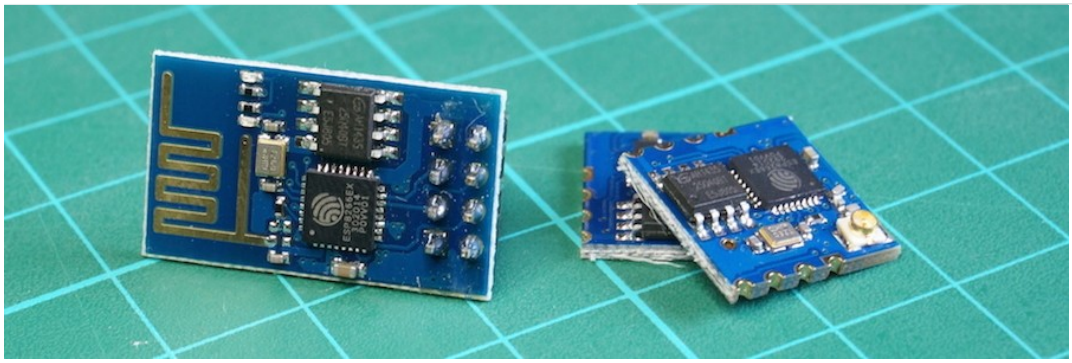


29 gennaio 2016
elettronica, guide
esp8266, esp-01, arduino, howto, iot
0 commenti

Programmare l'ESP8266: ovvero Arduino con il WiFi a meno di €2

È da un po' che gioco con questi modulini. Annoto di seguito le info che ho raccolto su di essi, necessarie per il loro utilizzo e la loro programmazione.

Cosa sono i moduli ESP8266?

Si tratta di modulini operanti a 3.3V che includono i SOC ESP8266 della cinese Espressif. Questi si distinguono per la connettività WiFi, le ridottissime dimensioni e il prezzo contenuto.

Il firmware di fabbrica permette di interfacciarsi tramite comandi AT su seriale, rendendolo un comodo **modem per Arduino**.

Inoltre la community ha sviluppato dei firmware che permettono di riprogrammarlo in LUA o con l'Arduino IDE. L'ESP8266 ha infatti abbastanza potenza di calcolo e memoria da poter eseguire sketch senza microcontrollori esterni. La compatibilità con l'Arduino IDE, inoltre, permette di sfruttare (almeno in parte) l'ampio bagaglio di librerie che ha reso popolari e accessibili i devices Arduino, che può essere usato per interfacciare questo chip a sensori e attuatori in poche righe di codice. In pratica **l'ESP8266 può fungere da Arduino**.

La famiglia di moduli ESP8266

I moduli più comuni

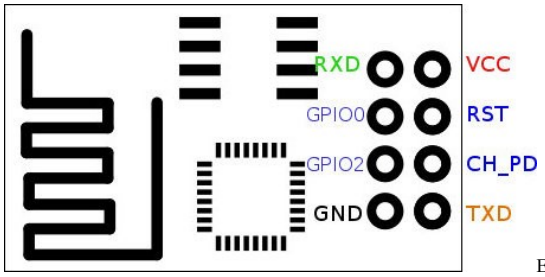
Vi sono un'infinità di moduli, dall'ESP-01, che mette a disposizione solo 4 pin digitali (comprese le porte seriali Tx e Rx), all'ESP-12f, con i suoi 11 pin digitali e 1 pin analogico.

A questi si aggiungono le board che espandono i sopra citati moduli con porte USB, regolatori di tensione, logic lever shifter a 5V e altre amenità del genere.

Personalmente ho optato per il più comune ed economico ESP-01, attualmente acquistabile anche a **meno di €1,70** su AliExpress. D'ora in poi **prenderò come riferimento l'ESP-01**, ma quanto scriverò può essere adattato a tutti i moduli basati su ESP8266.

I pin dell'ESP-01

Prima di procedere oltre è bene avere chiare la disposizione dei pin e le loro funzioni.



