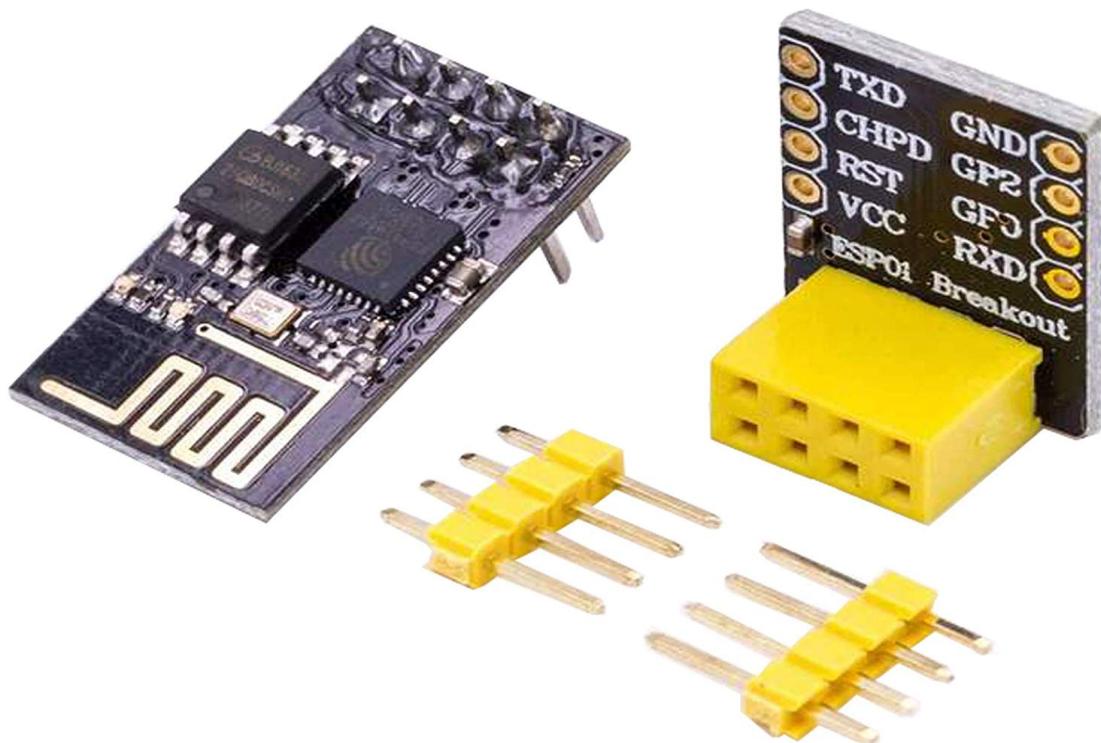


# AZ-Delivery

## Benvenuto!

Grazie per aver acquistato il nostro modulo con Adattatore per Breadboard AZ-Delivery ESP8266-01S. Nelle pagine seguenti, ti illustreremo come utilizzare e configurare questo pratico dispositivo.

**Buon divertimento!**

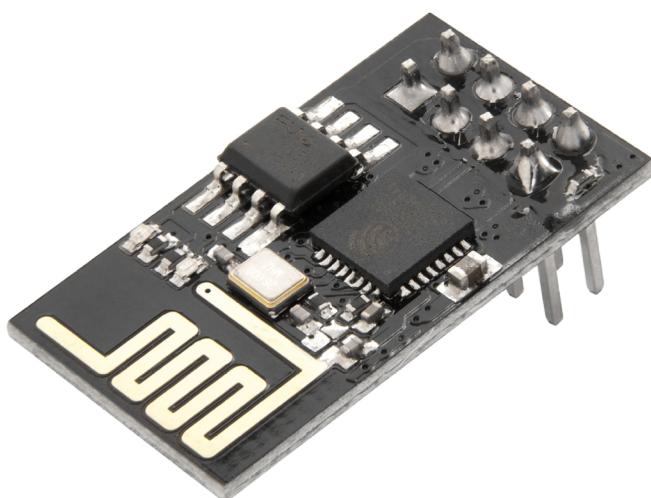


**AZ-Delivery**  
Ihr Experte für Mikroelektronik!

# Az-Delivery

Il modulo ESP8266 è un System on a Chip (SoC), prodotto dalla società cinese Espressif. Consiste in un microcontrollore Tensilica L106 a 32 bit e un ricetrasmettitore wifi. Ha 11 pin GPIO (General Purpose Input/Output) e anche un ingresso analogico. Ciò significa che puoi programmarlo come un qualsiasi normale Arduino o qualsiasi altro microcontrollore. E la cosa migliore dell'ESP8266 è che con esso ottieni una comunicazione wifi, quindi puoi usarlo per connetterti alla tua rete wifi, a Internet, ospitare un server web con pagine Web reali, lasciare che il tuo smartphone si colleghi ad esso, ecc.

L'ESP8266 è un fantastico chip WiFi che può essere utilizzato in diverse applicazioni di home automation. Grazie al potente processore da 80 MHz e all'ampia memoria da 1 MB, l'ESP8266 può funzionare anche in modo autonomo.



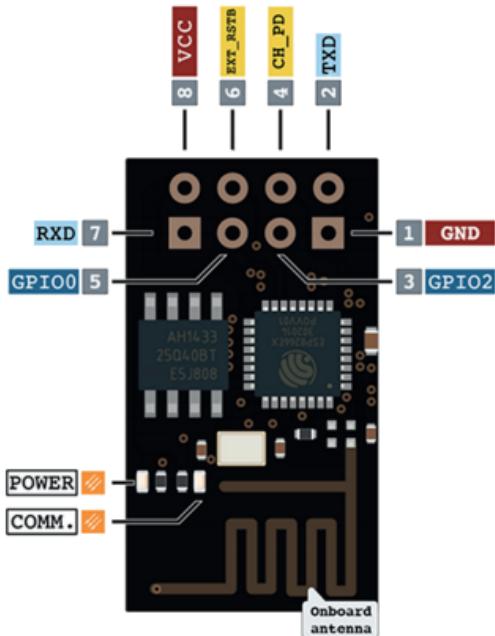


## Caratteristiche

- » 802.11 b/g/n
- » MCU a 32-bit a bassa potenza integrato
- » 10-bit ADC Integrato
- » Stack di protocollo TCP/IP integrato
- » Switch TR integrato, balun, LNA, amplificatore di potenza e rete corrispondente
- » PLL, regolatori e unità di gestione dell'alimentazione integrati
- » Supporta la diversità dell'antenna
- » Wi-Fi 2.4 GHz, supporta WPA/WPA2
- » Supporta modalità di funzionamento STA/AP/STA+AP
- » Supporta Funzione Smart Link sia per dispositivi Android che iOS
- » SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO
- » STBC, 1x1 MIMO, 2x1 MIMO
- » Aggregazione A-MPDU & A-MSDU e guard interval da 0.4s
- » Potenza sonno profondo <10uA, corrente dispersione power down < 5uA
- » Si avvia e trasmette pacchetti in < 2ms
- » Consumo in standby < 1.0mW (DTIM3)
- » +20dBm di potenza di uscita in modalità 802.11b
- » Intervallo operativo di temperatura: -40 °C ~ 125 °C

# Az-Delivery

Esistono molte schede di sviluppo diverse basate sull'ESP8266, quindi l'ESP8266-01S è una di queste. Ha 8 pin (designazione dei pin nell'immagine qui sotto).



