ISTITUTO TECNICO INDUSTRIALE STATALE
G.GALILEI DI SAN SECONDO

SISTEMI EMBEDDED
Il caso Arduino

"se ascolto dimentico, se vedo ricordo, se faccio capisco"

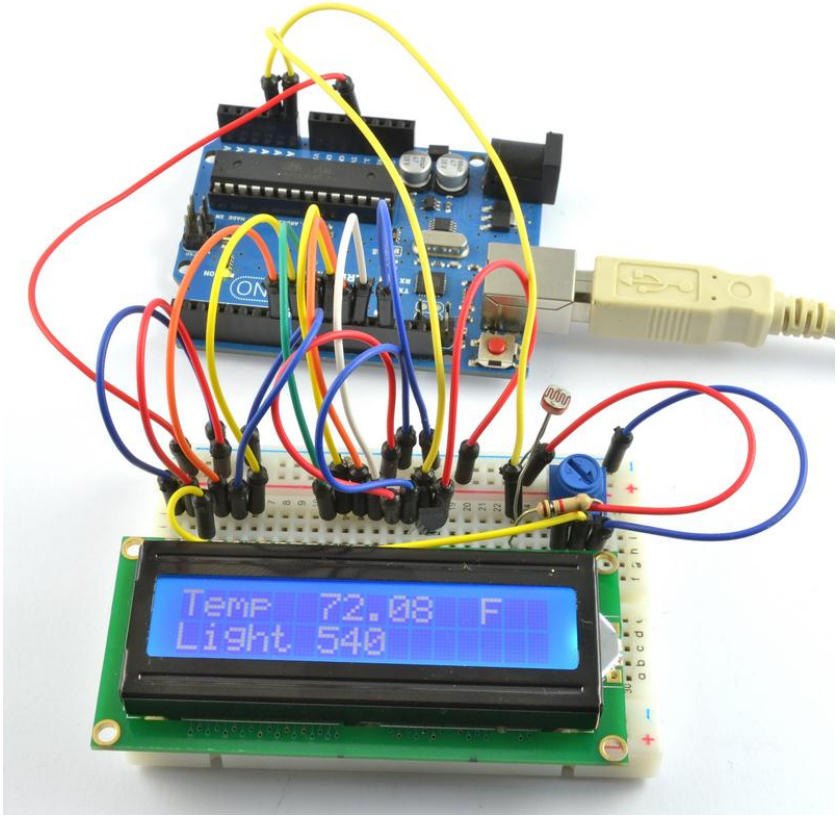
INDICE

- LCD
- Sistemi di comunicazione
  - Seriale
  - Ethernet
  - WiFi
  - Zigbee

LCD

Come utilizzare uno schermo LCD

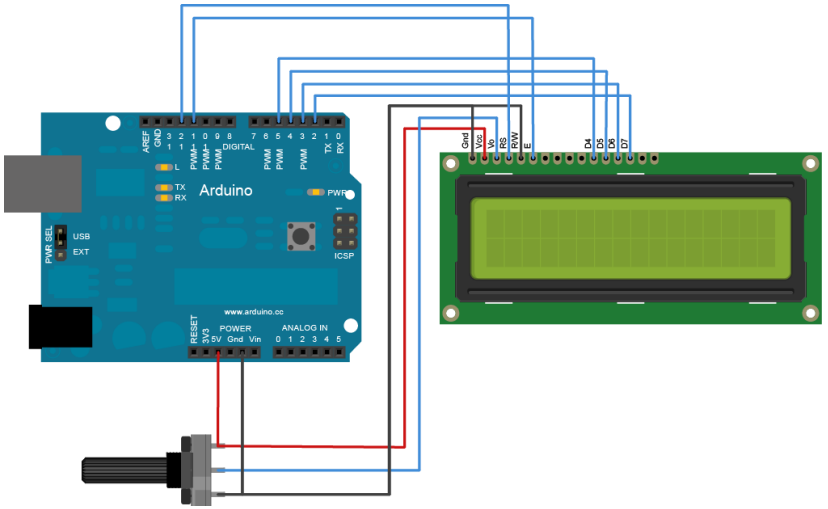
Si potrebbe pensare di utilizzare un display LCD per visualizzare dati provenienti da sensori collegati alla scheda Arduino (per esempio la temperatura o l'intensità luminosa).



LCD

Come utilizzare uno schermo LCD

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of
the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // set up LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis()/1000);
}
```

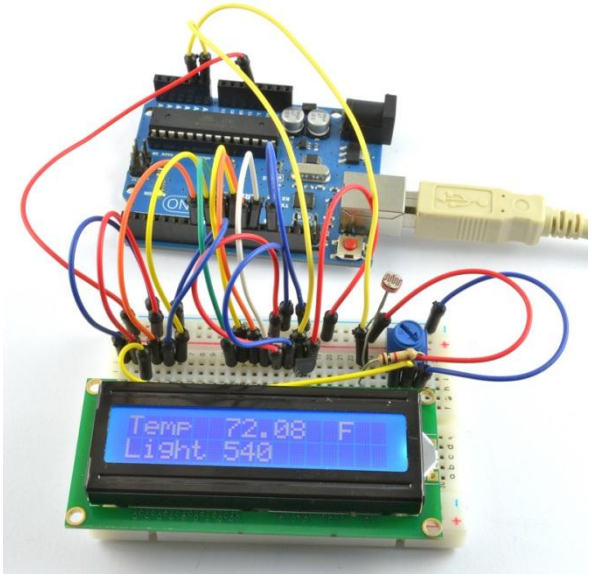


->LCD->HelloWorld

LCD

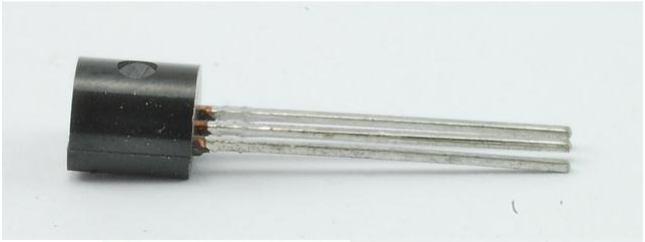
ESEMPIO

Il sensore di luminosità è una semplice fotoresistenza, mentre il sensore di temperatura è un sensore a tre pin, due per l'alimentazione (5V e GND), mentre l'altro si connette direttamente ad una porta analogica di Arduino.



Oggetti	Qt.
Display LCD	1
Resistenza variabile 10kΩ	1
Resistenza 1kΩ	1
Fotoresistenza	1
Sensore di temperatura TMP36	1

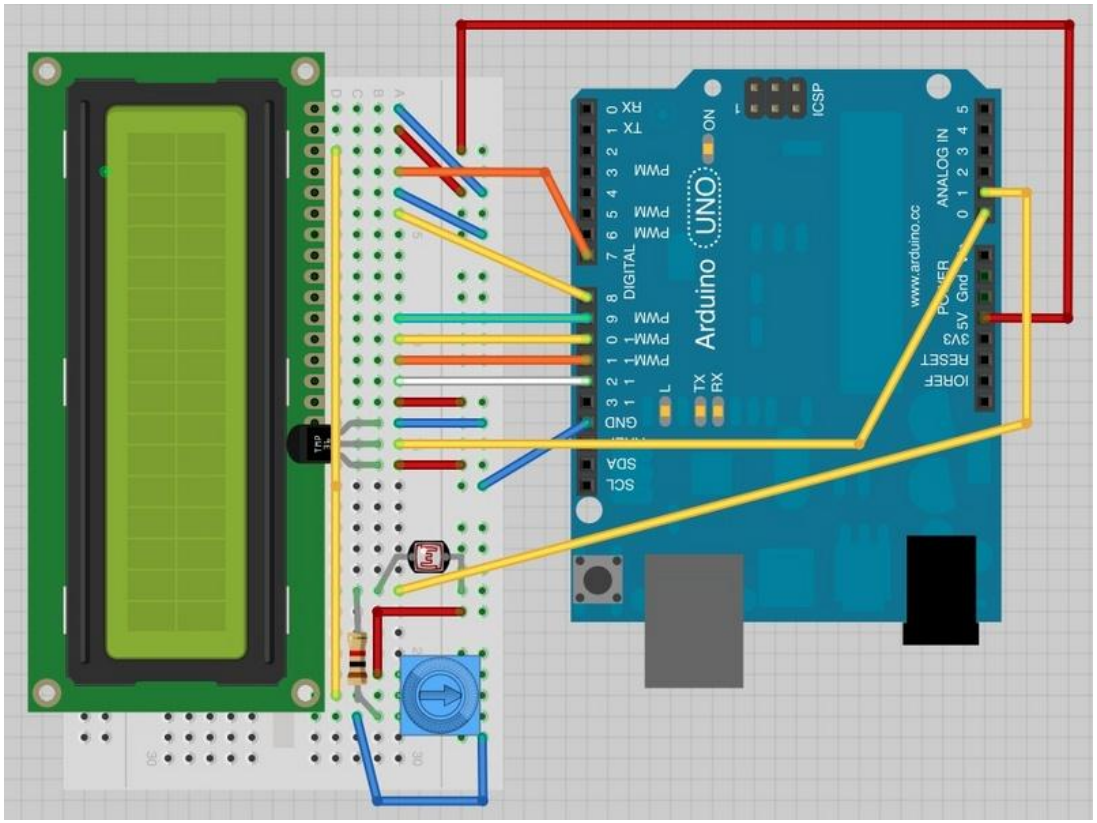
Come collegare il display ad Arduino? (link)



LCD

ESEMPIO

```
//Definisco i pin del LCD
// (BS E D4 D5 D6 D7)
LiquidCrystal
   lcd(7, 8, 9, 10, 11, 12);