ESP-01 pinout

PIN	Funzioni
GPIO_0	D0
TXD	D1, TX
GPIO_2	D2,
RXD	D3, RX

Funzionalità dei pin dell'ESP-01

Nella tabella `D0`, `D1`, etc stanno per "Digital Pin 0", "Digital Pin 1", etc. Nell'Arduino IDE sono chiamati semplicemente `0`, `1`, etc. Il pin analogico `ADC`, non presente su ESP-01, è chiamato `A0` nell'Arduino IDE.

`VCC` va collegato al polo positivo dell'alimentazione (3.3V) mentre `GND` a massa, `CH_PD` ai 3.3V e `RST` è il pin di reset e riavvia il chip se collegato a `GND`.

Alimentazione e collegamento seriale

I moduli ESP-01 operano a 3.3V.

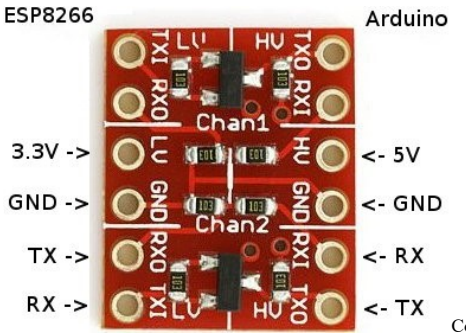
Importante

Voltaggi superiori potrebbero danneggiare il chip che, a differenza dei devices Arduino, non dispone di protezioni on board.

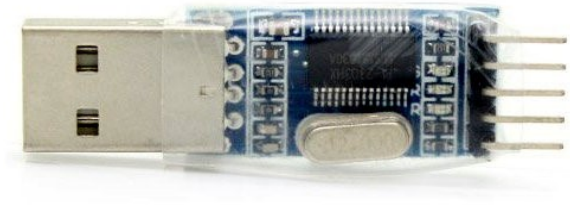
Questo non è vero per alcune board basate sui moduli ESP8266, a seconda di quanto indicato dal produttore.

Questo ha come conseguenza che non è possibile operare un collegamento seriale diretto (tramite i pin RXD -> TX e TXD -> RX) con i devices operanti a 5V (come l'Arduino Uno e Mega).

È comunque possibile operare tale collegamento se mediato da un convertitore logico bidirezionale, come mostrato in figura. Su AliExpress il costo è inferiore a €1.



Collegamento seriale con Arduino


Un debugger seriale

Probabilmente il modo migliore di avere alimentazione *e* collegamento seriale con un PC, è attraverso un debugger seriale che operi anche a 3.3V.

L'ESP8266 normalmente assorbe intorno ai 50 mA, ma può arrivare anche a superare i 200 mA (che è il massimo erogabile dai pin `VCC` dell'Arduino).

Suggerimento

È consigliabile non alimentarlo con il pin 3.3V dell'Arduino se non si vogliono rischiare sporadici riavvii.

Un modo economico di alimentare l'ESP-8266 in standalone è quello di riciclare un vecchio alimentatore USB per cellulare (tagliando il connettore terminale) e mediare il collegamento dai 5V dell'alimentatore al pin `VCC` con un regolatore di tensione da 3.3V.

Boot modes

I moduli basati su ESP8266 possono essere avviati in 2 modalità (funzionanti):

- **UART** o **Bootloading**: permette di caricare firmware e sketch
- **FLASH** o **Usage**: il chip esegue lo sketch o, nel caso del firmware di fabbrica, si comporta da modem WiFi controllabile tramite comandi AT

A queste si aggiunge la modalità **SDIO**, che però è non funzionante con i firmware disponibili. Una volta fornita alimentazione all'ESP8266, questo si avvierà in una delle precedenti modalità a seconda di come si sono collegati `GPIO_0` e `GPIO_2`, come mostrato nella tabella di seguito, dove `HIGH` corrisponde al logic high (3.3V), `LOW` al logic low (massa) e `FLOAT` significa non collegato (o collegato ad un circuito in cui non passa corrente). È consigliabile mediare questi collegamenti con resistenze da 10 kΩ.