Come puoi vedere, abbiamo due pin GPIO liberi, GPIO0 e GPIO2. Possiamo quindi usare GPIO0 e GPIO2 come ingressi o uscite digitali come in un normale microcontrollore. Nell'esempio (più avanti nell'eBook) useremo GPIO2 per leggere i dati dal modulo sensore di temperatura DHT22.

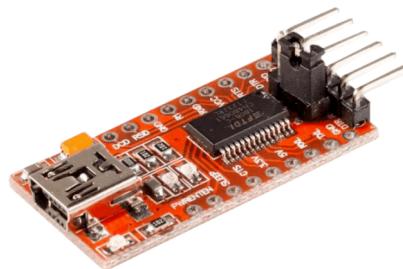
Possiamo programmare l'ESP8266-01S in molti metodi diversi, ma tratteremo qui solo il metodo che utilizza l'Arduino IDE. Se hai già usato le schede Arduino, allora per te ciò sarà davvero facile. Tieni presente che la programmazione non è limitata a questa opzione, ci sono molti altri modi per programmare l'ESP8266 (SDK ESP ufficiale per la programmazione C, interprete Lua, firmware MicroPython, sono solo alcuni dei tanti).

# Az-Delivery

Per programmare l'ESP8266-01S dobbiamo usare i pin RX e TX e collegarli al dispositivo con interfaccia seriale come il modulo FTDI o l'adattatore USB dedicato, o la scheda Arduino.

L'FTDI è assai popolare, perché può commutare tra 5V e 3.3V logici. **È essenziale che il convertitore da USB a Seriale funzioni a 3.3V!**

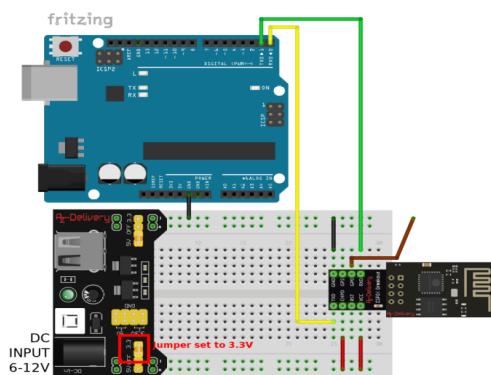
**Se fai funzionare l'ESP8266 con un modello 5V FTDI, danneggerai l'ESP8266!**



Adattatore FTDI

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

Se stai usando l'Arduino Uno per programmare l'ESP8266-01S, questo è lo schema di connessione (non lo useremo in questo eBook):



Abbiamo utilizzato questa connessione nel nostro eBook sull'ESP8266-01S con relè:

[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)

# Az-Delivery

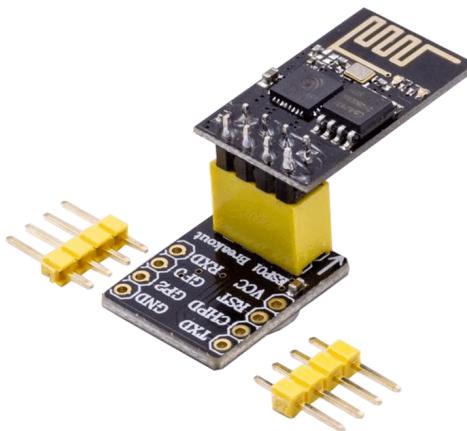
C'è un altro modo per programmare l'ESP8266-01S, ed è con un adattatore USB dedicato (immagine sotto), ma non ne parleremo nemmeno in questo eBook.



Adattatore USB con ESP8266-01S

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

Per programmare l'ESP8266-01S useremo la breakout board della breadboard e l'adattatore FTDI, perché è molto semplice, ma qualsiasi modalità tu scelga è simile.

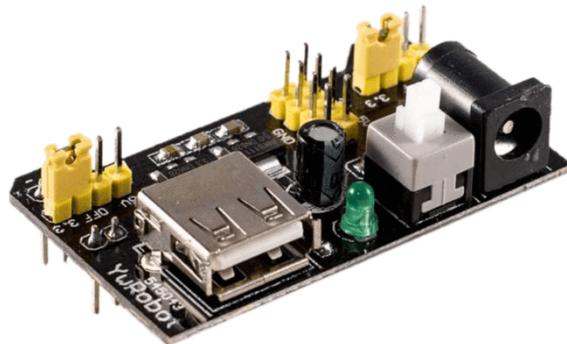


Breakout board breadboard con ESP8266-01S

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

# Az-Delivery

Assicurati di alimentare l'ESP8266-01S con un alimentatore separato. Puoi utilizzare il nostro alimentatore per breadboard MB102. È un alimentatore molto semplice e maneggevole che accetta input da 6 a 12V DC e può produrre sia + 3.3V che + 5V.



Alimentatore per breadboard MB102

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

Esistono due modi molto diversi di programmare ESP8266-01S. Il primo consiste nell'utilizzare i comandi "AT" (ATtention commands) tramite l'interfaccia seriale, e il secondo è in realtà la programmazione del chip ESP8266 come un microcontrollore (utilizzando l'IDE Arduino).

Poiché il secondo modo offre molta più libertà in ciò che è possibile creare, tratteremo solo questo modo di programmare ESP8266-01S in questo eBook.



