**Generic Class**

Methods and properties described in this section are specific to ESP8266. They are not covered in [Arduino WiFi library](https://www.arduino.cc/en/Reference/WiFi) documentation. Before they are fully documented please refer to information below.

Методи та властивості, описані нижче, є специфічними для ESP8266. Вони не розглядаються в документації [Arduino WiFi library](https://www.arduino.cc/en/Reference/WiFi). Перш ніж вони повністю документовані, будь ласка, зверніться до інформації нижче.

**onEvent**

void onEvent (WiFiEventCb cb, WiFiEvent\_t event**=**WIFI\_EVENT\_ANY) \_\_attribute\_\_((deprecated))

To see how to use onEvent please check example sketch [WiFiClientEvents.ino](https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/examples/WiFiClientEvents/WiFiClientEvents.ino) available inside examples folder of the ESP8266WiFi library.

Щоб дізнатись, як користуватися onEvent, будь ласка, перегляньте приклад [WiFiClientEvents.ino](https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/examples/WiFiClientEvents/WiFiClientEvents.ino), доступний у теці прикладів бібліотеки ESP8266WiFi.

**WiFiEventHandler**

WiFiEventHandler onStationModeConnected (std::function**<** void(const WiFiEventStationModeConnected **&**)**>**)

WiFiEventHandler onStationModeDisconnected (std::function**<** void(const WiFiEventStationModeDisconnected **&**)**>**)

WiFiEventHandler onStationModeAuthModeChanged (std::function**<** void(const WiFiEventStationModeAuthModeChanged **&**)**>**)

WiFiEventHandler onStationModeGotIP (std::function**<** void(const WiFiEventStationModeGotIP **&**)**>**)

WiFiEventHandler onStationModeDHCPTimeout (std::function**<** void(void)**>**)

WiFiEventHandler onSoftAPModeStationConnected (std::function**<** void(const WiFiEventSoftAPModeStationConnected **&**)**>**)

WiFiEventHandler onSoftAPModeStationDisconnected (std::function**<** void(const WiFiEventSoftAPModeStationDisconnected **&**)**>**)

To see a sample application with WiFiEventHandler, please check separate section with [examples :arrow\_right:](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/generic-examples.md) dedicated specifically to the Generic Class..

Приклад програми з використанням WiFiEventHandler буде роздивлятися в другій половині цього документу.

**persistent**

WiFi**.**persistent (persistent)

Module is able to reconnect to last used Wi-Fi network on power up or reset basing on settings stored in specific sectors of flash memory. By default these settings are written to flash each time they are used in functions like WiFi.begin(ssid, password). This happens no matter if SSID or password has been actually changed.

Модуль може повторно підключатись до останньої використаної мережі Wi-Fi при включенні живлення або скидання на основі налаштувань, що зберігаються у певних секторах флеш-пам'яті. За замовчуванням ці налаштування записуються на флеш кожного разу, коли вони використовуються в таких функціях, як WiFi.begin(ssid, password). Параметри SSID та її пароль будуть перезаписані навіть якщо їх не було змінено.

This might result in some wear of flash memory depending on how often such functions are called.

Частий виклик цих функцій може призвести до зносу флеш-пам'яті.

Setting persistent to false will get SSID / password written to flash only if currently used values do not match what is already stored in flash.

Якщо встановити значення persistent на false, SSID / пароль буде записано на флеш, лише якщо поточні значення не відповідають тому, що вже зберігається у флеш-пам’яті.

Please note that functions WiFi.disconnect or WiFi.softAPdisconnect reset currently used SSID / password. If persistent is set to false, then using these functions will not affect SSID / password stored in flash.

Зверніть увагу, що функції WiFi.disconnect або WiFi.softAPdisconnect скидають поточний SSID / пароль. Якщо для persistent встановлено значення false, то використання цих функцій не вплине на SSID / пароль, що зберігається у флеш-пам’яті.

To learn more about this functionality, and why it has been introduced, check issue report [#1054](https://github.com/esp8266/Arduino/issues/1054).

