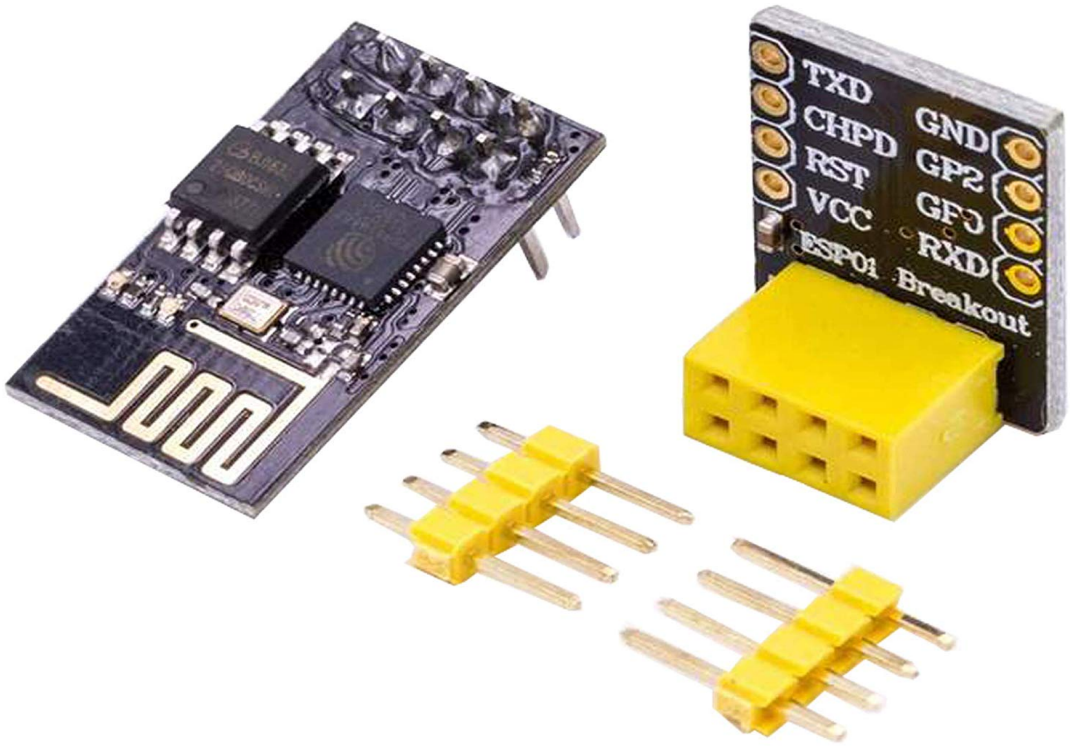
Welcome!

Thank you very much for purchasing our AZ-Delivery ESP8266-01S module with Breadboard Adapter. On the following pages, we will introduce you to how to use and setup this handy device.

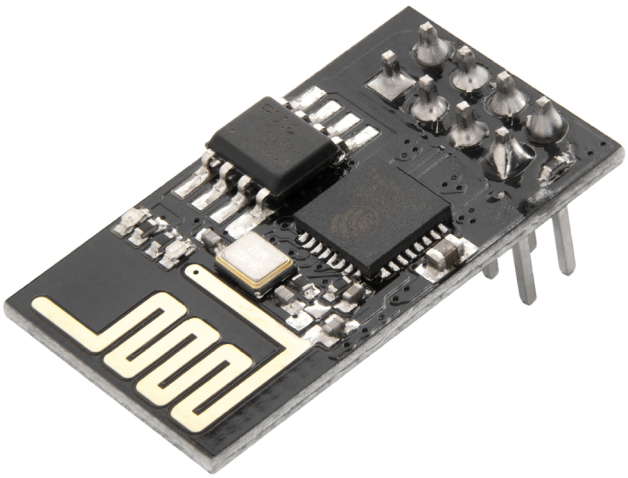
Have fun!



Ihr Experte für Mikroelektronik!

The ESP8266 module is a System on a Chip (SoC), manufactured by the Chinese company Espressif. It consists of a Tensilica L106 32-bit microcontroller and a wifi transceiver. It has 11 GPIO (General Purpose Input/Output) pins, and one analog input as well. This means that you can program it like any normal Arduino or any other microcontroller. And best thing about ESP8266 is that you get wifi communication with it, so you can use it to connect to your wifi network, connect to the internet, host a web server with real web pages, let your smartphone connect to it, etc.

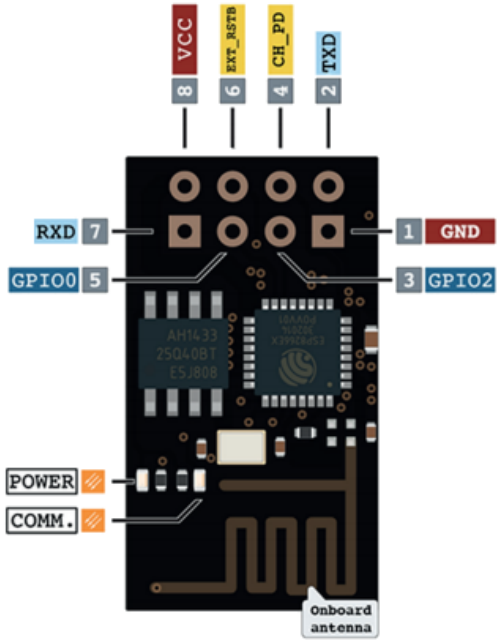
The ESP8266 is an amazing WiFi chip that can be used in several home automation applications. Thanks to the powerful 80MHz processor and a large 1MB memory, the ESP8266 can also be operated autonomously.



Features

- 802.11 b/g/n
- Integrated low power 32-bit MCU
- Integrated 10-bit ADC
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- Wi-Fi 2.4 GHz, support WPA/WPA2
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU & A-MSDU aggregation and 0.4s guard interval
- Deep sleep power <10uA, power down leakage current < 5uA
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)
- +20dBm output power in 802.11b mode
- Operating temperature range: -40 °C ~ 125 °C

There are many different development boards based on ESP8266, so ESP8266-01S is one of them. It has 8 pins (pin designation on image below).

