# Ulteriori schede di laboratorio

SCHEDA 7

## Serial device server

Si vuole collaudare un convertitore Ethernet seriale stabilendo un socket TCP/IP.

### Tibbo device server

Un serial device server presenta una porta Ethernet da un lato e una porta seriale asincrona sull'altro. Nel caso del dispositivo Tibbo DS203 (**fig. 1**) si tratta di una seriale RS 232.

Una volta alimentato il dispositivo con una tensione continua tra 9 e 25 V, collegarlo al PC da entrambi i lati, utilizzando due cavi cross, sia per la RS 232 sia per Ethernet (**fig. 2**).

Un cavo Ethernet cross incrocia le coppie 1-2 con 3-6, cioè il doppino di trasmissione con quello di ricezione, ed è da utilizzare solo per i collegamenti diretti tra due dispositivi finali, senza interposizione di Hub o altro.

Per poter eseguirne il collaudo, si tratta di configurarlo per una comunicazione diretta con un applicativo sul PC.

**Fig. 1.** *Tibbo serial device server DS203.*

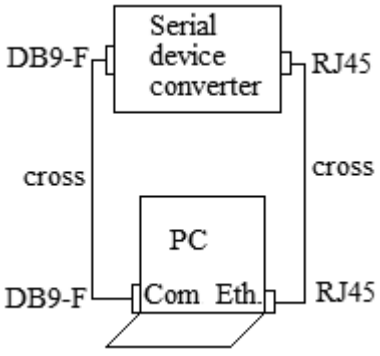



**Fig. 2.** *Connessioni cross.*

### IP address

Un dispositivo serial server è sempre fornito con un corredo di software utile per riconoscerlo e configurarlo; inoltre, dispone di una serie di comandi di monitor attivabili da seriale al reset.

Se si tratta di una prima esperienza e non si conosce l'indirizzo IP attuale del dispositivo, si può operare per esempio dalla seriale RS232, aprendo la Com del PC mediante una finestra Hyperterminal con i parametri indicati nel manuale (38400, 8, N, 1, Hw). Dopo aver premuto il tasto di reset del dispositivo, digitare il comando Get di richiesta dell'IP attuale:

| Ctrl-B GIP Cr |
|---|

in risposta al quale il sistema fornisce il proprio indirizzo, per esempio 1.0.0.1.

Inviare Ctrl-B O Cr per uscire dalla modalità comandi (oppure spegnere e riaccendere il dispositivo).

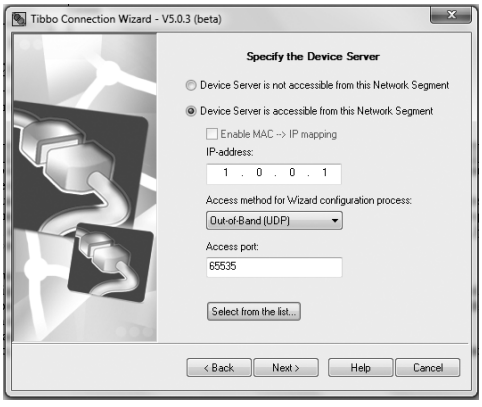



**Fig. 3.** *Tibbo Connection Wizard.*

Per poter configurare il dispositivo utilizzando il software da Ethernet, occorre predisporre l'indirizzo della scheda locale del PC per il medesimo segmento di rete, entrando in configurazione delle proprietà relative al protocollo TCP/IPv4 della scheda (in *Apri Centro connessioni di rete* ▸*Modifica impostazioni scheda* ▸*Connessione alla rete locale LAN* ▸*Proprietà* ▸*Rete* ▸*Protocollo Internet versione 4*) e impostando un indirizzo fisso compatibile, quale per esempio

| 1.0.0.2, con maschera 255.255.0.0 |
|---|

A questo punto, richiamando il software di configurazione in dotazione Tibbo Connection Wizard (**fig. 3**) e cliccando su *Select from the list*, parte una

ricerca dei dispositivi collegati che individua il serial server in esame.
Una volta selezionato, si procede indicando altri parametri, ossia chi è il primo ad inviare i dati (l'applicazione sul PC) e il protocollo utilizzato (TCP/IP sul port 1001).

## Collaudo

Aprire quindi un canale di collegamento (socket) con protocollo TCP/IP (Winsock), con il dispositivo di indirizzo 1.0.0.1 sul port 1001, utilizzando una seconda finestra Hyperterminal (**fig. 4**).
Con entrambe le finestre attive, se il dispositivo è funzionante, ciò che si scrive in una finestra deve comparire sull'altra e viceversa.

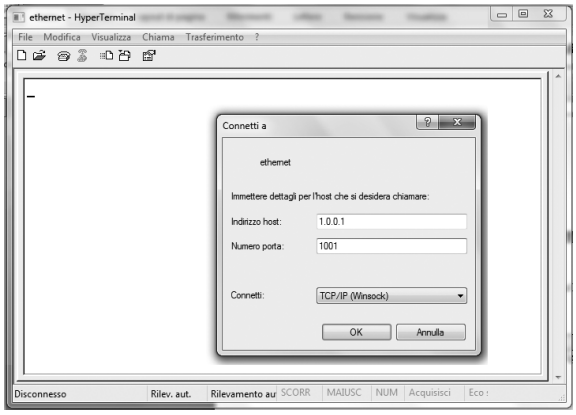



**Fig. 4.** *Tibbo Connection Wizard.*

## Rete di sensori

Assegnando ai diversi serial converter indirizzi IP differenti, appartenenti al medesimo segmento di rete del PC, è possibile comporre una rete di sensori in RS 232 (**fig. 5**) utilizzando hub o switch di diramazione.

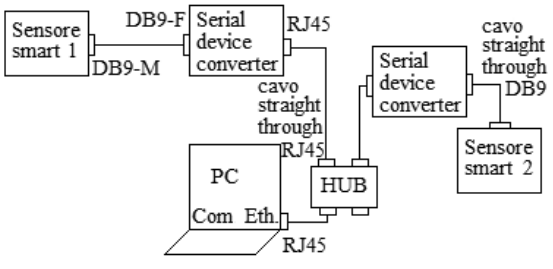



**Fig. 5.** *Rete di sensori smart in RS 232.*

I cavi da utilizzare sono tutti diritti (straight through), sia i seriali, con connettori maschio-femmina, sia gli Ethernet.
Per attivare la rete, l'applicazione sul PC deve prima aprire un canale TCP/IP (socket) con ciascuno di essi, indicando per ciascuno l'indirizzo IP (univoco) e il numero del port.

SCHEDA 8

# Gestione di Arduino da remoto

Per gestire una scheda Arduino da remoto, per esempio da PC tramite una linea seriale, sono necessarie tre cose:

- un insieme definito di comandi e risposte da scambiarsi tra il PC, che funziona da master (*client*) e la scheda slave Arduino (*server*);
- un programma (*sketch*) su Arduino, che attende i comandi in arrivo sulla linea seriale e li esegue;
- un software sul PC (una finestra *Hyperterminal* o un programma in *Visual Basic* o *C++*) che invia i comandi e visualizza le risposte.

Per facilitare l'elaborazione delle stringhe di comando si può utilizzare la funzionalità *String* di Arduino, mentre la finestra di dialogo sul PC può essere la stessa del monitor seriale a disposizione.

## Studio e collaudo della funzionalità *String* di Arduino

Una stringa è un array di elementi char, terminato con il carattere **null** (\0).
Il carattere null ha valore binario 0, notevolmente diverso rispetto al carattere ASCII '0', che ha valore decimale 48 ($30_H$). Per manipolare le stringhe (copiarle, concatenarle, calcolarne il numero dei caratteri) Arduino, oltre alle funzioni standard, mette a disposizione la funzionalità **String** (con il carattere iniziale 'S' maiuscolo), non compatibile con il linguaggio C standard.
Per comprenderne le caratteristiche, si esegua il programma che segue e si osservi quanto emesso nella finestra di monitor seriale.

```
//String_basic
String text1 = "prima stringa";
String text2 = " con aggiunta";
String text3; //da usare successivamente

void setup()
{
Serial.begin(9600);
Serial.print(" text1= ");
Serial.println( text1);
```

```
Serial.print(text1.length());
Serial.println(" caratteri significativi");
Serial.print("anche text2 contiene ");
Serial.print(text2.length());
Serial.println(" caratteri");
text1.concat(text2);
Serial.println("concatenandole, text1 di-
venta: ");
Serial.println(text1);
}
void loop()
{}
```

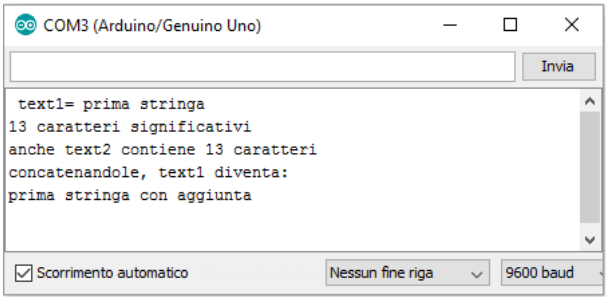



**Fig. 6.** *Collaudo della funzionalità String.*

Definendo la stringa text1 di tipo String è possibile ottenerne la lunghezza con la:

```
text1.length()
```

e concatenarla con la variabile String text2 mediante la:

```
text1.concat(text2)
```

Per combinare stringhe si può anche utilizzare l'operatore +. Per verificarlo basta aggiungere in coda al codice precedente le due righe seguenti.

```
text3 = text1 + " e altro ancora";
Serial.println(text3);
```

Per elaborare con la funzione String gli array di caratteri definiti in C, bisogna prima convertirli in oggetti String, come indicato di seguito.

```
charoldString[] = "Array di caratteri da
convertire in un oggetto String";
StringnewsString = oldString;
```

## Gestione da monitor seriale

Sulla scheda Arduino, la presa USB confluisce in un convertitore FT232, le cui uscite TXD e RXD sono incrociate con i segnali della porta seriale del microprocessore e terminano rispettivamente sui pin 0 (RX) e 1 (TX) del connettore Digital.