Щоб дізнатись більше про цю функцію та чому вона була введена, перевірте випуск звіту [#1054](https://github.com/esp8266/Arduino/issues/1054).

**mode**

WiFi**.**mode(m)

WiFi**.**getMode()

* WiFi.mode(m): для встановлення режиму на WIFI\_AP, WIFI\_STA, WIFI\_AP\_STA або WIFI\_OFF
* WiFi.getMode(): для повернення поточного режиму Wi-Fi (один з чотирьох режимів вище)

**Other Function Calls**

int32\_t channel (void)

bool setSleepMode (WiFiSleepType\_t type)

WiFiSleepType\_t getSleepMode ()

bool setPhyMode (WiFiPhyMode\_t mode)

WiFiPhyMode\_t getPhyMode ()

void setOutputPower (float dBm)

WiFiMode\_t getMode ()

bool enableSTA (bool enable)

bool enableAP (bool enable)

bool forceSleepBegin (uint32 sleepUs**=**0)

bool forceSleepWake ()

int hostByName (const char **\***aHostname, IPAddress **&**aResult)

Documentation for the above functions is not yet prepared.

Документація для вищезгаданих функцій ще не підготовлена, але ви можете зрозуміти як вони влаштовані та як з ними працювати переглянувши вихідний код (ви також можете допомогти розширити цю документацію).

За шляхом нижче ви знайдете бібліотеку ESP8266WiFiGeneric (h та cpp файли).

C:\Users\(ВАШ\_ПРОФІЛЬ)\AppData\Local\Arduino15\packages\esp8266\hardware\esp8266\2.4.0\libraries\ESP8266WiFi\src\

For code samples please refer to separate section with [examples](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/generic-examples.rst) dedicated specifically to the Generic Class.

**EXAMPLES**

# Generic

In the first [example](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/readme.md#quick-start) of the ESP8266WiFi library documentation we have discussed how to check when module connects to the Wi-Fi network. We were waiting until connection is established. If network is not available, the module could wait like that for ever doing nothing else. Another [example](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/scan-examples.md#async-scan) on the Wi-Fi asynchronous scan mode demonstrated how to wait for scan result and do in parallel something else - blink a LED not disturbing the blink pattern. Let’s apply similar functionality when connecting the module to an access point.

У першому прикладі документації бібліотеки ESP8266WiFi було обговорено, як перевірити, коли модуль під єднається до мережі Wi-Fi, і поки з'єднання не було встановлено, ми просто очікували. У випадку коли мережа не доступна, то модуль може чекати вічно. Інший приклад режиму асинхронного сканування Wi-Fi показав, як можна чекати результату сканування та паралельно зробити щось інше - блимати світлодіодом точно в такт. Давайте застосуємо подібну функціональність при підключенні модуля до точки доступу.

## Introduction

In example below we will show another cool example of getting ESP perform couple of tasks at the same time and with very little programming.

У наведеному нижче прикладі ми покажемо як налаштувати ESP виконувати декілька завдань одночасно з мінімальним програмним кодом.

## What are the Tasks?

We would like to write a code that will inform us that connection to Wi-Fi network has been established or lost. At the same time we want to perform some time critical task. We will simulate it with a blinking LED. Generic class provides specific, event driven methods, that will be executed asynchronously, depending on e.g. connection status, while we are already doing other tasks.

Уявімо, що нам треба написати код, який повідомлятиме нам про встановлення або загублення з'єднання з мережею Wi-Fi, а також у той же час буде виконувати декілька завдань критичних за часом? Змоделюємо ці завдання миготінням світлодіодом. Клас Generic надає конкретні методи, керування подіями, які будуть виконуватися асинхронно, наприклад, залежно від статусу підключення, в той час як виконуватимуться інші паралельні завдання.

## Event Driven Methods

The list of all such methods is provided in [Generic Class](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/generic-class.md) documentation.

Список усіх таких методів наведено в першій частині цієї.