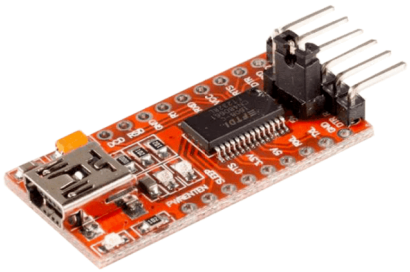


As you can see we have two free GPIO pins, GPIO0 and GPIO2. So we can use GPIO0 and GPIO2 as digital inputs or outputs like in normal microcontroller. In the example (later in the eBook) we will use GPIO2 to read the data from DHT22 temperature sensor module.

We can program the ESP8266-01S in many different methods, but we will only cover the method using the Arduino IDE. If you've used Arduino boards before, then this is really easy for you. Just keep in mind that it's not limited to this option, there are many other ways to program ESP8266 (official ESP SDK for C programming, Lua interpreter, MicroPython firmware, are few of many).

In order to program ESP8266-01S we need to use RX and TX pins, and connect them to the device with serial interface like the FTDI module or dedicated USB adapter, or Arduino board.

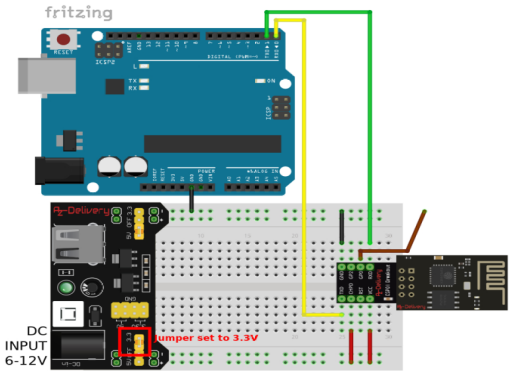
The FTDI is quite popular, because it can switch between 5V and 3.3V logic. **It is essential that the USB to Serial converter operates at 3.3V!**
If you run ESP8266 at a 5V FTDI model, you will damage the ESP8266!



FTDI adapter

https://www.az-delivery.de/products/ftdi-adapter-ft232rl?ls=en

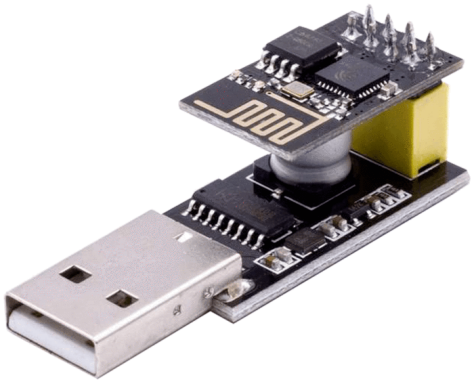
If you are using Arduino Uno to program the ESP8266-01S this is connection diagram (we won't use this in this eBook):



We used this in our eBook ESP8266-01S with Relay:

https://www.az-delivery.de/products/esp8266-01-mit-relais-kostenfreies-e-book?_pos=2&_sid=ffbf81394&_ss=r&ls=de

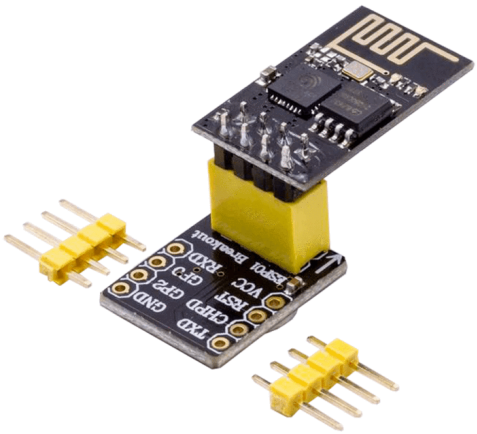
There is another way to program the ESP8266-01S and it is with dedicated USB adapter (image below), but we also won't cover it in this eBook.



USB adapter with ESP8266-01S

https://www.az-delivery.de/products/esp8266-01s-mit-usb-adapter?ls=en

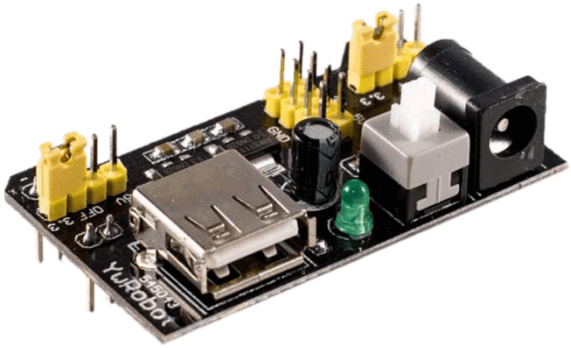
We will use breadboard breakout board and FTDI adapter to program ESP8266-01S, because it is very easy, but any way you choose is similar.



Breadboard breakout board with ESP8266-01S

https://www.az-delivery.de/products/esp8266-01s-mit-breadboardadapter?ls=en

Just make sure to power ESP8266-01S with separate power supply. You can use our MB102 breadboard power supply. It is very simple and handy power supply which accepts inputs from 6 to 12V DC and can output both +3.3V and +5V.



MB102 Power supply for breadboard
https://www.az-delivery.de/products/mb102-breadboard?ls=en

There are two very different ways to program ESP8266-01S. First is by using *"AT"* commands (ATtention commands) via Serial Interface, and second is by actually programming the ESP8266 chip like microcontroller (using Arduino IDE).

Because the second way gives much more freedom in what you can create, we will cover only that way of programming ESP8266-01S in this eBook.