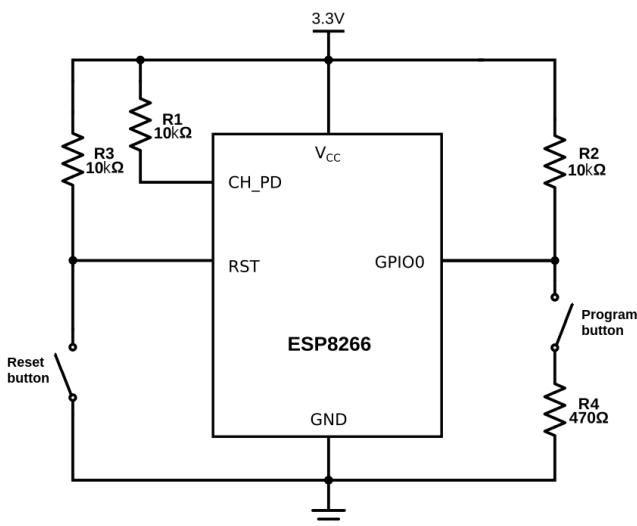
## Abilitare il chip

Dobbiamo aggiungere alcuni resistori per accendere l'ESP8266-01S e selezionare la modalità di avvio corretta (programmazione o modalità normale). Per garantire il corretto utilizzo del modulo ESP8266-01S, dobbiamo prima:

1. Abilitare il chip collegando il pin **CH\_PD** (Chip Power Down, a volte chiamato *CH\_EN*, or *CH\_PC*) AL **VCC** tramite una resistenza da **10kΩ**.
2. Selezionare la modalità *NORMAL* collegando **GPIO0** a **VCC** tramite una resistenza da **10KΩ**. (la colleghiamo a **GND** in modalità *PROGRAM*, e per la modalità *NORMAL* la lasciamo scollegata)
3. Preveniamo i rest casuali collegando il pin **RST** (reset) al **VCC** tramite una resistenza da **10KΩ**. (puoi aggiungerlo, ma non è necessario, lo lasciamo scollegato)

## Aggiunta dei pulsanti RESET e PROGRAM

Le nostre schede ESP8266-01S non hanno un pulsante di reset o un pulsante per la modalità di programmazione, ma è possibile (ma non è necessario) aggiungerli, come nell'immagine seguente.



Per il pulsante di reset, collegare un pulsante tra il pin RST e GND, ma aggiungere un resistore PULL-UP 10kΩ (R3 nell'immagine sopra) tra RST e VCC.

Per mettere il chip in modalità di programmazione, è necessario tirare il pin GPIO0 *LOW* durante l'avvio (collegarlo a GND). Se è *HIGH* o scollegato, il chip si avvierà in modalità normale. Poiché è possibile utilizzare GPIO0 come output, non è possibile cortocircuitarlo direttamente a terra, il che potrebbe danneggiare il chip. Per evitare ciò, collega la resistenza da 470Ω (R4 sull'immagine sopra) tra questo pulsante e il GND. È importante che questa resistenza sia abbastanza bassa, altrimenti verrà tirata in alto dal resistore da 10KΩ (R2 nell'immagine sopra) se si sceglie di aggiungere R2 come nel suggerimento nella pagina precedente (in 2.).



Ai fini di questo eBook, non stiamo utilizzando alcun pulsante. Per mettere ESP in modalità di programmazione, utilizziamo un cavo jumper, che collega *GPIO0* a *GND* per la modalità di programmazione e lo scollegiamo da *GND* in modalità normale

Per resettare l'ESP8266-01S, speghiamo e accendiamo l'alimentazione.

Ci sono alcune cose a cui devi fare attenzione quando usi un ESP8266-01S: La cosa più importante è che funziona a 3.3V, quindi se lo colleghi a un alimentatore a 5V, lo romperai. A differenza di alcune schede Arduino da 3,3 V, i pin I/O dell'ESP8266 non tollerano i 5 V, quindi se si utilizza un convertitore da USB a seriale da 5 V o sensori da 5 V ecc..

**Una seconda cosa da tenere a mente è che ESP8266 può solo generare o scendere di 12 mA per pin di uscita, rispetto a 20 - 40 mA per la maggior parte degli Arduino!**

Un'ultima cosa da tenere a mente è che l'ESP8266 deve condividere le risorse di sistema e il tempo della CPU tra lo sketch e il driver Wi-Fi. Inoltre, funzioni come PWM, interrupt o I<sup>2</sup>C sono emulate nel software, la maggior parte degli Arduino, d'altra parte, hanno parti hardware dedicate per queste attività. Per la maggior parte delle applicazioni, tuttavia, questo non è un grosso problema, ma devi pensarci quando progetti uno sketch.



## L'ESP8266 come microcontrollore - Hardware

Mentre l'ESP8266 è spesso usata come un ponte "stupido" da seriale a wifi, è un microcontrollore da solo molto potente .

### I/O Digitale

Proprio come un normale Arduino, l'ESP8266 ha pin di ingresso/uscita digitali o GPIO - Pin di ingresso/uscita per uso generico. Come suggerisce il nome, possono essere utilizzati come ingressi digitali per leggere una tensione digitale o come uscite digitali per emettere 0V (corrente di dispersione) o 3.3V (corrente di alimentazione).

L'ESP8266 è un microcontrollore da 3.3 V, quindi anche il suo I/O funziona a 3.3 V.

- I pin non tollerano 5 V, applicando più di 3.6V **SU QUALSIASI PIN romperà il chip!**
- La corrente massima che può essere prelevata da un singolo pin GPIO è 12mA!

### Pin utilizzabili

L'ESP8266 ha 17 pin GPIO (0-16), tuttavia è possibile utilizzarne solo 2, poiché per l'ESP8266-01S solo due sono disponibili sui pin di breakout. Se si tenta di utilizzare altri pin, è possibile che il programma si blocchi.

I pin *GPIO1* e *GPIO3* sono usati come *TX* e *RX* della Porta seriale dell'hardware (*UART*). Puoi usarli come GPIO, ma nella maggior parte dei casi non dovresti usarli come GPIO mentre si inviano/ricevono dati seriali.

### Modalità di avvio

# Az-Delivery

Come accennato nel capitolo precedente, alcuni pin GPIO hanno una funzione speciale durante l'avvio, selezionano 1 di 3 modalità di avvio:

GPIO0	GPIO2	Mode
0V	3.3V	Uart Bootloader (Modalità PROGRAM)
3.3V	3.3V	Sketch di Avvio (SPI flash)
x	x	Modalità SDIO (non usato per Arduino)

**Nota:** Non è necessario aggiungere un resistore pull-up esterno a GPIO2, quello interno è abilitato all'avvio.

Ci siamo assicurati che queste condizioni fossero soddisfatte aggiungendo dei resistori esterni nel capitolo precedente. Ciò ha alcune implicazioni, tuttavia:

- » GPIO0 è tirato su HIGH durante la normale modalità operativa, quindi non è possibile utilizzarlo come input Hi-Z.
- » GPIO2 non deve essere LOW all'avvio, quindi non è possibile collegare un'interruttore ad esso.

## Resistenze pull-up/down interne

Tutti i pin GPIO dell'ESP8266 hanno una resistenza pull-up integrata, proprio come in un Arduino. Tranne il GPIO16 ha un resistore pull-down incorporato (ma non è possibile accedervi nell'ESP8266-01S, quindi non importa).

