeping connection alive.

Існує кілька методів, доступних для діагностики та вирішення проблем із підключенням до Wi-Fi та підтримання зв'язку.

### Check Return Codes

Almost each function described in chapters above returns some diagnostic information.

Майже кожна функція, описана в розділах вище, повертає деяку діагностичну інформацію.

Such diagnostic may be provided as a simple boolean type true' orfalse` to indicate operation result. You may check this result as described in examples, for instance:

Така діагностика може бути надана у вигляді простого boolean  типу для визначення результату операції. Наприклад:

Serial.printf("Wi-Fi mode set to WIFI\_STA %s\n", WiFi.mode(WIFI\_STA) ? "" : "Failed!");

Some functions provide more than just a binary status information. A good example is WiFi.status().

Деякі функції більш інформативні. Хорошим прикладом є WiFi.status().

Serial**.**printf("Connection status: %d\n", WiFi**.**status());

This function returns following codes to describe what is going on with Wi-Fi connection:

Ця функція повертає наступні коди, щоб описати, стан з'єднання Wi-Fi:

* 0 : WL\_IDLE\_STATUS when Wi-Fi is in process of changing between statuses
* 1 : WL\_NO\_SSID\_AVAILin case configured SSID cannot be reached
* 3 : WL\_CONNECTED after successful connection is established
* 4 : WL\_CONNECT\_FAILED if password is incorrect
* 6 : WL\_DISCONNECTED if module is not configured in station mode
* 0 : WL\_IDLE\_STATUS – коли Wi-Fi в процесі переходу між статусами
* 1 : WL\_NO\_SSID\_AVAIL – у випадку неможливості налаштувати SSID
* 3 : WL\_CONNECTED – після успішного встановленого з'єднання
* 4 : WL\_CONNECT\_FAILED – якщо пароль неправильний
* 6 : WL\_DISCONNECTED – якщо модуль не налаштований в режимі станції

It is a good practice to display and check information returned by functions. Application development and troubleshooting will be easier with that.

Гарною практикою є відображення та перевірка інформації, яка повертається функціями. Розробка додатків та усунення неполадок буде легше.

### Use printDiag

There is a specific function available to print out key Wi-Fi diagnostic information:

Існує певна функція, для роздруківки основної діагностичної інформації Wi-Fi:

WiFi**.**printDiag(Serial);

A sample output of this function looks as follows:

Зразок роботи цієї функції виглядає наступним чином:

Mode: STA+AP

PHY mode: N

Channel: 11

AP id: 0

Status: 5

Auto connect: 1

SSID (10): sensor-net

Passphrase (12): 123!$#0&\*esP

BSSID set: 0

Use this function to provide snapshot of Wi-Fi status in these parts of application code, that you suspect may be failing.

Використовуйте цю функцію, щоб забезпечити знімок стану Wi-Fi у частинах коду програми, які, як ви вважаєте, можуть не коректно працювати.

### Enable Wi-Fi Diagnostic

By default the diagnostic output from Wi-Fi libraries is disabled when you call Serial.begin. To enable debug output again, call Serial.setDebugOutput(true). To redirect debug output to Serial1 instead, call Serial1.setDebugOutput(true). For additional details regarding diagnostics using serial ports please refer to [the documentation](http://arduino-esp8266.readthedocs.io/en/latest/reference.html).

За замовчанням вивід діагностичної інформації з бібліотек Wi-Fi вимикається під час виклику Serial.begin. Щоб відновити потік налагодження, викличте функцію –Serial.setDebugOutput(true). Щоб переспрямовувати вивід налагодження на Serial1, викличте функцію – Serial1.setDebugOutput(true). Додаткові відомості щодо діагностики з використанням послідовних портів див. [у документації](http://arduino-esp8266.readthedocs.io/en/latest/reference.html).

