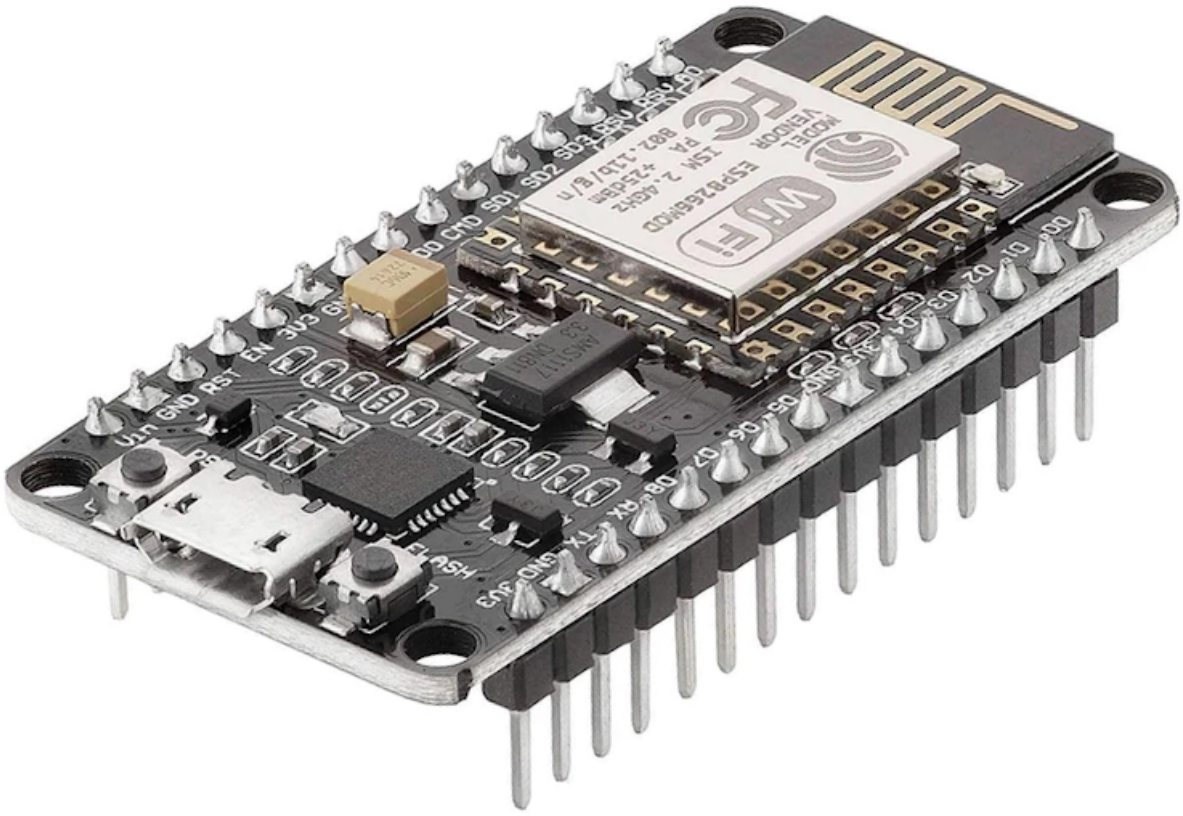
Welcome!

Thank you for purchasing our *AZ-Delivery NodeMCU LUA Amica V2*. On the following pages, we will introduce you to how to use and set-up this handy device.

Have fun!



NodeMCU LUA Amica V2 is a development board created around *ESP8266* chip, containing voltage regulator and USB programmer circuit for *ESP8266* chip, and a few other features.

For the development of applications you can choose between the Arduino IDE and LUA language. The user community is very active and supports platforms such as ESP8266.

NodeMCU comes with a pre-installed firmware which allows us to work with the LUA interpreted language, sending commands through the serial port (*CP2102* chip). The NodeMCU board is one of the most used platforms for Internet of Things (IoT) projects. NodeMCU LUA Amica V2 is fully compatable with Arduino IDE.

The NodeMCU board is specially designed to work on breadboard. It has a plate voltage regulator that allows it to feed directly from the USB port. The input/output pins work at *3.3V*. The *CP2102* chip is responsible for USB to serial communication.

