# Boards

## Adafruit HUZZAH ESP8266 (ESP-12)

Розглянемо лише плату NodeMCU, з якою ми будемо працювати, інші конфігурації ви зможете знайти за [посиланням](http://arduino-esp8266.readthedocs.io/en/2.4.0/boards.html).

## NodeMCU 0.9

### Pin mapping

Pin numbers written on the board itself do not correspond to ESP8266 GPIO pin numbers. Constants are defined to make using this board easier:

Номери контактів, написані на самій платі, не відповідають номерам PIN-кодів ESP8266 GPIO. Константи визначаються для полегшення використання цієї плати:

static const uint8\_t SDA = 4;

static const uint8\_t SCL = 5;

static const uint8\_t LED\_BUILTIN = 16;

static const uint8\_t BUILTIN\_LED = 16;

static const uint8\_t D0 = 16;

static const uint8\_t D1 = 5;

static const uint8\_t D2 = 4;

static const uint8\_t D3 = 0;

static const uint8\_t D4 = 2;

static const uint8\_t D5 = 14;

static const uint8\_t D6 = 12;

static const uint8\_t D7 = 13;

static const uint8\_t D8 = 15;

static const uint8\_t D9 = 3;

static const uint8\_t D10 = 1;

If you want to use NodeMCU pin 5, use D5 for pin number, and it will be translated to ‘real’ GPIO pin 14.

Якщо ви хочете використати NodeMCU pin 5, використайте D5 для PIN-коду, і він буде переведений на "реальний" PIN-код GPIO 14.

## NodeMCU 1.0

This module is sold under many names for around $6.50 on AliExpress and it’s one of the cheapest, fully integrated ESP8266 solutions.

Цей модуль продається під багатьма іменами за ціною близько $ 4,00 на AliExpress, і це один з найдешевших, з повністю інтегрованою ESP8266, рішень.

It’s an open hardware design with an ESP-12E core and 4 MB of SPI flash.

Це відкрита конструкція спроектована з ESP-12Е ядром і 4 МБ SPI флеш-пам'яті.

According to the manufacturer, “with a micro USB cable, you can connect NodeMCU devkit to your laptop and flash it without any trouble”. This is more or less true: the board comes with a CP2102 onboard USB to serial adapter which just works, well, the majority of the time. Sometimes flashing fails and you have to reset the board by holding down FLASH + RST, then releasing FLASH, then releasing RST. This forces the CP2102 device to power cycle and to be re-numbered by Linux.

За заявою виробника, "з мікро-USB-кабелем, ви можете підключити NodeMCU devkit до свого ноутбука і прошити, без проблем". Це більш-менш правда: плата поставляється з бортовим USB-послідовним адаптером CP2102 , або CH340 (у не офіційних збірках), який просто працює, більшу частину часу. Іноді прошити не вдається, і вам доведеться скинути плату, утримуючи FLASH + RST, потім відпустити FLASH, а за ним відпустити RST. Це змушує перезавантажити пристрій CP2102 (CH340) і переключитися на Linux.

The board also features a NCP1117 voltage regulator, a blue LED on GPIO16 and a 220k/100k Ohm voltage divider on the ADC input pin.

Плата також має регулятор напруги NCP1117, синій світлодіод на GPIO16 та подільник напруги 220к / 100к Ом на вхідному штирі АЦП.

Full pinout and PDF schematics can be found [here](https://github.com/nodemcu/nodemcu-devkit-v1.0)

Повну схему пінів та PDF-файлів можна знайти [тут](https://github.com/nodemcu/nodemcu-devkit-v1.0)

# Debugging

## Introduction

Since 2.1.0-rc1 the core includes a Debugging feature that is controllable over the IDE menu.

З версії 2.1.0-rc1 ядро містить функцію налагодження, яка управляється над меню IDE.

The new menu points manage the real-time Debug messages.

Нові пункти меню управляють повідомленнями в режимі реального часу.

### Requirements

For usage of the debugging a Serial connection is required (Serial or Serial1).

Для використання налагодження вимагається послідовний зв'язок (Serial або Serial1).

The Serial Interface need to be initialized in the setup().

Послідовний інтерфейс потрібно ініціалізувати в setup().

Set the Serial baud rate as high as possible for your Hardware setup.

Встановіть швидкість послідовної передачі якнайбільше для налаштування апаратного забезпечення.