//Acquisico i dati dai sensori
int tempReading =
   analogRead(tempPin);
float tempVolts =
   tempReading * 5.0 / 1024.0;
float tempC =
   (tempVolts - 0.5) * 100.0;
float tempF =
   tempC * 9.0 / 5.0 + 32.0;
```



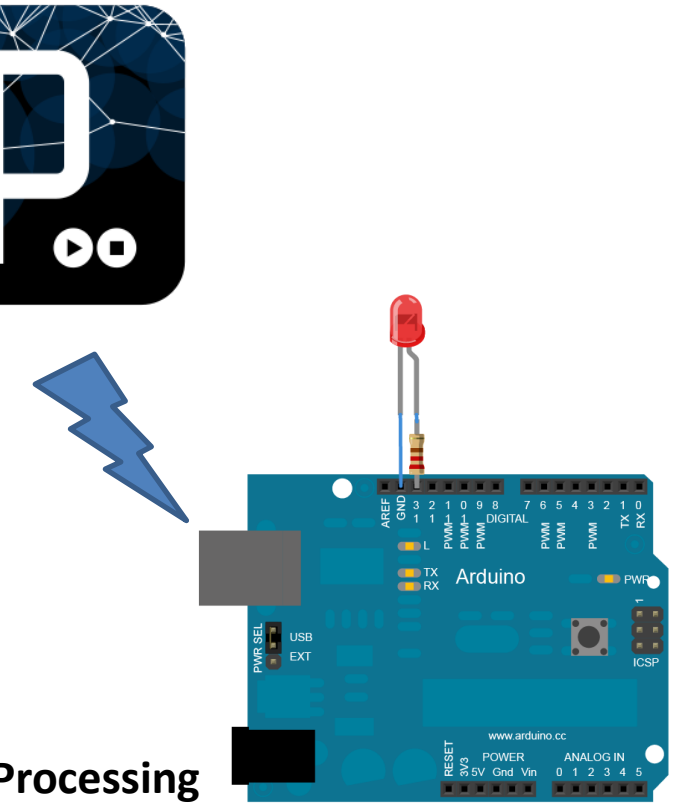
-> LCD->TemperatureLight

COMUNICAZIONE SERIALE – Arduino-to-PC

Questo esempio vuole mostrare come utilizzare Arduino per ricevere dati da un computer. Utilizziamo il circuito dell'esempio ->Blink e modifichiamo il codice come segue:

- *Se ricevo il carattere 'H' accendo il led*
- *Se ricevo il carattere 'L' spengo il led*

I dati comunicano sulla seriale e quindi possono essere gestiti da qualsiasi programma in grado di interfacciarsi con tale protocollo. Nel nostro caso usere Processing.



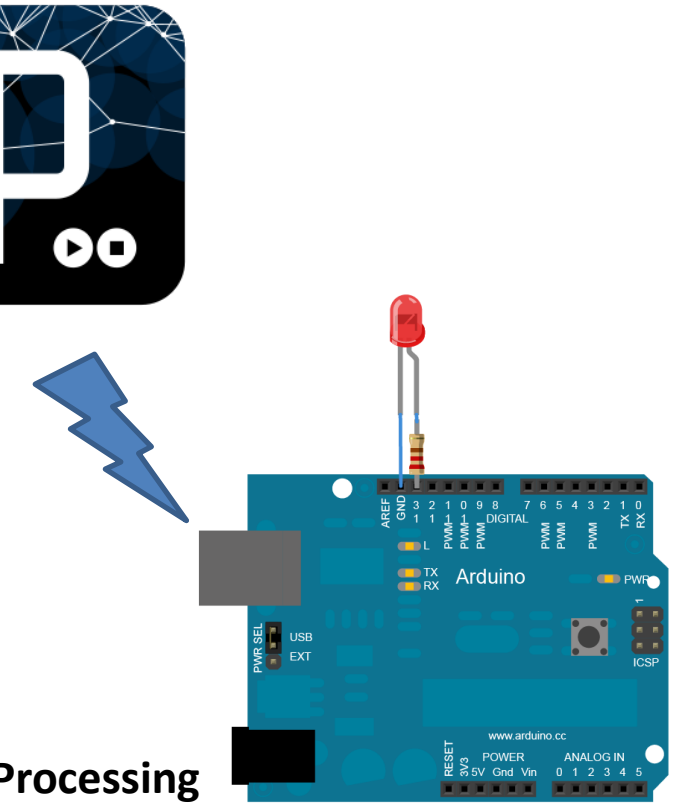
->Serial->ArduinoAndProcessing

COMUNICAZIONE SERIALE – Arduino-to-PC

Questo esempio vuole mostrare come utilizzare Arduino per ricevere dati da un computer. Utilizziamo il circuito dell'esempio ->Blink e modifichiamo il codice come segue:

- *Se ricevo il carattere 'H' accendo il led*
- *Se ricevo il carattere 'L' spengo il led*

I dati comunicano sulla seriale e quindi possono essere gestiti da qualsiasi programma in grado di interfacciarsi con tale protocollo. Nel nostro caso usere Processing.

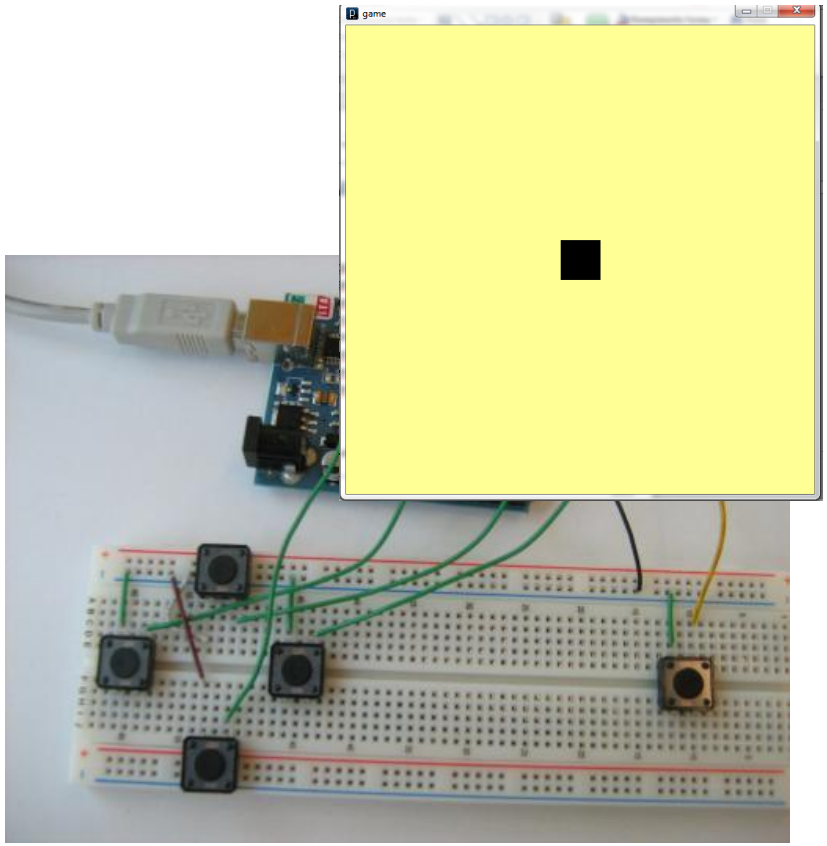


->Serial->ArduinoAndProcessing

COMUNICAZIONE SERIALE – Arduino-to-PC

ESERCIZIO

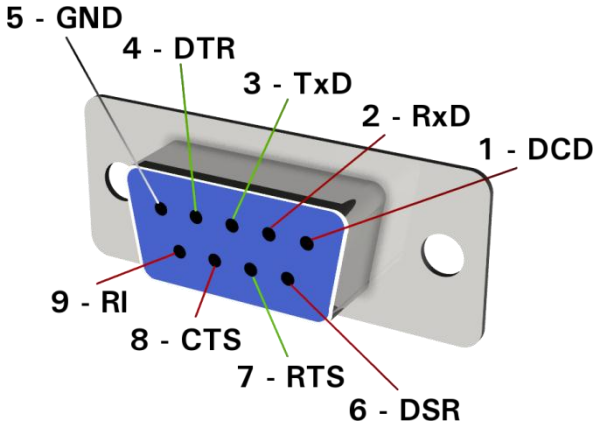
Fare un piccolo gioco di cui si dispone il “gamepad” fatto in Arduino. Si hanno a disposizione 4 bottoni che simulano le freccie direzionali, mentre un altro bottone simulerà un tasto per sparare. Con processing mostrare tali movimenti con un quadrato che si muove sullo schermo e quando si spara cambiare il colore del quadrato.



COMUNICAZIONE SERIALE – Arduino-to-Arduino

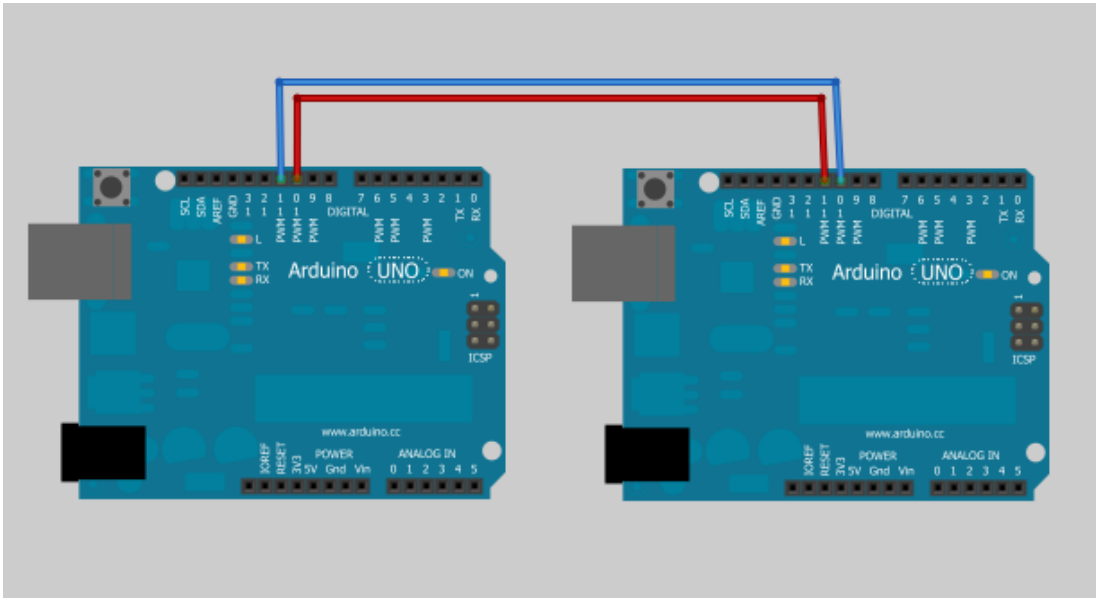
SoftwareSerial

Arduino dispone di pin digitali che possono essere utilizzati per simulare un ''Seriale Software'', cioè adibire un pin a ricevere dati e l'altro a trasmettere.



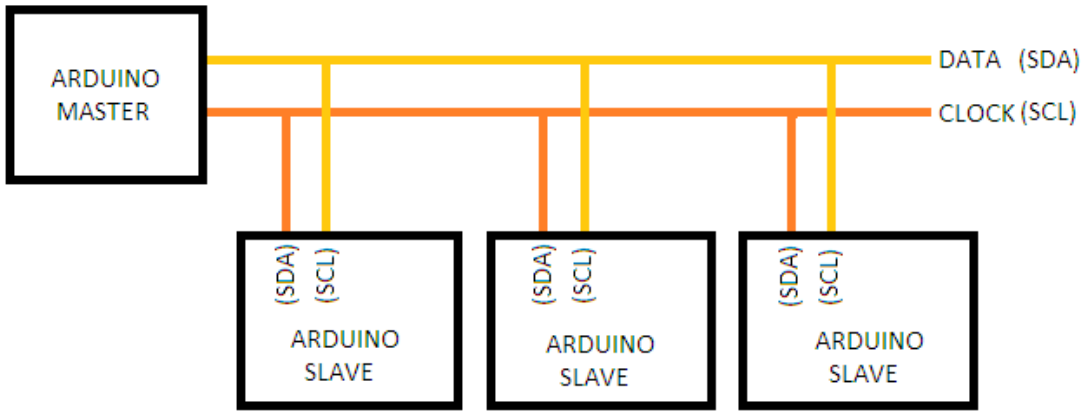
ATTENZIONE: la RS232 non funziona con segnali TTL. Ho bisogno di convertire il segnale

->Serial->SoftwareSerial

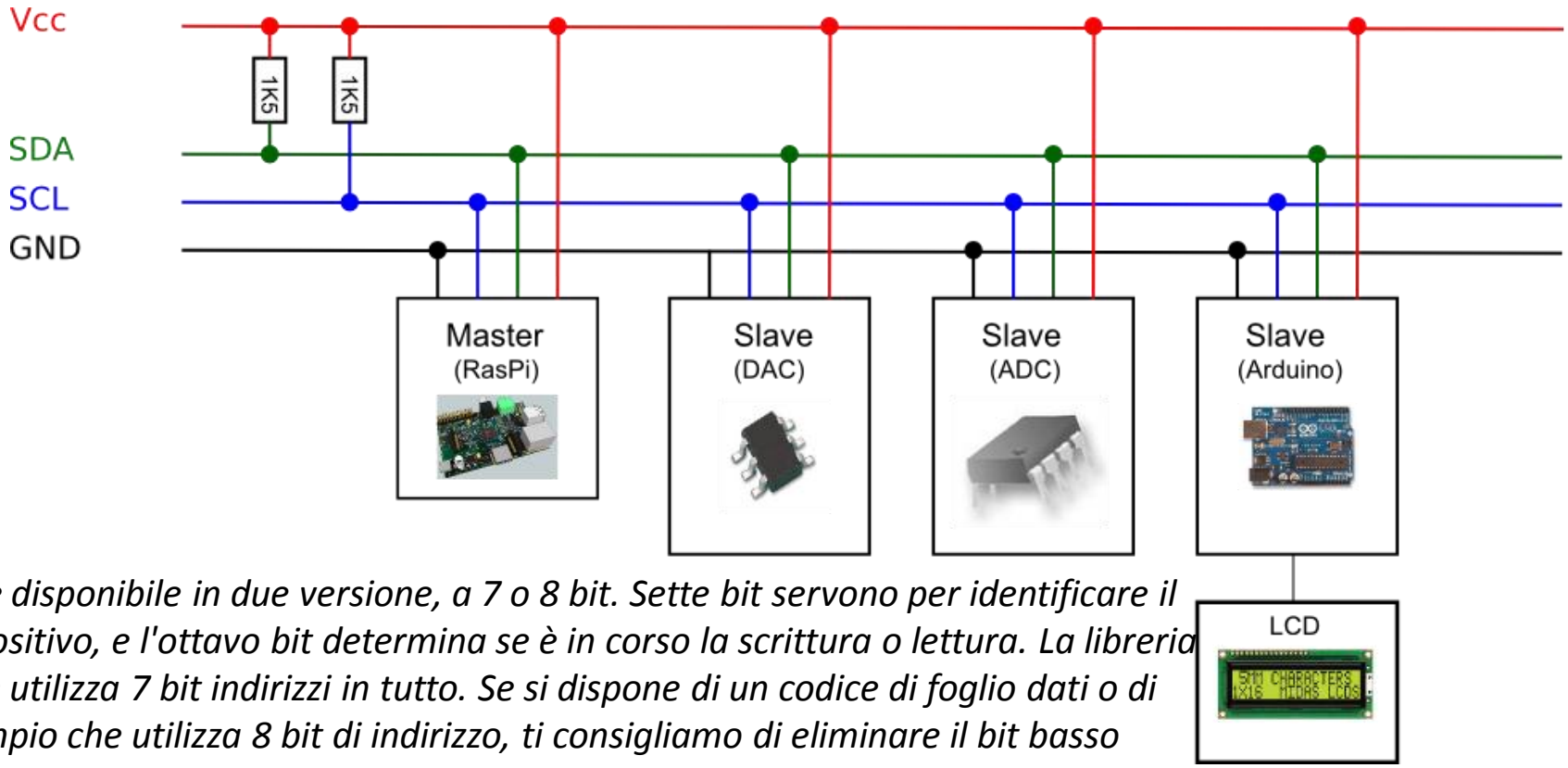


COMUNICAZIONE SERIALE – I2C

Esistono alcuni metodi che permettono di comunicare con altri dispositivi utilizzando come canale di trasmissione dati i protocolli seriali (I2C, SPI o RS232). Questo rende possibile demandare ad una scheda slave funzioni di elaborazione gravose, che non vogliamo far eseguire alla scheda master. Questo esempio usa il bus I2C per creare uno scambio dati tra un Arduino ed un chip ATmega328 standalone.



