# Бібліотеки

## WiFi (ESP8266WiFi library)

## Бібліотека [ESP8266WiFi](https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266WiFi) була розроблена на базі ESP8266 SDK, використовуючи конвенцію іменування та загальну філософію функціональності бібліотеки [Arduino WiFi Shield](https://www.arduino.cc/en/Reference/WiFi). З часом особливості функціоналу WiFi пакету ESP8266 SDK були перенесені до цієї бібліотеки. Відповідно це бібліотека стала більш насичена аніж [Arduino WiFi Shield](https://www.arduino.cc/en/Reference/WiFi) тому для неї було написано додаткову документацію – [ESP8266WiFi library documentation](http://arduino-esp8266.readthedocs.io/en/stable/esp8266wifi/readme.html).

## На тему WiFi далі буде виділено декілька уроків.

## Ticker

## Ця бібліотека призначена для програмного переривання за таймером.

## Щоб підключити переривання, для початку треба підключити бібліотеку за допомогою наступного рядка:

#include <Ticker.h>

## Після чого створюємо один або декілька екземплярів, залежно від кількості необхідних переривань.

Ticker name, name2;

## Для створених екземплярів name та name2 ми можемо підключити періодичні переривання, за допомогою двох функцій, які в свою чергу мають два [перевантаження](https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B2%D0%B0%D0%BD%D1%82%D0%B0%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%97):

## Для періодичності в секундах без аргументів функції callback:

name.**attach**(seconds, callback);

## Для періодичності в секундах з аргументом arg для функції callback:

name.**attach**(seconds, callback, arg);

## Для періодичності в мілісекундах без аргументів функції callback:

name.**attach\_ms**(miliseconds, callback);

## Для періодичності в мілісекундах з аргументом arg для функції callback:

name.**attach\_ms**(miliseconds, callback, arg);

## Як ви вже здогадалися callback це ім’я нашої функції яка буде викликатися при перериванні, а її аргумент arg може бути лише один.

## Для підключення переривання лиш раз треба використовувати функції, що дуже подібні до попередніх:

name.**once**(seconds, callback);

name.**once**(seconds, callback, arg);

name.**once\_ms**(miliseconds, callback);

name.**once\_ms**(miliseconds, callback, arg);

## Параметри seconds та miliseconds – це час, з моменту виклику функції підключення, до переривання.

## Існує ще дві не менш важливі функції – detach, що відключає переривання, та active, що повертає відомість активності переривання, логічного типу (bool).

name.detach();

bool check = name.active();

|  |
| --- |
|  |
|  |  |
|  |  |

Не рекомендується використовувати функцію callback (зворотного виклику) для блокування операцій входу/виходу (мережі, послідовного порту, файлу)!

Також існує бібліотека [TickerScheduler](https://github.com/Toshik/TickerScheduler) що базується на Ticker і дозволяє працювати з Task та допомагає уникнути проблем з WDT (сторожовий таймер).

Список посилань: https://github.com/esp8266/Arduino/tree/master/libraries/Ticker

## EEPROM

## Перед початком зчитування чи запису в енергонезалежну пам’ять вам потрібно викликати – EEPROM.begin(size), де size - це кількість байтів, які ви хочете використовувати. Розмір може варіюватися від 4 до 4096 байт.

Для запису в EEPROM пам'ять треба спочатку викликати команду EEPROM.write(address, value) , де змінна – address є адресою байту і задається в межах від 0 до size, а змінна value – це байт інформації. Команда write не виконує запис, а лише готує масив байтів до запису, тому після підготовки всіх потрібних даних треба викликати функцію EEPROM.commit() , або EEPROM.end() , що здійснюють сам запис, різниця цих команд в тому що остання звільняє RAM пам’ять, тому щоб потім продовжити працювати з EEPROM вам треба буде знову викликати EEPROM.begin(size).

Щоб зчитати дані з EEPROM застосуйте команду EEPROM.read(address) , де змінна – address є адресою байту і задається в межах від 0 до size, що повертає значення uint8\_t ,що еквівалентно byte.

В бібліотеці є ще декілька корисних функцій для легшої взаємодії…, але розглянутих функції в повній мірі вистачить для роботи з EEPROM пам’яттю.

Бібліотека EEPROM використовує один сектор flash пам’яті, що розташований відразу після SPIFFS.