Specification:

- » Power supply voltage (USB): 5V DC
- » Input/Output voltage: 3.3V DC
- » SoC: ESP8266 (ESP-12 Module)
- » CPU: Tensilica Xtensa LX3 (32 bit)
- » Clock frequency: 80MHz / 160MHz
- » Instruction RAM: 32kB
- » Data RAM: 96kB
- » External flash memory: 4MB
- » GPIO digital pins: 17 (can be configured as PWM at 3.3V)
- » ADC analog pin: 1 (BUT voltage range is: 0 - 1V)
- » UART: 2
- » USB serial chip: CP2102
- » PCB antenna
- » 802.11 b/g/n
- » Wi-Fi Direct (P2P), soft-AP
- » Integrated TCP/IP Protocol Stack
- » Output power of + 19.5dBm in 802.11b mode
- » Leakage current less than 10uA
- » Wake up and transmit packets in <2ms
- » Standby power consumption <1.0mW (DTIM3)

The ESP8266 series of Wi-Fi chips is produced by Espressif Systems, a semiconductor company from Shanghai (China). ESP8266 is affordable Wi-Fi module suited for DIY projects in the Internet of Things (IoT) field. This module comes with many GPIOs and support for a variety of protocols like SPI, I2C, UART, and more. The best part is that it comes with wireless networking included, which makes it apart from other microcontrollers like the Atmega328p. This means that you can easily control and monitor devices remotely via Wi-Fi at an affordable price.

ESP8266 is a system-on-chip (SoC) integrating a 32-bit Tensilica microcontroller, standard digital peripheral interfaces, antenna switches, RF balun, power amplifier, low noise receive amplifier, filters and power management modules into a small package. It provides *2.4GHz* Wi-Fi (802.11 b/g/n, supporting both WPA and WPA2), *17* GPIO pins, I2C (IIC) interface, analog to digital conversion (10-bit, on one pin), SPI interface, UART (on dedicated pins, plus a transmit-only UART can be enabled on GPIO2), and PWM (Pulse Width Modulation) in software on every GPIO pin.

The processor core, called “*L106*” by Espressif, is based on Tensilica Diamond Standard 106Micro 32-bit processor controller core and runs at 80MHz. It has a *64kB* boot ROM, *32kB* instruction RAM, and *80kB* user data RAM. External flash memory can be accessed through SPI interface.

Vendors have consequently created many of compact pcb modules based around the ESP8266 chip (like NodeMCU LUA Amica V2). Some of these modules have specific identifiers, such as “*ESP-01*” through “*ESP-14*”. ESP8266-based modules are shown to be low-cost, networkable and overall multpipurpose platform for facilitating end-point IoT development.

NodeMCU

The NodeMCU is an open-source firmware and development kit that helps you to prototype your IoT product within a few Lua script lines. NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266, and hardware which is based on the ESP-12 module. The term “NodeMCU” refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266.

Features of NodeMCU:

- Development Kit - based on ESP8266
- Atmega328p-like hardware IO
- Nodejs style network API - Event-driven API for network applications
- Lowest cost WI-FI