# Az-Delivery

## PWM

A differenza della maggior parte dei chip Atmel (Arduino), l'ESP8266 non supporta l'hardware PWM, tuttavia, il software PWM è supportato su tutti i pin digitali. L'intervallo PWM predefinito è 10 bit a 1 kHz, ma può essere modificato (fino a 14 bit a 1 kHz).

## Interfaccia seriale

l'ESP8266 ha due hardware *UARTS* (porte seriali), ma è possibile utilizzarne solo uno sul modulo ESP8266-01S, *UART0* sui pin 1 e 3 (*TX0* e *RX0* rispettivamente)



## L'ESP8266 come microcontrollore - Software

La maggior parte delle funzionalità del microcontrollore dell'ESP utilizza esattamente la stessa sintassi di un normale Arduino, rendendo davvero facile iniziare.

### I/O Digitale

Proprio come con un normale Arduino, puoi impostare la funzione di un pin usando `pinMode(pin, mode)` dove "pin" è il numero *GPIO* e "mode" può essere *INPUT* (che è l'impostazione predefinita), oppure *OUTPUT*, oppure *INPUT\_PULLUP* per abilitare le resistenze pull-up integrate per *GPIO 0-15*. Per abilitare il resistore pull-down per *GPIO16*, devi usare *INPUT\_PULLDOWN\_16* (per l'ESP8266-01S, non puoi accedere a *GPIO16* quindi questo vale solo per l'ESP8266 generale). Per impostare un pin di uscita *HIGH* (3.3 V) o *LOW* (0 V), utilizza `digitalWrite(pin, valore)` dove "pin" è il pin digitale e "valore" 1 o 0 (oppure *HIGH* e *LOW*). Per leggere un input, utilizza `digitalRead(pin)`. Per abilitare PWM su un determinato pin, utilizza `analogWrite(pin, value)` dove "pin" è il pin digitale e "value" un numero compreso tra 0 e 1023. È possibile modificare l'intervallo dell'uscita PWM utilizzando `analogWriteRange(new_range)`. La frequenza può essere modificata utilizzando `analogWriteFreq(new_frequency)`, "new\_frequency" dovrebbe essere compreso tra 100Hz e 1000Hz

### Comunicazione seriale

Per usare *UART0* (*TX* = *GPIO1*, *RX* = *GPIO3*), puoi usare l'oggetto Seriale, proprio come in un Arduino: `Serial.begin(baud_rate)`. Sono supportate anche tutte le funzioni di Arduino Stream, come `read`, `write`, `print`, `println`, ....



## Condivisione del tempo della CPU con la parte RF

Una cosa da tenere a mente durante la scrittura di programmi per ESP8266 è che lo sketch deve condividere le risorse (tempo e memoria della CPU) con gli stack wifi e TCP (il software che funziona in background e gestisce tutte le connessioni wifi e IP). Se l'esecuzione del codice impiega troppo tempo e non si lascia che gli stack TCP facciano le loro cose, potrebbe bloccarsi o perdere i dati. È meglio mantenere il tempo di esecuzione del tuo ciclo sotto un paio di centinaia di millisecondi. Ogni volta che il ciclo principale viene ripetuto, il tuo sketch cede al wifi e al TCP per gestire tutte le richieste wifi e TCP. Se il tuo ciclo impiega più tempo di questo, dovrà assegnare esplicitamente il tempo della CPU agli stack wifi/TCP, usando incluso *delay (0)* o *yield ()*. In caso contrario, la comunicazione di rete non funzionerà come previsto e se dura più di 3 secondi, il soft WDT (Watchdog Timer) ripristinerà l'ESP. Se il soft WDT è disabilitato, dopo poco più di 8 secondi, l'hardware WDT ripristinerà il chip. Dal punto di vista di un microcontrollore, tuttavia, 3 secondi è un tempo molto lungo (240 milioni di cicli di clock), quindi a meno che tu non esegua una serie di numeri estremamente pesante o invii stringhe estremamente lunghe sul seriale, non ne sarai influenzato. Tieni presente che hai aggiunto lo *yield()* all'interno dei tuoi cicli, il che potrebbero richiedere più tempo, diciamo 100ms.

# Az-Delivery

## Wifi

L'uso dell'ESP8266 come un semplice microcontrollore è fantastico, ma il motivo per cui la maggior parte delle persone lo usa è la sua capacità wifi. Supporta protocolli di rete come wifi, TCP, UDP, HTTP, DNS, ecc.

## Caricamento degli sketch nell'ESP8266

La procedura di upload per le schede ESP8266 è leggermente diversa dalla normale procedura di Arduino. La maggior parte delle schede Arduino si reimposterà automaticamente quando viene caricato un nuovo programma e entrerà automaticamente in modalità di programmazione. Sulle schede ESP8266-01S è necessario accedere manualmente alla modalità di programmazione e anche ripristinarli manualmente dopo aver programmato il modulo.



## RESET Manuale e PROGRAM Manuale

Se non si dispone di un convertitore da USB a seriale con linee DTR e RTS, è possibile utilizzare anche i pulsanti *RESET* e *PROGRAM* di cui abbiamo parlato in uno dei capitoli precedenti.

Per mettere l'ESP8266 in modalità *PROGRAM*, GPIO0 deve essere LOW durante l'avvio. Ai fini di questa guida, utilizziamo i cavetti jumper anziché i pulsanti, quindi questo è il processo con cui passare dalla modalità normale a quella programma:

» per entrare in modalità *PROGRAM*:

1. Spegnere il modulo (scollegare l'alimentazione)
2. Collegare il pin *GPIO0* a terra (GND) tramite cavetto jumper.
3. Accendere il modulo (collegare l'alimentazione), e l'ESP8266-01S è in modalità *PROGRAM*.

» per entrare in modalità *NORMAL*:

1. Spegnere il modulo (scollegare l'alimentazione)
2. Lasciare scollegato il pin *GPIO0*, o scollegarlo da GND se è stato precedentemente collegato. (assicurati solo che quel cavetto jumper non sia collegato a nulla!)
3. Accendere il modulo (collegare l'alimentazione), e l'ESP8266-01S è in modalità *NORMAL*.