Modalità	GPIO_0	GPIO_2	GPIO_15
Bootloading	LOW	FLOAT o HIGH	LOW
Usage	HIGH	FLOAT o HIGH	LOW
SDIO	FLOAT	FLOAT	HIGH

Boot modes e pin

È importante notare che ciò limita in parte le possibilità di utilizzo dell'ESP8266, in quanto non potremo collegare `GPIO_0` e `GPIO_2` a dispositivi che normalmente restituiscono un valore `LOW` (e nel caso di `GPIO_0` anche circuiti/dispositivi normalmente aperti) e `GPIO_15` (non presente su ESP-01) a dispositivi normalmente `HIGH`, altrimenti in caso di avvio (o riavvio) l'ESP8266 passa in una modalità diversa da **Usage**. In alcuni casi basta riconfigurare il circuito (ad esempio sostituire un pulsante pullup con uno pulldown), in altri casi occorre necessariamente cambiare pin.

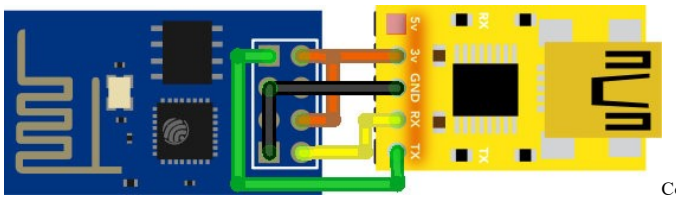
Caricare il firmware

Il firmware di fabbrica permette permette di comunicare via seriale tramite comandi AT. Sebbene tale possibilità può risultare comoda se si vuole aggiungere la connettività WiFi ad un progetto Arduino già completo, personalmente ho trovato questa modalità piuttosto frustrante a causa delle variazioni nei comandi con gli aggiornamenti dei firmware e a causa del fatto che difficilmente si può ottenere più di una richiesta GET (che è tutto ciò che serve se si vuole comunicare con ThingSpeak).

Il firmware NodeMCU invece incrementa drammaticamente le possibilità dell'ESP8266. In particolare permette, come già accennato, di programmare il chip con Arduino IDE, trasformandolo in una valida alternativa ad Arduino.

Riporto la procedura per installare il firmware su Linux. Assumo che la comunicazione con il PC sia mediata con un debugger seriale USB.

Per prima cosa collegare l'ESP8266 al debugger in modalità **Bootloading** e collegare il debugger ad una porta USB del PC.



Collegamento al debugger in modalità Bootloading

Scaricare l'ultima versione stabile de firmware NodeCMU:

```
wget https://github.com/nodemcu/nodemcu-firmware/releases/download/0.9.5_20150318/nodemcu_float_0.9.5_20150318.bin
```

Nota

Esistono build più nuove del firmware, ma la versione 0.9.5 è l'ultima versione *stabile* ed è anche l'ultima a supportare i vecchi batch di ESP8266 con 512KB di memoria. I moduli più recenti ne presentano 1024KB e sono compatibili anche con le versioni ≥0.9.6

Scaricare il programma per eseguire il flash:

```
 git clone https://github.com/themadinventor/esptool.git
```

Scrivere il firmware sul modulo:

```
cd esptool
sudo python esptool.py --port /dev/ttyUSB0  write_flash 0x00000 ../nodemcu_float_0.9.5_20150318.bin
```

Il programma dovrebbe restituire un output simile a questo:

```
Connecting...
Erasing flash...

Writing at 0x00000000... (0 %)
```

Se così non dovesse essere, verificare i collegamenti e riprovare.