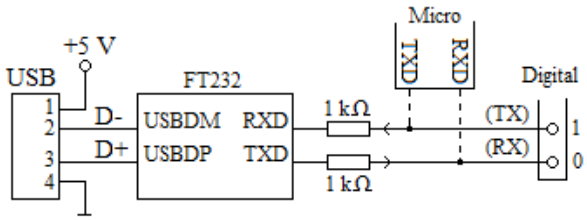



**Fig. 7.** *Linee seriali interne a Arduino.*

È possibile quindi governare le uscite di Arduino o leggere lo stato degli ingressi tramite monitor seriale, concordando i codici di attivazione da inviare e predisponendo uno sketch che interpreti il contenuto dei messaggi ricevuti dalla seriale.

Lo sketch che segue, per esempio, accende e spegne il LED presente su Arduino, connesso con il pin 13, inviando da monitor rispettivamente i caratteri A (accendi) e S (spegni).

```
/* Monitor_LED */
String comando ="";

void setup()
{
Serial.begin(9600);
pinMode(13,OUTPUT);
}
void loop()
{
while(Serial.available())
comando+=char(Serial.read());

 if(comando!=""){
  if(comando == "A"){
  digitalWrite(13, HIGH);
  Serial.println("Led ON");
  }
  if(comando == "S"){
  digitalWrite(13, LOW);
  Serial.println("Led OFF");
  }
 comando="";
 }
}
```

Ogni volta che il programma acquisisce dalla porta seriale un comando valido, aziona di conseguenza l'uscita 13 e restituisce un messaggio di conferma. Terminata l'elaborazione, la stringa di comando viene svuotata, così da renderla disponibile per una nuova acquisizione.

## Possibile continuazione

Predisporre uno sketch che gestisca i messaggi a dimensione fissa riportati in tabella, supponendo che i pin da 2 a 7 siano stati predisposti come ingressi con pull-up e da 8 a 13 come uscite.

| Protocollo di gestione da remoto | | |
|---|---|---|
| **Comando** | **Azione** | **Valori** |
| Hnn | Porta alto il pin nn | H08 – H13 |
| Lnn | Porta basso il pin nn | L08 – L13 |
| Rnn | Legge lo stato H/L del pin nn | R02 – R13 |
| An | Legge il livello dell'ingresso analogico n | A0 – A5 |

Per la soluzione, è utile considerare le due funzioni C standard di manipolazione delle stringhe evidenziate di seguito.

```
char *strstr (char *string1, char *string2)
```

Ricerca la presenza della stringa (dello spezzone) string2 all'interno di string1 e in caso affermativo ritorna un valore diverso da zero (in pratica ritorna il puntatore alla prima lettera corrispondente); per esempio:

```
if (strstr (str,"pin4=on")) digitalWrite(4, HIGH);
```

porta alto il pin 4 se nella stringa str è presente la scritta "pin4=on";

```
int sprintf (char * buf, char *format, arg-list)
```

Compone la stringa buf con le variabili listate al termine, utilizzando il format indicato; per esempio:

```
sprintf(str,"%d%dore",h/10, h%10);
```

compone str con la scritta indicante le decine e le unità delle ore in decimale, seguite dal testo "ore". Alcuni codici di formato sono: %c (singolo carattere), %d (decimale intero), %f (decimale con virgola), %x (esadecimale), %s (stringa di caratteri), ecc., con diverse varianti sul numero di cifre e di decimali da riportare.

SCHEDA 9

# WiFi con Arduino

Il modulo ricetrasmettitore (*transceiver*) ESP8266 ESP-01 WiFi fornisce un'interfaccia (*bridge*) tra una porta seriale asincrona e lo standard WiFi, rendendo possibile la comunicazione tra un qualunque microcontrollore fornito di UART e un computer, uno smartphone Android o un tablet dotato di porta WiFi.

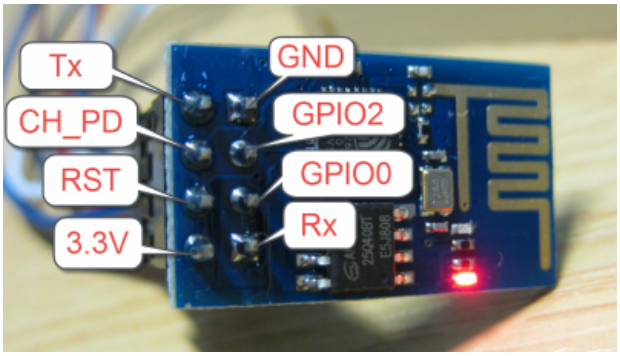



**Fig. 8.** *Modulo WiFi ESP8266 ESP-01(1 MB flash).*

Il modulo interagisce dal lato UART mediante comandi seriali (comandi AT) con velocità di default 115,2 kbps. I comandi disponibili sono molteplici e reperibili in rete ("ESP8266 AT commands set"). In tabella sono riportati i più utilizzati.

| Alcuni comandi AT per ESP8266 | | |
|---|---|---|
| **Funzione** | **Comando AT** | **Risposta** |
| Test | AT | OK |
| Reset | AT+RST | OK Ai-Thinker ready |
| Eco no/si | ATE0 ATE1 | OK |
| Get IP address | AT+CIFSR | 192.168.4.1 |
| WiFi Mode | AT+CWMODE? | Query |

Il dispositivo ESP8266 ESP-01 ha tre modalità operative:
1) **Station-Client** (**STA**);
2) **Access Point-Server** (**AP**);
3) **Both**.
In modalità **STA** (1, Client) il modulo può connettersi ad un Access Point e rendersi visibile a tutti i dispositivi presenti sulla rete. In modalità **Access Point** (2, Server), si permette ad altri dispositivi di connettersi con il modulo stesso e di stabilire una comunicazione dati bidirezionale via WiFi.
Con la terza modalità operativa, il comportamento sostiene entrambe le modalità precedenti.

## Connessioni elettriche tra ESP-01 e Arduino

Per esplorare il funzionamento dell'ESP-01 utilizzando il monitor seriale dell'ambiente di programmazione, si deve predisporre un collegamento del modulo con i segnali Tx (pin 1) e Rx (pin 0) di Arduino, connessi a loro volta alla USB del PC.
Nel circuito della scheda Arduino, il TXD proveniente dalla USB del PC finisce sul pin 0 (RX) del connettore Digital, al quale pertanto va connesso il segnale Rx del modulo, mentre il Tx del modulo va portato al pin 1 (TX), che prosegue verso RXD del convertitore USB FT232.

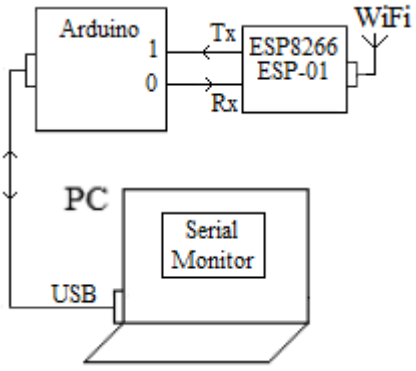



**Fig. 9.** *Connessione tra PC e modulo ESP-01.*

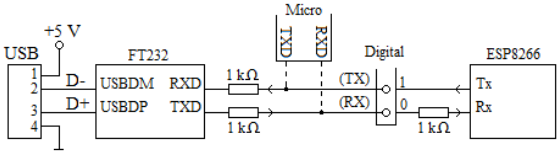



**Fig. 10.** *Linee seriali interne ad Arduino.*

Purtroppo, sui medesimi pin del connettore confluiscono anche i segnali della porta seriale del microprocessore ATMEL residente sulla scheda.
Pertanto, per evitare possibili conflitti che potrebbero insorgere per eventuali errori iniziali nei collegamenti o nella programmazione, si consiglia di predisporre delle protezioni. In un eventuale conflitto tra i segnali seriali, quelli provenienti dalla USB risultano già protetti dalla presenza di resistenze da 1 kΩ poste in serie.
Inoltre, poiché il modulo WiFi funziona a 3,3 V mentre il micro ATMEL di Arduino è a 5 V, è bene porre una resistenza di protezione da 1 kΩ in serie al segnale Rx del modulo.
Purtroppo, il passo del connettore non permette l'inserimento diretto del modulo ESP-01 in una breadboard, perciò i collegamenti vanno realizzati volanti.
In alternativa alla connessione diretta a 3,3 V, e per maggiore sicurezza, è disponibile l'adattatore ADP-01 level converter 3,3 V – 5 V, con passo compatibile.

