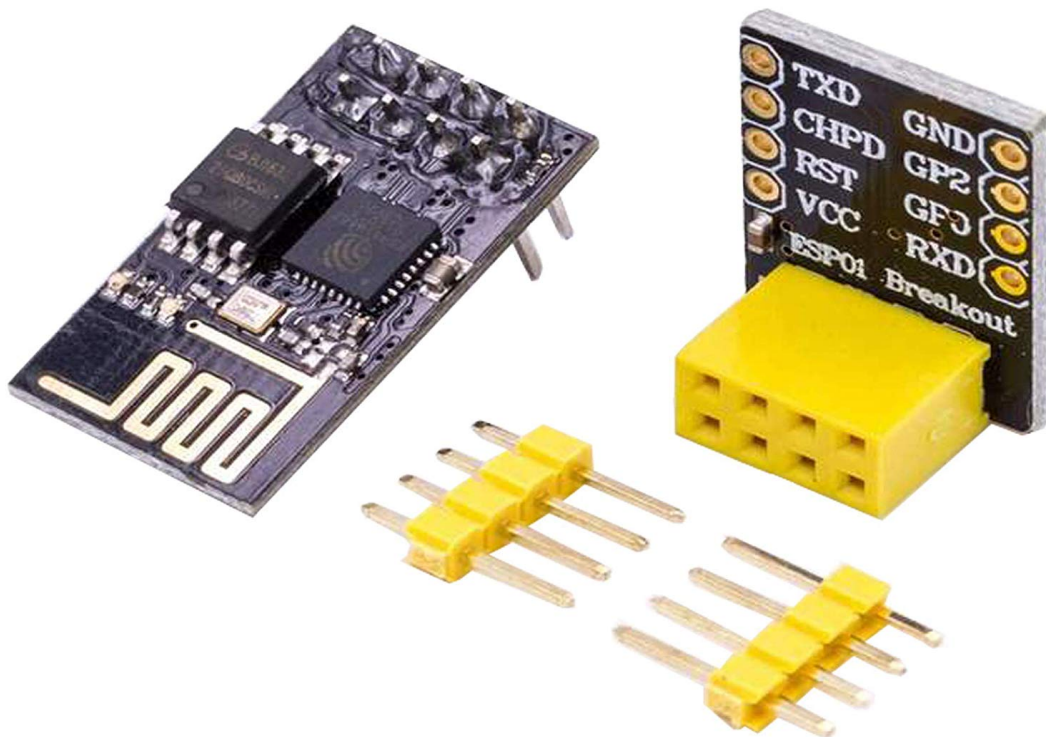


# AZ-Delivery

## Willkommen!

Vielen Dank, dass sie sich für unser "ESP8266-01S"-Modul mit einem Breadboardadapter von AZ- Delivery entschieden haben. In den nachfolgenden Seiten werden wir Ihnen erklären wie Sie das Gerät einrichten und nutzen können.

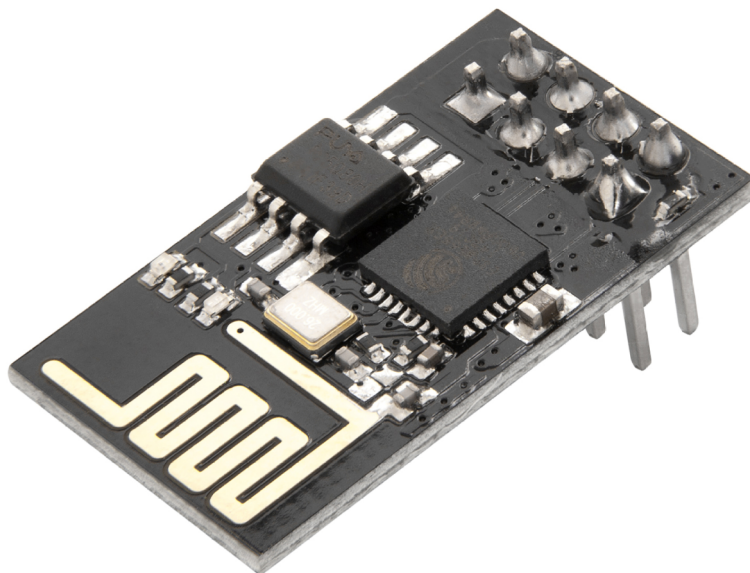
Viel Spaß!



# Az-Delivery

Das ESP8266-Modul ist ein "System on a Chip" (SoC), das von der chinesischen Firma Espressif hergestellt wird. Es besteht aus einem Tensilica L106 32-Bit-Mikrocontroller und einem Wifi-Transceiver. Es hat 11 GPIO-Pins (General Purpose Input/Output) und einen analogen Eingang. Das bedeutet, dass Sie ihn wie jeden normalen Arduino oder jeden anderen Mikrocontroller programmieren können. Und das Beste an dem ESP8266 ist, dass Sie mit ihm über Wifi kommunizieren können, so dass Sie ihn benutzen können, um eine Verbindung zu Ihrem Wifi-Netzwerk herzustellen, sich mit dem Internet zu verbinden, einen Webserver mit echten Webseiten zu hosten, Ihr Smartphone mit ihm verbinden zu lassen, usw.

Der ESP8266 ist ein erstaunlicher WiFi-Chip, der in verschiedenen Anwendungen der Heimautomatisierung eingesetzt werden kann. Dank des leistungsstarken 80MHz-Prozessors und eines großen 1MB-Speichers kann der ESP8266-01S auch autonom betrieben werden.



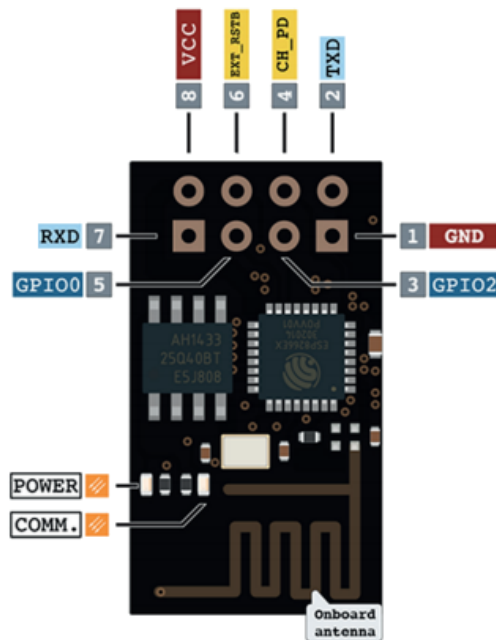


## Technische Daten

- » 802.11 b/g/n
- » Integrierte stromsparende 32-Bit-MCU
- » Integrierte 10-bit ADC
- » Integrierter TCP/IP Protokoll-Stapel
- » Integrierter TR-Schalter, Balun, LNA, Leistungsverstärker und Anpassungsnetzwerk
- » Integrierte PLL, Regler und Leistungsverwaltungseinheiten
- » Unterstützte Antennendiversität
- » Wi-Fi 2.4 GHz, unterstützt WPA/WPA2
- » Unterstützt STA/AP/STA+AP Betriebsmodi
- » Unterstützt Smart Link-Funktionen für Android- und iOS-Geräte
- » SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO
- » STBC, 1x1 MIMO, 2x1 MIMO
- » A-MPDU & A-MSDU-Aggregation und 0,4s Schutzintervall
- » Tiefschlafleistung <10uA, Abschaltleckstrom < 5uA
- » Aufwecken und Übertragen von Paketen in < 2ms
- » Standby-Leistungsaufnahme von < 1,0 mW (DTIM3)
- » +20dBm Ausgangsleistung im 802.11b-Modus
- » Betriebstemperaturbereich: -40 °C ~ 125 °C

# Az-Delivery

Es gibt viele verschiedene Entwicklungsboards, die auf dem ESP8266 basieren, so wie der ESP8266-01S. Er hat 8 Pins (Pin-Belegung wie unten abgebildet).