COMUNICAZIONE SERIALE – I2C



I2C è disponibile in due versione, a 7 o 8 bit. Sette bit servono per identificare il dispositivo, e l'ottavo bit determina se è in corso la scrittura o lettura. La libreria Wire utilizza 7 bit indirizzi in tutto. Se si dispone di un codice di foglio dati o di esempio che utilizza 8 bit di indirizzo, ti consigliamo di eliminare il bit basso (cioè spostare il valore di un bit a destra), ottenendo un indirizzo compreso tra 0 e 127. Il numero di periferiche I2C su uno stesso bus di solito dipende dalle capacità parassite introdotte sulla linea dai vari dispositivi, il limite è 400pF. (link)

COMUNICAZIONE SERIALE – I2C

Il bus I2C, basandosi su due fili, non permette la comunicazione contemporanea tra Master e Slave. Lo scambio dati deve essere gestito dal Master tramite gli indirizzi (univoci) degli slave. Il flusso può essere sintetizzato in questo modo

1. Il Master invia sul bus un bit di start
2. Il Master invia sul bus l'indirizzo dello slave con cui vuole comunicare
3. Il Master decide se scrivere o leggere dal dispositivo
4. Lo Slave legge o scrive in base alla richiesta del Master

La libreria Wire dispone di tutte le funzioni necessarie alla realizzazione Master-Slave tra due schede Arduino.

->I2C->Arduino_ATMEGA

->I2C->Arduino_Processing

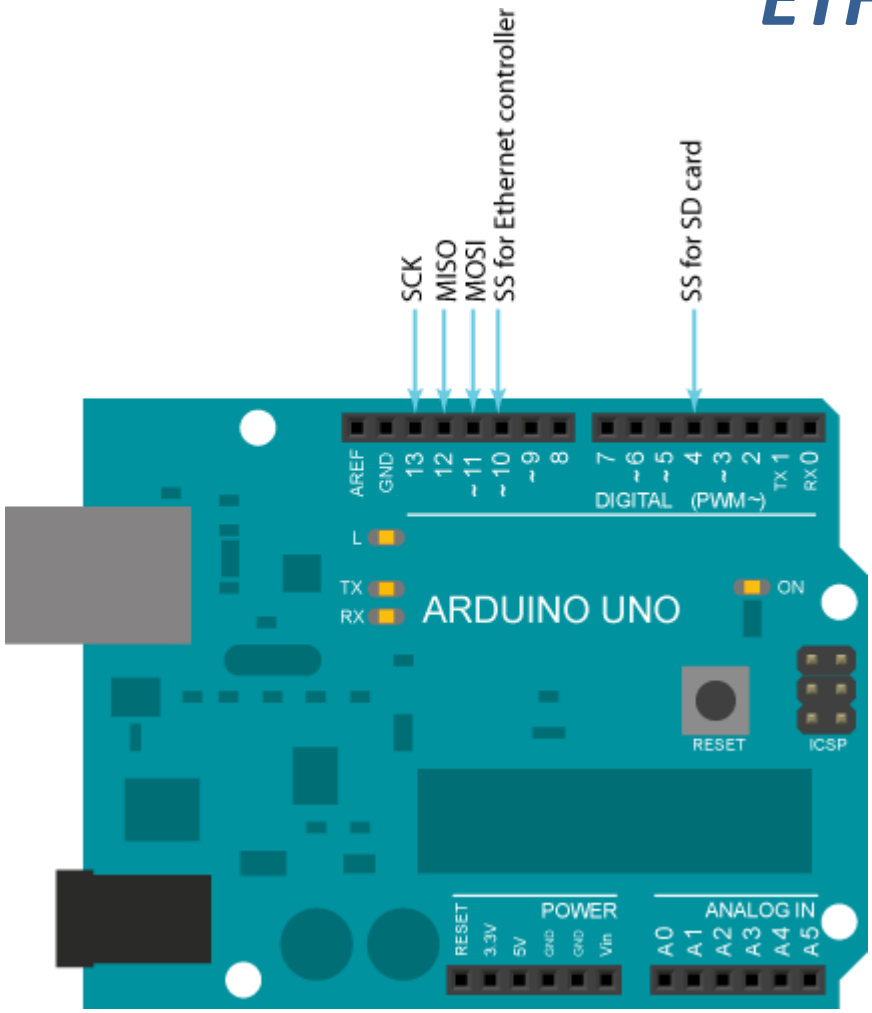
Board	I2C / TWI pins
Uno, Ethernet	A4 (SDA), A5 (SCL)
Mega2560	20 (SDA), 21 (SCL)
Leonardo	2 (SDA), 3 (SCL)
Due	20 (SDA), 21 (SCL), SDA1, SCL1

ETHERNET

Con Arduino è possibile collegare i nostri dispositivi in una rete attraverso i principali protocolli di comunicazione. Esempio utilizzando l'ethernet. La shield Ethernet permette di collegare il nostro Arduino ad un router o direttamente un pc e inviare/ricevere dati attraverso l'uso della libreria Ethernet.

Per collegare Arduino ad un router o un hub basta collegare un cavo Ethernet standard (CAT5 o CAT6), mentre per il collegamento a PC potrebbe servire un cavo incrociato

ETHERNET



Arduino comunica con la shield attraverso il bus SPI. Tale canale è sui pin 11, 12 e 13 sull'Arduino UNO mentre sull'Arduino MEGA è sui pin 50, 51 e 52. **Su entrambe le schede il pin 10 è usato per il controller SS.**

Disponendo anche di uno slot per SD, la shield Ethernet riserva il pin 4 per il controller della SD.

N.B. utilizzando la shield Ethernet su un Arduino UNO si riducono considerevolmente i pin di I/O utilizzabili.

ETHERNET

Alla shield deve essere assegnato un indirizzo MAC e un indirizzo IP con l'istruzione