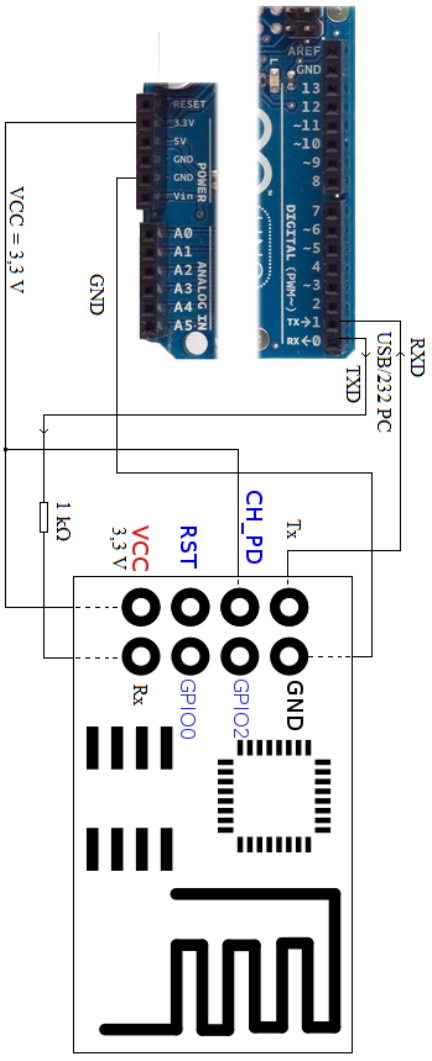



**Fig. 11.** *Collegamento diretto a 3,3 V.*

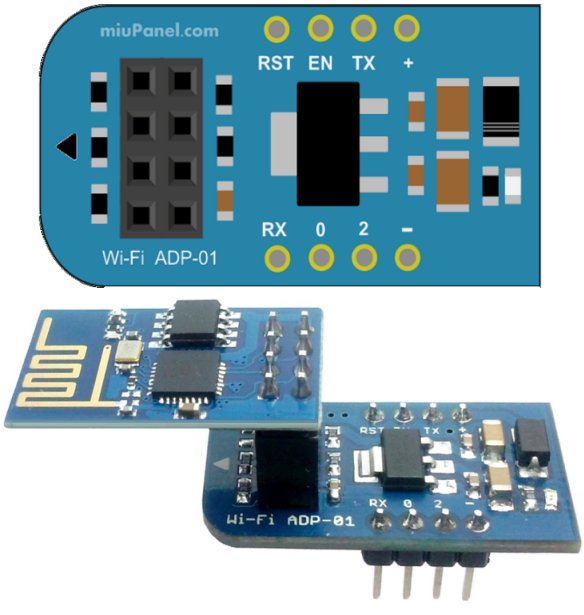



**Fig. 12.** *Adattatore level converter ADP-01.*

L'adattatore dispone al suo interno di uno stabilizzatore a 3,3 V, una resistenza di protezione da 1 kΩ in serie al segnale RX e resistenze di pull-up sui restanti segnali, pertanto può essere connesso direttamente ai segnali di Arduino e alimentato a 5 V.
Il pin "0" dell'adattatore (corrispondente al pin GPIO0 del modulo) va posto a massa, perché, se posto alto, attiva il caricamento di una nuova versione del firmware.

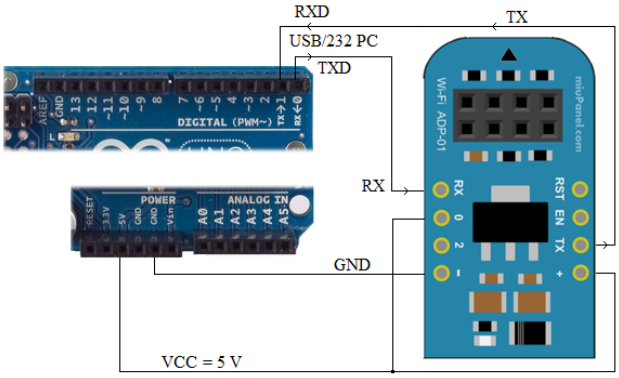



**Fig. 13.** *Connessioni a 5 V tra adattatore e Arduino.*

## Test dei comandi AT

Per testare i comandi AT del dispositivo, occorre realizzare una comunicazione diretta tra la finestra di monitor seriale e il modulo stesso, inibendo la seriale di Arduino.

- Caricare lo sketch che segue per impedire alla UART del micro di Arduino di interferire, ponendo in alta impedenza i due pin di comunicazione 0 e 1.

```
/* test ESP8266 */
void setup() {
pinMode (0, INPUT);
pinMode (1, INPUT);
}

voidloop()
{ }
```

- Configurare il monitor seriale per 115200 bit al secondo e con l'inserimento automatico dei due caratteri di ritorno carrello (CR) e di salta riga (NL) al termine di ogni comando inviato.

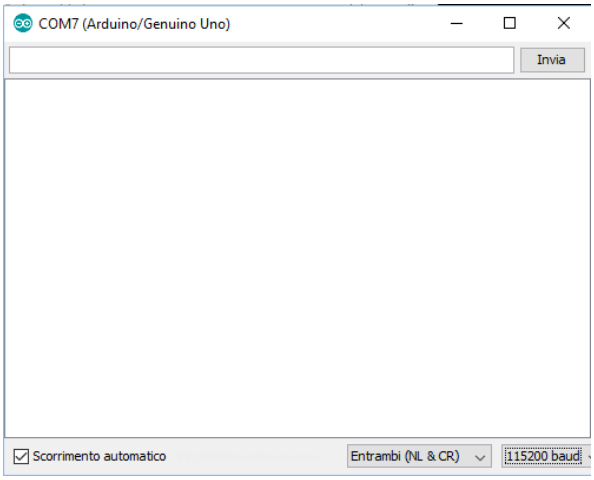



**Fig. 14.** *Predisposizione del monitor seriale.*

- Verificare la connessione inviando il comando "AT", al quale il modulo risponde con "OK".
  Di default il modulo esegue l'eco dei comandi ricevuti, quindi il monitor visualizza tra le risposte anche il comando. Se ciò dovesse dar fastidio, inserire "ATE0" (Eco = 0) per inibire l'eco.

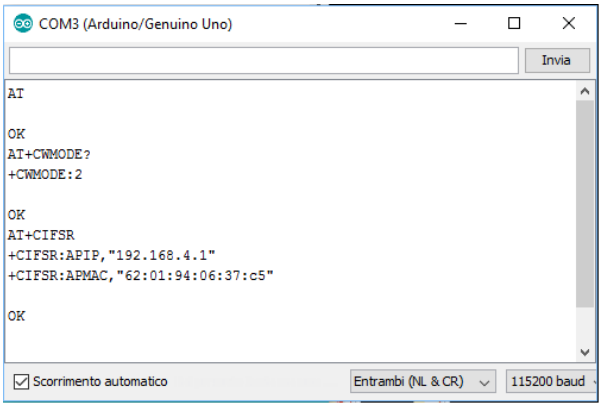



**Fig. 15.** *Comunicazioni con comandi AT.*

## Modalità Station-Client

In modalità Station-Client (**STA,** modalità1, Client) il modulo ESP8266 ESP-01 può connettersi ad un Access Point (come il router di casa) e rendersi visibile a tutti i dispositivi presenti sulla rete.
Alcuni dei comandi disponibili sono riportati in tabella.

