# Client Class

Methods documented for [Client](https://www.arduino.cc/en/Reference/WiFiClientConstructor) in [Arduino](https://github.com/arduino/Arduino)

Методи для класу [Client](https://www.arduino.cc/en/Reference/WiFiClientConstructor) документовані в [Arduino](https://github.com/arduino/Arduino)

1. [WiFiClient()](https://www.arduino.cc/en/Reference/WiFiClient)

#### Створює об’єкт клієнта, який може підключитися до вказаної IP-адреси та порту Інтернету, як це визначено в connect() нижче.

WiFiClient client;

1. [connected()](https://www.arduino.cc/en/Reference/WiFiClientConnected)

#### Для перевірки чи підключений клієнт. Зверніть увагу, що клієнт вважається з'єднаним, якщо з'єднання було закрито, але є ще непрочитані дані.

client.connected()

Повертає true, якщо клієнт підключений, false, якщо ні.

1. [connect()](https://www.arduino.cc/en/Reference/WiFiClientConnect)

Для підключення до вказаних IP-адреси та порту.

client.connect(ip, port)

#### Також існує перевантаження, яке підтримує пошук по DNS (доменного імені), наприклад, google.com.

client.connect(URL, port)

Зміст параметрів – наступний:

ip: IP-адреса, до якої клієнт підключиться (масив з 4 байтів)

URL: ім'я домену, до якого клієнт підключиться, типу String , наприклад: "carduino.cc"

port: порт, до якого клієнт підключиться, типу int

Повертає true, якщо з'єднання виконується, інакше false.

1. [write()](https://www.arduino.cc/en/Reference/WiFiClientWrite)

Write data to the server the client is connected to.

Для передачі даних на сервер, до якого клієнт підключений.

client.write(data)

Змінна – data це один байт інформації типу byte або char.

Повертає кількість написаних символів в типі byte.

1. [print()](https://www.arduino.cc/en/Reference/WiFiClientPrint)

Print data to the server that a client is connected to. Prints numbers as a sequence of digits, each an ASCII character (e.g. the number 123 is sent as the three characters '1', '2', '3').

Для надсилання даних у вигляді масиву символів на сервер, до якого підключено клієнт. Цей метод буде надсилати числа як послідовність символів ASCII (наприклад, номер 123 надсилається як три символи '1', '2', '3').

client.print(data)

Існує перевантаження функції яке задає формат виводу чисел, таких як DEC, OCT, HEX, BIN.

client.print(data, BASE)

Зміст параметрів – наступний:

data: дані для друку (char, byte, int, long, string)

BASE – база, в якій друкуватимуться номери : DEC для десяткової (база 10), OCT для вісімкової (база 8), HEX для шістнадцяткового (база 16), BIN для двійкової (база 2).

Повертає кількість написаних символів в типі byte.

1. [println()](https://www.arduino.cc/en/Reference/WiFiClientPrintln)

Аналогічний сенс що і в print() з відмінністю лише в тому, що в кінець повідомлення додається символ нового рядку ‘\n’.

#### Може використовуватися трьома різними перевантаженнями, наприклад:

client.println()   
client.println(data)   
client.println(data, BASE)

data: дані для друку (char, byte, int, long, string)

BASE – база, в якій друкуватимуться номери : DEC для десяткової (база 10), OCT для вісімкової (база 8), HEX для шістнадцяткового (база 16), BIN для двійкової (база 2).

Повертає кількість написаних символів в типі byte.

1. [available()](https://www.arduino.cc/en/Reference/WiFiClientAvailable)

Щоб дізнатися кількість байтів, доступних для читання (тобто кількість даних, написаних клієнтом на сервері, до якого він підключений).

available() успадкований від допоміжного класу [Stream](https://www.arduino.cc/en/Reference/Stream).

client.available()

#### Повертає кількість доступних байтів.

1. [read()](https://www.arduino.cc/en/Reference/WiFiClientRead)

Читає наступний байт, отриманий від сервера, до якого клієнт підключений (після останнього виклику read()).

read() успадкований від допоміжного класу [Stream](https://www.arduino.cc/en/Reference/Stream).

client.read()

#### Повертає наступний байт (char / byte), або -1, якщо ніхто не доступний.

1. [flush()](https://www.arduino.cc/en/Reference/WiFiClientFlush)

#### Description

Discard any bytes that have been written to the client but not yet read.

