Welcome!

And thank you for purchasing our AZ-Delivery ESP8266-01. On the following pages, we will go together with you through the first steps of the programming.

Enjoy!

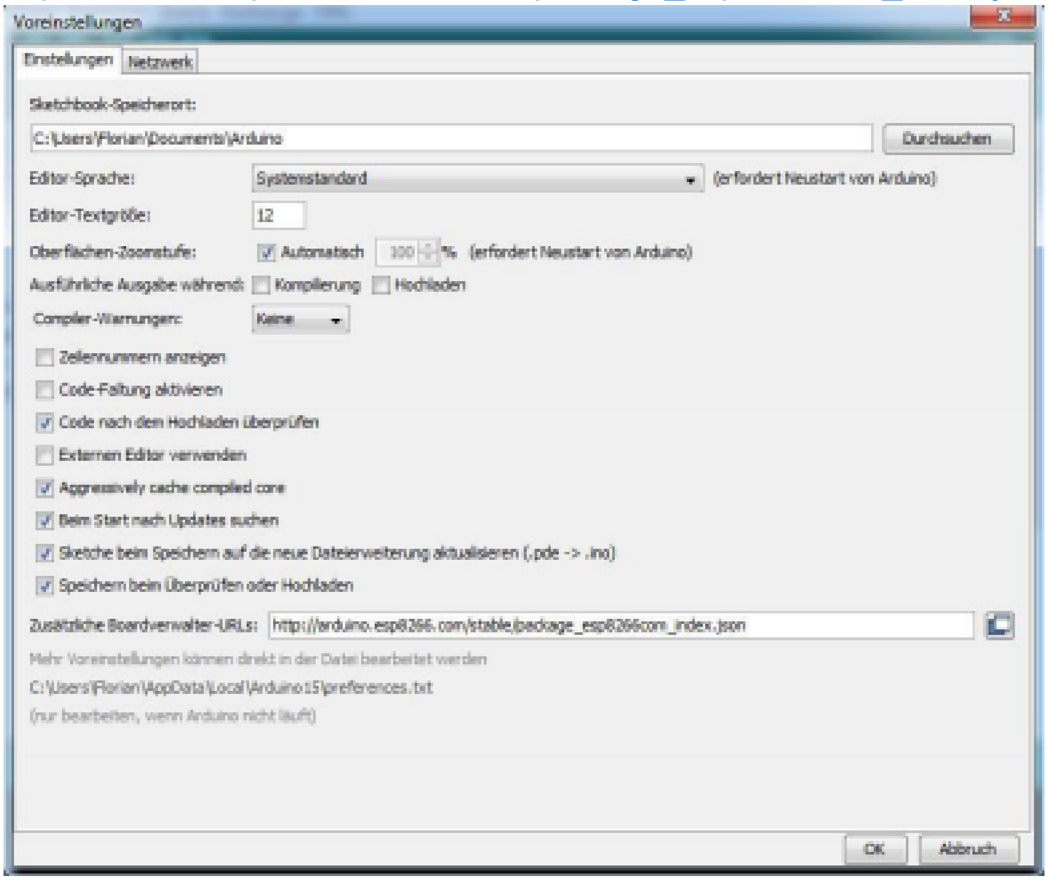
The ESP8622-01 has a wide range, thanks to its WLAN 802.11 b/g/n standard and can be used universally. The module supports three operating modes:
WLAN Router (AP), WLAN Client (STA), and both at the same time (AP + STA)! The powerful 80MHz processor and 1MB memory make the ESP8266-01 suitable for many applications. It is limited in contrast to its big brother (ESP8266-12E) by 2 GPIO pins.

Preparing the software:

In this step we see the Arduino software as *installed*, if that is not the case with you, you can always download it at https://www.arduino.cc/en/Main/Software# and install it on your PC. In addition, the drivers for the CH340 should have already been installed, if they are not, you can get the drivers here from us: CH340.