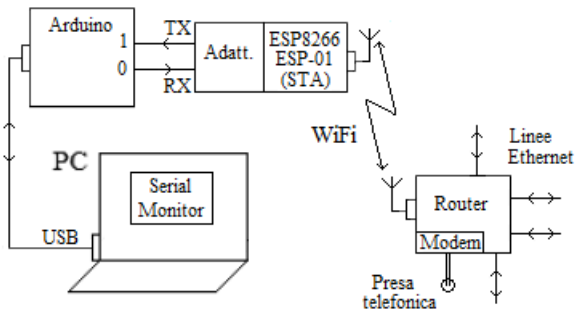



**Fig. 16.** *Connessione a un router.*

| Alcuni comandi per modalità STA-Client | | |
|---|---|---|
| **Funzione** | **Comando AT** | **Risposta** |
| WiFi Mode | AT+CWMODE?<br>AT+CWMODE=1<br>AT+CWMODE=2<br>AT+CWMODE=3 | Query<br>STA (Station-Client)<br>AP (Access Point)<br>BOTH |
| Get IP address | AT+CIFSR | 0.0.0.0 |
| Connessioni TCP/UDP | AT+CIPMUX?<br>AT+CIPMUX=0<br>AT+CIPMUX=1 | Query<br>Single<br>Multiple |
| List Access Points | AT+CWLAP | +CWLAP: …<br>+CWLAP: …<br>OK |
| Join Access Point | AT+CWJAP?<br>AT+CWJAP="SSID", "Password" | Query<br>OK |
| Quit Access Point | AT+CWQAP=?<br>AT+CWQAP | Query<br>OK |

- Impostare la modalità operativa STA (1) inviando il comando AT+CWMODE = 1 e verificarla.
  Per esempio:

```
AT+CWMODE=1
OK
AT+CWMODE?
+CWMODE:1
OK
```

- Chiedere l'indirizzo IP in atto `AT+CIFSR` e verificare che inizialmente il modulo ha indirizzo 0.0.0.0.
- Verificare le reti WiFi captate, con il comando `AT+CWLAP`.

  La risposta è del tipo:

```
+CWLAP:(3,"Vodafon-WiFi",-57," MAC address",1)
+CWLAP:(4,"NETGEAR14",-52," MAC address",7)
OK
```

- Associare (*join*) il modulo a una delle reti disponibili, con il comando `AT+CWJAP`, che accetta come primo parametro l'SSID della rete e come secondo parametro la password di accesso:

```
AT+CWJAP="NETGEAR14","MiaPassword"
```

  La risposta è:

```
WIFI CONNECTED
WIFI GOT IP
OK
```

  In particolare, con `WIFI GOT IP` il dispositivo segnala di aver ottenuto un indirizzo IP.

  Verificare il nuovo IP ricevuto, con il comando `AT+CIFSR`, e confrontare il risultato con quanto presente sul router.

| | | | | | |
|---|---|---|---|---|---|
| 802.11g | 54 Mbps | Eccellente | 32:24:fe:a5:07:de | 192.168.24.2 | Connesso |

**Fig. 17.** *Esempio di informazione fornita dal router.*

## Modalità Access Point

Impostando il dispositivo in modalità **Access Point** (2, Server), si permette ad altri dispositivi di connettersi e di stabilire una comunicazione dati bidirezionale via WiFi con il modulo stesso.
Durante l'esercitazione il collegamento WiFi è realizzato con il PC.

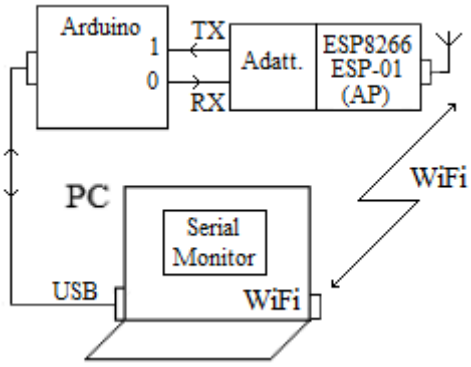



**Fig. 18.** *Modulo in modalità AP per la connessione WiFi ad un PC.*

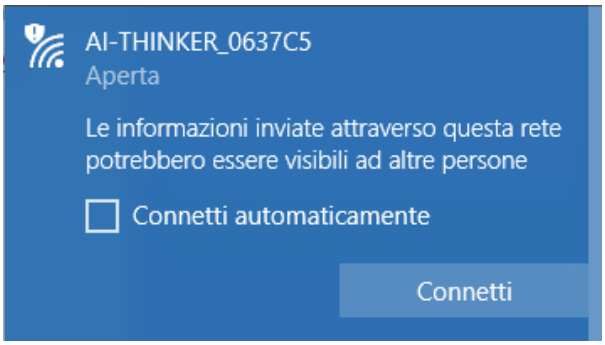
Impostare il modulo per la modalità operativa AP-Server, inviando il comando `AT+CWMODE=2`.
In modalità AP, il modulo ha un suo indirizzo IP di default, rilevabile con il comando `AT+CIFSR` (per es. 192.168.4.1) e può gestire fino a 5 comunicazioni sul port di default 333.

| Alcuni comandi per modalità AP-Server | | |
|---|---|---|
| **Funzione** | **Comando AT** | **Risposta** |
| Get IP address | AT+CIFSR | 192.168.4.1 |
| Connessioni TCP/UDP | AT+CIPMUX?<br>AT+CIPMUX=0<br>AT+CIPMUX=1 | Query<br>Single<br>Multiple |
| WiFi Mode | AT+CWMODE?<br>AT+CWMODE=1<br>AT+CWMODE=2<br>AT+CWMODE=3 | Query<br>STA (Station-Client)<br>AP (Access Point)<br>BOTH |
| Attiva/ disattiva server | AT+CIPSERVER = <mode>[,<port>] | mode = 0 chiude<br>mode = 1 attiva<br>default port = 333 |
| Server timeout | AT+CIPSTO?<br>AT+CIPSTO=<time> | Query<br>time = 0 – 28800 s<br>default = 180 s |
| Attiva una connessione TCP o UDP | AT+CIPSTART=?<br>(CIPMUX=0)<br>AT+CIPSTART=<type>,<addr>,<port><br>(CIPMUX=1)<br>AT+CIPSTART=<id><type>,<addr>,<port> | Query<br>id = 0-4<br>type = TCP/UDP<br>addr = IP addr |

- Abilitare il modulo per connessioni multiple (numerate da 0 a 4), con il comando `AT+CIPMUX=1`. Avviare, successivamente, il servizio server con il comando `AT+CIPSERVER=1,80`. Il primo numero del comando (1) indica al modulo che il server va aperto (0 per chiuderlo) e il secondo è il numero del port che un client dovrà usare per connettersi (80 è il port di default del protocollo http).
  Se non indicato, il numero di default del port rimane 333.
- Una volta in modalità AP, il modulo risulta presente (aperto) nell'elenco delle reti wireless captate dal PC (access point AI-THINKER in figura).


**Fig. 19.** *Modulo WiFi AI-Thinker rilevato.*

- Selezionare quindi la rete in oggetto e comandare la connessione.



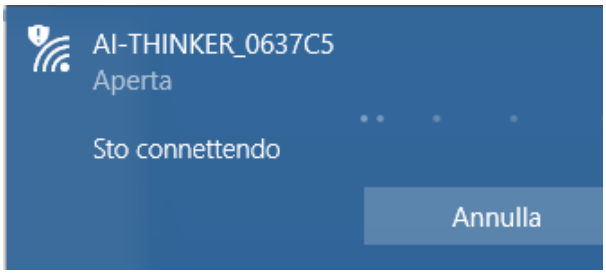




**Fig. 20.** *Connessione alla rete selezionata..*

- Una volta connessi, selezionando ancora la connessione, si può notare tra le "proprietà" che il WiFi del PC ha assunto un indirizzo IP (per es. 192.168.4.2) compatibile con quello presente nel modulo ESP-01 (192.168.4.1).


**Fig. 21.** *Rete connessa.*

- Verificare l'avvenuta connessione attraverso un ping del modulo ESP-01 (via WiFi) dal prompt dei comandi del PC, mediante il comando:

```
>ping 192.168.4.1
```

## Client da HyperTerminal

Per stabilire una connessione TCP/IP (un socket) tra un client e un server, il client deve conoscere l'indirizzo IP del server e il numero del port su cui è fornito il servizio.

- Porre il modulo ESP8266 in modalità Access Point-Server, dal monitor seriale, con i comandi:

```
AT+CWMODE=2
AT+CIPMUX=1
AT+CIPSERVER=1,80
```

- Richiedere alla rete wireless del PC la connessione con l'access point AI-Thinker, mediante una finestra di comunicazione client del tipo HyperTerminal.
- Aprire una connessione TCP/IP (Winsock) con l'host 192.168.4.1 (funzionante da server), sul port 80.

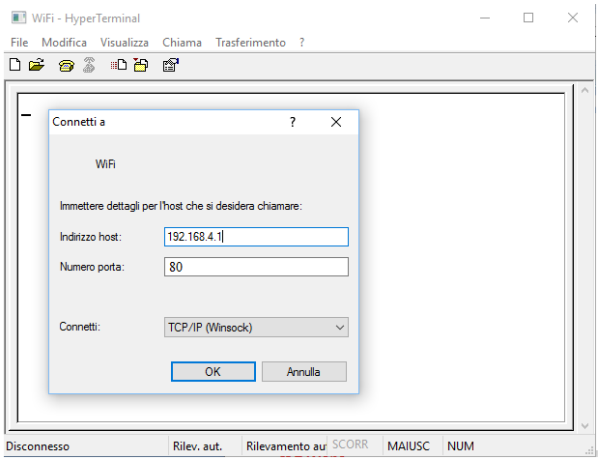



**Fig. 22.** *Client HyperTerminal.*

- Sulla finestra monitor seriale appare "0, CONNECT", per segnalare l'avvenuta apertura della connessione n. 0.