```
Ethernet.begin();
```

L'indirizzo MAC è un identificatore univoco globale per un particolare dispositivo collegato in rete. Le shield Ethernet attuali sono dotate di un adesivo che indica l'indirizzo MAC del dispositivo.

Indirizzi IP validi dipendono dalla configurazione della rete. E 'possibile utilizzare il protocollo DHCP per assegnare dinamicamente un indirizzo IP alla shield. In alternativa, è anche possibile specificare un gateway di rete e la sottorete.

```
Ethernet.begin(mac, ip, dns, gateway, subnet);
```

ETHERNET - Server

```
#include <Ethernet.h>
#include <SPI.h>

// network configuration.  gateway and subnet are optional.
// the media access control (ethernet hardware) address for the shield:
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
// the router's gateway address:
byte gateway[] = { 192, 168, 0, 1 };
// the subnet:
byte subnet[] = { 255, 255, 255, 0 };

EthernetServer server = EthernetServer(23);

//the IP address is dependent on your network
IPAddress ip(192,168,0,5);

void setup(){
  // initialize the ethernet device
  Ethernet.begin(mac, ip, gateway, subnet);

  // start listening for clients
  server.begin();
}
```

```
void loop(){
  //TODO something!!
  //print out the IP address
  Serial.println(myIPaddress);
}
```

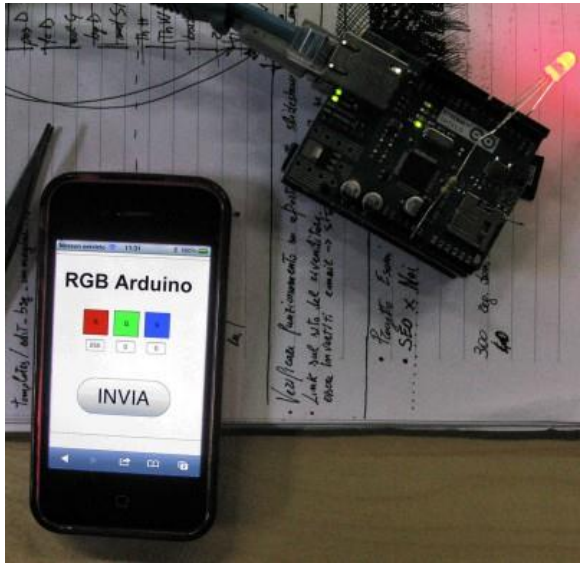
ETHERNET - Client

```
#include <Ethernet.h>
#include <SPI.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
byte ip[] = { 192, 168, 0, 6 };
byte server[] = { 192, 168, 0, 5 };

EthernetClient client;

void setup(){
  Ethernet.begin(mac, ip);
  Serial.begin(9600);

  delay(1000);

  Serial.println("connecting...");

  // it's connect to Arduino Server on port 8080
  if (client.connect(server, 8080)) {
    Serial.println("connected");
    client.println("GET / HTTP/1.0");
    client.println();
  } else {
    Serial.println("connection failed");
  }
}
```

```
void loop(){
  //If I get from the server
  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }

  //If client disconnected
  if (!client.connected()) {
    Serial.println();
    Serial.println("disconnecting.");
    client.stop();
    for(;;)
      ;
  }
}
```

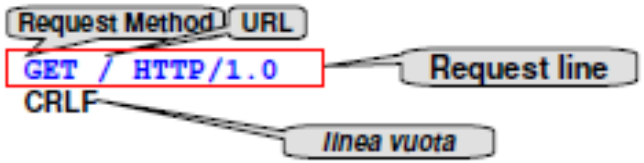
ETHERNET

->Ethernet->ServerWeb
->Ethernet->SchedaAttuatori
->Ethernet->LedRGBWeb



IL PROTOCOLLO HTTP

REQUEST MESSAGE:



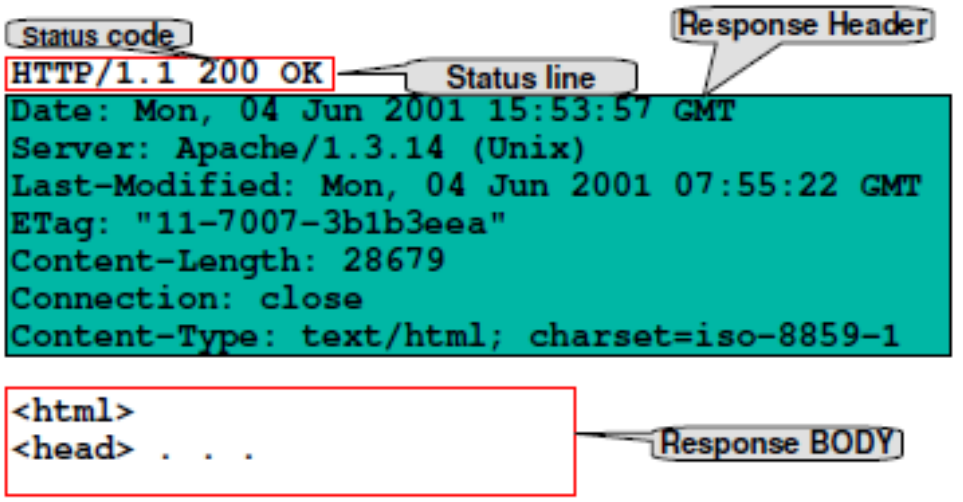
La *request line* è seguita immediatamente da linea vuota: questo messaggio non ha *header* (accettabile in 1.0);
Il metodo GET non prevede *Entity Body*, quindi il messaggio è terminato.

- Messaggio tipo **Request(CLIENT)**
  - Request Line (termina con CRLF)
  - [Header]
  - Linea vuota (CRLF)
  - [Entity Body] (stream di ottetti)

- Messaggio tipo **Response(SERVER)**
  - Status Line (termina con CRLF)
  - [Header]
  - Linea vuota (CRLF)
  - [Entity Body]

- Request Line / Status Line c'è sempre.
- Gli Header possono mancare (tranne Host in HTTP 1.1!).
- La linea vuota c'è sempre (separa Header e Body).
- Entity Body può mancare

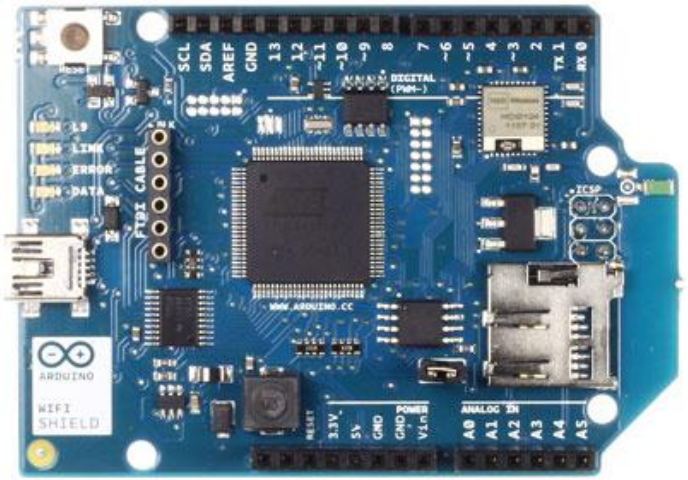
RESPONSE MESSAGE:

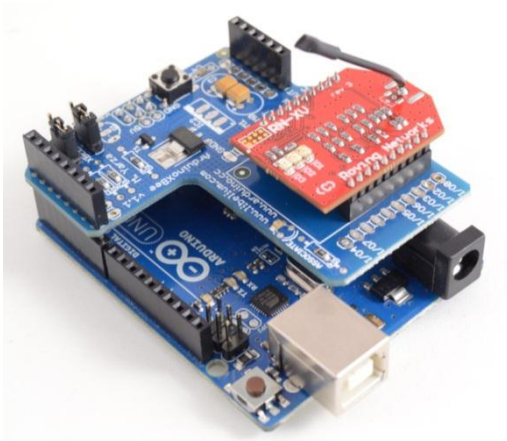


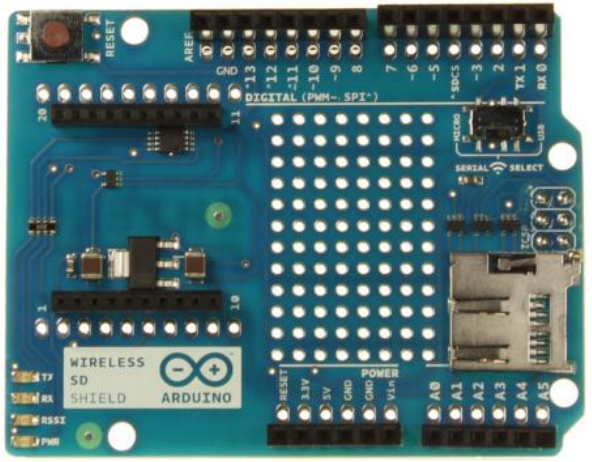
WIFI

Con l'Ethernet si è vincolati ad avere dei cavi per collegare i diversi dispositivi tra loro.
Una soluzione wireless per Arduino è possibile attraverso diverse soluzioni.

- Wireless SD Shield + modulo WiFi
- Shield con socket Xbee + modulo WiFi
- WiFi Shield

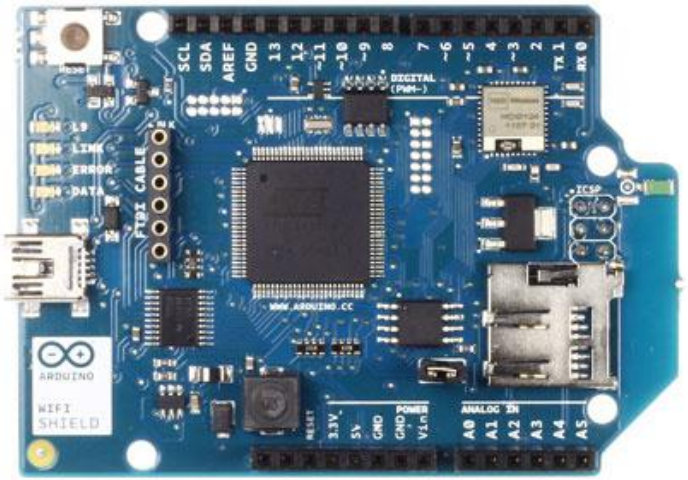


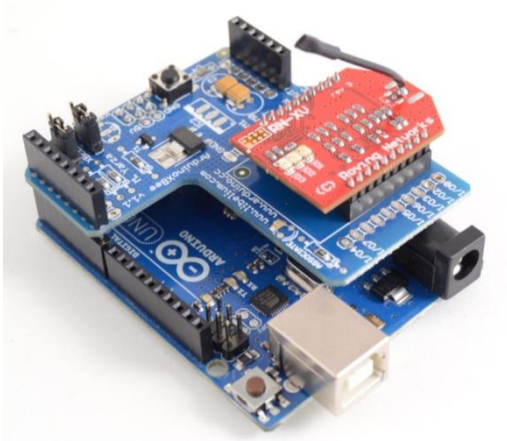


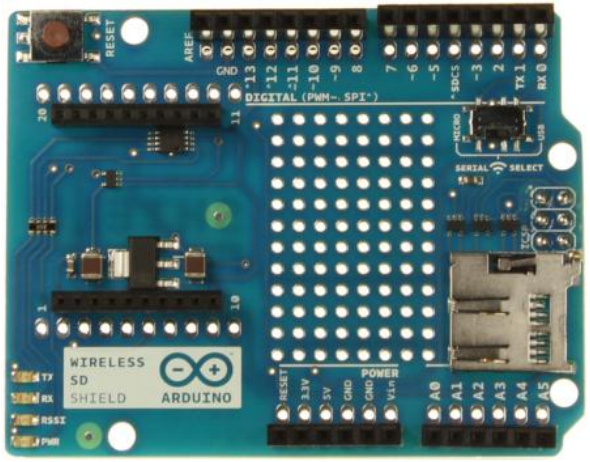


WIFI

Le prime due soluzioni sono le più versatili perché si possono sostituire i moduli WiFi nel caso di upgrade o rottura. Mentre la seconda è vincolata ad una shield e quindi in caso di danno o aggiornamto bisogna sostituire tutta la shield.

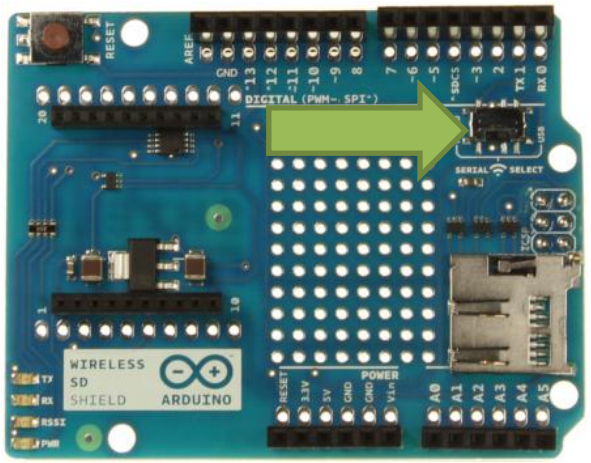






WIRELESS SD SHIELD

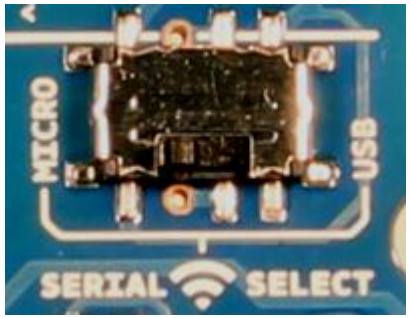
La wireless SD shield permette ad una scheda Arduino di comunicare senza fili con un modulo wifi. Si basa sui moduli Xbee Digi, ma può utilizzare qualsiasi modulo con lo stesso socket. Il modulo può comunicare fino a 100 metri al chiuso, all'aperto, o 300 metri.



Può essere usata come un seriale usb o lo si può mettere in una modalità di comando e configurarlo per una varietà di opzioni di rete di trasmissione e rete attraverso uno switch.
I
ncluso a bordo è presente uno slot per schede SD come per la shield Ethernet. Quando si utilizza la libreria SD per accedere alla scheda, Pin 4 è CS e non può essere utilizzata diversamente. SPI si basa anche in questo caso sui pin 11, 12, e 13 per la comunicazione.

WIRELESS SD SHIELD

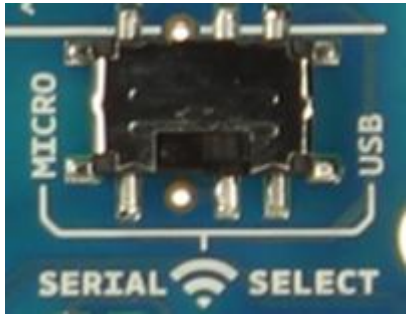
La shield ha un bordo interruttore con l'etichetta ''Serial Select''. Determina come la comunicazione seriale del modulo WiFi si collega alla comunicazione seriale tra il microcontrollore (ATmega8 o ATmega168) e USB-seriale chip sulla scheda Arduino.



Quando è nella posizione **Micro**, il pin DOUT del modulo wireless è connesso al pin RX del microcontrollore e DIN è collegato TX. Il modulo wireless comunicherà direttamente con il microcontrollore. Si noti che i pin RX e TX del microcontrollore sono ancora collegato ai pin TX e RX (rispettivamente) del USB-seriale convertitore. I dati inviati dal microcontrollore saranno trasmessi al computer tramite USB oltre ad essere inviati in modalità wireless dal modulo wireless. Il microcontrollore non sarà programmabile tramite USB in questa modalità.

WIRELESS SD SHIELD

Con l'interruttore in posizione **USB**, il pin DOUT del modulo wireless è connesso al pin RX del convertitore USB-seriale, e DIN sul modulo wireless è collegato al pin TX. Ciò significa che il modulo può comunicare direttamente con il computer. Il microcontrollore sulla scheda verrà bypassato.



Per utilizzare la shield in questa modalità, è necessario programmare il microcontrollore con uno sketch vuoto (vedi sotto), o rimuovere il chip dalla board.

Sketch vuoto:

```
void setup() {}
void loop() {}
```

Modulo WiFly

Un modulo utilizzabile con questa shield è il modulo **Roving Networks WiFly RN-XV**. Con questa configurazione è possibile creare un piccolo http client dal quale inviare i diversi dati al server. Il modulo wifi è già pronto all'uso (non ha bisogno di configurazioni particolari, salvo per necessità diverse).
Per sfruttarne le sue potenzialità si usano le librerie WiflyHQ.
Sono librerie pronte all'uso per Arduino e fornisco diversi esempi per creare una comunicazione sia HTTP che TCP/UDP.

Modulo WiFly

Configurazione e uso di una seriale software per interfacciarsi al dispositivo:

```
#include <SoftwareSerial.h>
SoftwareSerial wifiSerial(8,9);

wifiSerial.begin(9600);
wifly.begin(&wifiSerial);
```

Collegarsi ad una rete WiFi;

```
wifly.setSSID("mySSID");
wifly.setPassphrase("myWPApassword");
wifly.enableDHCP();
wifly.join();
```

Modulo WiFly

Creare una rete WiFi Ad Hoc con 10 canali:

```
wifly.createAdhocNetwork("myssid", 10);
```

Mandare pacchetti UDP:

```
wifly.setIpProtocol(WIFLY_PROTOCOL_UDP);
wifly.sendto("Hello, world", "192.168.1.100", 2042);
```

Aprire una connessione TCP, mandare alcuni dati, e chiudere la connessione:

```
wifly.setIpProtocol(WIFLY_PROTOCOL_TCP);
wifly.open("192.168.1.100",8042);
wifly.println("Hello, world!");
wifly.close();
```

Ricevere dati UDP e TCP (assumendo di avere una seriale software):

```
if (wifly.available() > 0) {
    Serial.write(wifly.read());
}
```

Modulo WiFly

Facile manipolazione di diverse opzioni ricevute con il metodo multiMatch_p()::