Enabling the chip

We have to add some resistors to turn on the ESP8266-01S, and to select the right boot mode (programming or normal mode). To ensure proper usage of ESP8266-01S module we first have to:

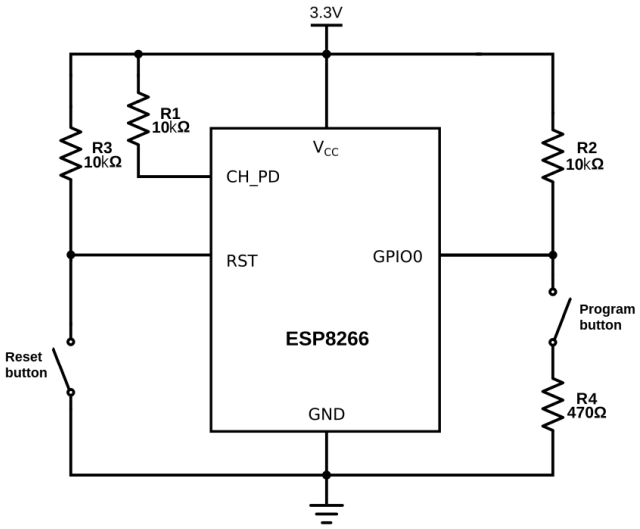
1. Enable the chip by connecting the **CH_PD** (Chip Power Down, sometimes labeled *CH_EN*, or *CH_PC*) pin to **VCC** through a **10kΩ** resistor.

2. Select *NORMAL* mode by connecting **GPIO0** to **VCC** through a **10KΩ** resistor. (we connect it to the **GND** in *PROGRAM* mode, and for *NORMAL* mode we leave it unconnected)

3. Prevent random resets by connecting the **RST** (reset) pin to **VCC** through a **10KΩ** resistor. (you can add it, but it is not necessary, we leave it unconnected)

Adding RESET and PROGRAM buttons

Our ESP8266-01S boards don't have a reset button or button for programming mode, but you could (but it is not necessary) add them, like on image below.



For reset button connect a push button between the *RST* pin and *GND*, but add one PULL-UP resistor 10kΩ (R3 on the image above) between RST and VCC.

To put the chip into programming mode, you have to pull *GPIO0* pin *LOW* during startup (connect it to the GND). If it is HIGH or unconnected, chip will start booting in normal mode. Because it's possible to use *GPIO0* as an output, we can't directly short it to ground, that could damage the chip. To prevent this, connect 470Ω resistor (R4 on the image above) between this button and the GND. It's important that this resistance is low enough, otherwise, it will be pulled high by the 10KΩ resistor (R2 on the image above) if you chose to add R2 like in suggestion on previous page (in 2.).

For purpose of this eBook, we are not using any buttons. To put ESP into programming mode, we use one jumper wire, which connects *GPIO0* to the *GND* for programming mode, and disconnect it from GND in normal mode.

To reset ESP8266-01S, we switch off/on power supply.

There's a few things you have to look out for when using an ESP8266-01S: The most important thing is that it runs at 3.3V, so if you connect it to a 5V power supply, you'll kill it. Unlike some 3.3V Arduino boards, the ESP8266's I/O pins are not 5V tolerant, so if you use a 5V USB to Serial converter, or 5V sensors etc. you'll kill it.

A second thing to keep in mind is that the ESP8266 can only source or sink 12mA per output pin, compared to 20 - 40mA for most Arduinos!

One last thing to keep in mind is that the ESP8266 has to share the system resources and CPU time between your sketch and the Wi-Fi driver. Also, features like PWM, interrupts or I²C are emulated in software, most Arduinos on the other hand, have dedicated hardware parts for these tasks. For most applications however, this is not too much of an issue, but you have to think about it when designing a sketch.

The ESP8266 as a microcontroller - Hardware

While the ESP8266 is often used as a "dumb" Serial-to-wifi bridge, it’s a very powerful microcontroller on its own.

Digital I/O

Just like a normal Arduino, the ESP8266 has digital input/output pins or GPIO - General Purpose Input/Output pins. As the name implies, they can be used as digital inputs to read a digital voltage, or as digital outputs to output either 0V (sink current) or 3.3V (source current).

The ESP8266 is a 3.3V microcontroller, so its I/O operates at 3.3V as well.

- The pins are not 5V tolerant, applying more than 3.6V **ON ANY PIN will kill the chip!**
- The maximum current that can be drawn from a single GPIO pin is 12mA!

Usable pins

The ESP8266 has 17 GPIO pins (0-16), however, you can only use only 2 of them, because for ESP8266-01S only two are availabe on breakout pins. If you try to use other pins, you might crash your program.

GPIO1 and *GPIO3* pins are used as *TX* and *RX* of the hardware Serial port (*UART*). You can use them as GPIOs but in most cases, you shouldn't use them as GPIOs while sending/receiving serial data.

Boot modes

As mentioned in the previous chapter, some GPIO pins have a special function during boot, they select 1 of 3 boot modes:

GPIO0	GPIO2	Mode
0V	3.3V	Uart Bootloader (PROGRAM mode)
3.3V	3.3V	Boot sketch (SPI flash)
x	x	SDIO mode (not used for Arduino)

Note: You don’t have to add an external pull-up resistor to GPIO2, the internal one is enabled at boot.

We made sure that these conditions are met by adding external resistors in the previous chapter. This has some implications, however:
» GPIO0 is pulled HIGH during normal operation mode, so you can’t use it as a Hi-Z input.
» GPIO2 must not be LOW at boot, so you can’t connect a switch to it.

Internal pull-up/-down resistors

All GPIO pins of ESP8266 have a built-in pull-up resistor, just like in an Arduino. Except GPIO16 has a built-in pull-down resistor (but you can't access it in ESP8266-01S, so it does not matter).

PWM

Unlike most Atmel chips (Arduino), the ESP8266 doesn’t support hardware PWM, however, software PWM is supported on all digital pins. The default PWM range is 10 bits at 1kHz, but this can be changed (up to >14 bit at 1kHz).

Serial Interface

The ESP8266 has two hardware *UARTS* (Serial ports), but you can use only one on ESP8266-01S module, *UART0* on pins 1 and 3 (*TX0* and *RX0* respectively)

The ESP8266 as a microcontroller - Software

Most of the microcontroller functionality of the ESP uses exactly the same syntax as a normal Arduino, making it really easy to get started.