After all basic requirements have been met, we can now start with the installation of the software. First of all, the Arduino software needs all the information about the ESP8266- 01S. We can do that by entering the

following address under "Preferences" > "Additional Board Administrator URLs":
http://arduino.esp8266.com/stable/package_esp8266com_index.json


Voreinstellungen

Einstellungen | Netzwerk

Sketchbook-Speicherort:
C:\Users\Florian\Documents\Arduino [Durchsuchen]

Editor-Sprache: Systemstandard (erfordert Neustart von Arduino)
Editor-Textgröße: 12
Oberflächen-Zoomstufe: ✓ Automatisch 100 % (erfordert Neustart von Arduino)
Ausführliche Ausgabe während: ☐ Kompilierung ☐ Hochladen
Compiler-Warnungen: Keine

- ☐ Zeilennummern anzeigen
- ☐ Code-Faltung aktivieren
- ✓ Code nach dem Hochladen überprüfen
- ☐ Externen Editor verwenden
- ✓ Aggressively cache compiled core
- ✓ Beim Start nach Updates suchen
- ✓ Sketche beim Speichern auf die neue Dateierweiterung aktualisieren (.pde -> .ino)
- ✓ Speichern beim Überprüfen oder Hochladen

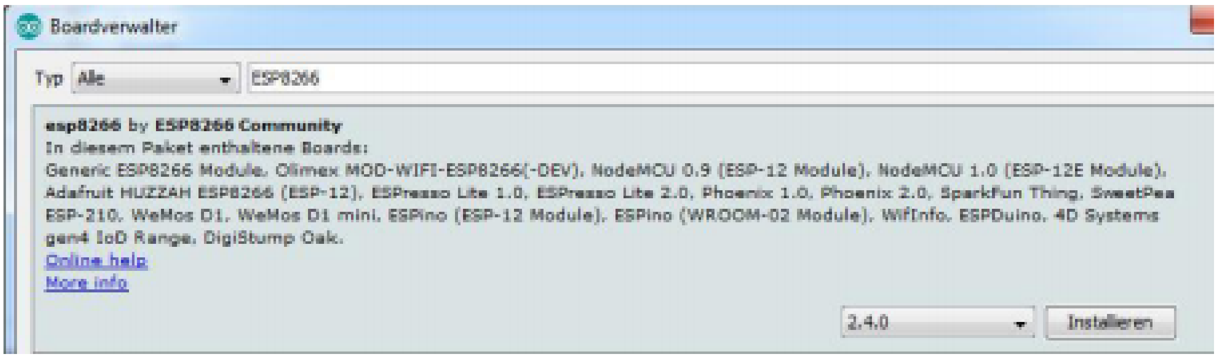
Zusätzliche Boardverwalter-URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

Mehr Voreinstellungen können direkt in der Datei bearbeitet werden
C:\Users\Florian\AppData\Local\Arduino15\preferences.txt
(nur bearbeiten, wenn Arduino nicht läuft)

OK | Abbruch

If you have already entered a link, click on the button and add a new line in the window.
Confirm the entry with "*OK*".

When that is completed, go to "Tools" > "Board" > "Board Administrator" and install the ESP8266 library. In the board manager, enter "ESP8266" in the search bar, which is located at the top right, and then the package of ESP8266 Community will be displayed. Select this package and then click on *Install*.


Boardverwalter

Typ Alle | ESP8266

esp8266 by **ESP8266 Community**
In diesem Paket enthaltene Boards:
Generic ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Adafruit HUZZAH ESP8266 (ESP-12), ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, SparkFun Thing, SweetPea ESP-210, WeMos D1, WeMos D1 mini, ESPino (ESP-12 Module), ESPino (WROOM-02 Module), WifInfo, ESPDuino, 4D Systems gen4 IoD Range, DigiStump Oak.
Online help
More info