```
if (wifly.available() > 0) {
          int match = wifly.multiMatch_P(100, 3,
                              F("button"), F("slider="), F("switch="));
          switch (match) {
          case 0: /* button */
                Serial.print(F("button: pressed"));
                break;
          case 1: /* slider */
                int slider = wifly.parseInt();
                Serial.print(F("slider: "));
                Serial.println(slider);
                break;
          case 2: /* switch */
                char ch = wifly.read();
                Serial.print(F("switch: "));
                Serial.println(ch);
                break;
          default: /* timeout */
                break;
          }
      }
```

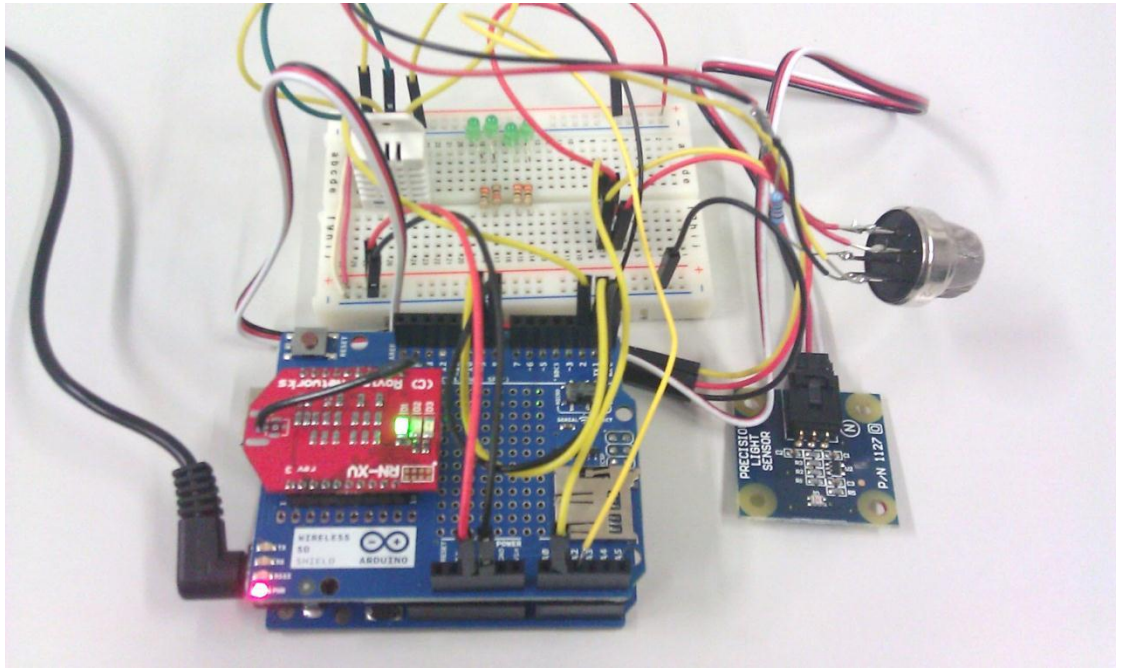
Modulo WiFly

Limitazioni con WiFly RN-XV rev 2.32 firmware:

1. Non si può determinare l'indirizzo IP del client TCP che si è connesso.
2. Cambiare la porta locale durante l'esecuzione non influenza nulla finchè non si salva e si riavvia il dispositivo.
3. Potrebbe non funzionare la chiusura di una connessione TCP. Il client deve rimanere connesso e inviare ulteriori dati.
4. è supportata solo una connessione TCP alla volta.
5. Cambiare il timeout di svuotamento (set comm time x) non porta cambiamenti fino al riavvio del dispositvo.

WIRELESS SHIELD - Esempio

Questo esempio mostra come è possibile creare una piccola stazione meteo wireless con alcuni sensori di rilevazione ambientale (temperatura, umidità, CO2, luce). I dati vengono inviati ad un server con il modulo WiFly montato sopra la Wireless Shield.

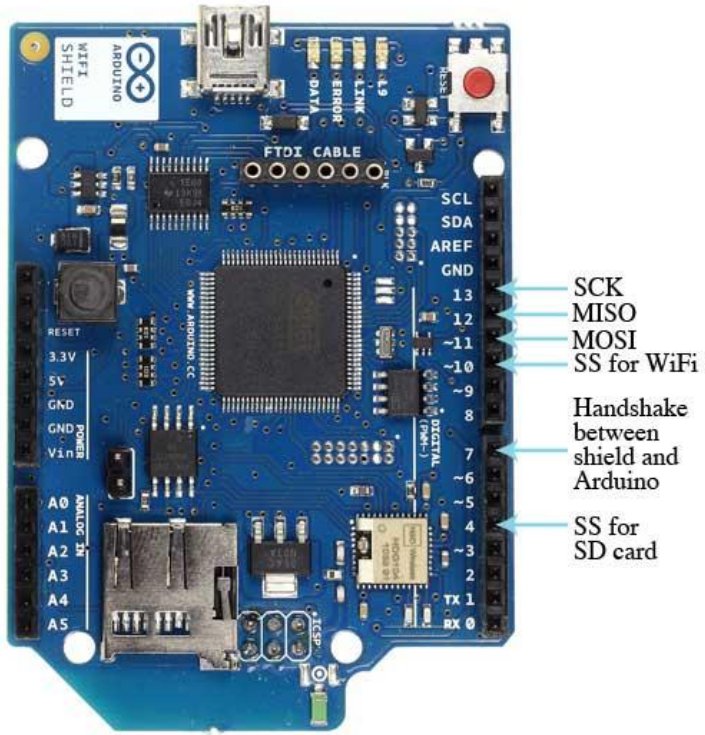


->StazioneMeteoWireless

WiFi Shield

La Arduino WiFi Shield collega Arduino a Internet in modalità wireless senza aver bisogno di ulteriori moduli.

- Tensione di lavoro 5 V (fornita dalla scheda Arduino)
- Collegamento a reti 802.11b / g
- Tipi di crittografia: WEP e WPA2 personal
- Collegamento con Arduino sulla porta SPI
- Slot micro SD on-board
- ICSP header
- Collegamento FTDI per il debug seriale
- Mini-USB per l'aggiornamento del firmware



Il pin digitale 7 è usato per un handshake tra la shield e Arduino e quindi non si può utilizzare.

Se si utilizza la shield con un Arduino prima del rev3 Uno, è necessario effettuare un ulteriore collegamento. La scheda Wi-Fi utilizza il pin IOREF sui nuovi layout pin di Arduino (Uno Rev3, Mega2560 rev3, e versioni successive) per rilevare la tensione di riferimento per i pin I / O della scheda a cui è attaccato. (link)

WiFi Shield

Per mandare dei comandi con la porta FTDI devo collegare un apposito cavo al connettore FTDI. La comunicazione avviene a 57600bps. Quando si invia un comando bisogna terminare con un a capo (CR).

Comandi supportati:

scan

connect <ssid>

setkey <key_idx (0-3)> <key in hex> ("**setkey** none" will delete all WEP keys)

status

debug <section> <level>

section: init, cm, spi, tcp , util, warn

level : 0 (off), 1 (on), 2 (verbose)

ttcp -t/-r [-options] host

-l length of bufs written to network (default 1024)

-n number of bufs written to network (default 1024)

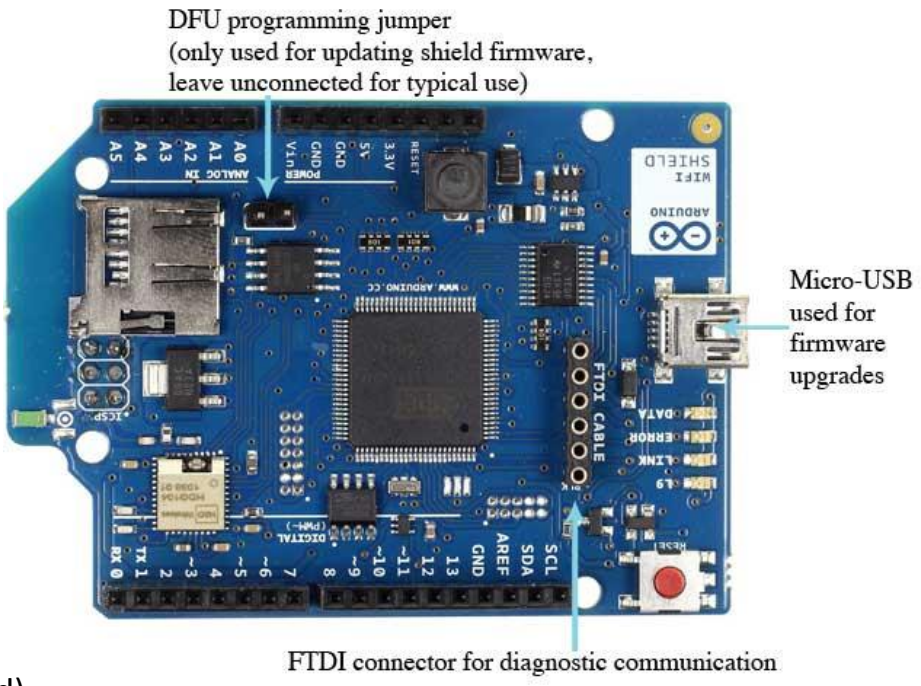
-p port number to send to (default 2000)

-u udp

-v verbose

wpass <ssid> <passphrase> (set passphrase associated to ssid)

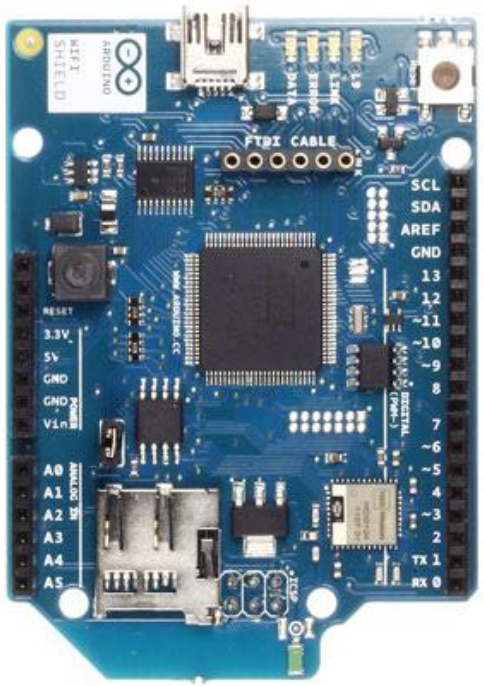
dpass <ssid> (delete passphrase associated to ssid)



WiFi Shield

Per utilizzare la shield faremo riferimento alla libreria WiFi.

```
#include <WiFi.h>

char ssid[] = "yourNetwork";      //  your network SSID (name)
char pass[] = "12345678";     // your network password
int status = WL_IDLE_STATUS;      // the Wifi radio's status

void setup() {
  // initialize serial:
  Serial.begin(9600);

  // attempt to connect using WPA2 encryption:
  Serial.println("Attempting to connect to WPA network...");
  status = WiFi.begin(ssid, pass);