Digital I/O

Just like with a regular Arduino, 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), or *OUTPUT*, or *INPUT_PULLUP* to enable the built-in pull-up resistors for *GPIO 0-15*. To enable the pull-down resistor for GPIO16, you have to use *INPUT_PULLDOWN_16* (for ESP8266-01S, you can't access GPIO16 so this is just for general ESP8266). 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).* 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 of the PWM output by using *analogWriteRange(new_range).* The frequency can be changed by using *analogWriteFreq(new_frequency)*, "*new_frequency"* should be between 100Hz and 1000Hz.

Serial communication

To use *UART0* (*TX* = *GPIO1*, *RX* = *GPIO3*), you can use the Serial object, just like on an Arduino: *Serial.begin(baud_reate)*. **All Arduino Stream functions, like *read*, *write*, *print*, *println*, ... are supported as well.**

Sharing CPU time with the RF part

One thing to keep in mind while writing programs for the ESP8266 is that your sketch has to share resources (CPU time and memory) with the wifi and TCP-stacks (the software that runs in the background and handles all wifi and IP connections). If your code takes too long to execute, and don't let the TCP stacks do their thing, it might crash, or you could lose data. It's best to keep the execution time of you loop under a couple of hundreds of milliseconds. Every time the main loop is repeated, your sketch yields to the wifi and TCP to handle all wifi and TCP requests. If your loop takes longer than this, you will have to explicitly give CPU time to the wifi/TCP stacks, by using including *delay(0)* or *yield()*. If you don't, network communication won't work as expected, and if it's longer than 3 seconds, the soft WDT (Watchdog Timer) will reset the ESP. If the soft WDT is disabled, after a little over 8 seconds, the hardware WDT will reset the chip. From a microcontroller's perspective however, 3 seconds is a very long time (240 million clock cycles), so unless you do some extremely heavy number crunching, or sending extremely long strings over serial, you won't be affected by this. Just keep in mind that you add the *yield()* inside your for or while loops that could take longer than, say 100ms.

Wifi

Using the ESP8266 as a simple microcontroller is great, but the reason why most people use it, is its wifi capabilities. It supports network protocols like wifi, TCP, UDP, HTTP, DNS, etc.

Uploading sketches to the ESP8266

The upload procedure for ESP8266 boards is a little different from the normal Arduino procedure. Most Arduino boards will automatically reset when a new program is being uploaded, and will automatically enter programming mode. On ESP8266-01S boards you have to manually enter programming mode, and you even have to reset them manually after programing the module.

Manual RESET and manual PROGRAM

If you don't have a USB-to-Serial converter with *DTR* and *RTS* lines, you could also just use the *RESET* and *PROGRAM* buttons we wrote about in one of the previous chapters.

To get the ESP8266 into *PROGRAM* mode, *GPIO0* must be LOW while booting. For purpose of this guide, we use jumper wires instead of buttons, so, this is process on how to switch between normal and program mode:

» To enter to *PROGRAM* mode:

1. Power down the module (disconnect power supply)
2. Connect *GPIO0* pin to ground (GND) via jumper wire.
3. Power up the module (connect power supply), and ESP8266-01S is in *PROGRAM* mode.

» To enter *NORMAL* mode:

1. Power down the module (disconnect power supply)
2. Leave unconnected *GPI0* pin, or unconnect it from GND if it was previously connected. (just make sure that jumper wire does not connect to anything!)
3. Power up the module (connect power supply), and ESP8266-01S is in *NORMAL* mode.

Of course you can add *RESET* and *PROGRAM* button, thus simplifying this process. If you choose to go this route, that the processes of entering *PROGRAM or NORMAL* modes are:

» To enter to *PROGRAM* mode:

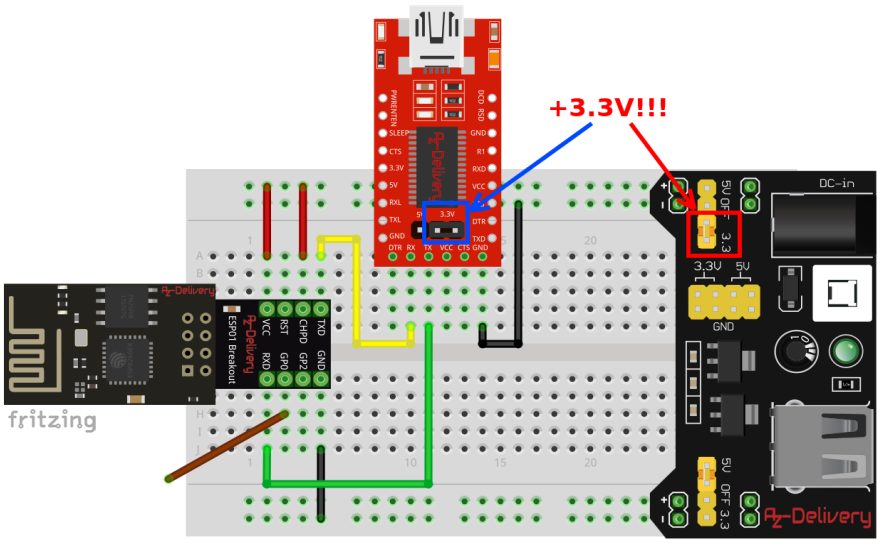
1. press and hold the *RESET* button
2. press and hold the *PROGRAM* button
3. release the *RESET* button, the ESP will boot in program mode
4. release the *PROGRAM* button
5. upload the sketch

» To enter *NORMAL* mode:

1. press and release *RESET* button to reset the ESP8266-01S. *PROGRAM* button should stay released during this process.

ESP8266-01S connection diagram

To make all suggestions like in previous chapters, connect ESP8266-01S as on connection diagram below:



For all three modules connect all GND pins together! (**Black wires**)

For FTDI and ESP8266-01S:

1. Make sure to set voltage reference jumper to 3.3V (as for FTDI this also applies for MB102)

2. Connect the **GND** pin of the FTDI to the **GND** pin of the ESP8266-01S.
3. Connect the **RX** pin of the FTDI to the **TX** pin of the ESP8266-01S. **Yellow wire** on connection diagram above.