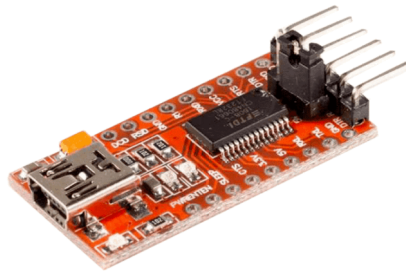
Wie Sie sehen können, haben wir zwei freie GPIO-Pins (GPIO0 und GPIO2). Wir können den GPIO0 und GPIO2 als digitale Eingänge oder Ausgänge verwenden, wie in einem Mikrocontroller. In unserem Beispiel (im späteren Verlauf des eBooks) verwenden wir den GPIO2 um die Daten von einem DHT22-Tempertursensormodul zu lesen.

Wir können den ESP8266-01S auf viele verschiedene Weisen programmieren, aber wir werden nur die Methode mit der Arduino-IDE behandeln. Wenn Sie bereits Arduino-Boards verwendet haben, dann wird dies sehr einfach für Sie. Denken Sie einfach daran, dass dies nicht die einzige Option ist. Es gibt viele andere Möglichkeiten den ESP8266 zu programmieren (offizielles ESP SDK für C-Programmierung, Lua-Interpreter, MicroPython-Firmware, nur einige von vielen).

# Az-Delivery

Um den ESP8266-01S zu programmieren, müssen wir RX- und TX-Pins an das Gerät mit serieller Schnittstelle wie das FTDI-Modul, den Arduino oder einen dedizierten USB-Adapter anschließen.

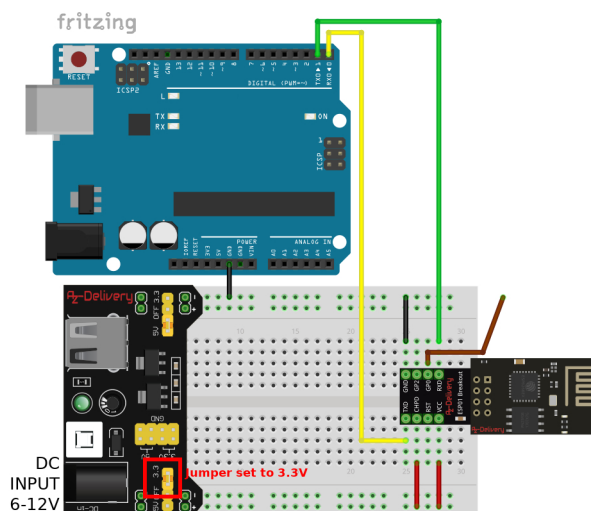
Der [FTDI](#) ist recht beliebt, da er zwischen 5V- und 3,3V-Logik umschalten kann. **Es ist wichtig, dass der USB-Seriell-Konverter mit 3,3V betrieben wird! Wenn Sie den ESP8266 mit einem 5V-FTDI-Modell betreiben, beschädigen Sie den ESP8266!!!**



FTDI adapter

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

Wenn Sie den Arduino Uno zum Programmieren des ESP8266-01S verwenden, ist dies der Anschlussplan (in diesem eBook nicht verwendet):



# Az-Delivery

Wir haben dies in unserem eBook für den ESP8266-01S mit Relais verwendet:

[https://www.az-delivery.de/products/esp8266-01-mit-relais-kostenfreies-e-book?\\_pos=2&\\_sid=ffbf81394&\\_ss=r&\\_ls=de](https://www.az-delivery.de/products/esp8266-01-mit-relais-kostenfreies-e-book?_pos=2&_sid=ffbf81394&_ss=r&_ls=de)

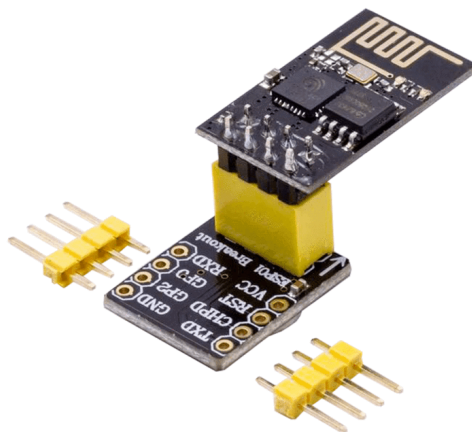
Man kann den auch ESP8266-01S mit einem dedizierten USB-Adapter (Bild unten) programmieren. Wird auch nicht in diesem eBook behandelt.



USB-Adapter mit ESP8266-01S

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

Wir werden ein Breadboard und einen FTDI-Adapter verwenden, um den ESP8266-01S zu programmieren, da dies der einfachste Weg ist. Andere Methoden sind ähnlich.

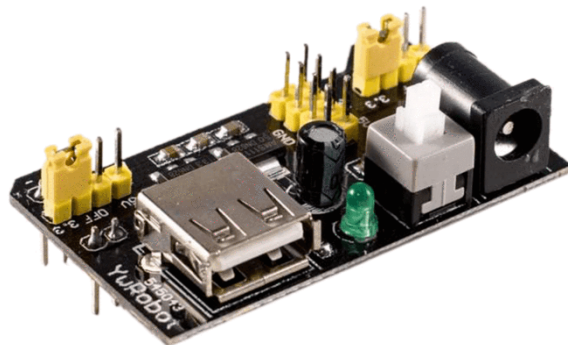


Breadboard-Breakout-Board mit ESP8266-01S

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

# Az-Delivery

Stellen Sie einfach sicher, dass der ESP8266-01S mit einem separaten Netzteil betrieben wird. Sie können unser Breadboard Netzteil MB102 verwenden. Es ist ein sehr einfaches und praktisches Netzteil, das Eingänge von 6 bis 12V DC toleriert und sowohl +3,3V als auch +5V ausgeben kann.



MB102 Netzteil für Breadboard

<https://www.az-delivery.de/products/mb102-breadboard?ls=en>

Es gibt zwei Möglichkeiten den ESP8266-01S zu programmieren. Erstens durch die Verwendung von AT-Befehlen (ATtention-Befehle) und zweitens durch die tatsächliche Programmierung des ESP8266-Chips mit Hilfe der Arduino-IDE.

Da die zweite Möglichkeit viel mehr Freiheit bei der Erstellung von Inhalten bietet, werden wir in diesem eBook nur diese Art der Programmierung behandeln.