2.4.0 | Installieren

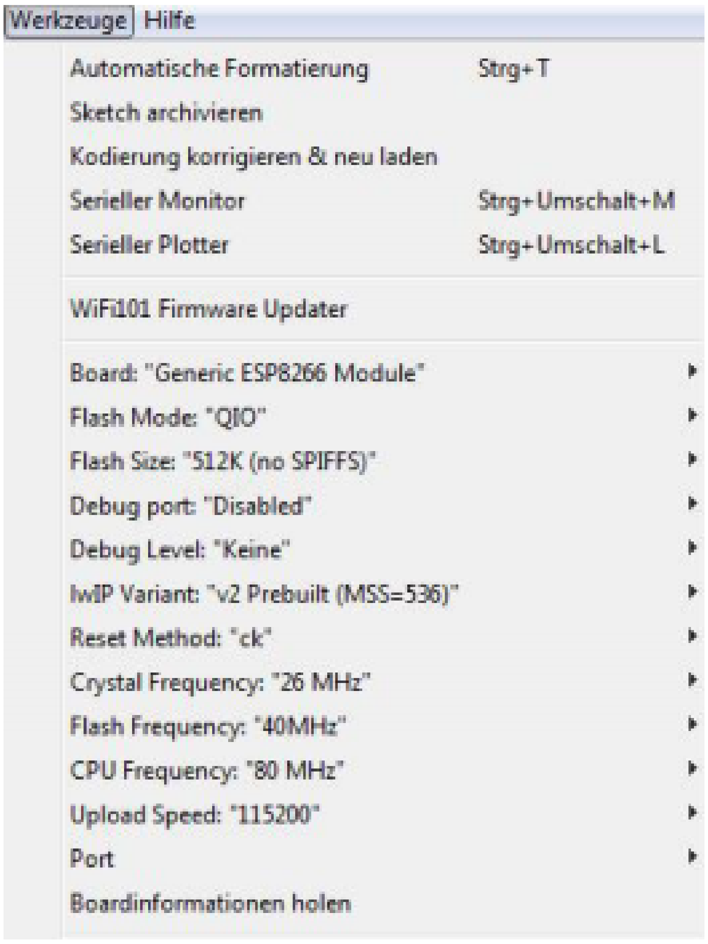
If the installation had been successful, then next to the package *INSTALLED*

will appear.
The next step is to select the correct board:

Under Tools >

Board: „Generic ESP8266 Module“

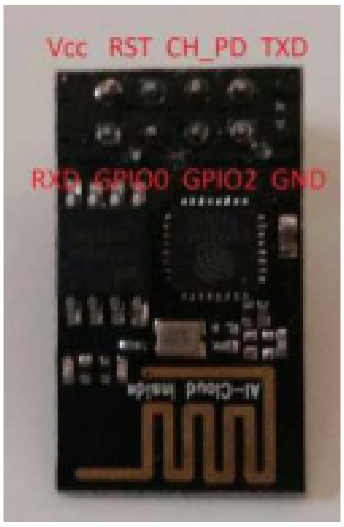
Port: “COMxx” (here is your port of the serial adapter)


Werkzeuge
Hilfe
Automatische Formatierung Strg+T
Sketch archivieren
Kodierung korrigieren & neu laden
Serieller Monitor Strg+Umschalt+M
Serieller Plotter Strg+Umschalt+L
WiFi101 Firmware Updater

Board: "Generic ESP8266 Module"
Flash Mode: "QIO"
Flash Size: "512K (no SPIFFS)"
Debug port: "Disabled"
Debug Level: "Keine"
lwIP Variant: "v2 Prebuilt (MSS=536)"
Reset Method: "ck"
Crystal Frequency: "26 MHz"
Flash Frequency: "40MHz"
CPU Frequency: "80 MHz"
Upload Speed: "115200"
Port
Boardinformationen holen

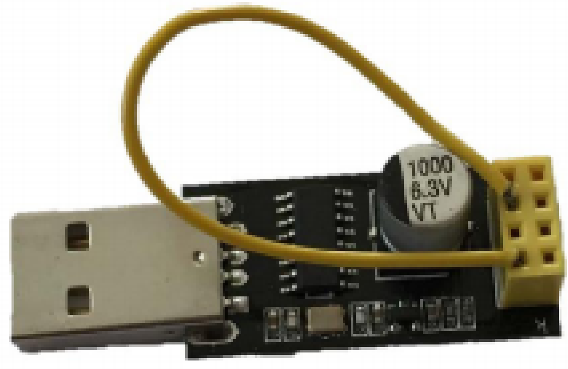
With that, all of the basic settings have been made, and now it is time for the wiring. To be able to program the ESP8266-01, firstly, it needs to be set to the programming mode.