4. Connect the **TX** pin of the FTDI to the **RX** pin of the ESP8266-01S. ** **Green wire** on connection diagram above.

VCC of ESP8266-01S is connected to the +3.3V (**Red wire**).
CHPD pin of ESP8266-01S has to be connected to the VCC (**Red wire**).

For normal mode *GPIO0* pin have to be disconnected on startup (**Brown wire**), but when we want to program ESP8266-01S, we first need to power down module (by disconnecting it from the power supply), connect *GPIO0* pin to the GND (**Brown wire**) and then to connect ESP back to the power supply.

In order to switch to normal mode again, we need to power down the module again, and then disconnect *GPIO0* pin (**Brown wire** leave unconnected like on connection diagram), and power on ESP module again, connect it back to the power supply.

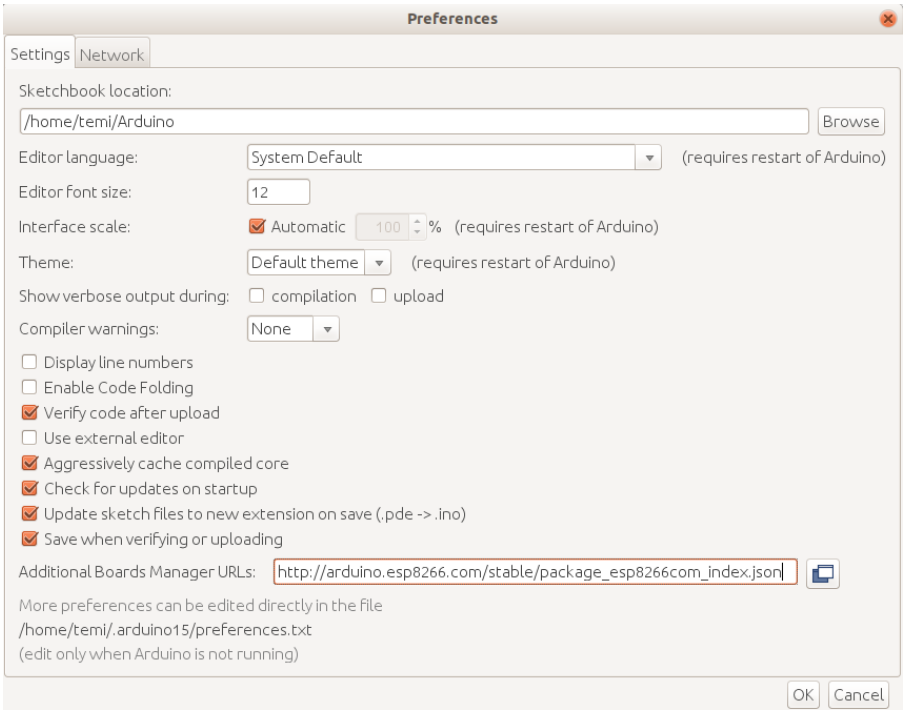
Software

We will only show you how to use ESP8266-01S with Arduino IDE.
The first step is to download and install the Arduino IDE. If you are using it already, great, if not, go on https://www.arduino.cc/en/Main/Software and download and install it.

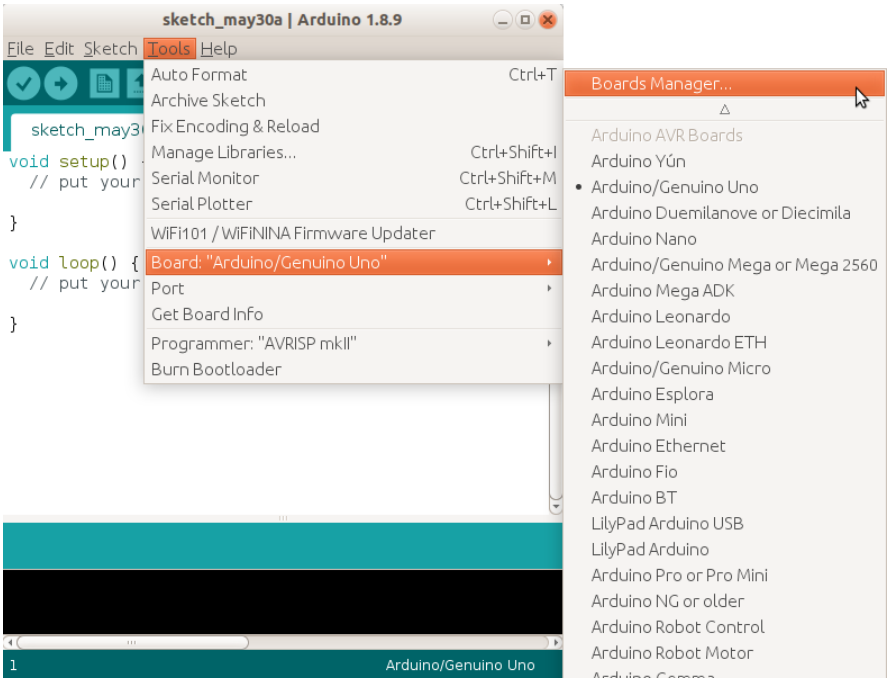
To program the ESP8266-01S, you'll need a plugin for the Arduino IDE, it can be downloaded from GitHub https://github.com/esp8266/Arduino manually, but it is easier to just add the URL in the Arduino IDE. Open the IDE, go to *File* > *Preferences*, paste the URL:

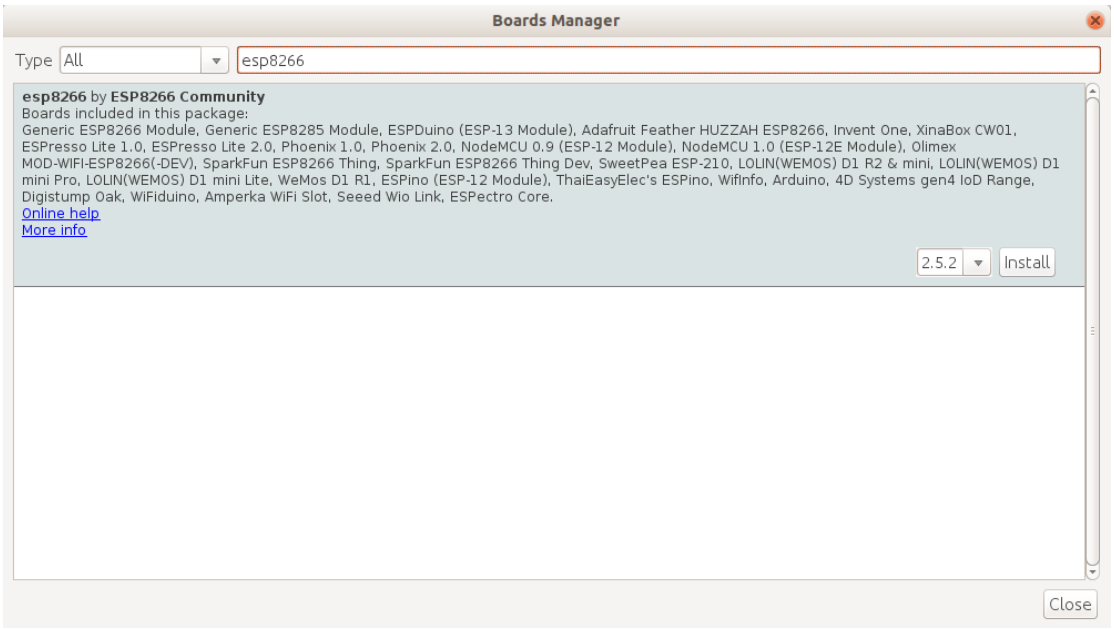
http://arduino.esp8266.com/stable/package_esp8266com_index.json

into the *"Additional Board Manager"* URLs field. (You can add multiple URLs, separating them with commas.)



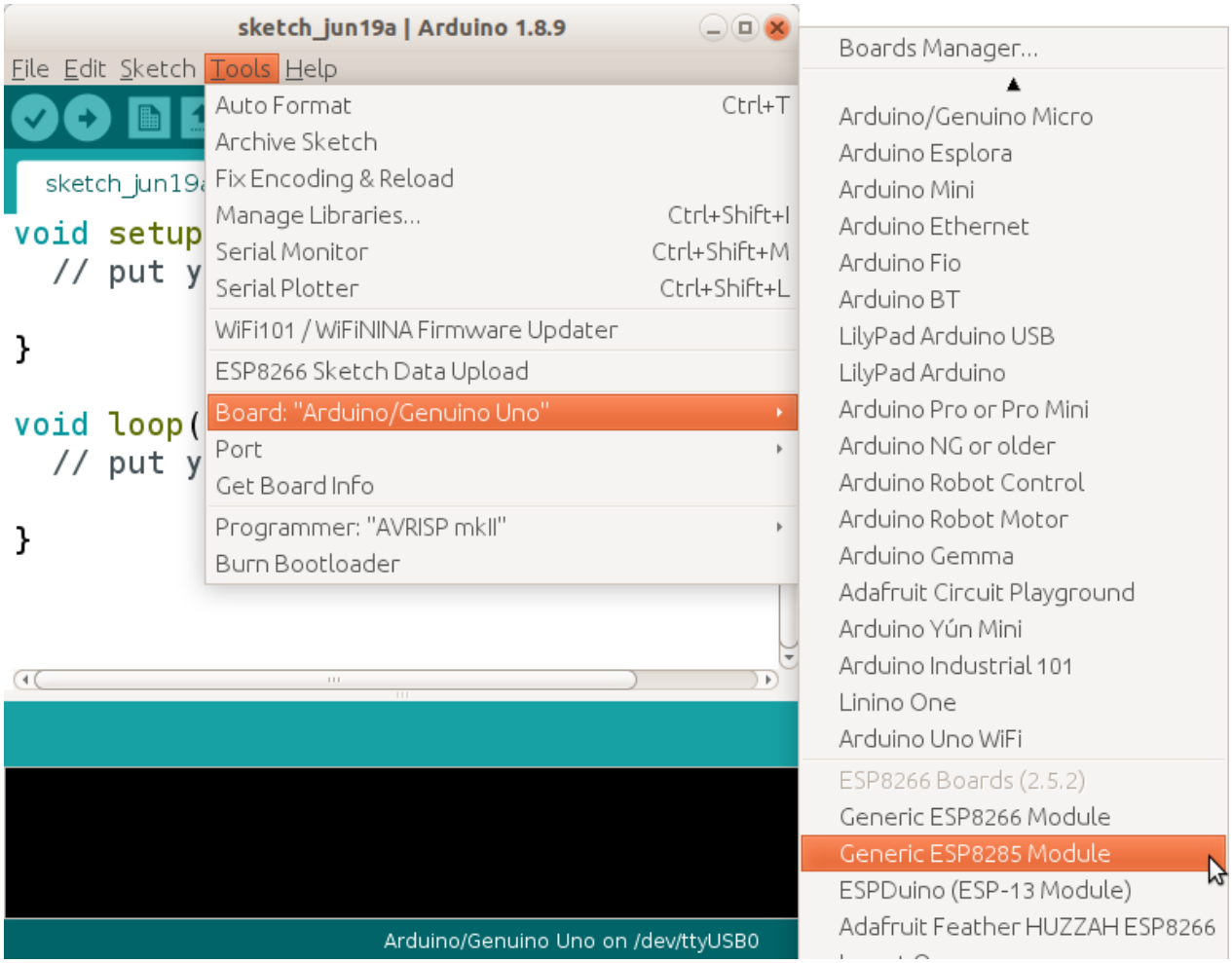
Then go to *Tools* > *Board* > *Board Manager* and search for "*esp8266*". Select the newest version, and click install. After this many sketch examples, and boards will be installed.