## Aktivieren des Chips

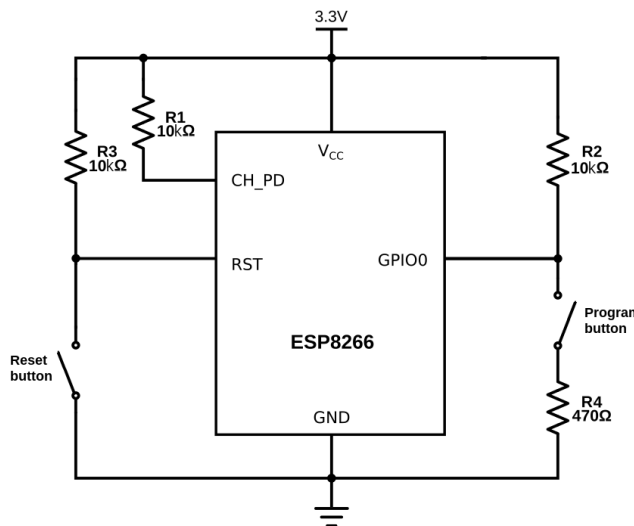
Wir müssen einige Widerstände hinzufügen, um den ESP8266-01S einzuschalten und den richtigen Bootmodus (Programmier- oder Normalmodus) zu wählen. Um die ordnungsgemäße Verwendung des ESP8266-01S-Moduls sicherzustellen, müssen wir dies zuerst tun:

1. Aktivieren Sie den Chip, indem Sie den **CH\_PD** (Chip Power Down, manchmal mit *CH\_EN* oder *CH\_PC* bezeichnet) Pin über einen **10kΩ** Widerstand an **VCC** anschließen. (R3 auf dem Breakout-Board).
2. Wählen Sie den Modus *NORMAL*, indem Sie den Pin **GPIO0** über einen Widerstand von **10kΩ** an **VCC** anschließen. (wir verbinden ihn mit **GND** im PROGRAMM-Modus; für den NORMAL-Modus lassen wir ihn unverbunden)
3. Verhindern Sie zufällige Resets, indem Sie den **RST**-(Reset-)Pin über einen **10kΩ** Widerstand an **VCC** anschließen. (Sie können ihn anschließen, aber es ist nicht zwingend notwendig; hier lassen wir ihn unverbunden)



## Hinzufügen der RESET- und PROGRAMM-Tasten

Unsere ESP8266-07-Boards haben keinen *RESET*-Knopf oder einen Knopf für den Programmier-Modus. Sie könnten (aber nicht zwingend nötig) einen hinzufügen, wie unten abgebildet:



Für den Reset-Knopf schließen Sie einen Druckknopf zwischen dem RST-Pin und GND an. Schalten Sie jedoch einen PULL-UP-Widerstand 10kΩ (R3 auf dem Bild oben) zwischen RST und VCC zu.

Um den Chip in den Programmiermodus zu versetzen, müssen Sie *GPIO0* während des Starts auf *LOW* schalten (mit GND verbunden). Wenn es auf *HIGH* steht oder unverbunden ist, wird der Chip im Normal-Modus gestartet. Da es möglich ist, *GPIO0* als Ausgang zu benutzen, können wir ihn nicht direkt mit Masse kurzschließen, da es den Chip beschädigen könnte. Um dies zu verhindern, schließen Sie einen Widerstand mit 470Ω in Reihe mit dem Schalter an. Es ist wichtig, dass dieser Widerstand niedrig genug ist, andernfalls wird er durch den 10KΩ Widerstand auf high geschaltet (R2 auf dem obigen Bild), wenn Sie sich dafür entschieden haben R2 hinzuzufügen.

# Az-Delivery

Für die Zwecke dieser Anleitung verwenden wir keinen Knopf. Um den ESP in den Programmiermodus zu versetzen, verwenden wir einen Überbrückungsdraht, der *GPIO0* mit *GND* für den Programmiermodus verbindet, und trennen ihn im normalen Modus von *GND*.

Um den ESP8266-07 zurückzusetzen, schalten wir die Stromversorgung aus und wieder ein.

Es gibt einige Dinge, auf die Sie bei der Verwendung eines ESP8266-01S achten müssen: Das Wichtigste ist, dass er mit 3,3V läuft. Wenn Sie ihn also an eine 5V-Stromversorgung anschließen, wird er zerstört. Im Gegensatz zu einigen 3,3V-Arduino-Boards sind die E/A-Pins des ESP8266 nicht 5V-tolerant. Wenn Sie also einen 5V-USB-zu-Seriell-Konverter oder 5V-Sensoren usw. verwenden, werden Sie es zerstören.

**Eine weitere Sache, die man im Auge behalten sollte, ist, dass der ESP8266 nur 12mA pro Ausgangspin erzeugen oder senken kann, im Vergleich zu 20 - 40mA bei den meisten Arduinos!**

Eine letzte Sache, die man bedenken sollte, ist, dass das ESP8266 die Systemressourcen und die CPU-Zeit zwischen Ihrem Sketch und dem Wi-Fi-Treiber aufteilen muss. Außerdem werden Funktionen wie PWM, Interrupts oder I<sup>2</sup>C in der Software emuliert. Die meisten Arduinos haben dagegen dedizierte Hardwareteile für diese Aufgaben. Für die meisten Anwendungen ist dies jedoch kein allzu großes Problem. Sie müssen allerdings darüber nachdenken, wenn Sie einen Sketch entwerfen.



## Der ESP8266 als Mikrocontroller - Hardware

Obwohl der ESP8266 oft als "einfache" Serial-to-Wifi-Brücke verwendet wird, ist er an sich ein sehr leistungsfähiger Mikrocontroller.

### Digitale E/A

Genau wie ein normales Arduino verfügt das ESP8266 über digitale Ein-/Ausgangspins oder GPIO - General Purpose Input/Output Pins. Wie der Name schon sagt, können sie als digitale Eingänge verwendet werden, um eine digitale Spannung zu lesen, oder als digitale Ausgänge, um entweder 0V (Senkenstrom) oder 3,3V (Quellenstrom) auszugeben.

Der ESP8266 ist ein 3,3V-Mikrocontroller. Die E/A arbeiten auch mit 3,3V.

- Die Pins sind nicht 5V-tolerant, das Anlegen von mehr als 3,6V **AN JEDEM PIN führt zur Zerstörung des Chips!**
- Der maximale Strom, der von einem einzelnen GPIO-Pin gezogen werden kann, beträgt 12mA!

### Verwendbare Pins

Der ESP8266 hat 17 GPIO-Pins (0-16), Sie können jedoch nur 2 davon verwenden, da der ESP8266-01S nur 2 verfügbare Breakout-Pins besitzt. Wenn Sie versuchen andere Pins zu verwenden, könnte Ihr Programm abstürzen. GPIO1 und GPIO3 werden als TX und RX der seriellen Schnittstelle (UART) der Hardware verwendet, so dass Sie sie in den meisten Fällen nicht als normale E/A beim Senden/Empfangen serieller Daten verwenden können.

## Boot-Modi

Wie im vorigen Kapitel erwähnt, haben einige E/A-Pins eine besondere Funktion während des Bootens: Sie wählen 1 von 3 Boot-Modi aus:

GPIO0	GPIO2	Modus
0V	3.3V	Uart Bootloader (PROGRAM mode)
3.3V	3.3V	Boot sketch (SPI flash)
x	x	SDIO Modus (wird nicht für den Arduino Uno verwendet)

**Hinweis:** Sie müssen keinen externen Pull-up-Widerstand zu GPIO2 hinzufügen, der Interne wird beim Booten aktiviert.

Wir erfüllen diese Bedingungen, indem wir (s. vorheriges Kapitel) externe Widerstände hinzugefügt haben. Dies hat jedoch einige Auswirkungen:

- » Der GPIO0 wird im normalen Betriebsmodus hochgezogen, so dass Sie ihn nicht als Hi-Z-Eingang verwenden können.
- » Der GPIO2 kann beim Hochfahren nicht niedrig geschalten werden, so dass Sie keinen Schalter anschließen können.

## Interne Pull-up/-down-Widerstände

Alle GPIO-Pins haben einen eingebauten Pull-up-Widerstand, wie bei einem Arduino. Der GPIO16 hat dagegen einen eingebauten Pull-down-Widerstand (Sie können aber im ESP8266-01S nicht auf diesen zugreifen).