Для відмови від будь-яких байтій, написаних клієнту, але ще не прочитаних.

flush() успадкований від допоміжного класу [Stream](https://www.arduino.cc/en/Reference/Stream).

client.flush()

#### Нічого не повертає.

1. [stop()](https://www.arduino.cc/en/Reference/WiFIClientStop)

Для від’єднання від серверу.

client.stop()

Нічого не поветає

Methods and properties described further down 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.

## setNoDelay

setNoDelay(nodelay)

With nodelay set to true, this function will to disable [Nagle algorithm](https://en.wikipedia.org/wiki/Nagle%27s_algorithm).

Якщо nodelay налаштовано на значення true, ця функція відключить [Nagle algorithm](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9D%D0%B5%D0%B9%D0%B3%D0%BB%D0%B0).

This algorithm is intended to reduce TCP/IP traffic of small packets sent over the network by combining a number of small outgoing messages, and sending them all at once. The downside of such approach is effectively delaying individual messages until a big enough packet is assembled.

Цей алгоритм призначений для зменшення трафіку TCP/IP для невеликих пакетів, що надсилаються по мережі, об'єднуючи їх в ряд невеликих вихідних повідомлень і відправляючи їх усім одночасно. Недоліком такого підходу є затримка окремих повідомлень, допоки не буде зібрано досить великий пакет.

Приклад:

client**.**setNoDelay(true);

## Other Function Calls

uint8\_t status ()

virtual size\_t write (const uint8\_t **\***buf, size\_t size)

size\_t write\_P (PGM\_P buf, size\_t size)

size\_t write (Stream **&**stream)

size\_t write (Stream **&**stream, size\_t unitSize) \_\_attribute\_\_((deprecated))

virtual int read (uint8\_t **\***buf, size\_t size)

virtual int peek ()

virtual size\_t peekBytes (uint8\_t **\***buffer, size\_t length)

size\_t peekBytes (char **\***buffer, size\_t length)

virtual operator bool ()

IPAddress remoteIP ()

uint16\_t remotePort ()

IPAddress localIP ()

uint16\_t localPort ()

bool getNoDelay ()

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 WiFiClient (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 :arrow\_right:](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/client-examples.md) dedicated specifically to the Client Class.

**EXAMPLES**

**Client**

Let’s write a simple client program to access a single web page and display its contents on a serial monitor. This is typical operation performed by a client to access server’s API to retrieve specific information. For instance we may want to contact GitHub’s API to periodically check the number of open issues reported on [esp8266/Arduino](https://github.com/esp8266/Arduino/issues) repository.

Давайте напишемо просту програму для доступу до однієї веб-сторінки та відображемо її вміст на послідовному моніторі. Це типова операція, яку виконує клієнт для доступу до API сервера для отримання конкретної інформації. Наприклад, ми можемо зв'язатися з API GitHub, щоб періодично перевіряти кількість відкритих проблем, повідомлених в репозиторії [esp8266/Arduino](https://github.com/esp8266/Arduino/issues).

**Introduction**

This time we are going to concentrate just on retrieving a web page contents sent by a server, to demonstrate basic client’s functionality. Once you are able to retrieve information from a server, you should be able to phrase it and extract specific data you need.

Цього разу ми збираємося зосередити увагу лише на отриманні вмісту веб-сторінки, надісланого сервером, для демонстрації функціональності основного клієнта. Після того, як ви можете отримати інформацію з сервера, ви повинні бути в змозі сформулювати його і витягти конкретні дані, які необхідні.

**Get Connected to Wi-Fi**

We should start with connecting the module to an access point to obtain an access to internet. The code to provide this functionality has been already discussed in chapter [Quick Start](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/readme.md#quick-start). Please refer to it for details.

Почнемо з підключення модуля до точки доступу, щоб отримати доступ до Інтернету. Код для надання цієї функції вже обговорюється в розділі "Quick Start". Передивіться його для додаткових деталей.