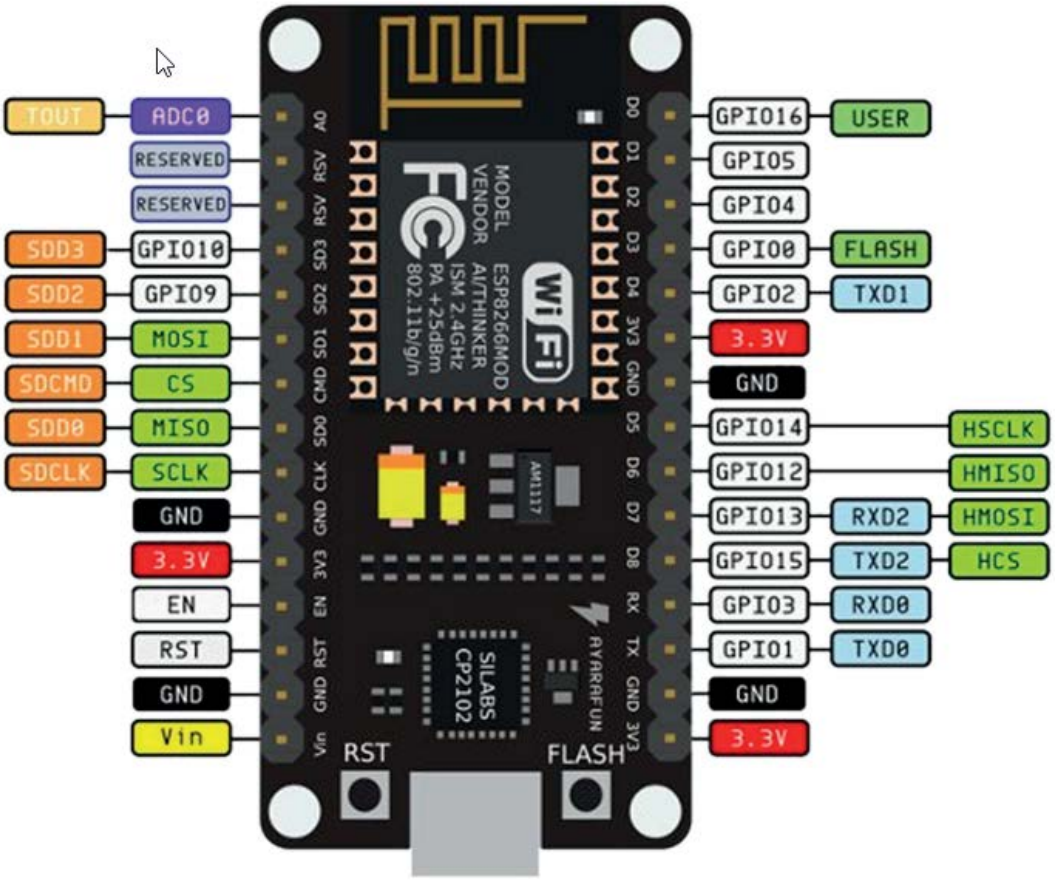
Difference between ESP8266 (NodeMCU) and Atmega328p

Specification	Mc	ESP8266
RAM:	4kB	80kB
FLASH memory:	32kB	4MB
Speed:	16MHz	80MHz
GPIOs (usable):	14	11
I/O voltage level:	5V	3.3V
ADC (resolution):	6 (10-bit)	1 (10-Bit)
Serial interface:	1	1
I2C interface:	1	1
SPI interface:	1	Used by flash chip
PWM, resolution:	6, 8 bit	All GPIO pins, 10 bit
WiFi	No	Yes 2MBps

Difference between ESP8266 and ESP32

Specification	ESP8266	ESP32
MCU:	Xtensa Single-core 32-bit L106	Xtensa Dual-Core 32-bit LX6
802.11 b/g/n Wi-Fi:	HT20	HT40
Bluetooth:	No	Bluetooth 4.2 and BLE
Typical frequency:	80MHz	160MHz
SRAM:	No	Yes
Flash:	No	Yes
GPIO pins:	17	36
HW/SW PWM:	None / 8 channels	None / 16 channels
SPI/I2C/I2S/UART:	2/1/2/2	4/2/2/2
ADC:	10-bit	12-bit
CAN:	No	Yes
Ethernet MAC interface:	No	Yes
Touch sensor:	No	Yes
Temperature sensor:	No	Yes
Hall effect sensor:	No	Yes
Working temperature:	-40ºC to 125ºC	-40ºC to 125ºC

NodeMCU LUA Amica V2 pinout



GPIO Pin Description

Just like a normal Atmega328p board, the ESP8266 has digital input/output pins (GPIO pins - General Purpose Input/Output pins). These digital input/outputs operate at *3.3V*.

5V voltage should not be connected to any ESP8266 chip pins!

The pins are not *5V* tolerant, applying more than *3.6V* on any pin will destroy the chip.

The maximum current that can be drawn from a single GPIO pin is *12mA*.