# Az-Delivery

## **PWM**

Im Gegensatz zu den meisten Atmel-Chips (Arduino) unterstützt der ESP8266 keine Hardware-PWM, jedoch wird Software-PWM auf allen digitalen Pins unterstützt. Der Standard-PWM-Bereich beträgt 10 Bit bei 1kHz, aber dies kann geändert werden (bis zu >14 Bit bei 1kHz).

## **Serial**

Der ESP8266 verfügt über zwei Hardware-*UARTS* (serielle Schnittstellen). Sie können aber beim ESP8266-01S nur eine verwenden. *UART0* an den Pins 1 und 3 (jeweils *TX0* bzw. *RX0*).



## Der ESP8266 als Mikrocontroller - Software

Der Großteil der Mikrocontroller-Funktionalität des ESP verwendet genau dieselbe Syntax wie ein normaler Arduino, was den Einstieg einfach macht.

### Digitale E/A

Genau wie bei einem Arduino können Sie die Funktion eines Pins mit `pinMode(pin, mode)` einstellen, wobei "pin" die *GPIO*-Nummer ist und "mode" entweder *INPUT* (was die Voreinstellung ist), *OUTPUT* oder *INPUT\_PULLUP* sein kann, um die eingebauten Pull-up-Widerstände für *GPIO* 0-15 zu aktivieren. Um den Pulldown-Widerstand für *GPIO16* zu aktivieren, müssen Sie *INPUT\_PULLDOWN\_16* verwenden (nur generell, da beim ESP8266-01S kein Zugriff auf *GPIO16* möglich ist). Um einen Ausgangspin auf *HIGH* (3,3V) oder *LOW* (0V) zu setzen, verwenden Sie `digitalWrite(pin, value)`, wobei "pin" der digitale Pin ist und "value" entweder 1 oder 0 (*HIGH* und *LOW*). Um einen Eingang zu lesen, verwenden Sie `digitalRead(pin)`. Um die PWM an einem bestimmten Pin zu aktivieren, verwenden Sie `analogWrite(pin, value)`, wobei "pin" der digitale Pin und "value" eine Zahl zwischen 0 und 1023 ist. Sie können den Bereich des PWM-Ausgangs ändern, indem Sie `analogWriteRange(new_range)` verwenden. Die Frequenz kann durch `analogWriteFreq(new_frequency)` geändert werden, "new\_frequency" und sollte bei 100Hz-1000Hz liegen.

### Serielle Kommunikation

Um *UART0* (*TX* = *GPIO1*, *RX* = *GPIO3*) zu verwenden, können Sie das Serial-Objekt wie bei einem Arduino verwenden: `Serial.begin(baud_reate)`.

# Az-Delivery

**Alle Arduino Stream-Funktionen, wie *Lesen, Schreiben, Drucken, ...* werden ebenfalls unterstützt.**

## **Gemeinsame Nutzung der CPU-Zeit mit dem RF-Teil**

Eine Sache, die Sie beim Schreiben von Programmen für den ESP8266 beachten müssen, ist, dass Ihr Sketch Ressourcen (CPU-Zeit und Speicher) mit dem WLAN- und TCP-Stack (der Software, die im Hintergrund läuft und alle WLAN- und IP-Verbindungen handhabt) teilen muss. Wenn die Ausführung Ihres Codes zu lange dauert und Sie die TCP-Stacks nicht machen lassen, könnte es zu einem Absturz kommen oder Sie könnten Daten verlieren. Es ist am besten, die Ausführungszeit Ihres Codes unter ein paar hundert Millisekunden zu halten. Jedes Mal, wenn der Haupt-loop wiederholt wird, gibt Ihr Sketch dem WLAN und TCP zugunsten nach, um alle WLAN- und TCP-Anfragen zu bearbeiten. Wenn Ihr loop länger dauert, müssen Sie den Wifi/TCP-Stacks explizit CPU-Zeit geben, indem Sie `include delay(0) oder yield()` verwenden. Wenn Sie das nicht tun, funktioniert die Netzwerkkommunikation nicht wie erwartet, und wenn sie länger als 3 Sekunden dauert, setzt der soft WDT (Watchdog-Timer) das ESP zurück. Wenn der soft WDT deaktiviert ist, setzt der Hardware-WDT nach etwas mehr als 8 Sekunden den Chip zurück. Aus der Sicht eines Mikrocontrollers sind 3 Sekunden jedoch eine sehr lange Zeit (240 Millionen Taktzyklen), so dass Sie davon nicht betroffen sind, es sei denn, Sie führen ein extremes Zahlencrunching durch oder senden extrem lange Strings über die serielle Schnittstelle. Denken Sie einfach daran, dass Sie `yield()` innerhalb Ihrer for- oder while-loops hinzufügen, die länger als, sagen wir, 100ms dauern könnten.

# Az-Delivery

## **Wifi**

Die Verwendung des ESP8266 als einfacher Mikrocontroller ist großartig, aber der Grund, warum die meisten Leute ihn verwenden, sind seine Wifi-Fähigkeiten. Er unterstützt Netzwerkprotokolle wie wifi, TCP, UDP, HTTP, DNS, usw.

## **Upload von Sketches auf den ESP8266**

Der Upload-Prozess für ESP8266-Boards unterscheidet sich ein wenig von der normalen Arduino-Prozedur. Die meisten Arduino-Boards werden automatisch zurückgesetzt, wenn ein neues Programm hochgeladen wird, und gehen automatisch in den Programmiermodus. Bei einigen ESP-Boards müssen Sie manuell in den Programmiermodus wechseln. Bei den Barebones-Modulen müssen Sie sie sogar manuell zurücksetzen.



## Manueller RESET und Manuelles PROGRAMM

Dies gilt nur für Boards ohne integrierten USB-Seriell-Wandler, das ESP87266-07 mit Breakout-Board, das wir verwenden.

Um den ESP8266 in den *PROGRAM*-Modus zu bringen, muss *GPIO0* beim Booten low sein. In dieser Anleitung handelt es sich um Überbrückungsdrähte anstelle von Knöpfen, d.h. es geht hier darum, wie man zwischen Normal- und dem Programm-Modus wechselt:

- » Um in den *PROGRAM*-Modus zu gelangen:
  1. Schalten Sie das Modul aus (Stromversorgung trennen)
  2. Verbinden Sie den *GPIO0*-Pin über Jumperdraht mit GND.
  3. Schalten Sie das Modul ein (Spannungsversorgung anschließen), und der ESP8266-07 befindet sich im *PROGRAM*-Modus.
  
- » Um in den *NORMAL*-Modus zu gelangen:
  1. Schalten Sie das Modul aus (Stromversorgung trennen).
  2. Lassen Sie den *GPIO*-Pin unbesetzt, oder trennen Sie ihn von GND, falls er zuvor angeschlossen war. (Stellen Sie nur sicher, dass der Jumperdraht mit nichts anderem verbunden ist!)
  3. Schalten Sie das Modul ein (Stromversorgung anschließen), und der ESP8266-07 befindet sich im *NORMAL*-Modus.

