**UDP Class**

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

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

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

Ініціалізує бібліотеку WiFi UDP та мережеві налаштування. Запускає сокет WiFiUDP, слухаючи на локальному порту PORT \*/

WiFiUDP.begin(port);

**Змінна –** port: це номер локального порту для прослуховування, типу int

Повертає true якщо порт успішно встановлений, та false якщо немає доступних портів для використання

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

Для отримання кількості байтів (символів), доступних для читання з буфера. Це дані, які вже надійшли.

Цю функцію можна успішно викликати лише після Wi-FiUDP.parsePacket().

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

WiFiUDP.available()

Повертає кількість байтів, доступних у поточному пакеті, у разі повернення 0 буде означати, що метод parsePacket() ще не був викликаний.

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

Запускає з'єднання для запису даних UDP до віддаленого підключення по імені хосту, у вигляді символьного рядка, та номеру його порту.

WiFiUDP.beginPacket(hostName, port);

Існує перевантаження функції де треба вказати IP-адресу та номер порту.

WiFiUDP.beginPacket(hostIp, port);

Зміст параметрів – наступний:

hostName: адреса віддаленого хосту у вигляді символьного рядка

hostIp: IP-адреса віддаленого з'єднання (4 байти)

port: порт віддаленого з'єднання типу int

Повертає true у разі успіху та false якщо виникла проблема з наданою IP-адресою або портом.

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

Викликається після запису даних UDP на віддалене з'єднання. Ця команда завершує пакет і відправляє його.

WiFiUDP.endPacket();

Повертає true у разі успіху та false якщо сталася помилка.

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

Writes UDP data to the remote connection. Must be wrapped between [beginPacket](https://www.arduino.cc/en/Reference/WiFiUDPBeginPacket)() and [endPacket](https://www.arduino.cc/en/Reference/WiFiUDPEndPacket)(). beginPacket() initializes the packet of data, it is not sent until endPacket() is called.

Записує дані UDP на віддалене з'єднання. Повинна викликатися між функціями – startPacket() і endPacket(). Функція beginPacket() ініціалізує пакет даних, що не відправляється, а очікує на виклик – endPacket().

Існує два перевантаження. Перше просто записує до пакету байт інформації:

WiFiUDP.write(byte);

Друге записує буфер символьного рядка конкретного розміру:

WiFiUDP.write(buffer, size);

Зміст параметрів – наступний:

byte: вихідний байт корисної інформації

buffer: вихідне повідомлення, складене з символів

size: розмір буфера повідомлення, пам’ятайте символьний рядок може закінчуватися символом переходу '\n'.

Повертає один байт у пакеті або розмір буфера в пакеті, в залежності від перевантаженої функції.

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

Для початку читання наступного доступного пакету, parsePacket() перевіряє наявність пакету UDP, і повідомляє про його розмір. ЇЇ слід викликати перед читанням буфера функцією – UDP.read().

UDP.parsePacket();

Повертає розмір пакета в байтах, або 0 якщо немає доступних пакетів.

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

Читає байт з файлу, не переходячи до наступного. Тобто, послідовні виклики для peek() повертають однин й той самий байт, що й наступний виклик read().

Ця функція успадковується від допоміжного класу [Stream](https://www.arduino.cc/en/Reference/Stream).

WiFiUDP.peek()

Повертає наступний байт або символ, або -1, якщо байт не доступний

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

Ця функція зчитує вхідне повідомлення і повинна бути викликана лише після Wi-FiUDP.parsePacket().

Існує два перевантаження цієї функції. Перше зчитує дані UDP в вказаний буфер.

WiFiUDP.read(buffer, len);

Друге – не має аргументів, і повертає наступний символ у буфері пакету.

WiFiUDP.read();

Зміст параметрів – наступний:

buffer: буфер типу char\* для зберігання вхідних пакетів

len: максимальний розмір буфера, типу int

Повертає кількість байт записаних в буфер або при другому перевантажені – наступний символ в буфері, і в обох випадках -1, якщо буфер скінчився

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

Відхиляє будь-які байти, написані клієнту, які ще не прочитані.

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

WiFiUDP.flush()

Нічого не повертає.

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

Для від’єднання від серверу. Звільнення будь-якого ресурсу, який використовується під час сеансу UDP.

WiFiUDP.stop()

Нічого не повертає.