ESP8266 has *17* GPIO pins, however, you can only use *11* of them, because *6* pins (GPIO *6* - *11*) are used to connect the flash memory chip. This is the small 8-pin IC right next to the ESP8266. If you try to use one of these pins, you might crash your program.

GPIO *1* and *3* are used as TX and RX of the hardware Serial port (UART), so in most cases, you can not use them as normal I/O while sending/receiving serial data.

Boot modes

Few I/O pins have a special function during boot, they select one of three boot modes:

GPIO15	GPIO0	GPIO2	Mode
0V	0V	3.3V	UART Bootloader
0V	3.3V	3.3V	Boot sketch (SPI flash)
3.3V	X	X	SDIO mode (not used for Atmega328p)

NOTE: External pull down resistor of *1k*Ω is required on GPIO0, external pull up resistor on GPIO2 is not required, the internal pull up on this pin is enabled at boot.

- GPIO15 is always pulled LOW, so you can not use the internal pull-up resistor on this pin. Keep this in mind when using GPIO15 as an input to read a switch or connect it to a device with an open-collector (or open-drain) output, like I2C.
- GPIO0 is pulled HIGH during normal operation.
- GPIO2 can not be LOW at boot, so you can not connect a switch to it.

Internal pull up/down resistors

GPIO 0-15 all have a built-in pull up resistor, just like in an Atmega328p. GPIO16 has a built-in pull down resistor.

Digital I/O pins

You can set the function of a pin using:

`pinMode(pin, mode)`

where *`pin`* is the GPIO number,

and *`mode`* can be either *`INPUT`*, which is the default, *`OUTPUT`*, or *`INPUT_PULLUP`* to enable the built-in pull up resistors for GPIO *0 - 15*. To enable the pull-down resistor for GPIO16, use *`INPUT_PULLDOWN_16`*.

To set an output pin HIGH (*3.3V*) or LOW (*0V*), use:

`digitalWrite(pin, value)`

where *`pin`* is the digital pin,

and *`value`* either *1* or *0* (or HIGH and LOW).

To read an input, use `digitalRead(pin)` .

PWM – Pulse-Width Modulation

ESP8266 supports software PWM on all digital pins. The default PWM resolution is 10-bits at 1kHz, but this can be changed. To enable PWM on a certain pin, use:

`analogWrite(pin, value)`

where *`pin`* is the digital pin,

and *`value`* a number between *0* and *1023*.

You can change the range (bit depth) of the PWM output by using `analogWriteRange(range)`.

The frequency can be changed by using

`analogWriteFreq(frequency)`

Frequency should be between *100Hz* and *1000Hz*.

Analog input

The ESP8266 has a single analog input pin, with an input voltage range from *0.0V* to *1.0V*. If you supply voltage of 3.3V, for example, you will damage the chip. The NodeMCU has an on-board resistive voltage divider, to get an easier *0* - *3.3V* range. The ADC (analog to digital converter) has a resolution of *10* bits. The ESP can also use the ADC to measure the supply voltage (VCC). To do this, include:

```
ADC_MODE(ADC_VCC)
```

at the top of your sketch, and use:

```
ESP.getVcc()
```

to actually get the voltage. If you use it to read the supply voltage, you can not connect anything else to the analog pin.

Serial Communication

The ESP8266 has two hardware UARTS (Serial ports):
- UART0 on pins *1* and *3* (TX0 and RX0 resp.), and
- UART1 on pins *2* and *8* (TX1 and RX1 resp.), however, GPIO8 is used to connect the flash chip. This means that UART1 can only transmit data.

Additionally UART0 has hardware flow control on pins *15* and *13* (RTS0 and CTS0 respectively). These two pins can also be used as alternative TX0 and RX0 pins.

To use UART0 (TX = GPIO1, RX = GPIO3), you can use the Serial object, just like on an Atmega328p: `Serial.begin(baud)`

To enable the alternative pins (TX = GPIO15, RX = GPIO13), use: `Serial.swap()` after `Serial.begin()`.

To enable UART1 (TX = GPIO2), use the `Serial1` object.