# Az-Delivery

Natürlich können Sie *RESET*- und *PROGRAM*-Tasten hinzufügen, um diesen Vorgang zu vereinfachen. Wenn Sie sich dafür entscheiden, können Sie folgendermaßen in die PROGRAMM/NORMAL-Modi wechseln :

» Für den *PROGRAM-Modus*:

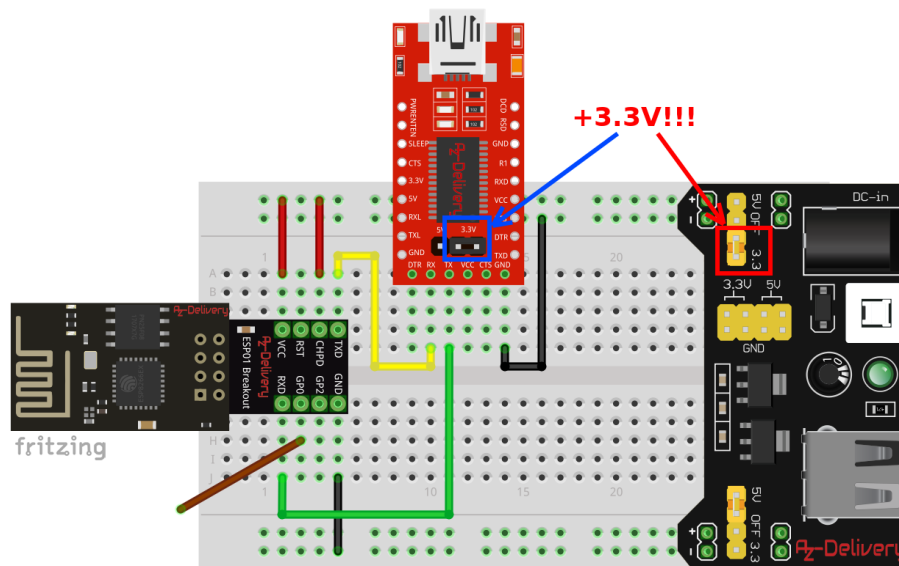
1. Drücken und halten Sie die Taste *RESET*
2. Drücken und halten Sie die *PROGRAM*-Taste
3. Lassen Sie die *RESET*-Taste los; ESP bootet im Programm-Modus
4. Lassen Sie die *PROGRAM*-Taste los.
5. den Sketch hochladen

» Für den *NORMAL-Modus*:

1. *RESET*-Taste drücken und loslassen, um das ESP zurückzusetzen. Der *PROGRAM*-Knopf sollte während dem Prozess nicht gedrückt sein.

## Anschlussplan des ESP8266-01S

Um den Vorschlägen der vorhergehenden Kapiteln zu folgen, schließen Sie ESP8266-01S wie im Anschlussplan unten an:



Verbinden Sie bei den drei Modulen alle GND-Pins miteinander!  
(Schwarze Drähte)

Für FTDI und ESP8266-01S:

1. Stellen Sie sicher, dass der Jumper für die Spannungsreferenz auf 3,3V gesetzt ist (wie bei FTDI gilt dies auch für den MB102)

# Az-Delivery

2. Verbinden Sie den **GND**-Pin vom FTDI mit **GND** vom ESP8266-01S.
3. Verbinden Sie den **RX**-Pin vom FTDI mit dem **TX**-Pin vom ESP8266-01S.  
**Gelber Draht** auf dem Anschlussschema.
4. Verbinden Sie den **TX**-Pin (FTDI) mit dem **RX**-Pin (ESP8266-01S). \*\*  
**Grüner Draht** auf dem Anschlussplan.

**VCC** von ESP8266-01S ist +3.3V angeschlossen (**Roter Draht**).

**CHPD**-Pin vom ESP8266-01S muss mit VCC verbunden sein (**Roter Draht**).

Für den Normal-Modus muss der GPIO0-Pin beim Start getrennt werden (**Brauner Draht**). Aber wenn wir den ESP8266-01S programmieren wollen, müssen wir das Modul zuerst herunterfahren (durch Trennen der Stromversorgung), den GPIO0-Pin mit GND (**Brauner Draht**) verbinden und dann wieder mit der Stromversorgung verbinden (starten).

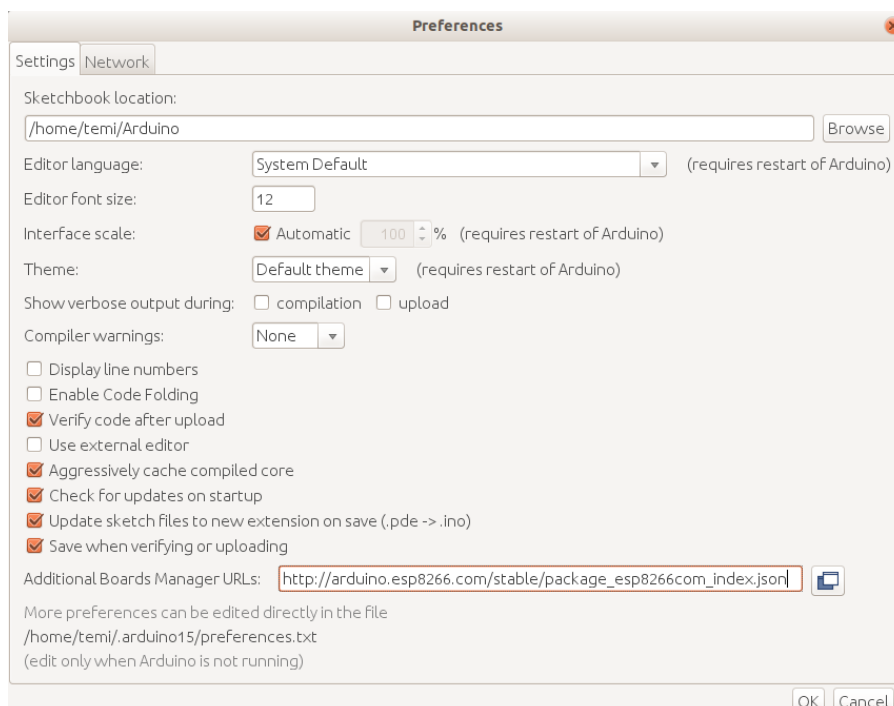
Um wieder in den Normal-Modus zu wechseln, müssen wir das Modul ausschalten, den GPIO0-Pin trennen (**Brauner Draht** unverbunden), das Modul wieder einschalten und es wieder mit der Stromversorgung verbinden (einschalten).