# Az-Delivery

Puoi ovviamente aggiungere i pulsanti *RESET* e *PROGRAM*, semplificando così il processo. Se scegli questo percorso, allora i processi per entrare nelle modalità *PROGRAM* o *NORMAL* sono:

» per entrare in modalità *PROGRAM*:

1. premi e tieni premuto il pulsante *RESET*

2. premi e tieni premuto il pulsante *PROGRAM*

3. rilascia il pulsante *RESET*, l'ESP si avvierà in modalità programmazione

4. rilascia il pulsante *PROGRAM*

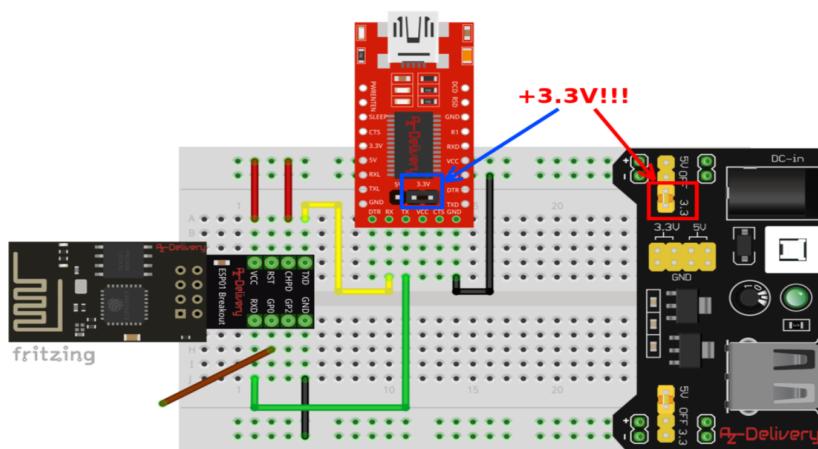
5. carica lo sketch

» Per entrare in modalità *NORMAL*:

1. premi e rilascia il pulsante *RESET* per resettare l'ESP8266-01S. Il pulsante *PROGRAM* deve rimanere non premuto durante questo processo.

## Schema di collegamento ESP8266-01S

Per seguire tutti i suggerimenti dei capitoli precedenti, collega l'ESP8266-01S come nello schema di collegamento seguente:



Per tutti e tre i moduli collegare tutti i pin GND insieme! (**Fili neri**)

Per FTDI e ESP8266-01S:

1. Assicurati di impostare il ponticello di riferimento della tensione su **3.3 V** (come per FTDI questo vale anche per MB102)
2. Collega il pin **GND** dell'FTDI al pin **GND** dell'ESP8266-01S.
3. Collega il pin **RX** dell'FTDI al pin **TX** dell'ESP8266-01S. **Filo giallo** dello schema di collegamento qui sopra.
4. Collega il pin **TX** dell'FTDI al pin **RX** dell'ESP8266-01S. \*\*  
**Filo verde** dello schema di collegamento qui sopra .

# Az-Delivery

Il **VCC** dell'ESP8266-01S è collegato al +3.3V (**Filo rosso**).

Il pin **CHPD** dell'ESP8266-01S deve essere collegato al VCC (**Filo rosso**).

Per la modalità normale, il *GPIO0* deve essere sollegato all'avvio (**Filo marrone**), ma quando vogliamo programmare l'ESP8266-01S, dobbiamo prima spegnere il modulo (scollegandolo dall'alimentazione), collegare il pin *GPIO0* al GND (**Filo marrone**) e quindi per ricollegare ESP all'alimentazione.

Per tornare alla modalità normale, è necessario spegnere di nuovo il modulo, quindi scollegare il pin *GPIO0* (il **Filo marrone** non viene collegato come nello schema di collegamento) e riaccendere il modulo ESP, ricollegarlo all'alimentazione.

## Software

Ti mostreremo solo come utilizzare l'ESP8266-01S con Arduino IDE.

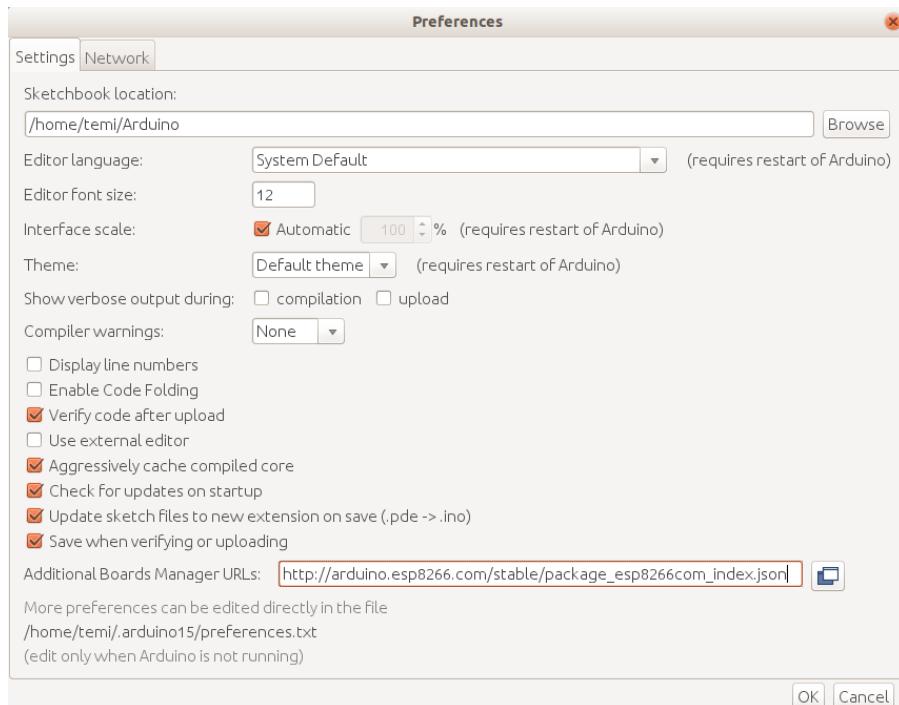
# Az-Delivery

Il primo passo è scaricare e installare l'IDE Arduino. Se lo stai già utilizzando, ottimo, in caso contrario, vai su <https://www.arduino.cc/en/Main/Software>, scaricala e installala.

Per programmare l'ESP8266-01S, è necessario un plug-in per l'IDE Arduino, che può essere scaricato da GitHub <https://github.com/esp8266/Arduino> manualmente, ma è più semplice aggiungere l'URL nell'IDE di Arduino. Apri l'IDE, vai su *File > Preferences*, copia l'URL:

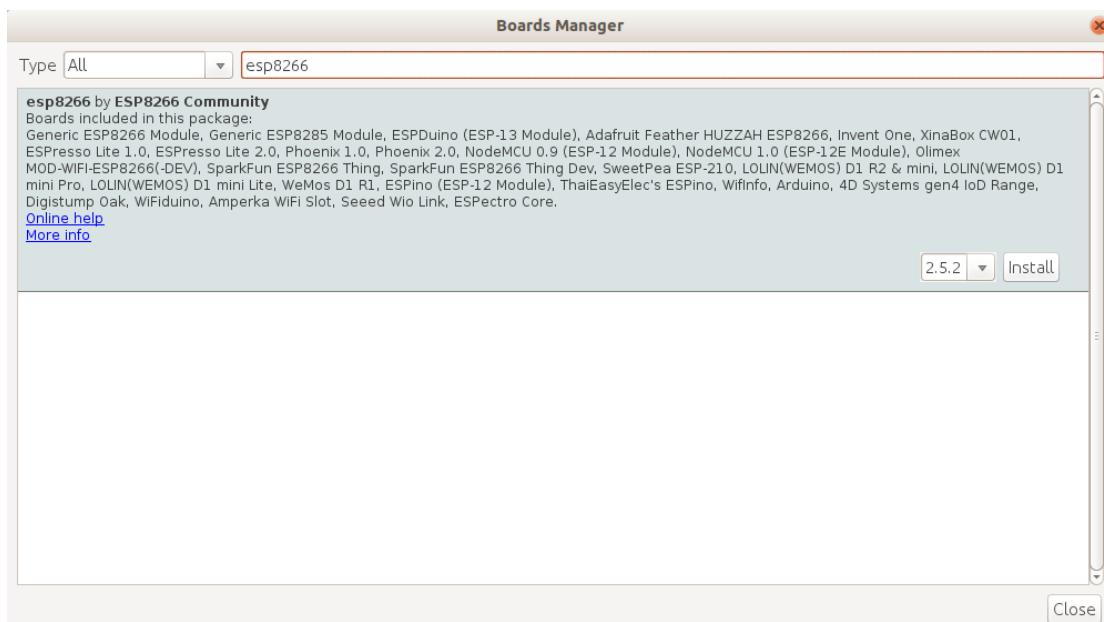
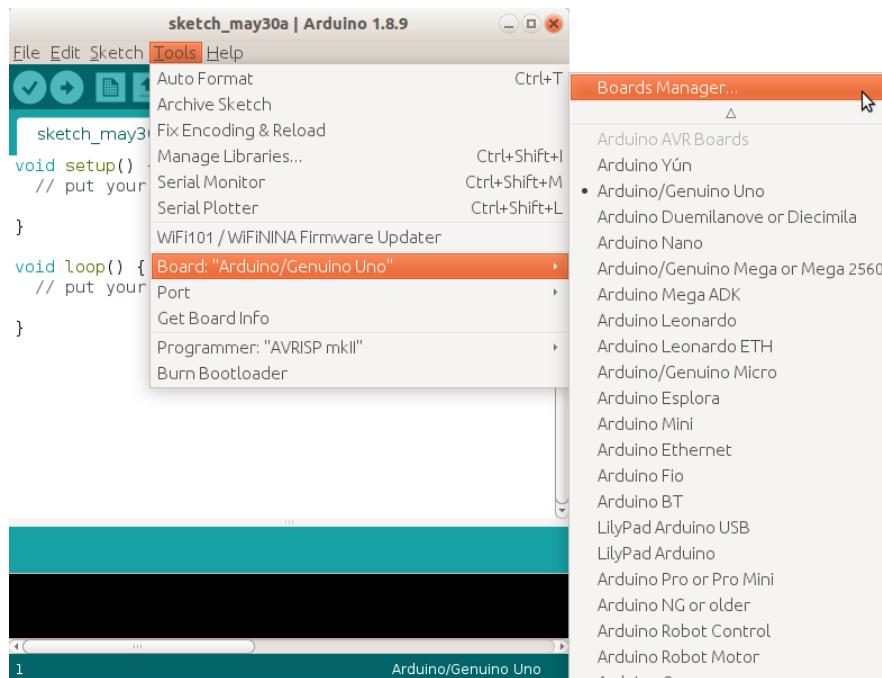
[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)

nel campo URL "*Additional Board Manager*" . (Puoi aggiungere più URL, separandoli con virgolette.)



# Az-Delivery

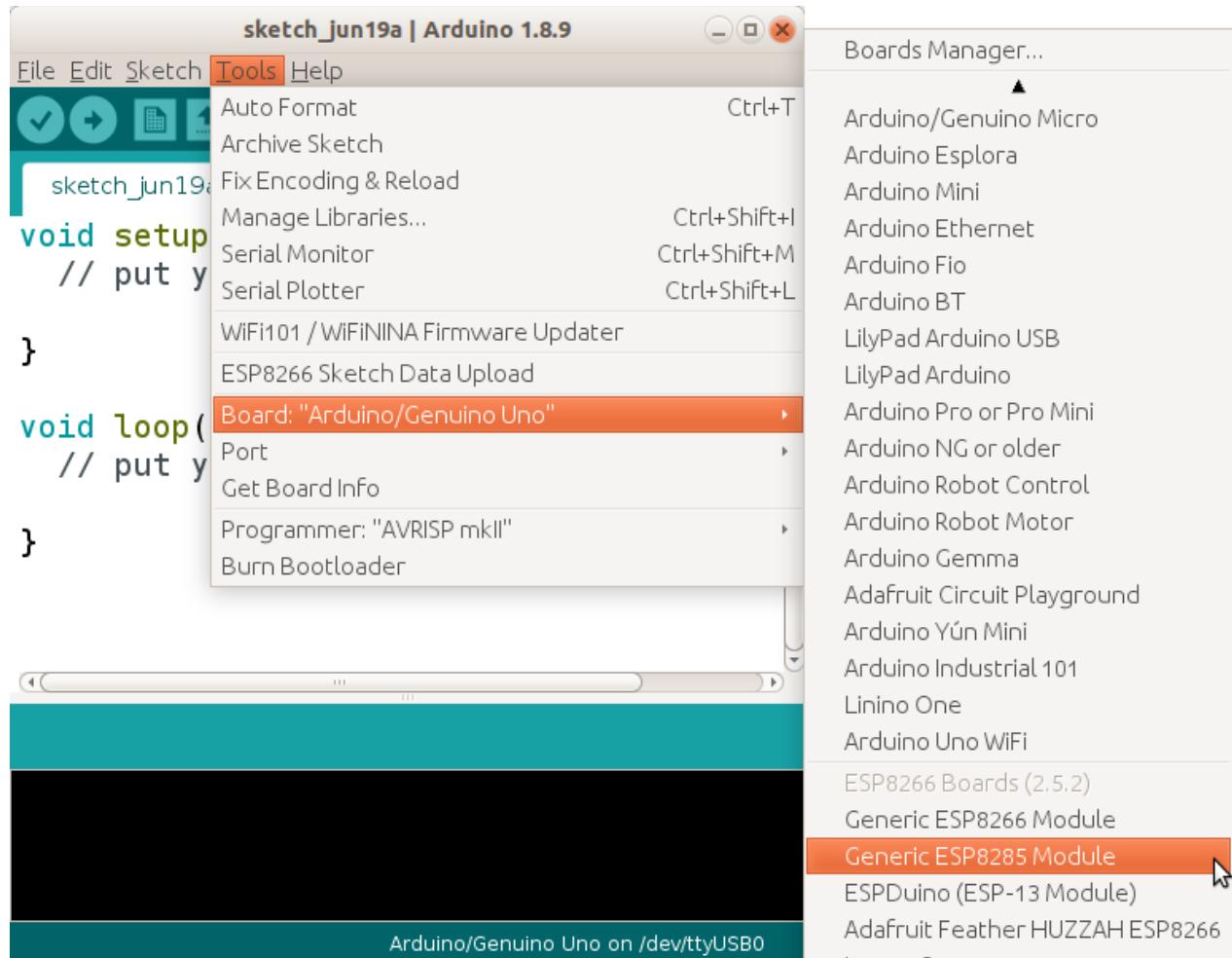
Vai poi su *Tools > Board > Board Manager* e cerca "esp8266". Seleziona la versione più recente e fai clic su Installa. Dopo questo molti esempi di sketch, verranno installate le schede.