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

Отримує IP-адресу віддаленого з'єднання. Ця функція повинна бути викликана після Wi-FiUDP.parsePacket().

WiFiUDP.remoteIP();

Повертає 4 байти: IP-адреси хоста, який відправив поточний вхідний пакет.

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

Gets the port of the remote UDP connection.

Отримує порт віддаленого UDP-з'єднання. Ця функція повинна бути викликана після Wi-FiUDP.parsePacket().

UDP.remotePort();

Повернення порту хосту, який відправив поточний вхідний пакет

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.

**Multicast UDP**

uint8\_t beginMulticast (IPAddress interfaceAddr, IPAddress multicast, uint16\_t port)

virtual int beginPacketMulticast (IPAddress multicastAddress, uint16\_t port, IPAddress interfaceAddress, int ttl**=**1)

IPAddress destinationIP ()

uint16\_t localPort ()

The WiFiUDP class supports sending and receiving multicast packets on STA interface. When sending a multicast packet, replace  udp.beginPacket(addr, port)  with  udp.beginPacketMulticast(addr, port, WiFi.localIP()). When listening to multicast packets, replace  udp.begin(port)  with  udp.beginMulticast(WiFi.localIP(), multicast\_ip\_addr, port). You can use  udp.destinationIP()  to tell whether the packet received was sent to the multicast or unicast address.

Клас WiFiUDP підтримує надсилання та отримання багатоадресних пакетів на інтерфейсі STA. При відправці багатоадресного пакету замініть udp.beginPacket(addr, port) на udp.beginPacketMulticast(addr, port, WiFi.localIP ()). Під час прослуховування пакетів багатоадресного замініть udp.begin(port) на udp.beginMulticast(WiFi.localIP (), multicast\_ip\_addr, port). Ви можете використовувати udp.destinationIP(), щоб визначити, чи отриманий пакет був надісланий на адресу багатоадресної або одноадресної пошти.

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

**EXAMPLES**

# UDP

The purpose of example application below is to demonstrate UDP communication between ESP8266 and an external client. The application (performing the role of a server) is checking inside the loop() for an UDP packet to arrive. When a valid packet is received, an acknowledge packet is sent back to the client to the same port it has been sent out.

## Declarations

At the beginning of sketch we need to include two libraries:

*#include <ESP8266WiFi.h>*

*#include <WiFiUdp.h>*

The first library ESP8266WiFi.h is required by default if we are using ESP8266’s Wi-Fi. The second one WiFiUdp.h is needed specifically for programming of UDP routines.

Once we have libraries in place we need to create a WiFiUDPobject. Then we should specify a port to listen to incoming packets. There are conventions on usage of port numbers, for information please refer to the [List of TCP and UDP port numbers](https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers). Finally we need to set up a buffer for incoming packets and define a reply message.

WiFiUDP Udp;

unsigned int localUdpPort **=** 4210;

char incomingPacket[255];

char replyPacekt[] **=** "Hi there! Got the message :-)";

## Wi-Fi Connection

At the beginning of setup() let’s implement typical code to connect to an access point. This has been discussed in [Quick Start](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/readme.md#quick-start). Please refer to it if required.

## UDP Setup

Once connection is established, you can start listening to incoming packets.

Udp**.**begin(localUdpPort);

That is all required preparation. We can move to the loop()that will be handling actual UDP communication.

## An UDP Packet Arrived!

Waiting for incoming UDP packed is done by the following code:

int packetSize **=** Udp**.**parsePacket();

**if** (packetSize)

{

Serial**.**printf("Received %d bytes from %s, port %d\n", packetSize, Udp**.**remoteIP()**.**toString()**.**c\_str(), Udp**.**remotePort());

int len **=** Udp**.**read(incomingPacket, 255);

**if** (len **>** 0)

{

incomingPacket[len] **=** 0;

}

Serial**.**printf("UDP packet contents: %s\n", incomingPacket);

(**...**)

}

Once a packet is received, the code will printing out the IP address and port of the sender as well as the length of received packet. If the packet is not empty, its contents will be printed out as well.

## An Acknowledge Send Out

For each received packet we are sending back an acknowledge packet:

Udp**.**beginPacket(Udp**.**remoteIP(), Udp**.**remotePort());

Udp**.**write(replyPacekt);

Udp**.**endPacket();

Please note we are sending reply to the IP and port of the sender by using Udp.remoteIP() and Udp.remotePort().