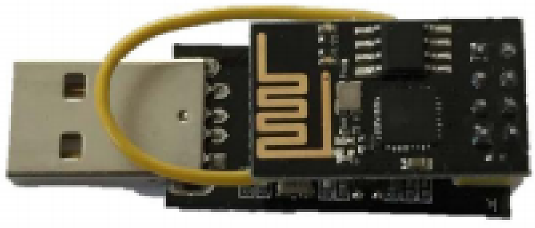
Wiring the module with the serial adapter:

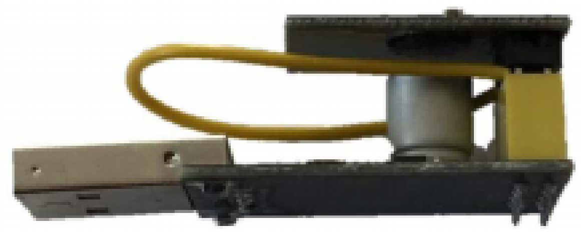


GPIO02 must be grounded while booting, in order to allow the chip to enter programming mode. To do this create a bridge between GPIO2 and GND.

This bridge must be removed after programming.



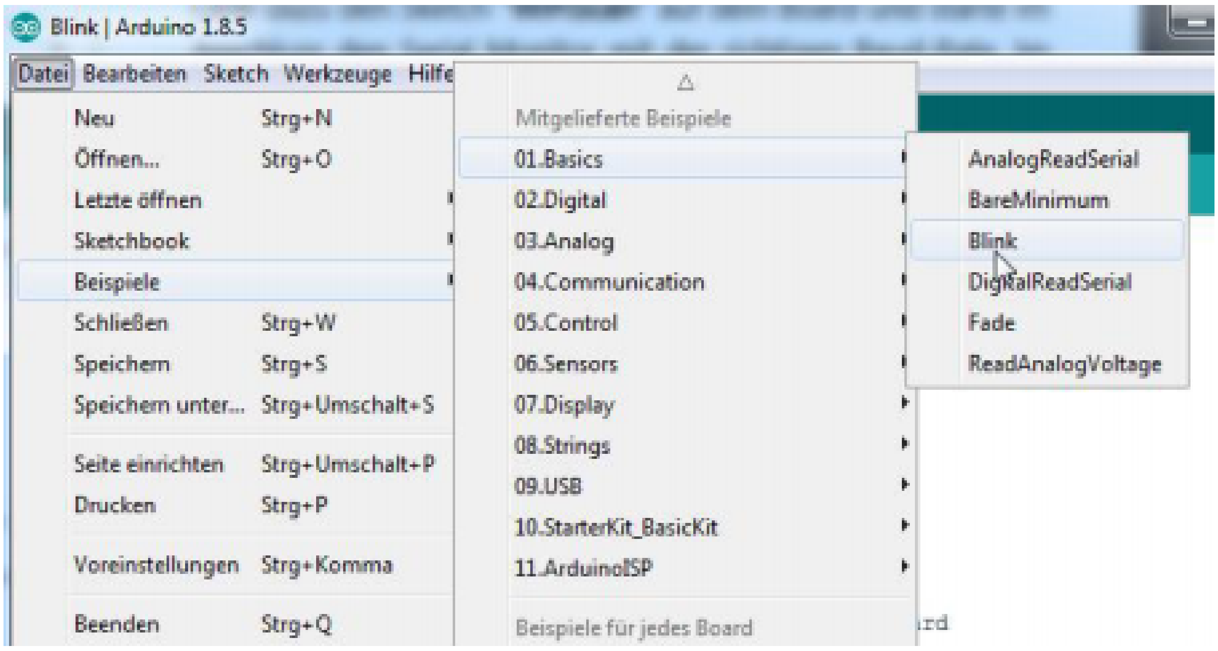




The Code:

Now that the wiring has been done and the adapter plugged in, we can commence with writing our first code. Let the LED flash directly on the ESP8266.

To do this, click on File > Examples > 01.Basics > **Blink**.