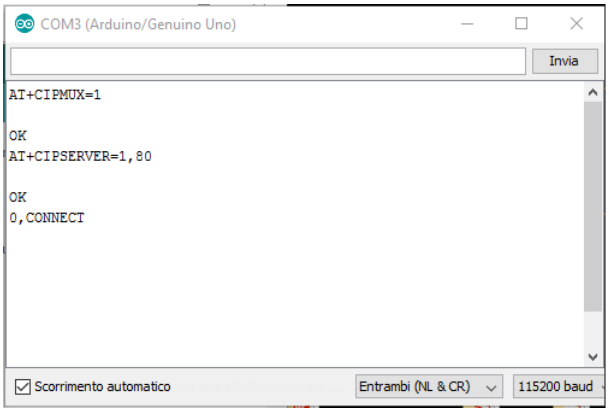



**Fig. 23.** *Connessione avvenuta.*

- Inviando da HyperTerminal la stringa "CIAO", i caratteri appaiono uno per volta sul monitor seriale:

```
+IPD,0,1:C
+IPD,0,1:I
+IPD,0,1:A
+IPD,0,1:O
```

Dopo la scritta "+IPD," appare il numero della connessione ("0"), il numero dei caratteri trasferiti ("1") e il carattere.

- Per inviare caratteri dal monitor seriale del modulo al PC, per esempio la scritta "HELLO", bisogna usare il comando `AT+CIPSEND=0,5` dove lo "0" indica il canale di trasferimento dei dati e "5" il numero dei caratteri da trasferire. Quando appare il simbolo ">", è possibile scrivere la stringa "HELLO" che apparirà successivamente nella finestra HyperTerminal.

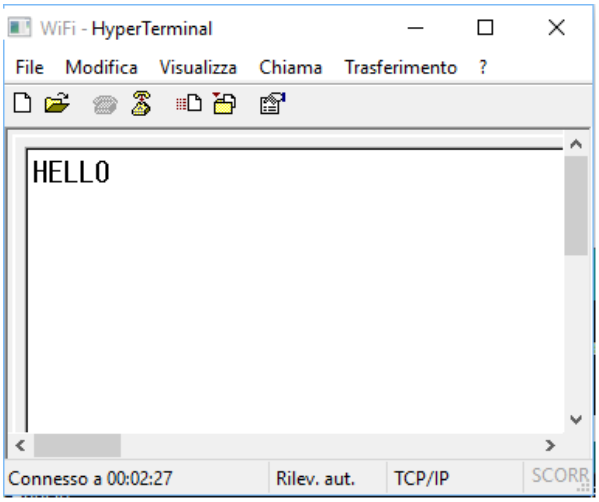



**Fig. 24.** *Messaggio ricevuto.*

- La connessione può essere chiusa chiudendo HyperTerminal (sul monitor appare "0, CLOSED"), oppure dal server con il comando

`AT+CIPCLOSE=0`.

Per chiudere definitivamente il servizio server, inviare:

`AT+CIPSERVER=0`.

## Client da SocketTest

- Predisporre il modulo ESP-01 in modalità AP-Server, con i comandi:

```
AT+CWMODE=2
AT+CIPMUX=1
AT+CIPSERVER=1 (il port di default è 333)
```

- Tra le connessioni WiFi del PC, ricercare Ai-Thinker e connettersi.
- Aprire SocketTest come Client, connettersi all'indirizzo IP del modulo (192.168.4.1) port 333 e inviare "CIAO".

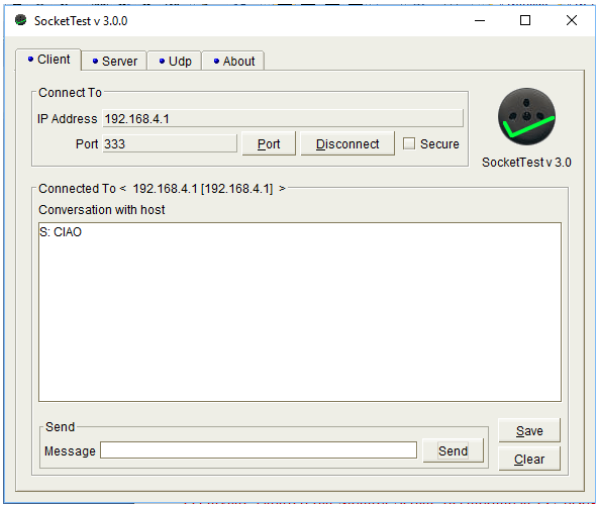



**Fig. 25.** *Finestra SocketTest.*

- Sul monitor seriale compare `+IPD,0,6:CIAO`, con "0" ad indicare il numero del canale aperto dal quale è arrivato il messaggio.

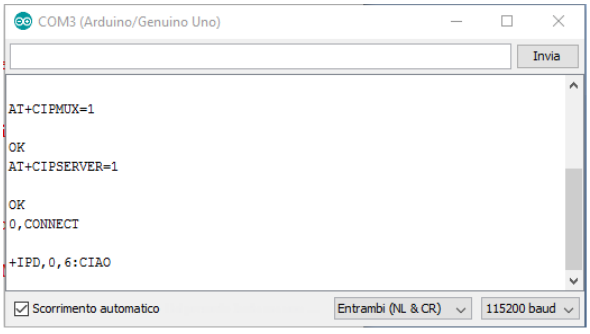



**Fig. 26.** *Finestra monitor seriale.*

- Per l'invio di caratteri dal monitor seriale e per la chiusura del canale valgono i comandi sperimentati in precedenza con HyperTerminal.
- Aprire ulteriori finestre SocketTest come Client all'indirizzo IP 192.168.4.1, port 333, fino a rilevare sul monitor seriale il massimo numero di connessioni gestibili in contemporanea dal modulo.

## SCHEDA 10
# Arduino web server WiFi

### Messaggi da browser web

Prima di procedere alla composizione di un vero web server WiFi su Arduino, è bene sperimentare in quale modo i messaggi inviati da un browser residente su un PC client vengono riportati sulla porta seriale del modulo ESP8266, utilizzando il monitor seriale disponibile nell'ambiente di Arduino.
Per impedire al micro di Arduino di interferire con la comunicazione diretta tra la finestra di monitor seriale e il modulo WiFi, al solito, si pongono in alta impedenza i suoi due pin di comunicazione caricando lo sketch che segue (**la compilazione degli sketch risulta più veloce se il canale WiFi tra il PC e il modulo risulta sconnesso**).

```
/* test ESP8266 */
void setup() {
pinMode (0, INPUT);
pinMode (1, INPUT);
}

voidloop() { }
```

Il monitor seriale va configurato per 115200 baud e con l'inserimento automatico dei due caratteri di ritorno carrello (CR) e di salta riga (NL) al termine di ogni comando inviato (**fig. 27**).

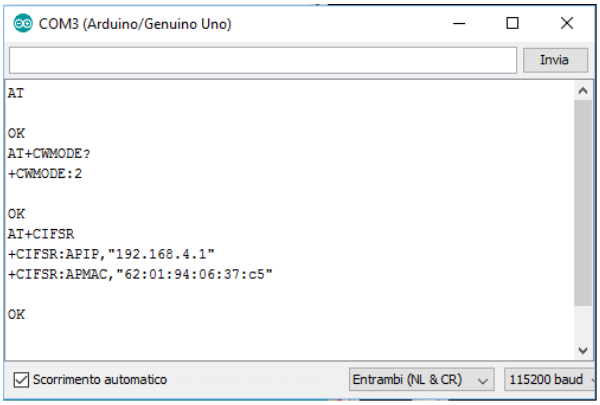



**Fig. 27.** *Predisposizione del monitor seriale.*

Connettere quindi il modulo ESP8266 con i pin 0 e 1 di Arduino, come già nell'esperienza precedente (**fig. 28**) e configurarlo come AP-Server sul port 80, con i comandi:

```
AT+CWMODE=2
AT+CIPMUX=1
AT+CIPSERVER=1,80
```

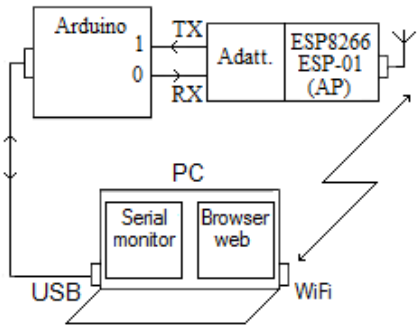



**Fig. 28.** *Modulo connesso al serial monitor.*

In modalità AP, il modulo ha un suo indirizzo IP di default (rilevabile con il comando AT+CIFSR) di valore 192.168.4.1.
Dopo aver richiesto e ottenuto dalla rete wireless del PC la connessione con l'access point AI-Thinker, aprendo, mediante browser web, una connessione http con l'indirizzo 192.168.4.1 (**fig. 29**), sul monitor di Arduino appaiono alcune interessanti informazioni (**fig. 30**).