Below is an example of output for sample sketch discussed in [Quick Start](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html#quick-start) above with Serial.setDebugOutput(true):

Нижче наведений приклад виводу для програми зразка, описаного у розділі "Quick Start" вище, з викликом – Serial.setDebugOutput(true):

Connectingscandone

state: 0 **->** 2 (b0)

state: 2 **->** 3 (0)

state: 3 **->** 5 (10)

add 0

aid 1

cnt

connected **with** sensor**-**net, channel 6

dhcp client start**...**

chg\_B1:**-**40

**...**ip:192.168**.**1.10,mask:255.255**.**255.0,gw:192.168**.**1.9

**.**

Connected, IP address: 192.168**.**1.10

The same sketch without Serial.setDebugOutput(true) will print out only the following:

Той же приклад без виклику – Serial.setDebugOutput(true) буде роздруковувати лише наступне:

Connecting**....**

Connected, IP address: 192.168**.**1.10

### Enable Debugging in IDE

Arduino IDE provides convenient method to [enable debugging](https://github.com/esp8266/Arduino/blob/master/doc/Troubleshooting/debugging.md)for specific libraries.

Arduino IDE забезпечує зручний спосіб [увімкнути налагодження](https://github.com/esp8266/Arduino/blob/master/doc/Troubleshooting/debugging.md) для певних бібліотек.

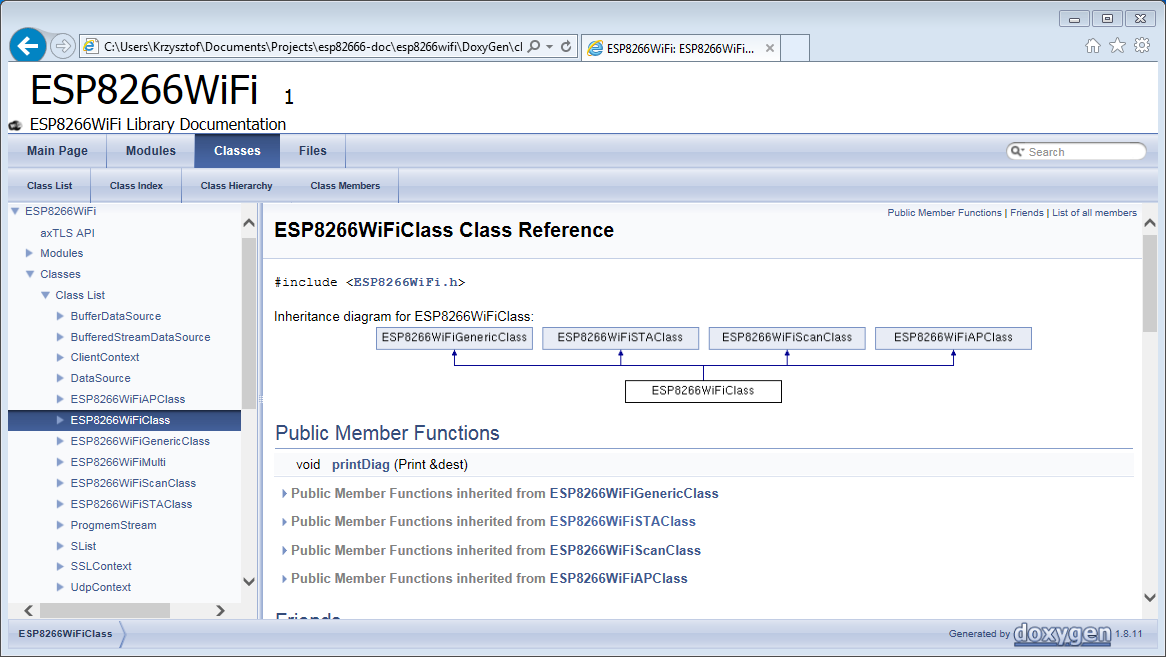
## Додаток What’s Inside?

If you like to analyze in detail what is inside of the ESP8266WiFi library, go directly to the [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi/src) folder of esp8266 / Arduino repository on the GitHub.

Якщо ви хочете детально проаналізувати, що знаходиться в бібліотеці ESP8266WiFi, перейдіть безпосередньо до теки [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi/src) репозиторію esp8266 / Arduino на GitHub.