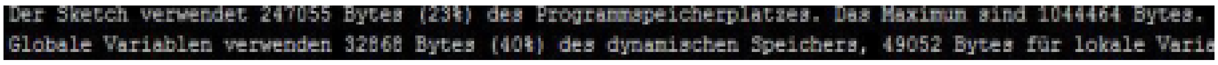


```
void setup() {
   pinMode(LED_BUILTIN, OUTPUT);
}
void loop() { digitalWrite
    (LED_BUILTIN, HIGH);
   delay(1000); digitalWrite
    (LED_BUILTIN, LOW);
   delay(1000);
}
```

If the LED with "LED_BUILTIN" does not flash, you have to replace LED_BUILTIN with port 1 and try again:

```
void setup() {
   pinMode(1, OUTPUT);
}
void loop() {
   digitalWrite (1,
   HIGH); delay(1000);
   digitalWrite (1,
   LOW); delay(1000);
}
```

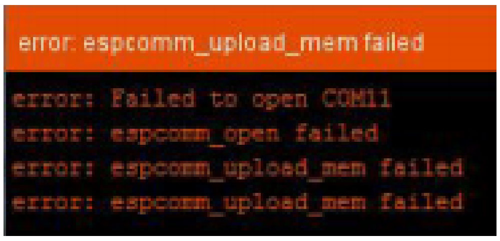
After we have opened or changed the code, we click above on and verify our program: If everything is correct and our program contains no errors,



```
Der Sketch verwendet 247055 Bytes (23%) des Programmspeicherplatzes. Das Maximum sind 1044464 Bytes.
Globale Variablen verwenden 32868 Bytes (40%) des dynamischen Speichers, 49052 Bytes für lokale Varia
```

We can upload it to the ESP8622-01. For that, click above on .

If our ESP8266-01 is not in programming mode, then we get the following message:



```
error: espcomm_upload_mem failed

error: Failed to open COM11
error: espcomm_open failed
error: espcomm_upload_mem failed
error: espcomm_upload_mem failed
```

Then simply disconnect the power supply, for a brief moment, and then reconnect or reset it again and check whether the GPIO02 is grounded.

Start the upload again and now it should work.

After the programming process, as already mentioned, disconnect again the light blue marked cables!

Now the LED will flash every second, but this is only a small part of what your ESP8266-01 can do.

Using the ESP8266-01 as a WLAN module for the microcontroller

Basics of communication via AT commands

The AZ-Delivery esp8266 WLAN module comes preinstalled with firmware that allows the connected devices, such as an Arduino, to communicate with it via AT commands. The AT instruction set is relatively old (~ 30 years) and was originally developed for communication with modems.

However, its simplicity and economy of resources make it ideal for communication with the ESP8266-01.

All AT commands begin with "**AT+**" followed by the actual command. It is also important that all AT commands end with a line break (CR and NL). For this reason, if you send a command over the serial connection between ESP8266 and Arduino, you should always use *println,* so that the line break is sent and the module is told to execute the command.

The AT commands can be found in the datasheet of ESPRESSIF:

https://www.espressif.com/sites/default/files/documentation/4a
esp8266_at_instruction_set_en.pdf

Download a Website

In this example, we would like to inform AZ-Delivery ESP8266 -01 about AT commands that allow the download of a website. To do this, the WLAN module must first be set to "WLAN Client" mode via the **AT + CWMODE = 1** command. Then it can be connected to a Wi-Fi network via **AT + CWJAP = "SSID", "Password"**.

Now you can use **AT + CIPSTART = "TCP", "website.de", 80** to connect to the target server. Then you have to inform the Wi-Fi module via **AT + CIPSEND = NUMBER**, how long the request is, which you want to send to the server. You still have to add a few characters, because the line breaks have to be counted at the end.

The request to the server is based on the principle *GET /Paddy_to_Website.html*. Now you may have to insert spaces until you have reached the previously specified number of characters, whereupon the ESP8266 sends the request automatically to the server, and automatically sends the answer (HTML code of the website) back to the Arduino. As long as the specified number of characters is not reached, the ESP8266 does nothing. Should the query be longer than specified, it will output "*busy*".

Due to the complexity of this procedure, it should first be carried out on a PC via the serial console, before translating the procedure into the Arduino code, otherwise, troubleshooting will be very difficult.

You did it! Now you can program and actualize your own projects with the ESP8266-01!

Now it is time for experimenting!

And for more hardware our online store is always at your disposal: https://az-delivery.de

Enjoy!
Imprint

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