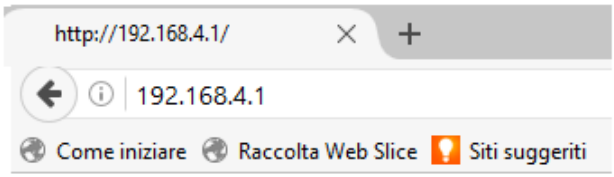



**Fig. 29.** *Apertura di un browser web.*

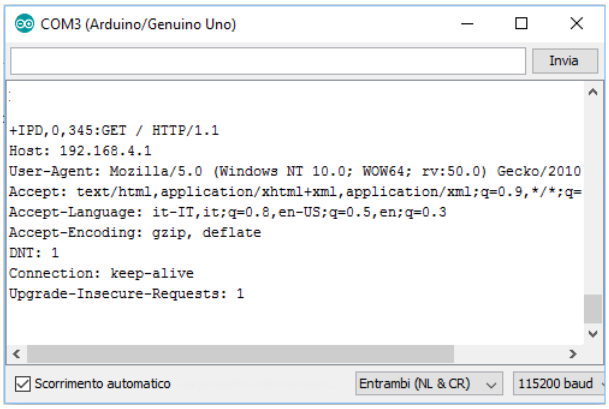



**Fig. 30.** *Informazioni relative al browser.*

Dopo i 5 caratteri "**+IPD,**", il numero della connessione ("**0**"), il numero dei caratteri trasferiti ("**345**") e il carattere "**:**" si ritrova il solito messaggio di apertura che inizia con i 5 caratteri "**GET /**" e altre informazioni, quali nome e versione del browser, il sistema operativo utilizzato, il tipo di file atteso, il linguaggio preferito, ecc.
Per inviare caratteri dal monitor seriale al browser del PC, digitare il solito comando:

```
AT+CIPSEND=0,5
```

e, all'apparire del carattere > (**fig. 31**), la stringa di 5 caratteri "ciaoo".

Sebbene il monitor avvisi che l'invio è andato a buon fine, sulla finestra del browser non appare nulla, finché (come richiesto dal protocollo) non si chiude il canale con il comando:

```
AT+CIPCLOSE=0
```

dove "**0**" indica appunto il numero del canale da chiudere.

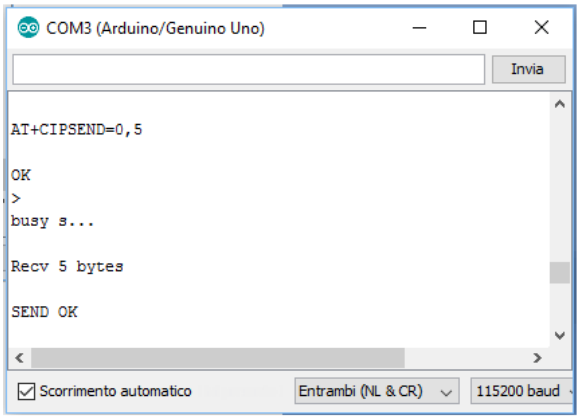



**Fig. 31.** *Invio al browser di 5 caratteri.*

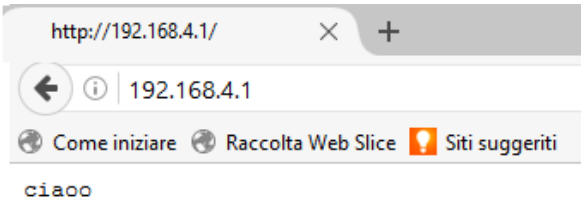



**Fig. 32.** *Messaggio visualizzato nella finestra del browser web.*

## Invio comandi da pagina web

Per controllare Arduino mediante una pagina web con connessione WiFi, oltre a configurare il modulo ESP8266 in modalità operativa AP e avviare il servizio server sul port 80, serve un file HTML che, interpretato dal browser del PC, disegni sullo schermo una pagina grafica nella quale, per semplificare, premendo un bottone, si modifica lo stato di un'uscita di Arduino (**fig. 33**).

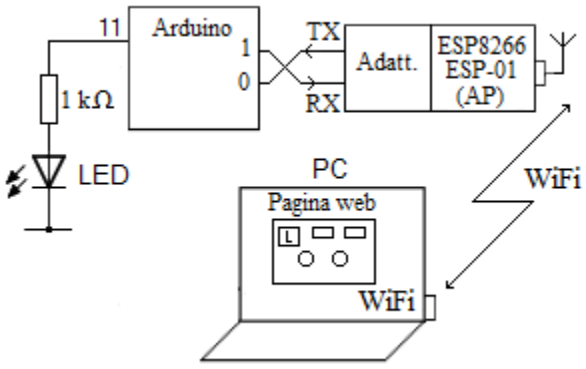



**Fig. 33.** *Comandi da pagina web.*

Questa volta, anziché il monitor seriale del PC, è il programma su Arduino che deve interagire in tempo reale con il modulo WiFi, pertanto **i segnali di comunicazione vanno incrociati**, come indicato in **fig. 34**, oltre a **sollevare il filo TX del modulo durante il caricamento del programma in Arduino**, per non confliggere con la TXD del PC.

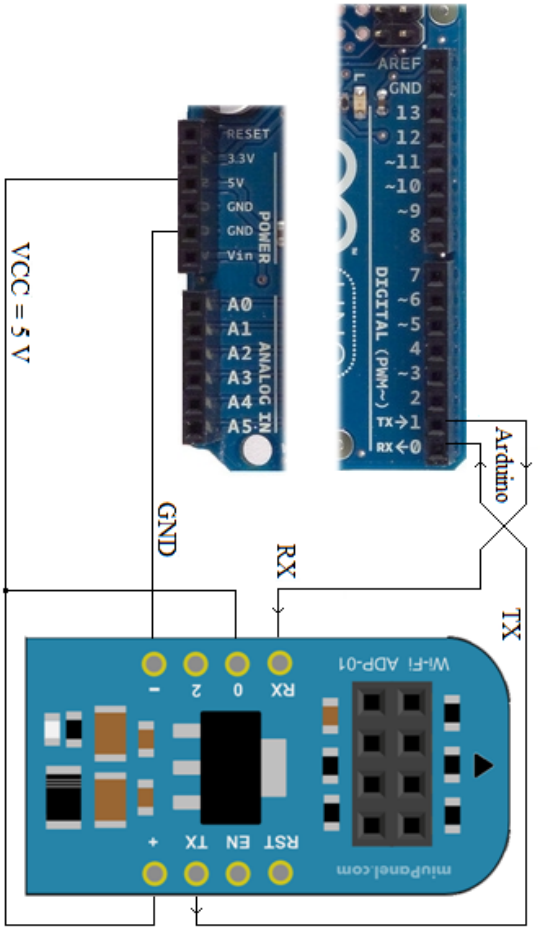



**Fig. 34.** *Connessioni seriali per funzionamento web server.*

Lo sketch per Arduino, riportato nel listato a lato, attende solo comandi contenenti la scritta “pin=11” per rovesciare lo stato logico di uscita del pin 11.

```
/* WiFi_html */
void setup()
{
digitalWrite(11,LOW);pinMode(11,OUTPUT);
Serial.begin(115200); // comunicazione con
ESP8266
Serial.print(“AT+RST\r\n”);
delay(1000); // resetta modulo
Serial.print(“AT+CWMODE=2\r\n”);
delay(500); // configura come access point
Serial.print(“AT+CIPMUX=1\r\n”);
delay(500); // configura per connessioni
multiple
Serial.print(“AT+CIPSERVER=1,80\r\n”);
delay(500); // avvia server sulla porta 80
}

voidloop()
{
if(Serial.available()) // controlla se il
modulo sta comunicando
 {
 if(Serial.find(“+IPD,”)) //Cerca nella
 stringa il testo “+IPD,” inviato premendo
 il bottone nella pagina html
  {
  delay(20);
  int connectionId = Serial.read()-48; //al
  connection id va sottratto 48 (“0” ASCII)
  Serial.find(“pin=”); // adesso cerca “pin=”
  int pinNumber = (Serial.read()-48)*10; //
  il primo numero sono le decine
  pinNumber += (Serial.read()-48); //il se-
  condo numero (unità) va sommato al primo
  digitalWrite(pinNumber,
  !digitalRead(pinNumber)); //rovescia lo
  stato del pin
  // compone il comando di chiusura da in-
  viare al modulo
  String closeCommand = “AT+CIPCLOSE=”;
  closeCommand+=connectionId; // attacca il
  connection id
  closeCommand+=”\r\n”;
  Serial.print(closeCommand);
  delay(20);// attesa invio del comando di
  chiusura
  }
 }
}
```

Una volta caricato il programma (tenendo staccato il TX del modulo), ricollegare TX e aprire la finestra di monitor sul PC. Resettando la scheda, Arduino invia i comandi di configurazione al modulo WiFi (**fig. 35**) e resta in attesa.