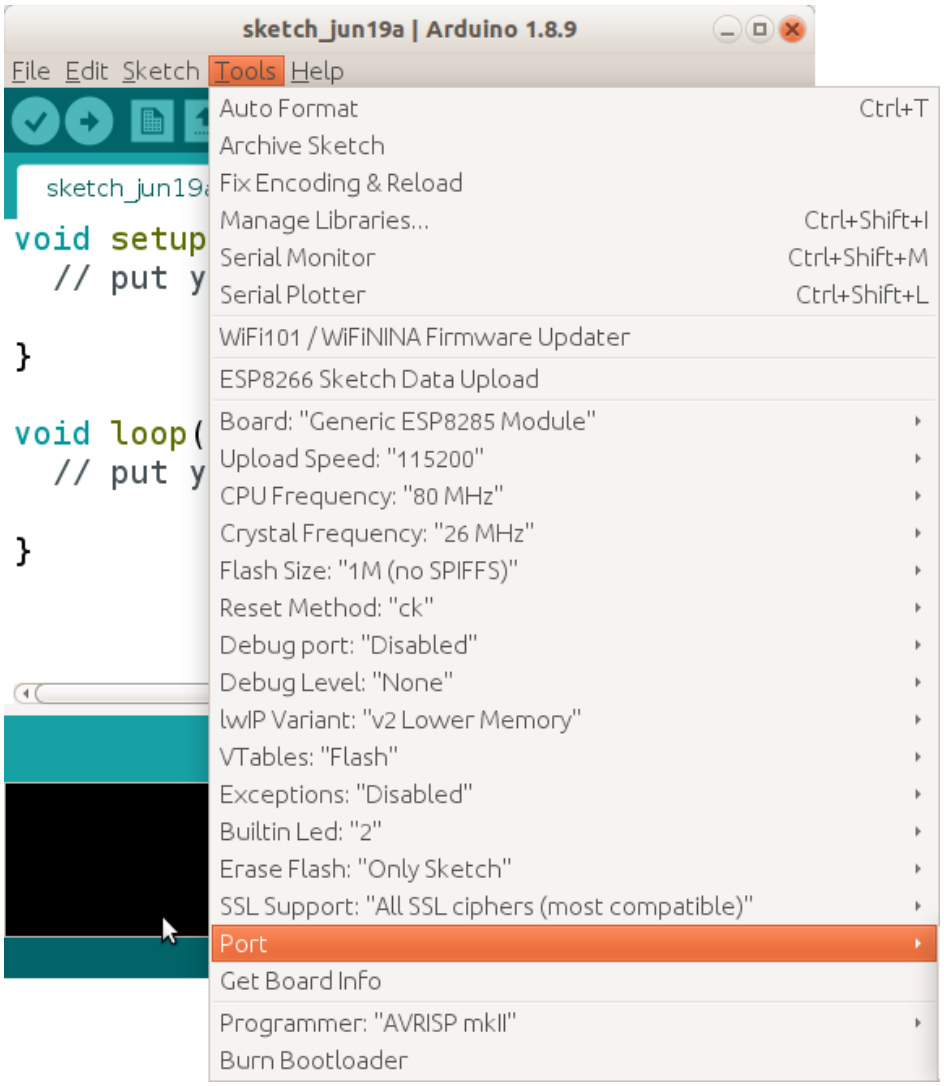


Now we have to setup which board to program.

Go to *Tools* > *Board* > *{board name}* and choose *"Generic ESP8266 Module".*



When you choose this board there are some other options you can setup too. When you open drop down menus *"Tools"* after you choose *"Generic ESP8266 Module"*, there will be several options which you can setup. For purpose of this eBook leave it as it is, because we won't go into details with it. Just choose port on which your board is connected.



Application example

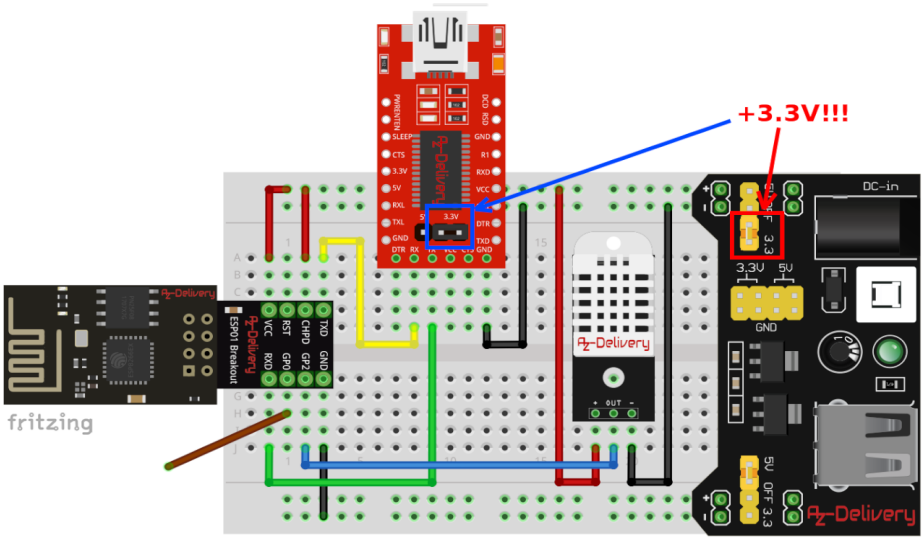
In this example we will show you how to connect ESP8266-01S with DHT22 temperature and humidity sensor.

We wrote about DHT22 in our eBook Quick Starter Guide DHT22 sensor https://www.az-delivery.de/products/dht-22-modul-kostenfreies-e-book?_pos=3&_sid=8bfd4bac3&_ss=r&ls=en.
So we won't go into details with this sensor. All you have to do is to download *"SimpleDHT"* library from, and add it to the Arduino IDE if you didn't use DHT22 before this. To download this library go to link: https://github.com/winlinvip/SimpleDHT .
When you download the *".zip"* file, open Arduino Uno and go to:
Sketch > *Include Library* > *Add .ZIP Library*, and add the downloaded zip file. After this go to: *File* > *Examples* > *SimpleDHT* > *DHT22Default* and open that sketch. We will use code from this sketch in our sketch.