Список посилань: https://github.com/esp8266/Arduino/tree/master/libraries/EEPROM

## I2C (Wire library)

## Для цього інтерфейсу реалізовано лише режим ведучого (master), частота орієнтовно до 450 кГц. Перед використанням шини I2C, потрібно вибрати контакти SDA і SCL шляхом виклику функції Wire.begin(int sda, int scl), наприклад Wire.begin(0, 2) для модуля ESP-01. Для інших модулів піни за замовчуванням 4 (SDA) та 5 (SCL).

## SPI

## Бібліотека SPI підтримує весь Arduino SPI API, включаючи транзакції, в тому числі фазу синхронізації (CPHA). Clock polarity (CPOL) поки не підтримується (SPI\_MODE2 і SPI\_MODE3 не працюють).

Зазвичай SPI піни це:

* MOSI = GPIO13
* MISO = GPIO12
* SCLK = GPIO14

Існує розширений режим, за допомогою якого ви можете переключитися на SPI0. Це досягається шляхом виклику SPI.pins(6, 7, 8, 0) перед функцією SPI.begin(). Контакти зміняться на:

* MOSI = SD1
* MISO = SD0
* SCLK = CLK
* HWCS = GPIO0

Цей режим поділяє контакти SPI з контролером, який зчитує код програми з flash пам’яті та керується апаратним арбітром (flash пам’ять завжди має вищий пріоритет). Для цього режиму CS буде керуватися апаратним способом, оскільки ви не можете обробляти лінію CS з GPIO, ви ніколи не знаєте, коли арбітр буде надавати вам доступ до шини, так що ви повинні дозволити йому обробляти CS автоматично.

Список посилань: <https://github.com/esp8266/Arduino/tree/master/libraries/SPI>

P.S. Ви також можете переглянути приклад програми для світлодіодної матриці в директорії – EXAMPLES.

## SoftwareSerial

## Порт ESP8266 бібліотеки SoftwareSerial підтримує швидкість передачі до 115200 та множинні екземпляри SoftwareSerial.

## Список посилань: https://github.com/plerup/espsoftwareserial

## ESP-specific APIs

## Деякі API специфічних можливостей ESP, пов'язані з режимом глибокого сну, RTC (точного часу) і флеш-пам'яті доступні в об'єкті – ESP. Так, наприклад, функція ESP.deepSleep(microseconds, mode) переводить модуль в режим глибокого сну. Параметр mode може приймати значення: WAKE\_RF\_DEFAULT,  WAKE\_RFCAL,  WAKE\_NO\_RFCAL,  WAKE\_RF\_DISABLED. Для виходу з режиму глибокого сну, треба з'єднати GPIO16 з RESET.

## ESP.rtcUserMemoryWrite (offset, & data, sizeof (data)) та ESP.rtcUserMemoryRead (offset, & data, sizeof (data)) дозволяють записувати дані та зчитувати їх з пам'яті RTC відповідно. Загальний розмір користувальницької пам'яті RTC складає 512 байт, тому offset + sizeof(data) не повинні перевищувати 512.Змінна – data повинна бути рівна 4-м байтам. Збережені дані можуть зберігатися між циклами глибокого сну. Однак ці дані можуть бути втрачені після скидання живлення на чіпі.

## Функції ESP.wdtEnable() , ESP.wdtDisable() , і ESP.wdtFeed() керують сторожовим таймером.

## ESP.reset() перезавантажує модуль

ESP.getResetReason() повертає String, що містить останню причину скидання в читабельному вигляді.

## ESP.getFreeHeap() повертає розмір вільної пам'яті

ESP.getCoreVersion() повертає String, що містить версію ядра.

ESP.getSdkVersion() повертає версію SDK як char.

ESP.getCpuFreqMHz() повертає частоту процесора в МГц як uint 8-bit.

ESP.getSketchSize() повертає розмір поточного скетчу як uint 32-bit.

ESP.getFreeSketchSpace() повертає вільне простір ескізу як uint 32-bit.

ESP.getSketchMD5() повертає нижній регістр String що містить MD5 поточного скетчу.

## ESP.getChipId() повертає ESP8266 chip IDE, int 32bit

## ESP.getFlashChipId() повертає flash chip ID, int 32bit

## ESP.getFlashChipSize() повертає розмір флеш пам'яті в байтах, так, як його визначає SDK (може бути менше реального розміру).

ESP.getFlashChipRealSize() повертає дійсний розмір чіпа в байтах на основі flash chip ID.