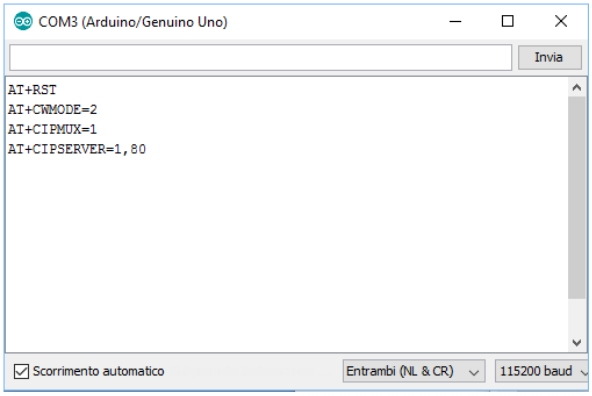



**Fig. 35.** *Comandi che Arduino invia al modulo WiFi.*

Occorre quindi:

- copiare il codice riportato in un documento mediante Notepad o qualsiasi altro text editor e salvarlo in formato testo in una cartella (per es. "*control.txt*");
- rinominare il file, modificandone l'estensione in ".html" (per es. "*control.html*");
- importare nella medesima cartella il file java *jquery.min* (**fig. 36**), copiandolo da http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js

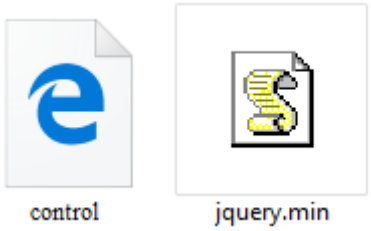



**Fig. 36.** *File .html e .min.*

Il file HTML che, interpretato dal browser, comanda da PC (via WiFi) l'uscita 11 di Arduino è riportato di seguito.

```
<html>
<head>
<title>ESP8266 LED Control</title>
</head>
<body>
<!-- in the <button> tags below the ID attribute is the value sent to the arduino -->
<button id="11" class="led">Toggle Pin 11</button><!-- button for pin 11 -->

<script src="jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
$(".led").click(function(){
var p = $(this).attr('id'); // get id value (i.e. pin13, pin12, or pin11)
// invia la richiesta HTTP GET all'IP address con il parametro "pin" di valore "p"
$.get("http://192.168.4.1:80/", {pin:p});
// execute get request
});
});
</script>

</body>
</html>
```

Il codice HTML utilizzato usa difatti la libreria JavascriptJQuery, richiamata con l'istruzione:

```
<script src="jquery.min.js"></script>
```

A questo punto, dopo aver richiesto e ottenuto dalla rete wireless del PC la connessione con l'access point AI-Thinker, facendo doppio clic sul file *control.html*, compare la finestra di **fig. 37**.

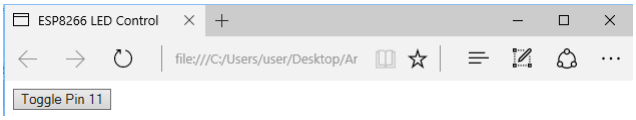



**Fig. 37.** *Pagina web visualizzata.*

Cliccando sul bottone "Toggle Pin 11", il programma invia una GET request al modulo ESP8266, mediante l'istruzione:

```
$.get("http://192.168.4.1:80/", {pin:p});
```

indicandone l'indirizzo IP e il port (80), con il numero 11 all'interno della variabile *p*.

Tramite la seriale, Arduino riceve una stringa di comunicazione del tipo:

```
+IPD,0,345:GET /?pin=11 HTTP/1.1
Host: 192.168.4.1
Connection: keep-alive
Accept:text/html,…
```

Al sopraggiungere di messaggi dalla linea seriale

```
if(Serial.available())
```

Arduino identifica i comandi da elaborare controllando la presenza dello spezzone "+IPD," nel buffer ricevuto, mediante la funzione `Serial.find`

```
if(Serial.find("+IPD,"))
```

Se ciò avviene, il codice legge il carattere successivo, corrispondente all'ID della connessione ("0" nell'esempio riportato), necessario per la corretta chiusura successiva della connessione stessa (richieste simultanee hanno ID differenti).

```
int connectionId = Serial.read()-48;
```

Successivamente il codice scorre la stringa alla ricerca dello spezzone "pin=" e acquisisce il numero del pin interessato, prima le decine e poi le unità:

```
Serial.find("pin=");
int pinNumber = (Serial.read()-48)*10;
pinNumber += (Serial.read()-48);
```

Al termine, rovescia il livello logico del pin indicato:

```
digitalWrite(pinNumber,
!digitalRead(pinNumber));
```

e chiude la connessione (**fig. 38**):

```
String closeCommand = "AT+CIPCLOSE=";
closeCommand+=connectionId; // attacca
il connection id
closeCommand+="\r\n";
Serial.print(closeCommand);
```

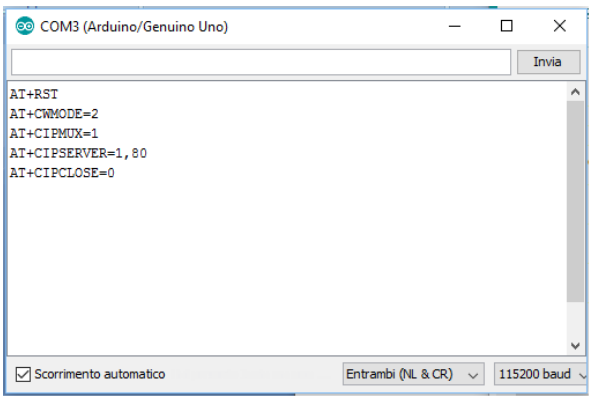



**Fig. 38.** *Invio del comando di chiusura.*

## Possibile continuazione

Modificare lo sketch di Arduino e aggiungere altri due bottoni nella pagina HTML, in modo da controllare anche le uscite 12 e 13.

## Web server WiFi

Un dispositivo web server deve possedere una pagina HTML, da spedire all'indietro al client che chiede la connessione http, con all'interno le possibilità di interazione offerte al client.
Un server web è un dispositivo sempre in attesa di una richiesta da parte di un client, la serve e la chiude ogni volta.
Poiché in questo caso è interposta una connessione WiFi, Arduino deve gestire l'invio della pagina rispettando l'ordine previsto dal protocollo seriale del modulo ESP8266, ovvero deve prima inviare il comando:

```
AT+CIPSEND=c,n
```

con "c" l'identificativo del canale aperto e "n" il numero dei caratteri della pagina da trasferire, e quando appare il simbolo ">" (o con opportuno ritardo) inviare la pagina stessa.
Nello sketch che segue, l'invio di una pagina è delegato alla funzione **Send_webPage ()**, utilizzando il "**connectionId**" letto all'interno del messaggio ricevuto dal browser subito dopo la scritta "**+IPD,**".

```
/* Arduino WiFi server web */
int connectionId;
StringwebPage;
voidsetup()
{
Serial.begin(115200); // comunicazione con
ESP8266
Serial.print("AT+RST\r\n");
delay(1000); // resetta modulo
Serial.print("AT+CWMODE=2\r\n");
delay(500); // configura come access point
Serial.print("AT+CIPMUX=1\r\n");
delay(500); // configura per connessioni
multiple
Serial.print("AT+CIPSERVER=1,80\r\n");
delay(500); // avvia server sulla porta 80
}

voidloop()
{
if(Serial.available()) //controlla se il
modulo sta comunicando
 {
```

```
 if(Serial.find("+IPD,"))
  {
  delay(20);
  connectionId = Serial.read()-48;
  webPage = "<h1>Hello</h1><h2>World!</
  h2><button>LED1</button>";
  Send_webPage ();
  webPage="<button>LED2</button>";
  Send_webPage ();
  Send_closeCommand();
  }
 }
}

void Send_webPage(){
String sendCommand = "AT+CIPSEND=";
sendCommand += connectionId;
sendCommand += ",";
sendCommand +=webPage.length();
sendCommand +="\r\n";
Serial.print(sendCommand);
delay(500);
Serial.print(webPage);
delay(500);
}
void Send_closeCommand(){
String closeCommand = "AT+CIPCLOSE=";
closeCommand+=connectionId;
closeCommand+="\r\n";
Serial.print(closeCommand);
delay(500);
}
```

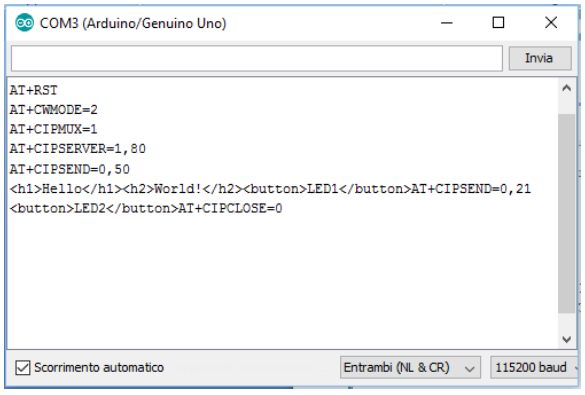



**Fig. 39.** *Comandi inviati da Arduino.*

Una volta caricato il programma (tenendo staccato il TX del modulo), ricollegare TX e aprire la finestra di monitor sul PC. Resettando la scheda, Arduino invia i comandi di configurazione al modulo WiFi e resta in attesa.
A questo punto, attivare la connessione WiFi tra il PC e AI-Thinker e aprire un browser all'indirizzo 192.168.4.1.