WiFi Communication

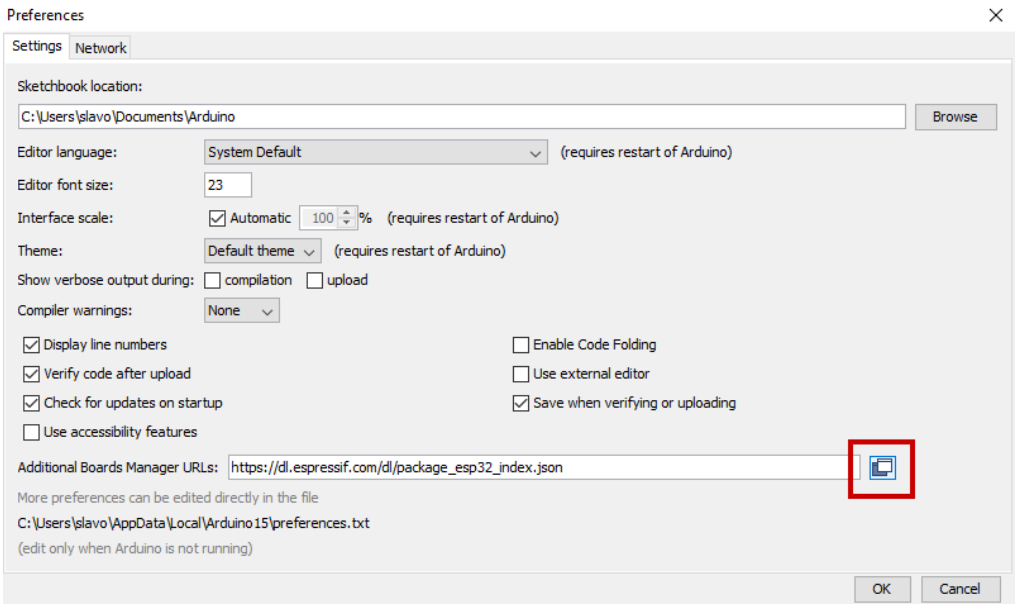
ESP can operate in three different modes: Wi-Fi station, Wi-Fi access point, and both at the same time.

Other features

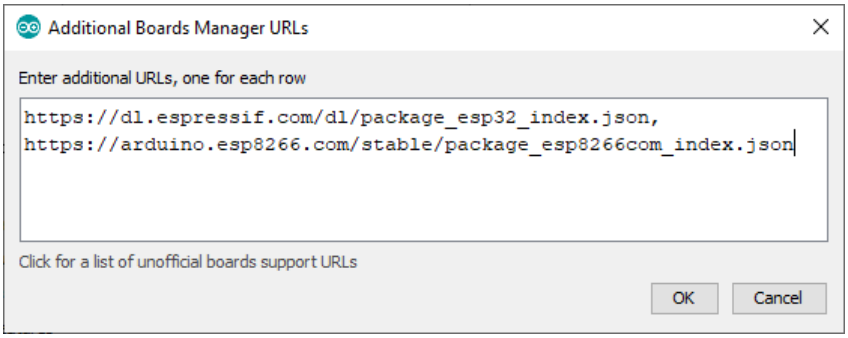
One of features of NodeMCU LUA Amica V2 is its ability to work as Access Point, or hotspot for your Wi Fi project. Also, you can run a web server on the NodeMCU. Another feature enables uploading code to NodeMCU LUA Amica V2 via internet. This is called OTA - Over-The-Air programming, and this is a process which allows devices to upgrade their firmware or software wirelessly without any physical access (over Wi-Fi, Bluetooth, GPRS or 4G/3G). These features are not covered in this eBook.

How to Use ESP8266/NodeMCU with Arduino IDE

In order to use NodeMCU with Arduino IDE, follow few easy steps. First is to install ESP8266 core. To install it, open Arduino IDE and go to:
File > Preferences,
and find Additional URLs field.