## ESP.getFlashChipSpeed​​(void) повертає частоту флеш пам'яті, в Гц.

## ESP.getCycleCount() повертає кількість циклів CPU з моменту старту, unsigned 32-bit. Може бути корисна для точного таймінгу дуже коротких операцій

ESP.getVcc() може використовуватися для вимірювання напруги живлення. ESP має переналаштувати АЦП під час запуску, щоб ця функція була доступною. Додайте наступний рядок у верхній частині скетча, щоб скористатися цією функцією:

ADC\_MODE(ADC\_VCC);

TOUT (ADC) пін повинен бути не задіяний периферією в цьому режимі.

Зверніть увагу, що по замовчанню ADC налаштовується для зчитування напруги і використання analogRead(A0) та ESP.getVCC() недоступне.

## mDNS and DNS-SD responder (ESP8266mDNS library)

Дозволяє ескізу реагувати на багатоадресові запити DNS для доменних імен, таких як "foo.local" та запитів DNS-SD (служба виявлення). Докладніше див. Приклади

## SSDP responder (ESP8266SSDP)

## SSDP - це ще один протокол виявлення сервісів, який підтримується на Windows із коробки. Доданий приклад для довідки.

## DNS server (DNSServer library)

## Включає простий DNS-сервер, який можна використовувати як у режимах STA, так і в режимі AP. На даний момент DNS-сервер підтримує лише один домен (для всіх інших доменів він відповідатиме з NXDOMAIN або користувацьким кодом стану). Завдяки цьому клієнти можуть відкривати веб-сервер, що працює на ESP8266, використовуючи ім'я домену, а не IP-адресу.

## Servo

## Бібліотека дозволяє управляти сервомоторами. Підтримує до 24 сервоприводів на будь-яких доступних GPIO. За замовчуванням перші 12 сервоприводів використовуватимуть Timer0 і будуть незалежні від будь-яких інших процесів. Наступні 12 сервоприводів використовуватимуть Timer1 і будуть розділяти ресурси з іншими функціями, які використовують Timer1. Більшість сервоприводів працюватимуть з керуючим сигналом ESP8266 3,3V, але не зможуть працювати від джерела напруги 3,3V і зажадають окреме джерело живлення. Не забудьте з'єднати загальний провід GND цього джерела з GND ESP8266.

## Other libraries (не включені по замовчанню в IDE)

Бібліотеки, які не використовують низький рівень доступу до регістрів AVR, повинні працювати добре. Ось декілька бібліотек, які були перевірені на роботу:

* [Adafruit\_ILI9341](https://github.com/Links2004/Adafruit_ILI9341) - Порт Adafruit ILI9341 для ESP8266
* [arduinoVNC](https://github.com/Links2004/arduinoVNC) - Клієнт VNC для Arduino
* [arduinoWebSockets](https://github.com/Links2004/arduinoWebSockets) - WebSocket Server та Client сумісний з ESP8266 (RFC6455)
* [aREST](https://github.com/marcoschwartz/aREST) - REST API бібліотека обробника.
* [Blynk](https://github.com/blynkkk/blynk-library) - проста структура IoT для розробників (check out the [Kickstarter page](http://tiny.cc/blynk-kick)).
* [DallasTemperature](https://github.com/milesburton/Arduino-Temperature-Control-Library.git)
* [DHT-sensor-library](https://github.com/adafruit/DHT-sensor-library) - Бібліотека Arduino для датчиків температури та вологості DHT11 / DHT22. Завантажте останню бібліотеку v1.1.1, і ніяких змін не потрібно. Старіші версії повинні ініціалізувати DHT наступним чином: DHT dht(DHTPIN, DHTTYPE, 15)
* [DimSwitch](https://github.com/krzychb/DimSwitch) - Контролюйте електронні регулюючі баласти для флуоресцентних ламп дистанційно, як за допомогою перемикача на стіні.
* [Encoder](https://github.com/PaulStoffregen/Encoder) - Ардюно бібліотека для ротаційних кодерів. Версія 1.4 підтримує ESP8266.
* [esp8266\_mdns](https://github.com/mrdunk/esp8266_mdns) - mDNS запити та відповіді на esp8266. Або описати це іншим способом: клієнт mDNS або бібліотека Bonjour Client для esp8266.
* [ESPAsyncTCP](https://github.com/me-no-dev/ESPAsyncTCP) - Асинхронна бібліотека TCP для ESP8266 та ESP32/31B
* [ESPAsyncWebServer](https://github.com/me-no-dev/ESPAsyncWebServer) - Бібліотека асинхронних веб-серверів для ESP8266 та ESP32/31B
* [Homie for ESP8266](https://github.com/marvinroger/homie-esp8266) - Arduino framework for ESP8266, що впроваджує Homie, MQTT-конвенція для IoT.
* [NeoPixel](https://github.com/adafruit/Adafruit_NeoPixel) - Бібліотека NeoPixel Adafruit, тепер із підтримкою ESP8266 (використовуйте версію 1.0.2 або вище від менеджера бібліотеки Arduino).
* [NeoPixelBus](https://github.com/Makuna/NeoPixelBus) - Arduino NeoPixel бібліотека сумісна з ESP8266. Використовуйте гілки "DmaDriven" або "UartDriven" для ESP8266. Включає підтримку кольорів HSL та багато іншого.
* [PubSubClient](https://github.com/Imroy/pubsubclient) - MQTT бібліотека від @Imroy.
* [RTC](https://github.com/Makuna/Rtc) - Arduino Library для Ds1307 & Ds3231 сумісний з ESP8266.
* [Souliss, Smart Home](https://github.com/souliss/souliss) - Рамки Smart Home на основі Arduino, Android і openHAB.
* [ST7735](https://github.com/nzmichaelh/Adafruit-ST7735-Library) - Бібліотека Adafruit ST7735 модифікується для сумісності з ESP8266. Просто обов'язково змініть піни в прикладах, оскільки вони все ще є специфічними для AVR.
* [Task](https://github.com/Makuna/Task) - Arduino Non Preemptive багатозадачна бібліотека. Подібно до бібліотеки, що входить до складу бібліотеки, в наданих функціях ця бібліотека призначена для сумісності Arduino.
* [TickerScheduler](https://github.com/Toshik/TickerScheduler) Бібліотека надає простий планувальник для Ticker щоб уникнути перезавантаження WDT (сторожового таймеру).
* [Teleinfo](https://github.com/hallard/LibTeleinfo) - Generic French Power Meter library to read Teleinfo energy monitoring data such as consuption, contract, power, period, … This library is cross platform, ESP8266, Arduino, Particle, and simple C++. French dedicated [post](https://hallard.me/libteleinfo/) on author’s blog and all related information about [Teleinfo](https://hallard.me/category/tinfo/) also available.
* [UTFT-ESP8266](https://github.com/gnulabis/UTFT-ESP8266) - Бібліотека дисплеїв UTFT з підтримкою ESP8266. На даний час підтримуються лише дисплеї серійного інтерфейсу (SPI) (без 8-бітового паралельного режиму тощо). Також включає підтримку апаратного контролера SPI ESP8266.
* [WiFiManager](https://github.com/tzapu/WiFiManager) - Менеджер з'єднання з Інтернетом за допомогою веб-порталу. Якщо він не може з'єднатися, він запускає режим АР та портал конфігурації, щоб ви могли вибрати та ввести в обліковий запис Wi-Fi.
* [OneWire](https://github.com/PaulStoffregen/OneWire) - Бібліотека для Dallas/Maxim 1-Wire Chips.
* [Adafruit-PCD8544-Nokia-5110-LCD-Library](https://github.com/WereCatf/Adafruit-PCD8544-Nokia-5110-LCD-library) - Порт Adafruit PCD8544 - бібліотека для ESP8266.
* [PCF8574\_ESP](https://github.com/WereCatf/PCF8574_ESP) - Дуже спрощена бібліотека для використання 8-контактного GPIO-розширювача PCF857 / / PCF8574A I2C.
* [Dot Matrix Display Library 2](https://github.com/freetronics/DMD2) - Freetronics DMD & Generic 16-дюймова 32-графічна матрична бібліотека P10
* [SdFat-beta](https://github.com/greiman/SdFat-beta) - Бібліотека SD-карти з підтримкою довгих імен файлів, програмного та апаратного забезпечення SPI та багато іншого.
* [FastLED](https://github.com/FastLED/FastLED) - Бібліотека для зручного та ефективного управління різноманітними світлодіодними чіпсетами, такими як Neopixel (WS2812B), DotStar, LPD8806 та багато інших. Включає функції згладжування, градієнта, перетворення кольорів.
* [OLED](https://github.com/klarsys/esp8266-OLED) - Бібліотека для керування з'єднаними OLED-дисплеями I2C. Тестували з 0,96-дюймовий OLED-графічним дисплеєм.
* [MFRC522](https://github.com/miguelbalboa/rfid) - Бібліотека для використання читача / письма RFID-мітки Mifare RC522.
* [Ping](https://github.com/dancol90/ESP8266Ping) - дозволяє ESP8266 налаштовувати віддалену машину.
* [AsyncPing](https://github.com/akaJes/AsyncPing) - Повністю асинхронна бібліотека Ping (мати повну статистичну статистику та апаратну MAC-адресу).