  // if you're not connected, stop here:
  if ( status != WL_CONNECTED) {
    Serial.println("Couldn't get a wifi connection");
    while(true);
  }
  // if you are connected, print out info about the connection:
  else {
    Serial.println("Connected to network");
  }
}
```



WiFi Shield - Esempi

Esempio di modello client/server con due Arduino con WiFi shield.

Unica differenza rispetto alla classe Ethernet è che **non posso assegnare un indirizzo IP fisso** al server e quindi sono vincolato tutte le volte a sapere l'indirizzo del server e riprogrammare l'Arduino Client.

->WiFiShield->WiFiServer
->WiFiShield->WiFiClient

ZIGBEE

I moduli **XBee** sono una vera manna dal cielo per chi adora lavorare con i microcontrollori ed il pc. Tali moduli, davvero molto piccoli e con prezzi abbordabili, per poter comunicare tra loro utilizzano le specifiche **ZigBee** e operano in Italia sulla frequenza di **2,4GHz**. Tale specifica prevede che i dispositivi che la utilizzano siano a basso consumo e quindi per contro non abbiano transfer rate elevati.

In rete si possono trovare tutte le informazioni che si vogliono su tale protocollo di comunicazione:
Wikipedia – Zigbee
Zigbee.org

ZIGBEE

I Moduli XBee, quindi, sfruttano il protocollo ZigBee ed implementano una comunicazione seriale come quella presente sulla **porta RS232** del pc (ovviamente con diversi livelli di tensione). Esistono vari tipi di moduli ZigBee e ogni tipo di modulo può essere richiesto con un particolare tipo di antenna: cambiano le caratteristiche, le potenze in gioco ecc. Alcuni tipi di moduli, anche se diversi, possono comunicare tra loro.

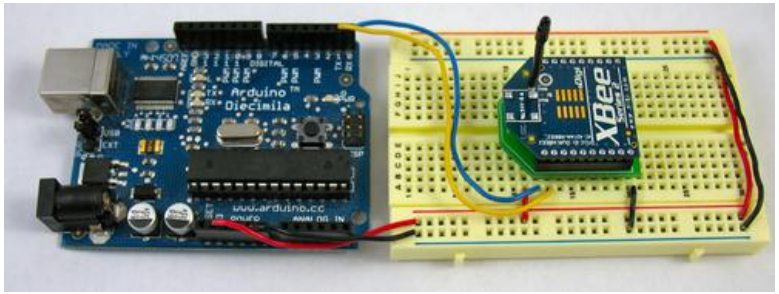
L'utilizzo degli XBee, poi, è davvero semplice: la codifica/decodifica di trasmissione è già eseguita via hardware; noi non dovremo fare assolutamente nulla se non fornire il dato seriale da trasmettere via usart.

XBEE – Come utilizzarli

Al primo utilizzo con un modulo XBee dovremo scontrarci con le prime due "difficoltà":

- I Moduli XBee hanno una **piedinatura con passo da 2mm**, più piccola della classica da 2,54mm alla quale siamo abituati normalmente.
- I Moduli XBee funzionano a **3,3Volts**, per cui anche i livelli logici vanno di pari passo.

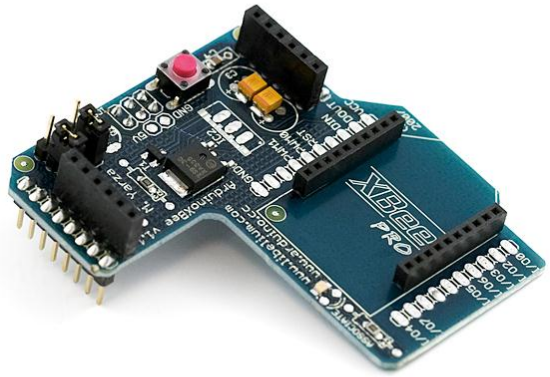
Questi due piccoli-grandi ostacoli possono essere aggirati molto facilmente dal momento che esistono apposite schede di adattamento in commercio. Esistono degli adattatori, che ci permettono di utilizzare tali moduli senza fare fatica e addirittura su una basetta breadboard!



XBEE – Come utilizzarli

Ovviamente esistono anche delle shield apposta da utilizzare con Arduino. Oltre alla Wireless Shield, esiste anche la Xbee shield che permette di utilizzare tali moduli senza aver bisogno di una breadboard. Questa shield accetta i vari tipi di moduli XBee mantenendo dimensioni molto contenute (può essere montato tranquillamente sopra una scheda Arduino UNO).

Ovviamente non si tratta di una "semplice" scheda di adattamento: ha a bordo i condensatori e il regolatore di tensione a 3,3V (accetta una tensione in ingresso fino a 18Volts sul pin denominato Vin), led per monitorare la comunicazione e pin per interfacciarsi con Arduino.



XBEE – I modelli

SERIE 1
Serie 1 (Xbee 802.15.4 OEM Low Power RF Modules)
Serie 1 PRO (Xbee-PRO 802.15.4 OEM Extended Range RF Modules)
Serie 1 Digimesh (Xbee Digimesh Low Power RF Modules)
Serie 1 PRO Digimesh (Xbee-PRO Digimesh Extended Range RF Modules)

SERIE 2
Serie 2 (Xbee ZB ZigBee Low Power RF Modules)
Serie 2B PRO (Xbee-PRO ZB S2B ZigBee Extended Range RF Modules)
Serie 2B PRO Programmabile (Programmable Xbee-PRO ZB S2B ZigBee Extended Range RF Modules)

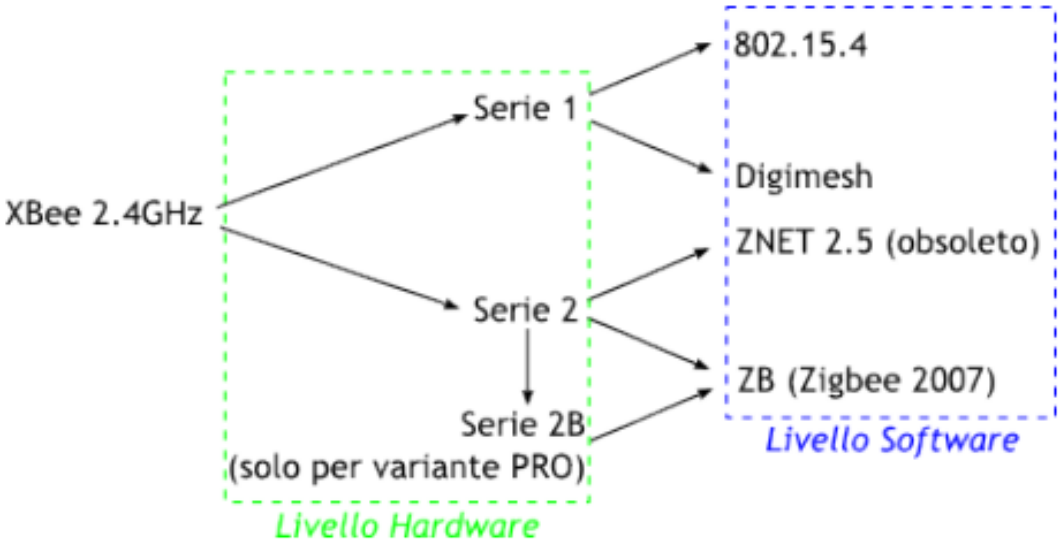
Entrambe le versioni montano tre tipi di antenne:

- A chip
- A filo
- Con connettore U.FL

XBEE – Serie

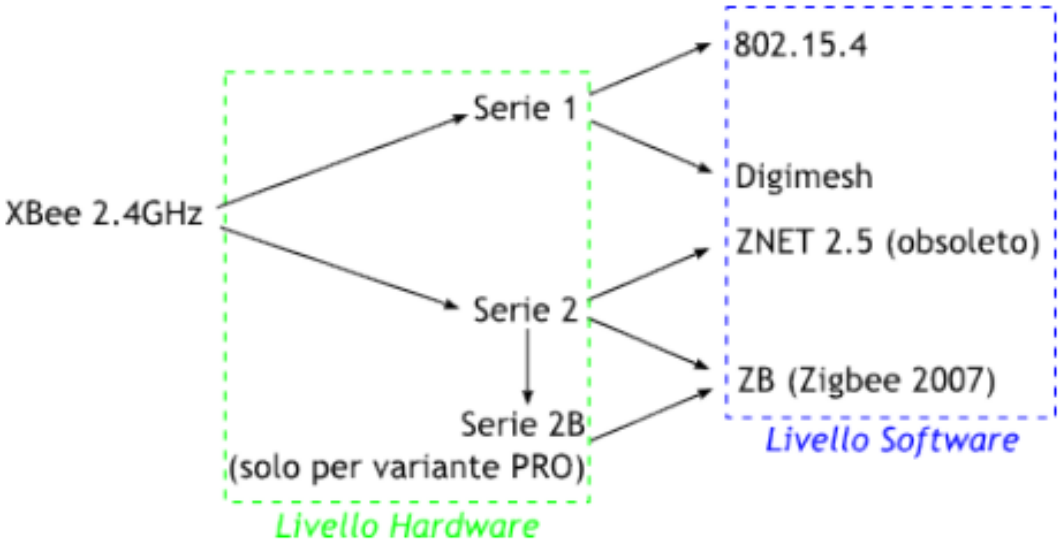
Questa denominazione non è indicativa delle varie famiglie di XBee ma è piuttosto un **riferimento all'hardware montato a bordo** che permette loro di funzionare in un modo piuttosto che in un altro.

N.B. Gli Xbee Serie 2 non sono ''migliori'' di quelli di Serie 1. Sono due tipologie di moduli che operano su tecnologie differenti.



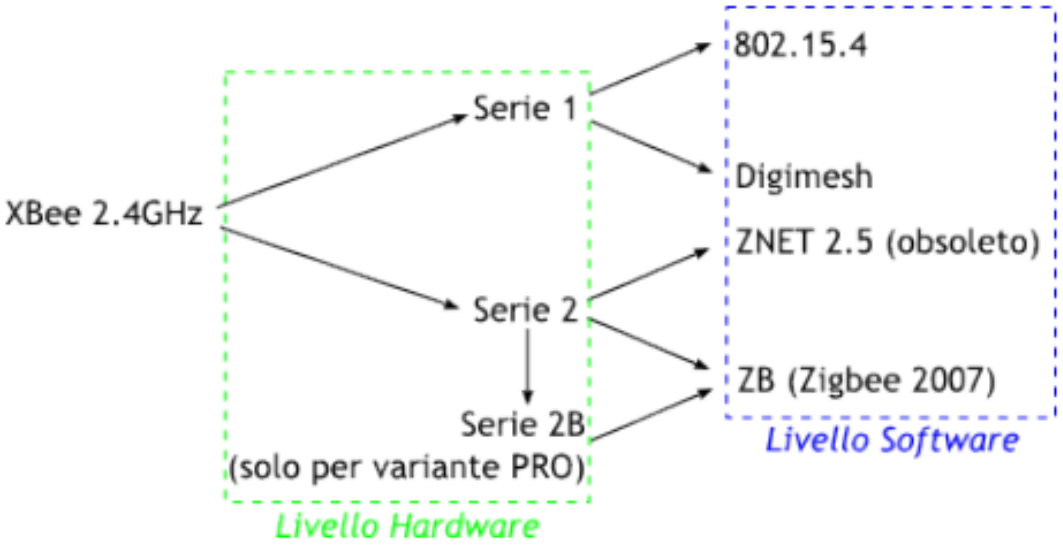
XBEE – Serie 1

Nei casi in cui si desidera realizzare una **comunicazione point-to-point** (ovvero devono comunicare tra loro soltanto due dispositivi) è sicuramente conveniente scegliere i moduli appartenenti alla famiglia **XBee 802.15.4 OEM RF modules (ovvero la Serie 1).**



XBEE – Serie 2

Per una comunicazione punto punto possono andar bene anche gli **XBee serie 2** ma sono più difficili da utilizzare in quanto andrebbero opportunamente configurati dato che di serie sono impostati per lavorare in una **rete Mesh** secondo il protocollo Zigbee.

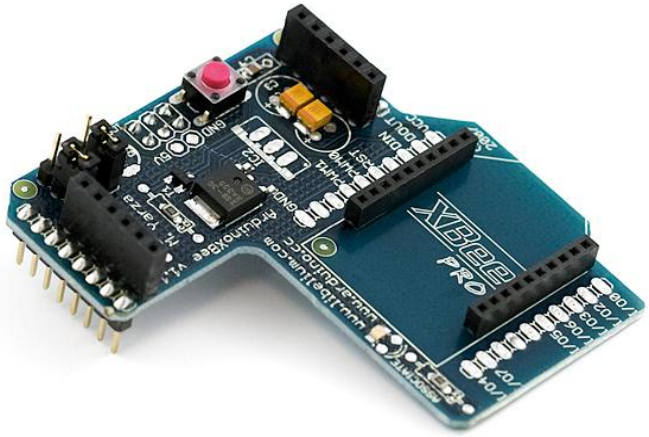


(per ulteriori chiarimenti rimando alla guida)

XBEE – Come utilizzarli

Ovviamente esistono anche delle shield apposta da utilizzare con Arduino. Oltre alla Wireless Shield, è presente anche la Xbee shield che permette di utilizzare tali moduli senza aver bisogno di una breadboard. Questa shield accetta i vari tipi di moduli XBee mantenendo dimensioni molto contenute (può essere montato tranquillamente sopra una scheda Arduino UNO).

Ovviamente non si tratta di una "semplice" scheda di adattamento: ha a bordo i condensatori e il regolatore di tensione a 3,3V (accetta una tensione in ingresso fino a 18Volts sul pin denominato Vin), led per monitorare la comunicazione e pin per interfacciarsi con Arduino.

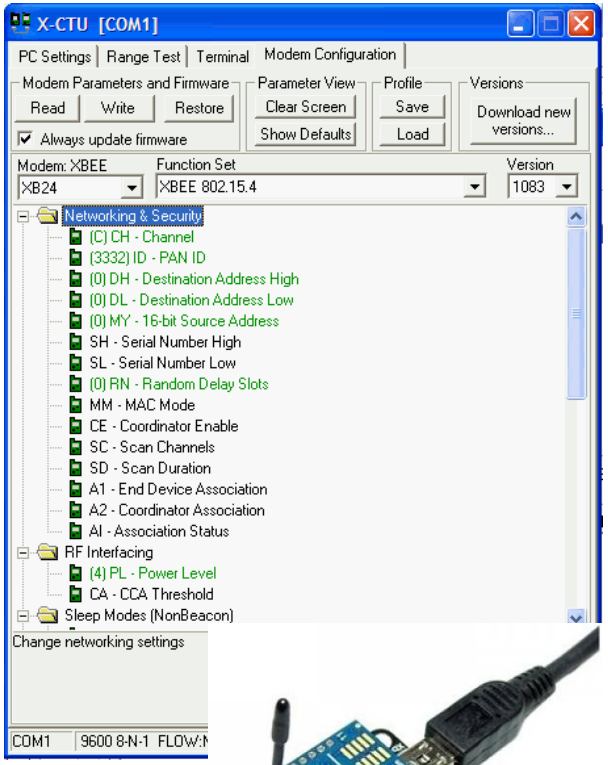


XBEE - Configurazione

1. Ogni Xbee deve essere configurato attraverso l'apposito software **X-CTU** via USB. Fortunatamente i moduli serie 1 sono già autoconfigurati per comunicare tra loro ad un baud rate di 9600bps in modalità trasparente. (link1)(link2)

2. Caricare un semplice sketch di prova su Arduino (senza modulo Xbee installato sopra) per la comunicazione. Il trasmittente userà l'istruzione Serial.print() mentre il ricevitore userà l'istruzione Serial.read().

3. Mettere il modulo sopra Arduino con i pin in modalità "Xbee".



Per approfondire

XBEE – Hello Xbee

TRASMETTITORE