**Select a Server**

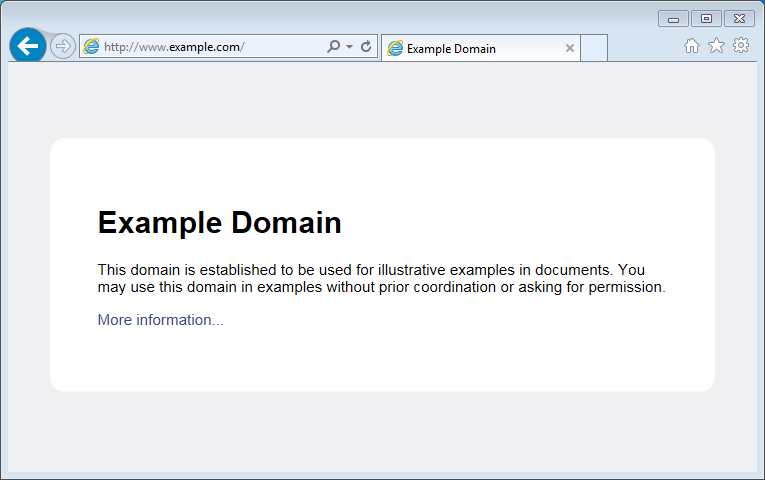
Once connected to the network we should connect to the specific server. Web address of this server is declared in host character string as below.

Після підключення до мережі ми повинні підключитися до певного сервера. Веб-адреса цього сервера оголошується в рядку символів host, як показано нижче.

const char**\*** host **=** "www.example.com";

I have selected www.example.com domain name and you can select any other. Just check if you can access it using a web browser.

Я вибрав ім'я домену www.example.com, а ви можете обрати будь-який інший. Просто перевірте, чи можете ви отримати доступ до нього за допомогою веб-браузера.



*alt text*

**Instantiate the Client**

Now we should declare a client that will be contacting the host (server):

Тепер ми повинні оголосити клієнта, який зв'яжеться з хостом (сервером):

WiFiClient client;

**Get Connected to the Server**

In next line we will connect to the host and check the connection result. Note 80, that is the standard port number used for web access.

У наступному рядку ми зв'яжемося з хостом і перевіримо результат підключення. Примітка, 80 це стандартний номер порту, який використовується для доступу до Інтернету.

if (client.connect(host, 80))

{

// we are connected to the host!

}

else

{

// connection failure

}

**Request the Data**

If connection is successful, we should send request the host to provide specific information we need. This is done using the [HTTP GET](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods) request as in the following lines:

Якщо з'єднання успішне, ми повинні надіслати запит хосту, надати конкретну інформацію, яка нам потрібна. Це робиться за допомогою запиту HTTP GET, як у наступних рядках:

client**.**print(String("GET /") **+** " HTTP/1.1\r\n" **+**

"Host: " **+** host **+** "\r\n" **+**

"Connection: close\r\n" **+**

"\r\n"

);

**Read Reply from the Server**

Then, while connection by our client is still alive (while (client.connected()), see below) we can read line by line and print out server’s response:

Потім, в той час як зв'язок нашого клієнта все ще живий (while (client.connected()), дивіться нижче) ми можемо читати по рядках та друкувати відповідь сервера:

**while** (client**.**connected())

{

**if** (client**.**available())

{

String line **=** client**.**readStringUntil('\n');

Serial**.**println(line);

}

}

The inner if (client.available()) is checking if there are any data available from the server. If so, then they are printed out.

Внутрішня структура if (client.available ()) перевіряє наявність будь-яких даних з сервера. Якщо відповідь true, то вони роздруковуються.

Once server sends all requested data it will disconnect and program will exit the while loop.

Після того, як сервер надішле всі запрошені дані, він буде від'єднаний, і програма завершить цикл while.

**Now to the Sketch**

Complete sketch, including a case when contention to the server fails, is presented below.

Нижче наведено повний приклад, у тому числі випадок, коли сервер вийде з ладу.

*#include <ESP8266WiFi.h>*

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

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

const char**\*** host **=** "www.example.com";

void setup()

{

Serial**.**begin(115200);

Serial**.**println();

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

WiFi**.**begin(ssid, password);

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

{

delay(500);

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

}

Serial**.**println(" connected");

}

void loop()

{

WiFiClient client;

Serial**.**printf("\n[Connecting to %s ... ", host);

**if** (client**.**connect(host, 80))

{

Serial**.**println("connected]");

Serial**.**println("[Sending a request]");

client**.**print(String("GET /") **+** " HTTP/1.1\r\n" **+**

"Host: " **+** host **+** "\r\n" **+**

"Connection: close\r\n" **+**

"\r\n"

);

Serial**.**println("[Response:]");

**while** (client**.**connected())

{

**if** (client**.**available())

{

String line **=** client**.**readStringUntil('\n');

Serial**.**println(line);

}

}

client**.**stop();

Serial**.**println("\n[Disconnected]");

}

**else**

{

Serial**.**println("connection failed!]");

client**.**stop();

}

delay(5000);

}

**Test it Live**

Upload sketch the module and open serial monitor. You should see a log similar to presented below.