DHT22 sensor can work on both +3.3V and +5V so we connect DHT22 to +3.3V because ESP works on +3.3V and will be destroyed if DHT22 is connected to +5V. When DHT22 is connected to the +5V it will output +5V signal on OUT pin.

So make sure to connect DHT22 sensor to +3.3V power supply!

Connect everything like on connection diagram below:



Every connection is the same as on last connection diagram in this eBook, except the part with DHT22.

DHT22 pin	**>**	**ESP pin and power supply pins**		
"+" pin	>	+3.3V	[power supply]	**Red wire**
OUT pin	>	GPIO pin 2	[ESP]	**Blue wire**
"-" pin	>	GND	[power supply]	**Black wire**

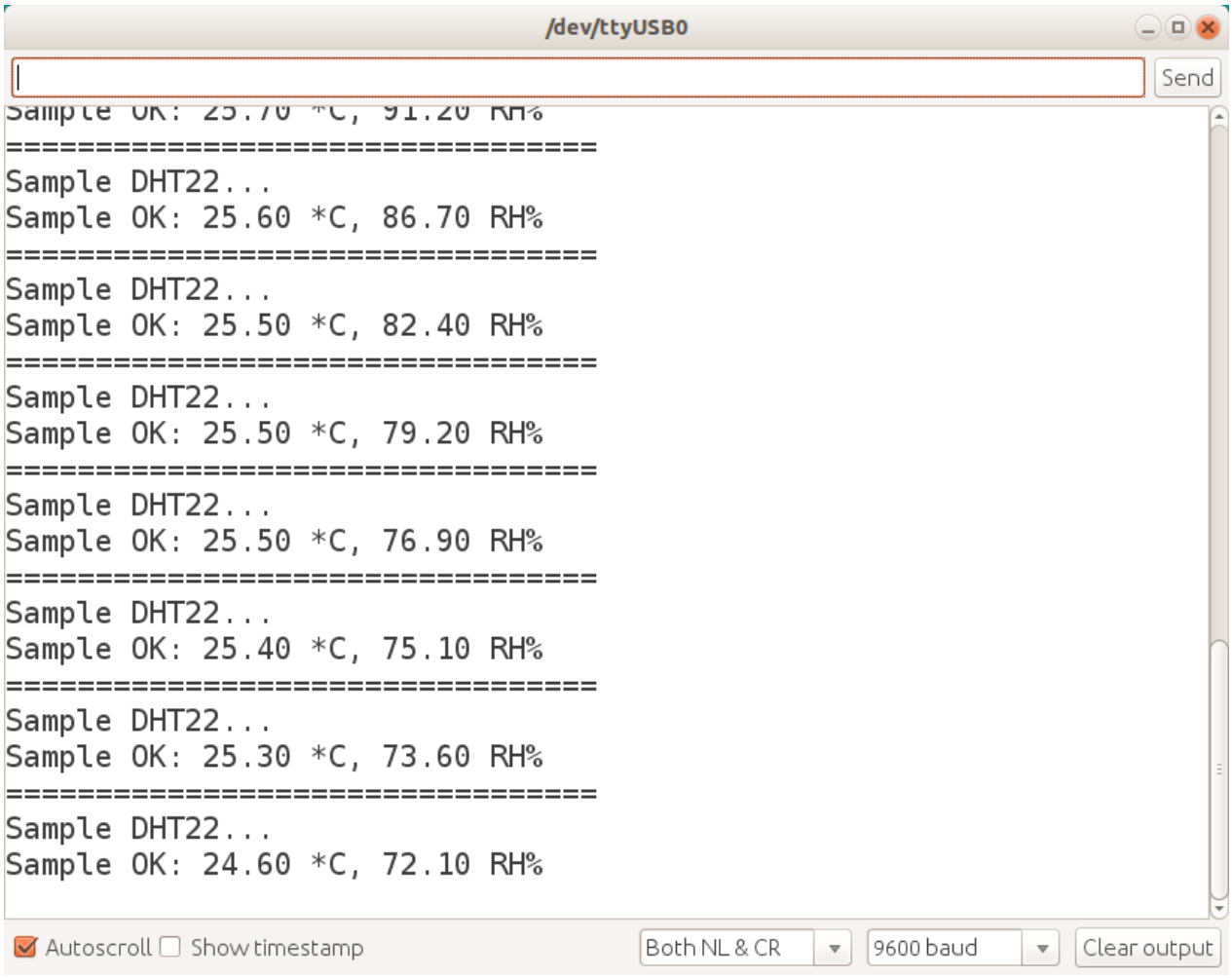
When you open this sketch *"DHT22Default"*, and then choose in *Tools > Board > {board name}*, first *“Generic ESP8266 Module”*. Then when you start uploading the sketch it will be compiled for ESP8266.

We just changed a few things, just for better readability, but the sketch is the same. Here is our sketch code:

```
#include <SimpleDHT.h>
int pinDHT22 = 2;
SimpleDHT22 dht22(pinDHT22);
float temperature = 0;
float humidity = 0;
void setup() {
  pinMode(pinDHT22, INPUT);
  Serial.begin(9600);
}
void loop() {
  temperature = 0;
  humidity = 0;
  Serial.println("=================================");
  Serial.println("Sample DHT22...");
  int err = SimpleDHTErrSuccess;
    if((err = dht22.read2(&temperature, &humidity, NULL)) !=
SimpleDHTErrSuccess) {
    Serial.print("Read DHT22 failed, err=");
    Serial.println(err);
    delay(2000);
    return;
  }
  Serial.print("Sample OK: ");
  Serial.print((float)temperature); Serial.print(" *C, ");
  Serial.print((float)humidity); Serial.println(" RH%");
  delay(2500);
```

}

And when you start Serial Monitor (*Tools* > *Serial Monitor*) the output should be something like this: (like with Arduino or Raspberry Pi)



We used 9600 baud rate in the sketch, so make sure that in Serial Output, the 9600 baud rate is set.

You've done it, you can now use your module for your projects.

Now it is 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 products for Arduino and Raspberry Pi, 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