# Az-Delivery

## Software

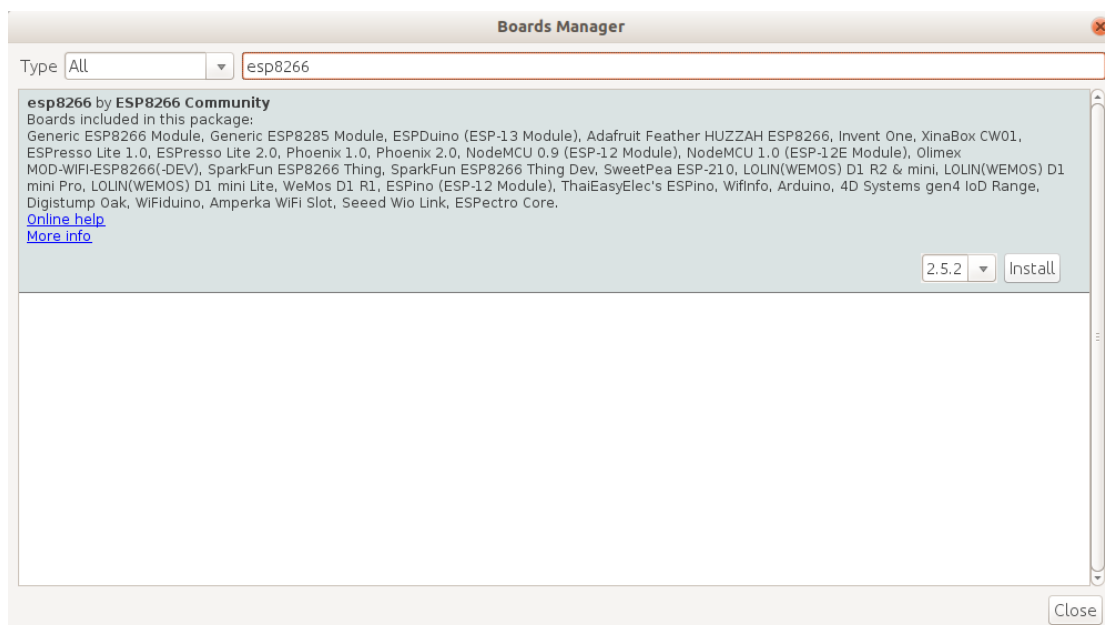
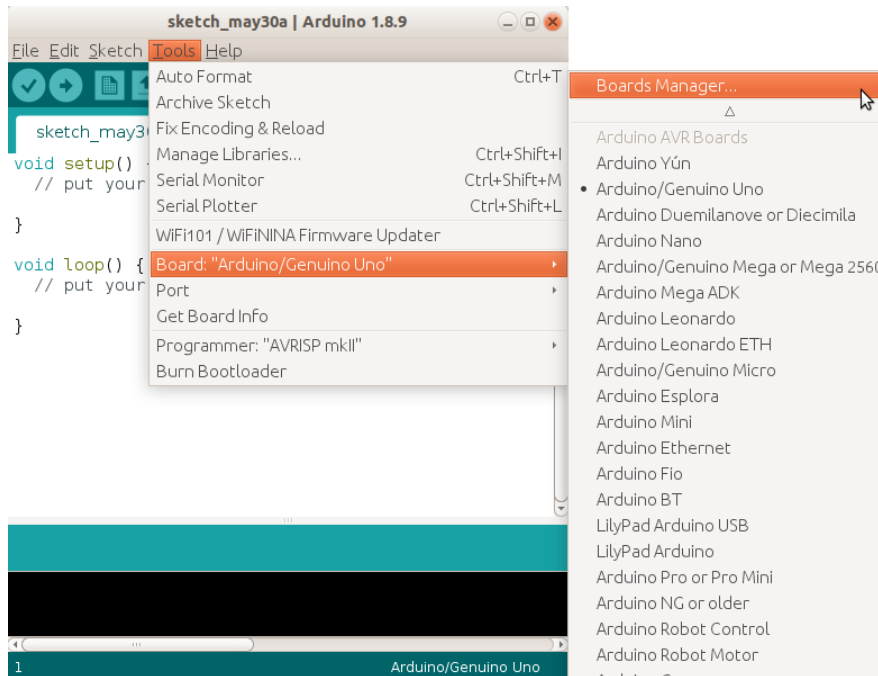
Um den ESP8266 mit der Arduino IDE zu verwenden, müssen wir zuerst die Arduino IDE einrichten. Der erste Schritt ist das Herunterladen und Installieren der Arduino IDE. Sollte sie noch nicht installiert sein, gehen Sie auf <https://www.arduino.cc/en/Main/Software> und downloaden und installieren Sie sie.

Um den ESP8266 zu programmieren, benötigen Sie ein Plugin für die Arduino IDE. Es kann manuell von [GitHub](#) heruntergeladen werden, aber einfacher ist es die URL in der Arduino IDE hinzuzufügen. Öffnen Sie die Arduino IDE, gehen Sie zu *File > Preferences*, kopieren Sie die URL [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json) in das respektive Feld vom "*Additional Board Manager*" (Zusätzliche URLs des Board-Managers). Sie können mehrere URLs hinzufügen, indem Sie sie durch Kommata trennen.



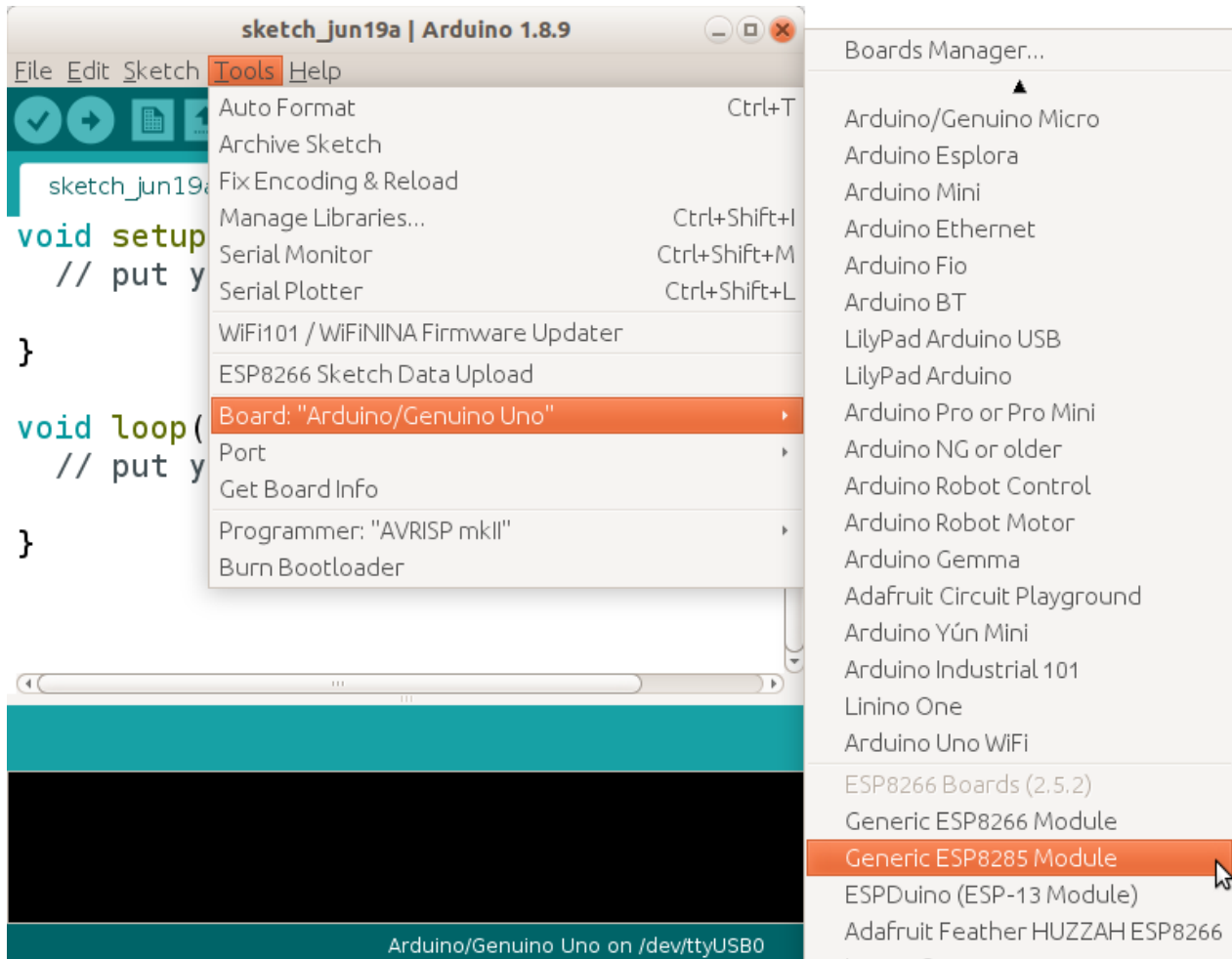
# Az-Delivery

Gehen Sie zu *Tools > Board > Board Manager* und suchen Sie "esp8266". Wählen Sie die neueste Version und klicken Sie auf "Installieren". Danach werden viele Sketchbeispiele und Boards installiert.