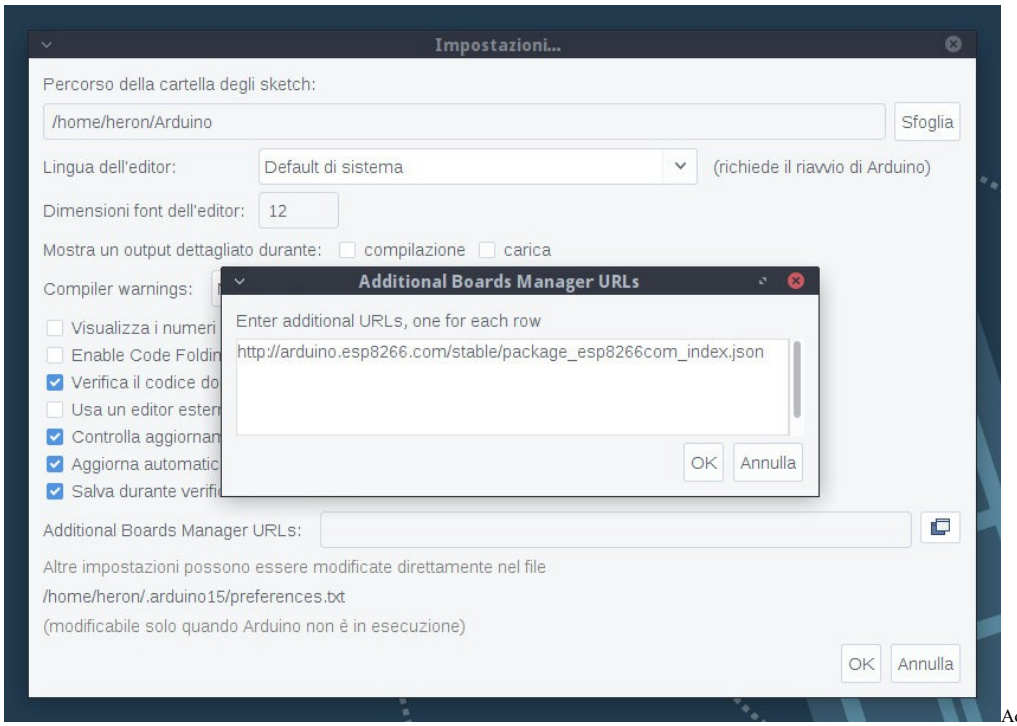
Configurare Arduino IDE

Nota

Nel momento in cui scrivo questo articolo solo la versione 1.6.5 di Arduino IDE è considerata compatibile. Questo potrebbe cambiare col tempo.

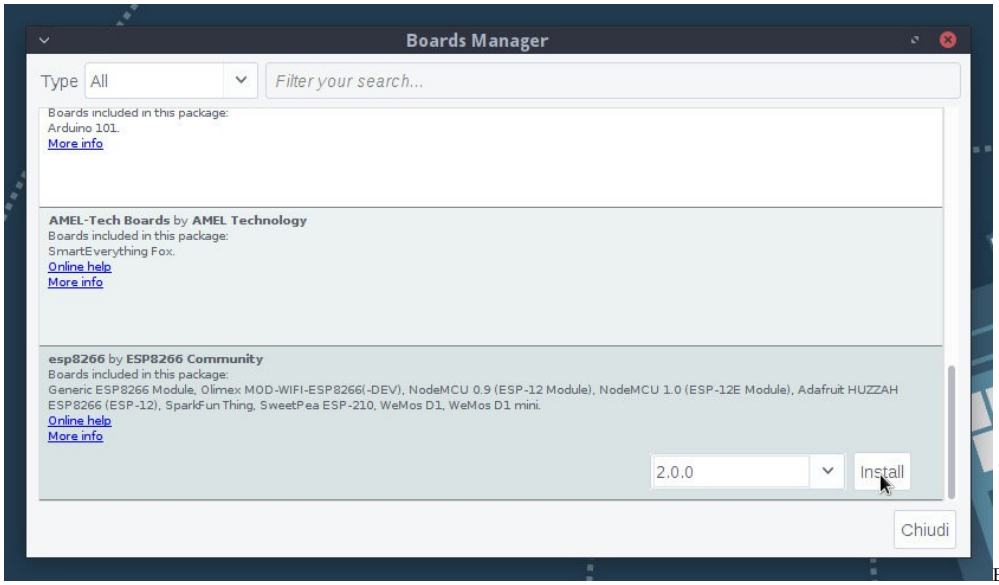
Scaricare e installare la versione 1.6.5 di Arduino IDE dal (vero) sito ufficiale Arduino.cc.