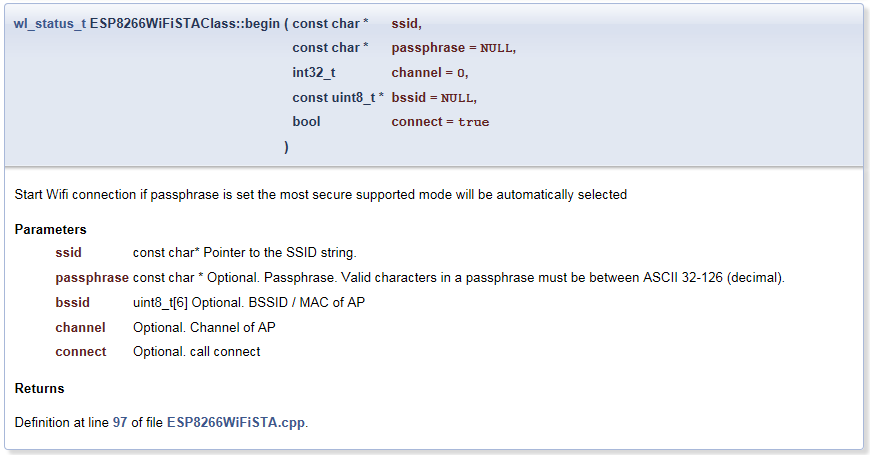
To make the analysis easier, rather than looking into individual header or source files, use one of free tools to automatically generate documentation. The class index in chapter [Class Description](http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/class-description) above has been prepared in no time using great [Doxygen](http://www.stack.nl/~dimitri/doxygen/), that is the de facto standard tool for generating documentation from annotated C++ sources.

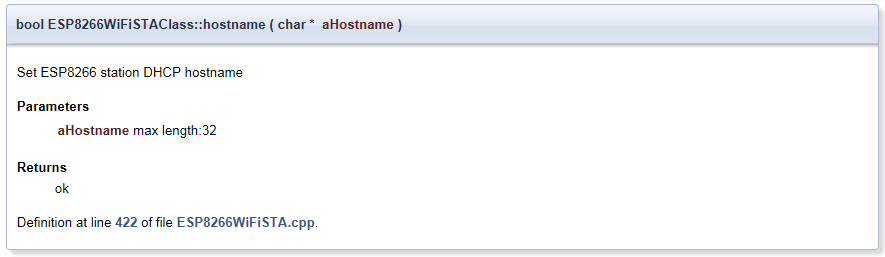
Щоб полегшити аналіз, а не переглядати окремі заголовки чи вихідні файли, використовуйте один з вільних інструментів для автоматичного створення документації. Індекс класу в розділі " Class Description" вище було підготовлено використовуючи великий [Doxygen](http://www.stack.nl/~dimitri/doxygen/), тобто де-факто стандартний інструмент для створення документації з анотованих джерел C++.



The tool crawls through all header and source files collecting information from formatted comment blocks. If developer of particular class annotated the code, you will see it like in examples below.

Інструмент виконує складування через всі заголовки та вихідні файли, збираючи інформацію з блоків коментарів. Якщо розробник певного класу анотований за кодом, ви побачите його як у прикладах нижче.





If code is not annotated, you will still see the function prototype including types of arguments, and can use provided links to jump straight to the source code to check it out on your own. Doxygen provides really excellent navigation between members of library.

Якщо код не анотований, ви все одно побачите прототип функції, включаючи типи аргументів, і можете використовувати надані посилання, щоб перейти прямо до вихідного коду, щоб перевірити його самостійно. Doxygen надає дійсно чудову навігацію між членами бібліотеки.



Several classes of [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) are not annotated. When preparing this document, [Doxygen](http://www.stack.nl/~dimitri/doxygen/) has been tremendous help to quickly navigate through almost 30 files that make this library.

Кілька класів [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi/src) не анотовані. Під час підготовки цього документа [Doxygen](http://www.stack.nl/~dimitri/doxygen/) надав величезну допомогу в швидкому переміщенні майже 30 файлів, які створюють цю бібліотеку.