We would like to use two of them: \* onStationModeGotIP called when station is assigned IP address. This assignment may be done by DHCP client or by executing WiFi.config(...). \* onStationModeDisconnected called when station is disconnected from Wi-Fi network. The reason of disconnection does not matter. Event will be triggered both if disconnection is done from the code by executing WiFi.disconnect(), because the Wi-Fi signal is weak, or because the access point is switched off.

Ми хотіли б використовувати два з них:

onStationModeGotIP викликається, коли станції призначена IP-адреса. Це призначення може виконуватись DHCP-клієнтом або викликом WiFi.config(...).

OnStationModeDisconnected викликається, коли станція відключена від мережі Wi-Fi. Причина відключення не має значення. Подія буде активована як при добровільному відключенні за допомогою – WiFi.disconnect() (наприклад, сигнал Wi-Fi є слабким), так і при зникненні точки доступу.

## Register the Events

To get events to work we need to complete just two steps:

Щоб події працювали, нам потрібно виконати лише два етапи:

1. Оголосити обробника події:

WiFiEventHandler disconnectedEventHandler;

1. Виберіть конкретну подію (у даному випадку StationModeDisconnected) і додайте код, який буде виконуватися, коли подія настане.

disconnectedEventHandler = WiFi.onStationModeDisconnected([](const WiFiEventStationModeDisconnected& event)   { Serial.println("Station disconnected"); });

If this event is fired the code will print out information that station has been disconnected.

Якщо ця подія настала, код буде роздруковувати інформацію про те, що станція була від'єднана.

That’s it. It is all we need to do.

Це все, що нам потрібно зробити.

## The Code

The complete code, including both methods discussed at the beginning, is provided below.

Повний код, включаючи обидва методи, що розглядаються на початку, наведено нижче.

#include <ESP8266WiFi.h>

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

const char\* password = "\*\*\*\*\*\*\*\*";

WiFiEventHandler gotIpEventHandler, disconnectedEventHandler;

bool ledState;

void setup()

{

Serial.begin(115200);

Serial.println();

pinMode(LED\_BUILTIN, OUTPUT);

gotIpEventHandler = WiFi.onStationModeGotIP([](const WiFiEventStationModeGotIP& event)

{

Serial.print("Station connected, IP: ");

Serial.println(WiFi.localIP());

});

disconnectedEventHandler = WiFi.onStationModeDisconnected([](const WiFiEventStationModeDisconnected& event)

{

Serial.println("Station disconnected");

});

Serial.printf("Connecting to %s ...\n", ssid);

WiFi.begin(ssid, password);

}

void loop()

{

digitalWrite(LED\_BUILTIN, ledState);

ledState = !ledState;

delay(250);

}

## Check the Code

After uploading above sketch and opening a serial monitor we should see a similar log:

Після завантаження скетчу, цього прикладу, та відкриття послідовного монітора ми повинні побачити подібний журнал подій:

Connecting to sensor**-**net **...**

Station connected, IP: 192.168**.**1.10

If you switch off the access point, and put it back on, you will see the following:

Якщо ви вимкнете точку доступу і знову ввімкнете його, ви побачите таке:

Station disconnected

Station disconnected

Station disconnected

Station connected, IP: 192.168**.**1.10

The process of connection, disconnection and printing messages is done in background of the loop() that is responsible for blinking the LED. Therefore the blink pattern all the time remains undisturbed.

Процес з'єднання, відключення та друкування повідомлень здійснюється у фоновому режимі циклу loop(), який відповідає за мигання світлодіодів. Тому модель мерехтіння, весь час залишається непорушненою.

## Conclusion

Check out events from generic class. They will help you to write more compact code. Use them to practice splitting your code into separate tasks that are executed asynchronously.

Передивіться всі події з класу Generic. Вони допоможуть вам написати більш компактний код. Використовуйте їх для практики розбиття вашого коду на окремі завдання, які виконуються асинхронно.

For review of functions included in generic class, please refer to the [Generic Class](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/generic-class.md) documentation.