## Complete Sketch

The sketch performing all described functionality is presented below:

*#include <ESP8266WiFi.h>*

*#include <WiFiUdp.h>*

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

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

WiFiUDP Udp;

unsigned int localUdpPort **=** 4210; **//** local port to listen on

char incomingPacket[255]; **//** buffer **for** incoming packets

char replyPacekt[] **=** "Hi there! Got the message :-)"; **//** a reply string to send back

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");

Udp**.**begin(localUdpPort);

Serial**.**printf("Now listening at IP %s, UDP port %d\n", WiFi**.**localIP()**.**toString()**.**c\_str(), localUdpPort);

}

void loop()

{

int packetSize **=** Udp**.**parsePacket();

**if** (packetSize)

{

**//** receive incoming UDP packets

Serial**.**printf("Received %d bytes from %s, port %d\n", packetSize, Udp**.**remoteIP()**.**toString()**.**c\_str(), Udp**.**remotePort());

int len **=** Udp**.**read(incomingPacket, 255);

**if** (len **>** 0)

{

incomingPacket[len] **=** 0;

}

Serial**.**printf("UDP packet contents: %s\n", incomingPacket);

**//** send back a reply, to the IP address **and** port we got the packet **from**

Udp.beginPacket(Udp**.**remoteIP(), Udp**.**remotePort());

Udp**.**write(replyPacekt);

Udp**.**endPacket();

}

}

## How to Check It?

Upload sketch to module and open serial monitor. You should see confirmation that ESP has connected to Wi-Fi and started listening to UDP packets:

Connecting to twc**-**net**-**3 **........** connected

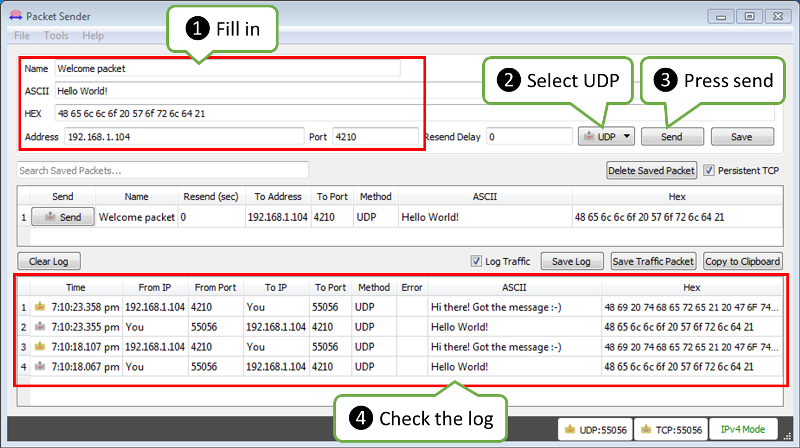
Now listening at IP 192.168**.**1.104, UDP port 4210

Now we need another application to send some packets to IP and port shown by ESP above.

Instead of programming another ESP, let’s make it easier and use a purpose build application. I have selected the [Packet Sender](https://packetsender.com/download). It is available for popular operating systems. Download, install and execute it.

Once Packet Sender’s window show up enter the following information: \* Name of the packet \* ASCII text of the message to be send inside the packet \* IP Address shown by our ESP \* Port shown by the ESP \* Select UDP

What I have entered is shown below:



*alt text*

Now click Send.

Immediately after that you should see the following on ESP’s serial monitor:

Received 12 bytes from 192.168.1.106, port 55056

UDP packet contents: Hello World!

The text 192.168.1.106, port 55056 identifies a PC where the packet is send from. You will likely see different values.

As ESP sends an acknowledge packet back, you should see it in the log in the bottom part of the Packet Sender’s window.

## Conclusion

This simple example shows how to send and receive UDP packets between ESP and an external application. Once tested in this minimal set up, you should be able to program ESP to talk to any other UDP device. In case of issues to establish communication with a new device, use the [Packet Sender](https://packetsender.com/) or other similar program for troubleshooting

For review of functions provided to send and receive UDP packets, please refer to the [UDP Class](http://arduino-esp8266.readthedocs.io/en/2.4.0/esp8266wifi/udp-class.html) documentation.