Avviare Arduino IDE ed aprire "File" > "Impostazioni" ed inserire http://arduino.esp8266.com/stable/package_esp8266com_index.json in "Additional Boards Manager URLs":



Additional Boards Manager URLs

Aprire "Strumenti" > "Scheda" > "Boards Manager" ed installare "esp8266 by ESP8266 Community":



Boards Manager

Caricare il primo sketch

Collegare l'ESP8266 al debugger in modalità **Bootloading** e collegare il debugger ad una porta USB del PC. Ora siamo pronti per caricare il primo sketch.

Aprire "File" > "Esempi" > "01.Basics" > "Blink". All'interno dello sketch sostituire modificare la riga 20

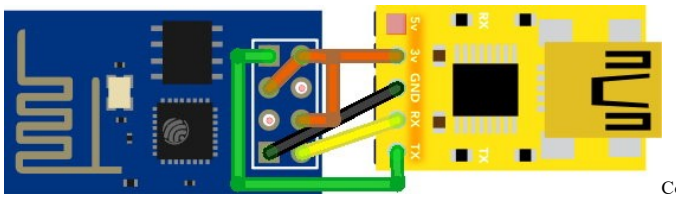
```
pinMode(13, OUTPUT);
```

in

```
pinMode(1, OUTPUT);
```

Quindi selezionare "Strumenti" > "Scheda" > "Generic ESP8266 Module" e "Strumenti" > "Porta" > "USB0" (che portrebbe avere un nome diverso). Infine cliccare su Carica.

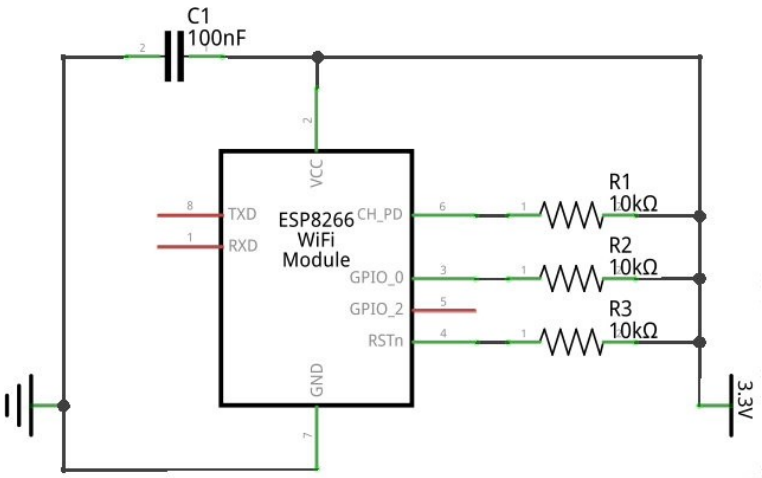
Se si avvia l'ESP8266 in **Usage** mode, è possibile notare che il led (blu) si accende e spegne ad intervalli regolari di 1 secondo.



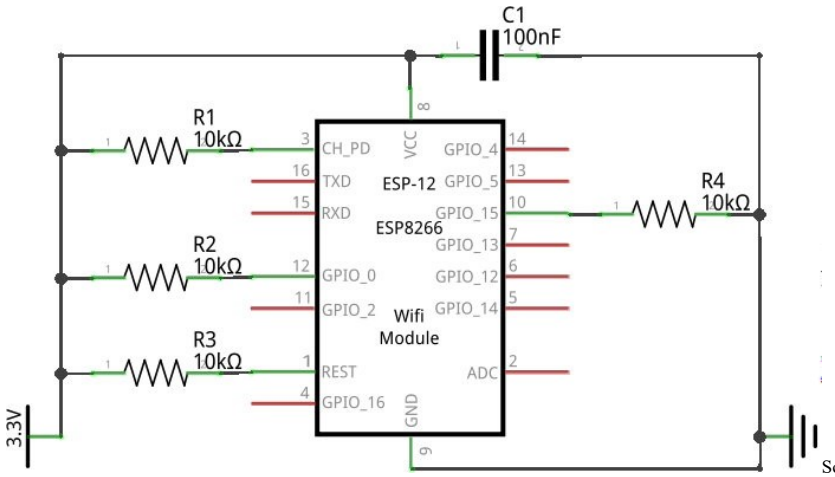
Collegamento al debugger in modalità Usage

Migliorare la stabilità

I collegamenti mostrati fin'ora possono andar bene durante la fase di testing, ma nel finalizzare il progetto (magari su PCB) è opportuno adottare configurazioni atte a migliorare la stabilità del chip. Riporto di seguito lo schema base raccomandato.