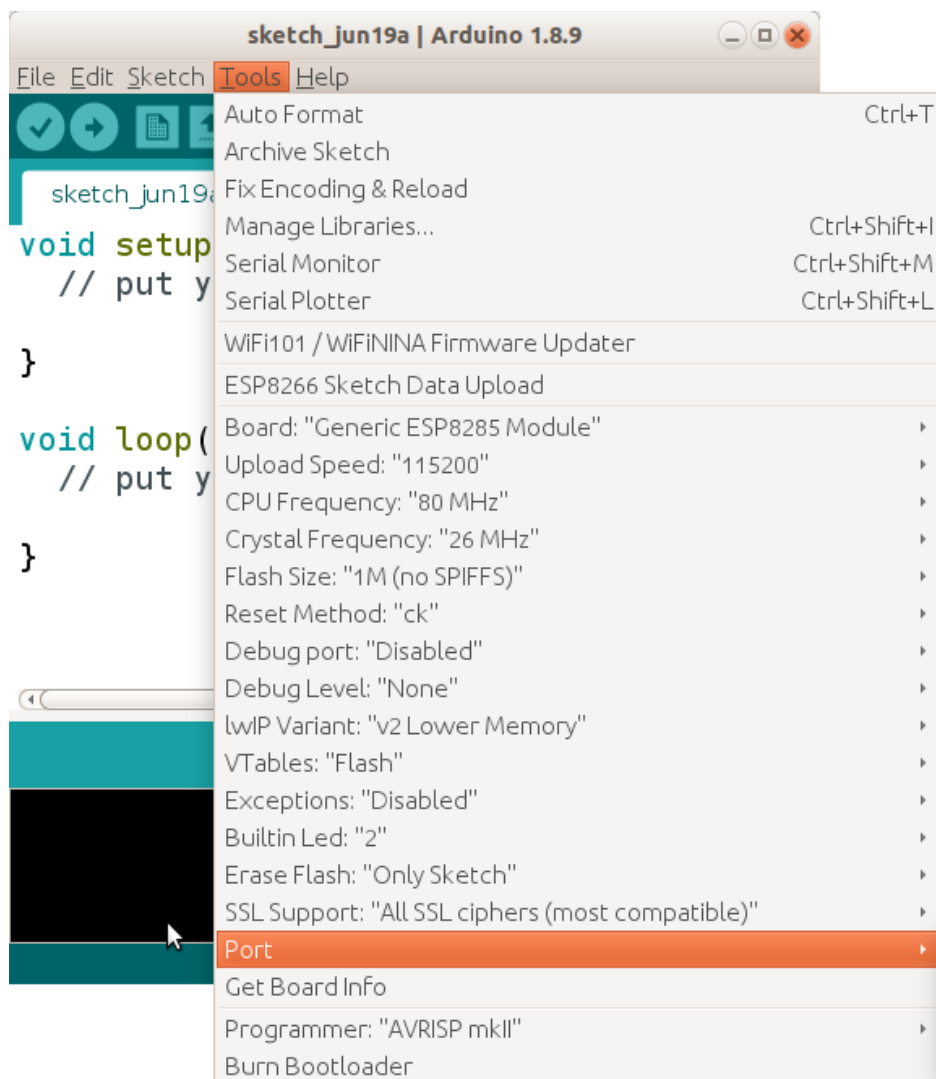
# Az-Delivery

Jetzt müssen wir festlegen, welches Board programmiert werden soll. Gehe Sie zu **Tools > Board > {board name}** und wählen "Generic ESP8266 Module" aus.



# Az-Delivery

Wenn Sie sich für dieses Board entscheiden, können Sie auch einige andere Optionen einstellen. Wenn Sie das Dropdown-Menü "Tools" öffnen, nachdem Sie das "Generic ESP8266 Modul" gewählt haben, gibt es mehrere Optionen, die Sie einstellen können. Für die Zwecke dieses eBooks ändern Sie diese Einstellungen nicht, weil wir weitere Einstellungen dahingehend nicht näher behandeln werden. Wählen Sie einfach den Port aus, an dem Ihr Board angeschlossen ist.





# Az-Delivery

## Anwendungsbeispiel

In diesem Beispiel zeigen wir Ihnen, wie Sie den ESP8266-01S mit dem DHT22 Temperatur- und Feuchtigkeitssensor verbinden.

Wir haben bereits den "DHT22" bereits näher behandelt:

[https://www.az-delivery.de/products/dht-22-modul-kostenfreies-e-book?\\_pos=3&\\_sid=8bfd4bac3&\\_ss=r&ls=en](https://www.az-delivery.de/products/dht-22-modul-kostenfreies-e-book?_pos=3&_sid=8bfd4bac3&_ss=r&ls=en).

Deshalb werden wir bei diesem Sensor nicht ins Detail gehen. Alles, was Sie tun müssen, ist die "SimpleDHT"-Library herunterzuladen und sie zur Arduino-IDE hinzuzufügen, falls Sie vorher DHT22 noch nicht verwendet haben. Um diese Library herunterzuladen, gehen Sie auf den Link:

<https://github.com/winlinvip/SimpleDHT> .

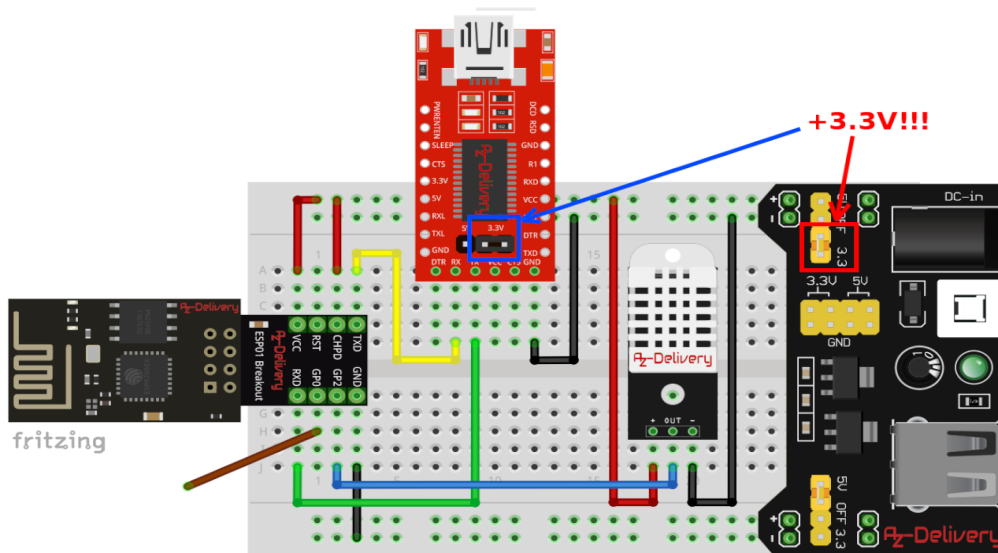
Wenn Sie die .ZIP-Datei herunterladen, öffnen Sie den Arduino Uno und gehen auf: *Sketch > Include Library > Fügen Sie die .ZIP Library* und die heruntergeladene .ZIP-Datei hinzu. Danach gehen Sie zu: *File > Examples > SimpleDHT > DHT22Default* und öffnen die Skizze. Das sollte so aussehen:

Der DHT22-Sensor kann sowohl mit +3,3V als auch mit +5V betrieben werden, daher schließen wir ihn an +3,3V an, weil der ESP mit +3,3V arbeitet, aber beschädigt wird, wenn der DHT22 an +5V angeschlossen wird. Wenn der DHT22 an +5V angeschlossen wird, gibt er ein +5V-Signal am OUT-Pin aus.