Minimum sketch to use debugging:

Мінімальний код для використання налагодження:

void setup() {

Serial**.**begin(115200);

}

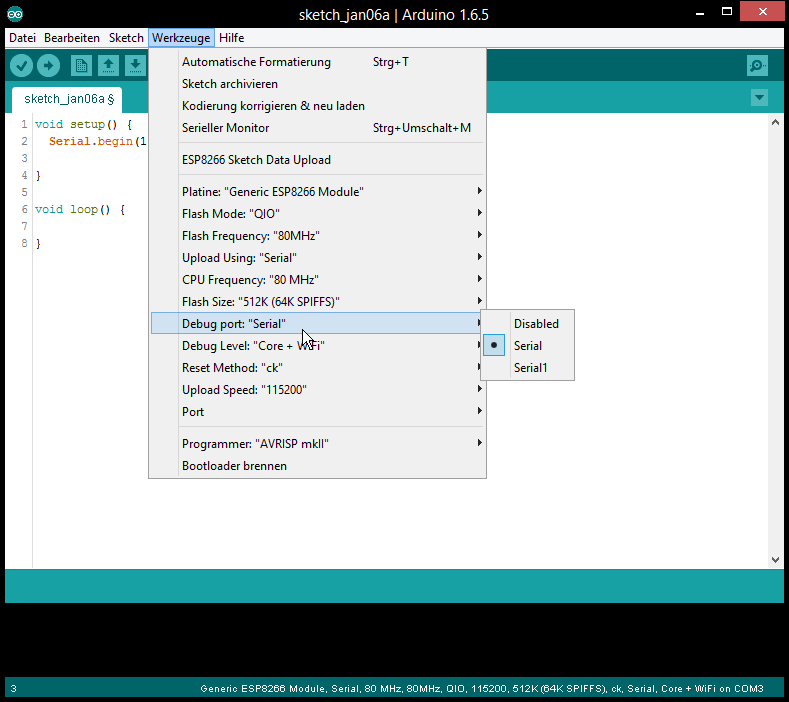
void loop() {

}

### Usage

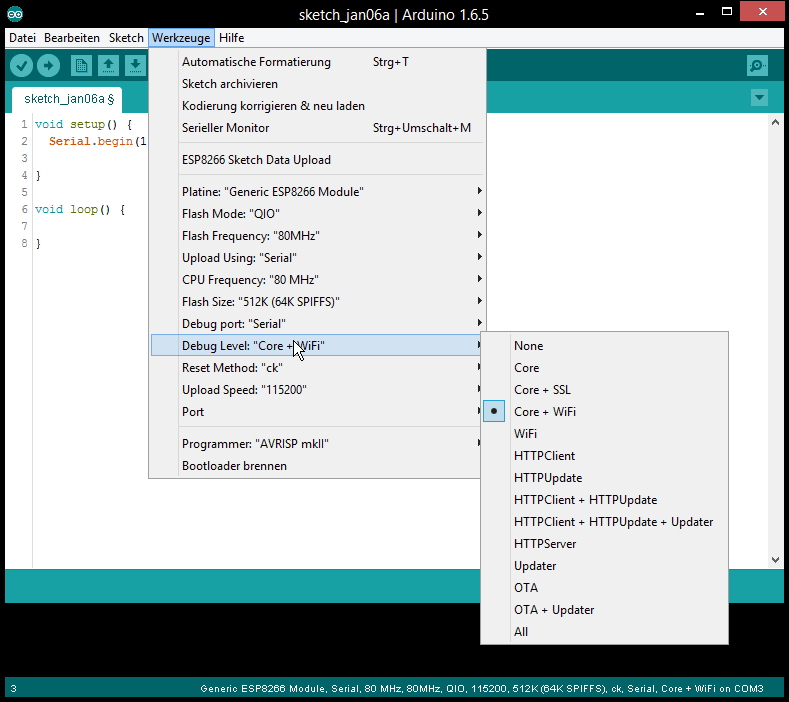
Select the Serial interface for the Debugging messages:

1. Виберіть послідовний інтерфейс для повідомлень налагодженя:



Select which type / level you want debug messages for:

1. Виберіть, який тип / рівень налагоджувальних повідомлення ви хочете:



Check if the Serial interface is initialized in setup() (see[Requirements](http://arduino-esp8266.readthedocs.io/en/2.4.0/Troubleshooting/debugging.html#requirements))

1. Перевірте, чи інтерфейс послідовного інтерфейсу ініціалізовано в setup() (див. Requirements).

Flash sketch

1. Прошийте скетч

Check the Serial Output

1. Перевірте послідовний вихід

## Informations

It work with every sketch that enables the Serial interface that is selected as debug port.

Він працює з кожним скетчом, що дозволяє використовувати послідовний інтерфейс, який вибирається як налагоджувальний порт.

The Serial interface can still be used normal in the Sketch.

Послідовний інтерфейс у Скетчі може використовуватись як звичайний.

The debug output is additional and will not disable any interface from usage in the sketch.

Вивід налагодження є додатковим і не відключає будь-який інтерфейс від використання в скетчі.

### For Developers

For the debug handling uses defines.

Для обробки налагодження використовуються визначення (defines).

The defined are set by command line.