Schema base ESP-01



Schema base per gli alti moduli

Commenti

SEMPRE SU **HERON'S BLOG**

Il mio primo progetto con Arduino: un …
7 anni fa • 13 commenti
a bunch of random stuff.

heron's blog diventa self-hosted passa a …
5 anni fa • 4 commenti
a bunch of random stuff.

heron's blog
7 anni fa • 14 commenti
a bunch of random stuff.

Sostituire un …
7 anni fa • 7
a bunch of

44 Commenti heron's blog Privacy Policy di Disqus Accedi

Consiglia 2 Tweet Condividi Ordina dal migliore

Partecipa alla discussione...

ENTRA CON O REGISTRATI SU DISQUS

Nome

Mario Buccoliero • un anno fa
Buongiorno a tutti,
avrei un quesito che, navigando sul web, non sono riuscito a soddisfare.
Ho messo a punto (ma non ottimizzato) un semplice circuito con Arduino uno per la lettura di alcune variabili tramite sensori (temperatura, umidità, livello acqua) e l'azionamento di una pompa per irrigare tramite relé, adesso però mi piacerebbe inviare e ricevere con arduino parte di queste letture. Ho letto che l' ESP8266 0-01 è praticamente un'interfaccia wi-fi adatta a questo ma non ho trovato nessun esempio pratico sul come inserire le variabili all'interno dello sketch. E possibile? Avete qualche dritta da darmi?
Grazie in anticipo
• Rispondi • Condividi ›

Silvia Sanna • 2 anni fa
Salve, io ho un esp8266 e qualunque programma faccia partire ho sempre questi errori: warning: espcomm_sync failed
error: espcomm_open failed
error: espcomm_upload_mem failed
error: espcomm_upload_mem failed

come posos risolvere?
• Rispondi • Condividi ›

Antonio Sainato ↗ Silvia Sanna • 2 anni fa
Ho avuto lo stesso problema con un tipo di interfaccia USB, poi ho usato questa; https://it.aliexpress.com/i..., ed ha funzionato
• Rispondi • Condividi ›

Federico ↗ Silvia Sanna • 2 anni fa
controlla di avere i permessi sulla porta usb
• Rispondi • Condividi ›

Francesco Pellegrino • 4 anni fa
salve ho un problema, ho programmato l'esp-01 con codice blynk collegandolo alla mia wifi. ora mi succede una cosa se avvio l'esp-01 non collegando niente all'uscita gpio0 si avvia e blynk lo vede, dopo l'avvio collego gpio0 al pin del mio rele e tutto funziona alla grande..... cosa diversa se do corrente con gpio0 collegato al pin in del rele, il tutto non funziona..... blynk lo vede disconnesso.... come potrei ovviare a tutto cio...
• Rispondi • Condividi ›

Augusto Ciuffoletti • 4 anni fa
Ok, funziona! Bravo per aver puntualizzato le differenze di alimentazione e pilotaggio. Lo fanno in pochi, e ci sono anche quelli che dicono che non è poi così importante :-/

Ho usato un USB-TTL/USB-STC-ISP da pochi euro come quello che hai indicato tu (comprato su Amazon), ancora seguendo le tue indicazioni ho usato il "gestore" ESP8266 (in versione 2.3.0) della IDE Arduino 1.6.11 indicando come programmer AVR ISP, e mi sono fatto costruire il firmware da https://nodemcu-build.com/ (strepitoso...), l'ho caricato con il software esptool.py scaricato come da tue indicazioni. Il blink funziona subito, senza necessità di riavviare l'ESP. Scollegando l'alimentazione e riavviando bisogna mettere in Usage.

Grazie!
• Rispondi • Condividi ›