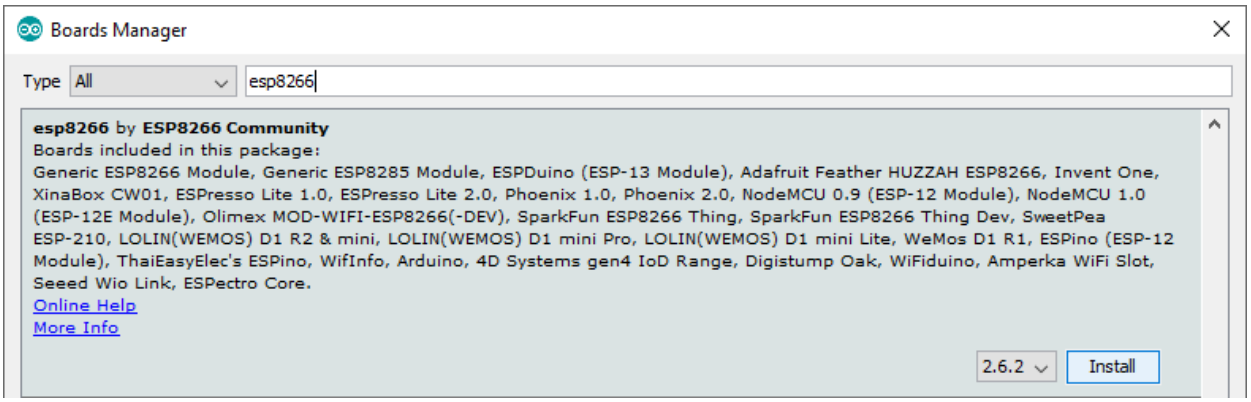
Then copy the following URL:
https://arduino.esp8266.com/stable/package_esp8266com_index.json
Paste this link in the Additional URLs field. If you already have one or more links inside this field, just add one comma after the last link, paste new link after comma and click the *OK* button. Then close the Arduino IDE.



Open Arduino IDE again and go to:

Tools > Board > Boards Manager

New window will open, type “*esp8266*” in the search box and install the board called “*esp8266*” made by "*ESP8266 Community*", as shown on the image below:



Now you have ESP8266 core installed.

To select NodeMCU LUA Amica V2 board, go to:

Tools > Board > NodeMCU 1.0 (ESP - 12E Module)

To upload the sketch code to the NodeMCU board, first select port on which you connected the board. Go to:

Tools > Port > {port name}

Blinking two LEDs on-board NodeMCU LUA Amica V2

There are two LEDs on-board NodeMCU LUA Amica V2. One LED is connected to the GPIO pin *2* and is located on the ESP8266 board. The other LED is connected to the GPIO pin *16* and is located on the NodeMCU board. If you use default *Blink* sketch example that comes with Arduino IDE, *LED_BUILTIN* macro represents the LED connected to the GPIO pin *2*. To make these two LEDs to blink, the following code is the sketch example:

```
#define led_built_in_ESP    2;
#define led_built_in_Node   16;
void setup() {
  pinMode(led_built_in_ESP, OUTPUT);
  pinMode(led_built_in_Node, OUTPUT);
}
void loop() {
  digitalWrite(led_built_in_ESP,  HIGH);
  digitalWrite(led_built_in_Node, LOW);
  delay(1000);
  digitalWrite(led_built_in_ESP,  LOW);
  digitalWrite(led_built_in_Node, HIGH);
  delay(1000);
}
```

PWM – Pulse Width Modulation

We will use PWM to fade an LED connected to GPIO pin 2. Sketch code:

```
#define LED 2
uint16_t brightness = 0; // how bright the LED is
uint8_t fadeAmount = 5; // how many points to fade the LED by
void setup() {
  pinMode(LED, OUTPUT);
}
void loop() {
  analogWrite(LED, 0);
  delay(2000);
  analogWrite(LED, 512);
  delay(2000);
  analogWrite(LED, 1023);
  delay(2000);
  while(1) {
    analogWrite(LED, brightness);
    brightness = brightness + fadeAmount;
    if(brightness <= 0 || brightness >= 1023) {
      fadeAmount = -fadeAmount;
    }
    delay(15);
  }
}
```

You've done it!

Now you can use your module for various projects.

Now is the time to learn and make the Projects on your own. You can do that with the help of many example scripts and other tutorials, which you can find on the internet.

If you are looking for the high quality microelectronics and accessories, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

https://az-delivery.de

Have Fun!

Impressum

https://az-delivery.de/pages/about-us