Завантажте скетч в модуль та відкрийте послідовний монітор. Ви повинні побачити журнал, подібний до наведеного нижче.

First, after establishing Wi-Fi connection, you should see confirmation, that client connected to the server and send the request:

По-перше, після встановлення Wi-Fi-з'єднання, ви повинні побачити підтвердження, що клієнт підключений до сервера та надсилає запит:

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

[Connecting to www**.**example**.**com **...** connected]

[Sending a request]

Then, after getting the request, server will first respond with a header that specifies what type of information will follow (e.g. Content-Type: text/html), how long it is (like Content-Length: 1270), etc.:

Тоді, після отримання запиту, сервер спочатку відповість заголовком, який визначає, який тип інформації буде слідувати (наприклад, Content-Type: text/html), як довго (наприклад, Content-Length: 1270) та ін .:

[Response:]

HTTP**/**1.1 200 OK

Cache**-**Control: max**-**age**=**604800

Content**-**Type: text**/**html

Date: Sat, 30 Jul 2016 12:30:45 GMT

Etag: "359670651+ident"

Expires: Sat, 06 Aug 2016 12:30:45 GMT

Last**-**Modified: Fri, 09 Aug 2013 23:54:35 GMT

Server: ECS (ewr**/**15BD)

Vary: Accept**-**Encoding

X**-**Cache: HIT

x**-**ec**-**custom**-**error: 1

Content**-**Length: 1270

Connection: close

End of header is marked with an empty line and then you should see the HTML code of requested web page.

Кінець заголовка позначено порожньою лінією, після чого ви повинні побачити HTML-код потрібної веб-сторінки.

<!doctype html>

<html>

<head>

<title>Example Domain</title>

<meta charset="utf-8" />

<meta http-equiv="Content-type" content="text/html; charset=utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<style type="text/css">

(...)

</head>

<body>

<div>

<h1>Example Domain</h1>

<p>This domain is established to be used for illustrative examples in documents. You may use this

domain in examples without prior coordination or asking for permission.</p>

<p><a href="http://www.iana.org/domains/example">More information...</a></p>

</div>

</body>

</html>

[Disconnected]

**Test it More**

In case server’s web address is incorrect, or server is not accessible, you should see the following short and simple message on the serial monitor:

Якщо веб-адреса сервера невірна або сервер недоступний, ви повинні побачити таке коротке та просте повідомлення на послідовному моніторі:

Connecting to sensor-net ........ connected

[Connecting to www.wrong-example.com ... connection failed!]

**Conclusion**

With this simple example we have demonstrated how to set up a client program, connect it to a server, request a web page and retrieve it. Now you should be able to write your own client program for ESP8266 and move to more advanced dialogue with a server, like e.g. using [HTTPS](https://en.wikipedia.org/wiki/HTTPS) protocol with the [Client Secure](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/client-secure-examples.html) .

За допомогою цього простого прикладу ми продемонстрували, як налаштувати клієнтську програму, підключити її до сервера, запитати веб-сторінку та отримати її. Тепер ви маєте можливість написати власну клієнтську програму для ESP8266 і перейти до більш розширеного діалогу з сервером, наприклад, використовуючи протокол [HTTPS](https://uk.wikipedia.org/wiki/HTTPS) з клієнтом Secure.

For more client examples please check

Для інших клієнтських прикладів, будь ласка, перегляньте:

* [WiFiClientBasic.ino](https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/examples/WiFiClientBasic/WiFiClientBasic.ino) - простий скетч, який надсилає повідомлення на сервер TCP
* [WiFiClient.ino](https://github.com/esp8266/Arduino/blob/master/libraries/ESP8266WiFi/examples/WiFiClient/WiFiClient.ino) - цей скетч надсилає дані через запити HTTP GET на службу data.sparkfun.com.

For the list of functions provided to manage clients, please refer to the [Client Class :arrow\_right:](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/client-class.md) documentation.