# Az-Delivery

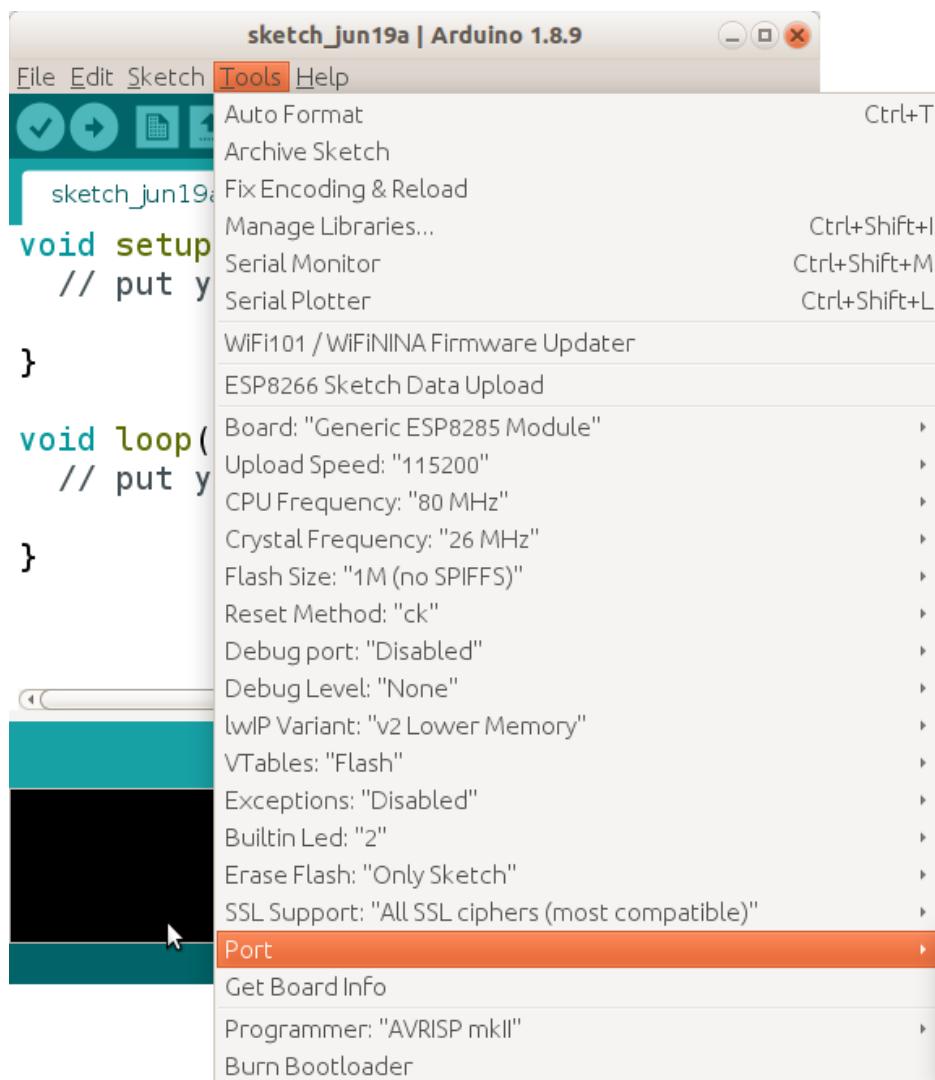
Ora dobbiamo impostare quale scheda programmare.

Vai su *Tools > Board > {board name}* e scegli "*Generic ESP8266 Module*".



# Az-Delivery

Quando scegli questa scheda ci sono anche altre opzioni che puoi impostare. Quando si aprono i menu a discesa "Tools" dopo che avrai scelto "Generic ESP8266 Module", ci saranno diverse opzioni che puoi impostare. Ai fini di questo eBook lascia tutto così com'è, perché non entreremo nei dettagli. Basta scegliere la porta su cui è connessa la tua schedad.





## Esempio di applicazione

In questo esempio ti mostreremo come collegare l'ESP8266-01S con il sensore di temperatura e umidità DHT22.

Abbiamo scritto sul DHT22 nel nostro eBook del Sensore DHT22

[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).

Non ci addentreremo quindi nei dettagli di questo sensore. Tutto ciò che devi fare è scaricare la libreria "*SimpleDHT*", e aggiungila all'IDE di Arduino se non hai usato DHT22 prima di ora. Per scaricare questa libreria vai al link: <https://github.com/winlinvip/SimpleDHT> .

Dopo aver scaricato il file ".zip", apri l'Arduino Uno e vai su:

*Sketch > Include Library > Add .ZIP Library*, e aggiungi il file zip scaricato.

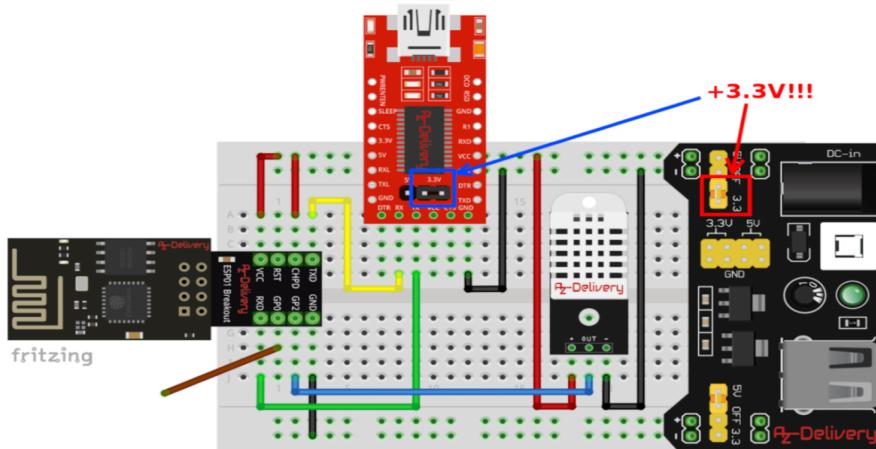
Dopo di questo vai su: *File > Examples > SimpleDHT > DHT22Default* e apri lo sketch. Useremo il codice di questo sketch nel nostro sketch.