Sul monitor seriale compare la sequenza di invio della pagina di risposta (**fig. 39**) e sul browser la sua interpretazione grafica (**fig. 40**).

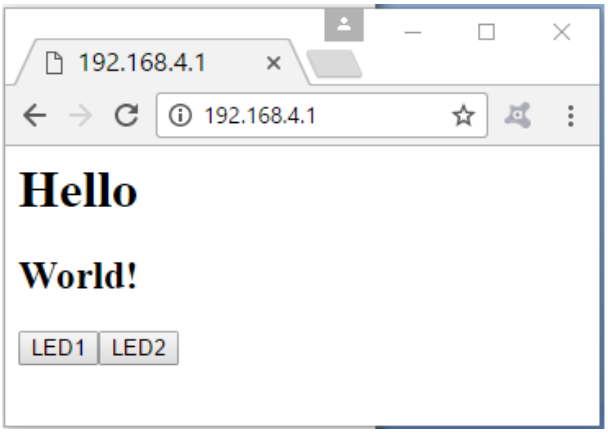



**Fig. 40.** *Pagina visualizzata dal browser.*

## Pagina web WiFi interattiva

Per ottenere una connessione che permetta il controllo del processo su Arduino, i bottoni vanno resi capaci di scatenare una richiesta Get e il browser del client (PC) va istruito in modo da inviare una richiesta http con periodicità fissa e in modo automatico, per l'aggiornamento in tempo reale dello stato degli ingressi residenti sul server (Arduino), includendo nella pagina HTML l'istruzione **Refresh**.

Il programma che segue, per esempio, gestisce le richieste provenienti da due bottoni per il governo di un LED posto sul pin 4 di Arduino (**fig. 41**), e aggiorna ogni 5 secondi (Refresh: 5) il client sullo stato dell'ingresso digitale 2 e dell'ingresso analogico A0.

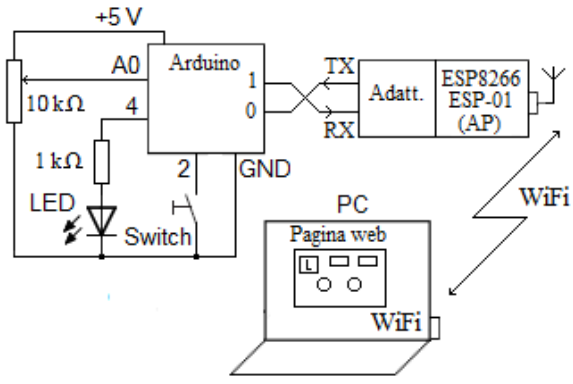



**Fig. 41.** *Web server WiFi.*

```
/* Arduino WiFi server web_2 */
int connectionId, i;
String webPage;
char str[100];
char* StatoLed4;

voidsetup()
{
Serial.begin(115200); // comunicazione con
ESP8266
Serial.print("AT+RST\r\n");
delay(1000); // resetta modulo
Serial.print("AT+CWMODE=2\r\n");
delay(500); // configura come access point
Serial.print("AT+CIPMUX=1\r\n");
delay(500); // configura per connessioni
multiple
Serial.print("AT+CIPSERVER=1,80\r\n");
delay(500); // avvia server sulla porta 80
pinMode(2, INPUT_PULLUP); //pin 2 input con
pull-up
  //digitalWrite(2, HIGH);
pinMode(4, OUTPUT);
digitalWrite(4, LOW);
}

voidloop()
{
if(Serial.available()) //controlla se il
modulo sta comunicando
 {
 if(Serial.find("+IPD,"))
  {
  delay(20);
  connectionId = Serial.read()-48;
   if(Serial.find("GET /?")) {
    for(i=0;i<7;i++)str[i]=Serial.read();
    str[i]=0;
     if(strstr(str,"pin4=on"))digi-
     talWrite(4, HIGH);
     else if(strstr(str,"pin4=of"))digi-
     talWrite(4, LOW);
     }
   if(digitalRead(4))StatoLed4="on";
   else StatoLed4="off";
   homePage();// prepara la pagina
   Send_webPage ();
   Send_closeCommand();
   }
  }
}
```

```
void Send_webPage(){
String sendCommand = "AT+CIPSEND=";
sendCommand += connectionId;
sendCommand += ",";
sendCommand +=webPage.length();
sendCommand +="\r\n";
Serial.print(sendCommand);
  while (Serial.read ()!= '>');
Serial.print(webPage);
delay(200);
}

void Send_closeCommand(){
String closeCommand = "AT+CIPCLOSE=";
closeCommand+=connectionId;
closeCommand+="\r\n";
Serial.print(closeCommand);
delay(300);
}

static word homePage() {
long t = millis() / 1000;
word h = t / 3600;
byte m = (t / 60) % 60;
byte s = t % 60;
webPage="HTTP/1.0 200 OK\r\n"
"Content-Type: text/html\r\nPragma: no-
cache\r\nRefresh: 5\r\n\r\n"
"<html><head><title>ENC28J60</title></head>"
"<body>"
"<div style='text-align:center;'>"
"<h1>ENC28J60 web server</h1>"
"<h4>Durata della connessione :";
sprintf(str,"%d%d:%d%d:%d%d</h4>",h/10,
h%10, m/10, m%10, s/10, s%10);
webPage.concat(str);
sprintf(str,"<br />Input 2 = %d<br />",digi-
talRead(2));
webPage.concat(str);
sprintf(str,"<br /><br />Stato del LED4:
%s<br />",StatoLed4);
webPage.concat(str);
webPage+="<a  href='/?pin4=on'><input
type='button' value='OUT 4 ON'></a>"
"<a href='/?pin4=off'><input type='button'
value='OUT 4 OFF'></a>"
"<br /><br />";
sprintf(str,"<br /><br />Potenziometro: %d
(su 1024)",analogRead(0));
webPage.concat(str);
webPage+="<br /><br /></body></html>";
}
```

Una volta caricato lo sketch, aprendo la connessione WiFi e attivando il browser web all'indirizzo 192.168.4.1, comparirà la schermata riportata in **fig. 42**, con il tempo di connessione, lo stato logico dell'ingresso digitale, la possibilità di modificare lo stato del LED e il livello rilevato su A0.

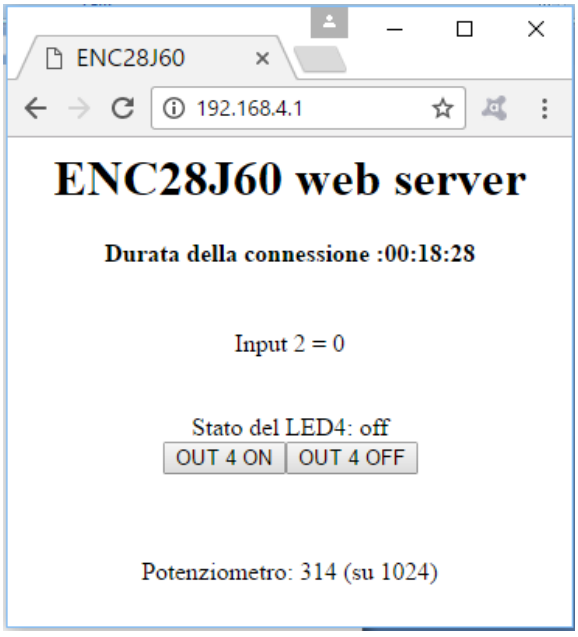



**Fig. 42.** *Pagina visualizzata.*

Ogni 5 secondi o premendo uno dei pulsanti, il PC attiva una richiesta di connessione e invia un comando. Se nella stringa inviata dal client (**fig. 43**) lo sketch rileva lo spezzone "+IPD,", salva il numero della connessione, gestisce l'eventuale richiesta GET/? (se presente), invia una pagina HTML con i valori aggiornati e chiude la comunicazione.

```
+IPD,1,437:GET /?pin4=off HTTP/1.1
Host: 192.168.4.1 …
```

**Fig. 43.** *Esempio parziale di stringa ricevuta da Arduino.*

## Possibile continuazione

Aggiungere un ulteriore switch attivo basso sul pin 3, un ulteriore LED sul pin 5 e modificare il programma in modo da renderli entrambi interattivi come già fatto per i precedenti.