Визначення задаються командним рядком

#### Debug Port

The port has the define DEBUG\_ESP\_PORT possible value: - Disabled: define not existing - Serial: Serial - Serial1: Serial1

Порт має значення DEBUG\_ESP\_PORT, яке можна визначити:

Disabled: визначеня не існує

Serial: Serial

Serial1: Serial1

#### Debug Level

All defines for the different levels starts with DEBUG\_ESP\_

Всі визначення для різних рівнів починаються з DEBUG\_ESP\_

a full list can be found here in the [boards.txt](https://github.com/esp8266/Arduino/blob/master/boards.txt#L180)

повний список можна знайти тут у файлі [boards.txt](https://github.com/esp8266/Arduino/blob/master/boards.txt#L180)

#### Example for own debug messages

The debug messages will be only shown when the Debug Port in the IDE menu is set.

Налагоджувальні повідомлення будуть відображатися тільки тоді, коли встановлено порт для відладки в меню IDE.

*#ifdef DEBUG\_ESP\_PORT*

*#define DEBUG\_MSG(...) DEBUG\_ESP\_PORT.printf( \_\_VA\_ARGS\_\_ )*

*#else*

*#define DEBUG\_MSG(...)*

*#endif*

void setup() {

Serial**.**begin(115200);

delay(3000);

DEBUG\_MSG("bootup...\n");

}

void loop() {

DEBUG\_MSG("loop %d\n", millis());

delay(1000);

}

# Stack Dumps

## Introduction

If the ESP crash the Exception Cause will be shown and the current stack will be dumped.

Якщо ESP дасть збій, з'явиться повідомлення про виключення, а поточний стек буде скинутий.

Example:

**Exception** (0): epc1**=**0x402103f4 epc2**=**0x00000000 epc3**=**0x00000000 excvaddr**=**0x00000000 depc**=**0x00000000

ctx: sys

sp: 3ffffc10 end: 3fffffb0 offset: 01a0

**>>>**stack**>>>**

3ffffdb0: 40223e00 3fff6f50 00000010 60000600

3ffffdc0: 00000001 4021f774 3fffc250 4000050c

3ffffdd0: 400043d5 00000030 00000016 ffffffff

3ffffde0: 400044ab 3fffc718 3ffffed0 08000000

3ffffdf0: 60000200 08000000 00000003 00000000

3ffffe00: 0000ffff 00000001 04000002 003fd000

3ffffe10: 3fff7188 000003fd 3fff2564 00000030

3ffffe20: 40101709 00000008 00000008 00000020

3ffffe30: c1948db3 394c5e70 7f2060f2 c6ba0c87

3ffffe40: 3fff7058 00000001 40238d41 3fff6ff0

3ffffe50: 3fff6f50 00000010 60000600 00000020

3ffffe60: 402301a8 3fff7098 3fff7014 40238c77

3ffffe70: 4022fb6c 40230ebe 3fff1a5b 3fff6f00

3ffffe80: 3ffffec8 00000010 40231061 3fff0f90

3ffffe90: 3fff6848 3ffed0c0 60000600 3fff6ae0

3ffffea0: 3fff0f90 3fff0f90 3fff6848 3fff6d40

3ffffeb0: 3fff28e8 40101233 d634fe1a fffeffff

3ffffec0: 00000001 00000000 4022d5d6 3fff6848

3ffffed0: 00000002 4000410f 3fff2394 3fff6848

3ffffee0: 3fffc718 40004a3c 000003fd 3fff7188

3ffffef0: 3fffc718 40101510 00000378 3fff1a5b

3fffff00: 000003fd 4021d2e7 00000378 000003ff

3fffff10: 00001000 4021d37d 3fff2564 000003ff

3fffff20: 000003fd 60000600 003fd000 3fff2564

3fffff30: ffffff00 55aa55aa 00000312 0000001c

3fffff40: 0000001c 0000008a 0000006d 000003ff

3fffff50: 4021d224 3ffecf90 00000000 3ffed0c0

3fffff60: 00000001 4021c2e9 00000003 3fff1238

3fffff70: 4021c071 3ffecf84 3ffecf30 0026a2b0

3fffff80: 4021c0b6 3fffdab0 00000000 3fffdcb0

3fffff90: 3ffecf40 3fffdab0 00000000 3fffdcc0

3fffffa0: 40000f49 40000f49 3fffdab0 40000f49

**<<<**stack**<<<**

The first number after Exception gives the cause of the reset. a full ist of all causes can be found [here](http://arduino-esp8266.readthedocs.io/en/2.4.0/exception_causes.html) the hex after are the stack dump.

Перше число після Exception  дає причину скидання. Повний перелік всіх причин можна знайти [тут](http://arduino-esp8266.readthedocs.io/en/2.4.0/exception_causes.html), шіснадцятковий шифр після це - stack dump.

### Decode

It’s possible to decode the Stack to readable information. For more info see the [Esp Exception Decoder](https://github.com/me-no-dev/EspExceptionDecoder) tool.

Можна декодувати Стек до читабельної інформації. Для отримання додаткової інформації див. інструмент [Esp Exception Decoder](https://github.com/me-no-dev/EspExceptionDecoder).