**Stellen Sie also sicher, dass der DHT22-Sensor an die +3,3V-Stromversorgung angeschlossen wird.**

# Az-Delivery

Verbinden Sie alles, wie unten abgebildet:



Alles wird genauso wie beim letzten Anschlussplan (s.o.) angeschlossen, außer bei dem Teil mit dem DHT22.

**DHT22 pin** > **ESP pin and power supply pins**

"+" pin > +3.3V [Stromversorgung] **Roter Draht**

OUT pin > GPIO pin 2 [ESP] **Blauer Draht**

"-" pin > GND [Stromversorgung] **Schwarzer Draht**

Wenn Sie den Sketch "*DHT22Default*" öffnen, wählen Sie in *Tools* > *Board* > {*board name*}, zunächst "*Generic ESP8266 Module*" aus. Wenn Sie dann

# Az-Delivery

mit dem Hochladen des Sketches beginnen, wird diese für den ESP8266 kompiliert.

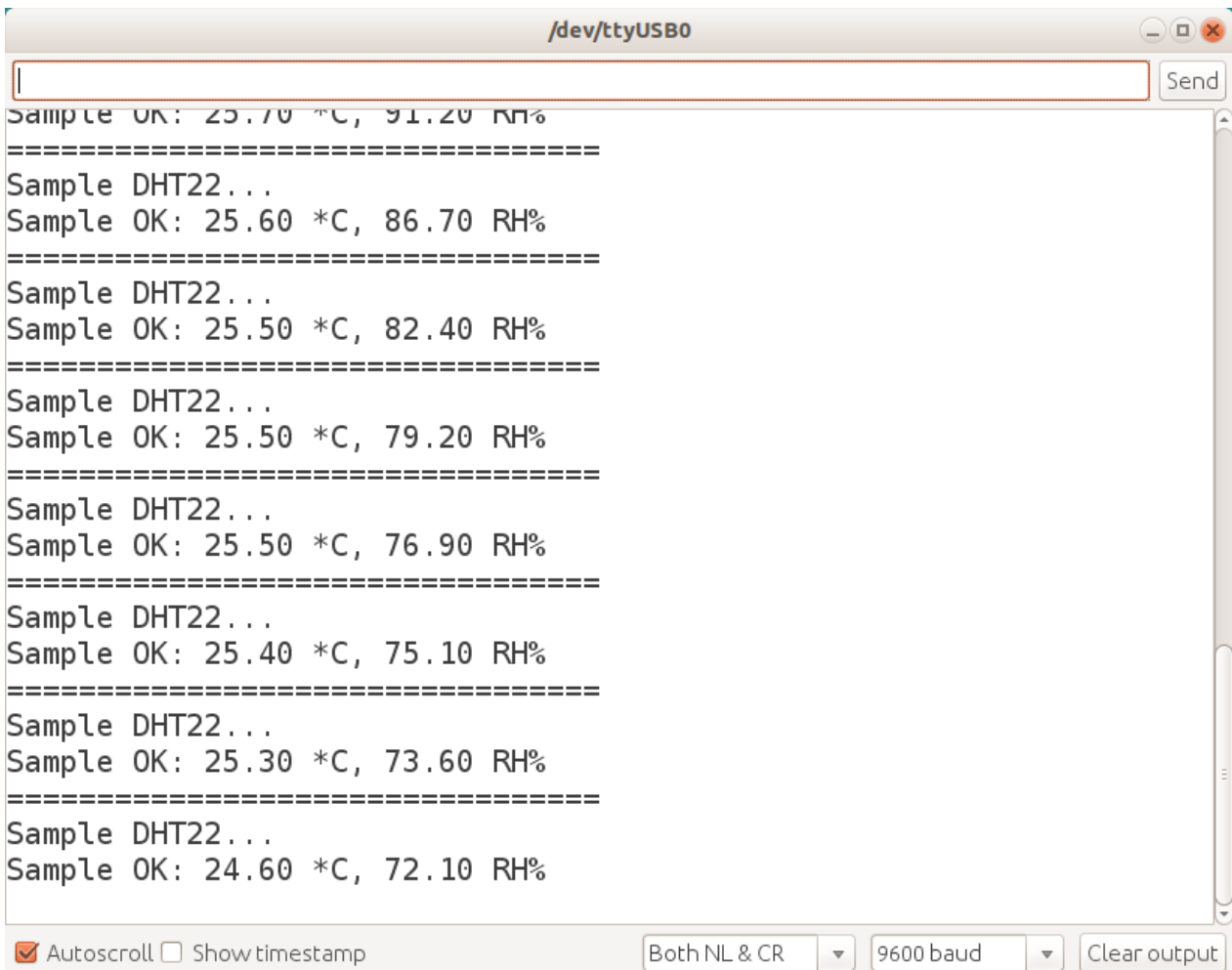
Wir haben ein paar Dinge zur besseren Lesbarkeit geändert, aber der Sketch ist der gleiche. Hier ist unser Sketchcode:

```
#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);
}
```

# Az-Delivery

}

Wenn Sie den Serial Monitor starten (*Tools > Serial Monitor*) sollte die Ausgabe wie folgt aussehen (Arduino oder Raspberry Pi gleich:



The screenshot shows a Serial Monitor window titled "/dev/ttyUSB0". The window contains a text area with the following output:

```
Sample OK: 25.70 *C, 91.20 RH%
=====
Sample DHT22...
Sample OK: 25.60 *C, 86.70 RH%
=====
Sample DHT22...
Sample OK: 25.50 *C, 82.40 RH%
=====
Sample DHT22...
Sample OK: 25.50 *C, 79.20 RH%
=====
Sample DHT22...
Sample OK: 25.50 *C, 76.90 RH%
=====
Sample DHT22...
Sample OK: 25.40 *C, 75.10 RH%
=====
Sample DHT22...
Sample OK: 25.30 *C, 73.60 RH%
=====
Sample DHT22...
Sample OK: 24.60 *C, 72.10 RH%
```

At the bottom of the window, there are several controls: a checked "Autoscroll" checkbox, an unchecked "Show timestamp" checkbox, a dropdown menu set to "Both NL & CR", a dropdown menu set to "9600 baud", and a "Clear output" button.

Wir haben in dem Sketch ein Baudrate von 9600 verwendet, stellen Sie also sicher, dass bei der seriellen Ausgabe die Baudrate von 9600 eingestellt ist.

# Az-Delivery

**Sie haben es geschafft. Sie können jetzt unser Modul  
nun für Ihre Projekte nutzen.**

# AZ-Delivery

Jetzt sind Sie dran! Entwickeln Sie Ihre eigenen Projekte und Smart-Home Installationen. Wie Sie das bewerkstelligen können, zeigen wir Ihnen unkompliziert und verständlich auf unserem Blog. Dort bieten wir Ihnen Beispielskripte und Tutorials mit interessanten kleinen Projekten an, um schnell in die Welt der Mikroelektronik einzusteigen. Zusätzlich bietet Ihnen auch das Internet unzählige Möglichkeiten, um sich in Sachen Mikroelektronik weiterzubilden.

**Falls Sie nach noch weiteren hochwertigen Produkten für Arduino und Raspberry Pi suchen, sind Sie bei AZ-Delivery Vertriebs GmbH goldrichtig. Wir bieten Ihnen zahlreiche Anwendungsbeispiele, ausführliche Installationsanleitungen, E-Books, Bibliotheken und natürlich die Unterstützung unserer technischen Experten.**

<https://az-delivery.de>

Viel Spaß!

Impressum

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