Il sensore DHT22 può funzionare sia a + 3.3V che a + 5V, quindi collegiamo il DHT22 a + 3.3 V perché l'ESP funziona a + 3.3V e verrà distrutto se il DHT22 è collegato a +5V. Quando il DHT22 è collegato a + 5V emetterà un segnale + 5V sul pin OUT.

**Quindi assicurati di collegare il sensore DHT22 all'alimentazione + 3.3V!**

# Az-Delivery

Collega tutto come nello schema qui sotto:



Ogni connessione è la stessa dell'ultimo schema di connessione in questo eBook, ad eccezione della parte con DHT22.

## Pin DHT22 > Pin ESP e pin alimentazione

Pin "+"	> +3.3V	[alimentazione]	Filo rosso
Pin OUT	> GPIO pin 2 [ESP]		Filo blu
Pin "-"	> GND	[alimentazione]	Filo nero

Apri quindi questo sketch "DHT22Default", e poi scegli in *Tools > Board > {board name}*, primo "Generic ESP8266 Module". Quindi, quando si avvia il caricamento dello sketch, questo verrà compilato per l'ESP8266.

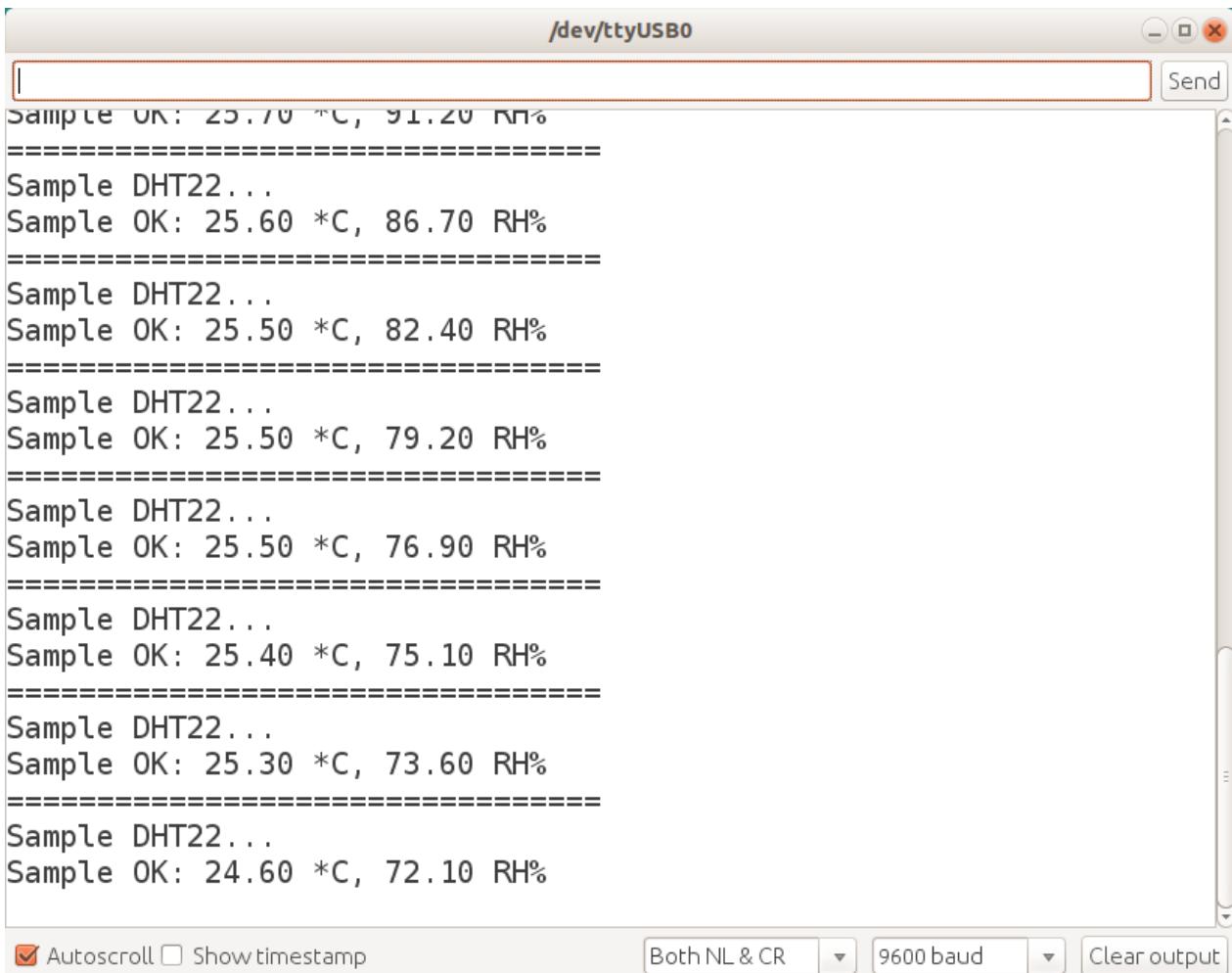
# Az-Delivery

Abbiamo cambiato appena poche cose, solo per una migliore leggibilità, ma lo sketch è lo stesso. Ecco il nostro codice di sketch:

```
#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

E quando avvii il Monitor Seriale (*Tools > Serial Monitor*) l'output dovrebbe essere così: (come nell'Arduino o Raspberry Pi)



The screenshot shows the Arduino Serial Monitor window titled '/dev/ttyUSB0'. The window displays a series of temperature and humidity readings from a DHT22 sensor. Each reading consists of two lines: 'Sample OK: [temperature] \*C, [humidity] RH%' followed by '======' at the end of each sample. The readings are as follows:

```
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 three checkboxes: 'Autoscroll' (checked), 'Show timestamp' (unchecked), and 'Clear output'. There are also dropdown menus for 'Both NL & CR' and '9600 baud'.

Abbiamo utilizzato un baud rate di 9600 nello sketch, quindi assicurati che in Uscita Seriale sia impostato il baud rate di 9600.

**Ce l'hai fatta, ora puoi usare il modulo per i tuoi progetti.**



E ora è tempo di imparare e di creare dei Progetti da solo. Lo puoi fare con l'aiuto di molti script di esempio e altri tutorial, che puoi trovare in internet.

**Se stai cercando dei prodotti di alta qualità per il tuo Arduino e Raspberry Pi, AZ-Delivery Vertriebs GmbH è l'azienda giusta dove potrai trovarli. Ti forniremo numerosi esempi di applicazioni, guide di installazione complete, e-book, librerie e l'assistenza dei nostri esperti tecnici.**

<https://az-delivery.de>

Buon divertimento!

Impressum

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