```
void setup(){
   Serial.begin(19200);
}

void loop(){
   Serial.println('a');
   delay(1000);
   Serial.println('b');
   delay(1000);
}
```

RICEVITORE


```
// LED connected to digital pin 13
int ledPin = 13;

int val = 'a';
void setup(){
   // sets the digital pin as output
   pinMode(ledPin, OUTPUT);
   Serial.begin(19200);
}

void loop(){
   val = Serial.read();
   if (val=='a'){digitalWrite(13,LOW);}
   if (val=='b'){digitalWrite(13,HIGH);}
}
```

XBEE – AT Comands

La **modalità comandi** è quella normalmente utilizzata da XCTU per comunicare con il modulo XBee e permette di inviare comandi AT (lo standard utilizzato per tutti i modem).
Per entrare in modalità comandi, di default **basta rispettare 3 semplici regole**:

- Attendere almeno un secondo dall'ultimo dato ricevuto
- Inviare al modulo la stringa +++ (senza ritorno a capo!)
- Attendere un secondo

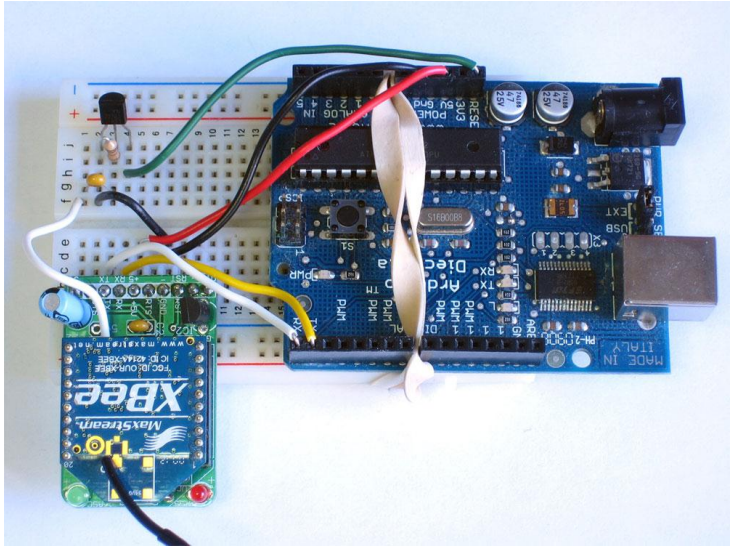
Il modulo **deve rispondere con "OK".**

Il secondo da attendere sia prima che dopo è definito dal parametro GT – Guard Times, di default impostato al valore 0x03E8 (1000 in decimale), mentre il carattere da inviare per 3 volte per entrare in modalità comandi è definito dal parametro CC – Command Sequence Character, impostato di default al valore 0x2B (codice ascii del +). (Lista comandi)

XBEE Library

I moduli Xbee possono essere utilizzati anche in modalità API. Arduino mette a disposizione una libreria per comunicare con i moduli in tale modalità.
La libreria supporta sia i moduli Serie 1 (802.15.4) che Serie 2 (ZB Pro / ZNet).
Questa libreria Include il supporto per la maggior parte dei tipi di pacchetti, tra cui: TX / RX, comando AT, remoto AT, I / O Campioni e lo stato del modem.

Nota: Questo software richiede modalità API, impostando AP = 2. Se si utilizza Series 2 XBee, è necessario installare firmware API (che spediscono con AT firmware), quindi impostare AP = 2. Questo software non funziona correttamente con AP = 1. Fare riferimento a XBeeConfiguration e WhyApiMode per maggiori informazioni.

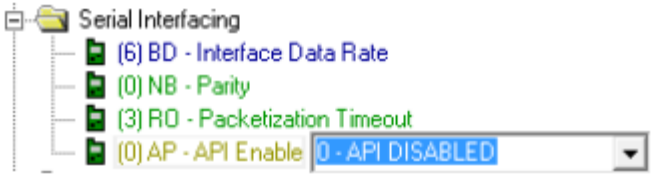


XBEE – API

La modalità API (Application Programming Interface) è un'alternativa alla modalità trasparente che consente, oltre che trasmettere e ricevere dati, di **interagire ad un livello più basso con i moduli XBee** consentendo, tra le altre cose, di:

- Cambiare i parametri di configurazione (inviare comandi AT) senza entrare in
- modalità comandi
- Conoscere l' RSSI (Received Signal Strength Indicator – Indicatore di forza del
- segnale ricevuto)
- Ricevere una conferma di pacchetto dati consegnato correttamente per ogni
- pacchetto trasmesso o un'indicazione di consegna fallita
- Trasmettere dati a più destinatari
- Identificare l'indirizzo di chi ha trasmesso il pacchetto dati

Si capisce quindi perché la **modalità trasparente** abbia tale nome: i dati vengono inviati sulla seriale tali e quali, in **modalità API** invece i dati vengono interpretati.
La modalità trasparente viene impostata, tramite XCTU, agendo sul parametro *AP – API Enable*. Il valore 0 identifica la modalità trasparente (API Disabled).



XBEE – Esempio

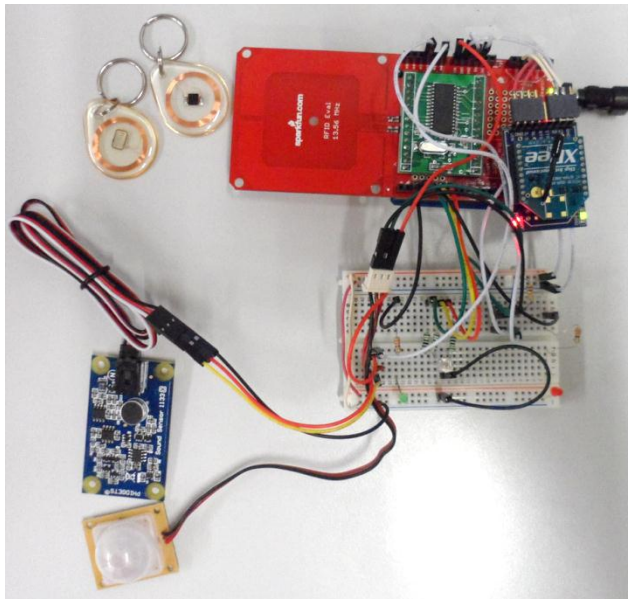
Il modo più semplice per comunicare con i moduli XBee è in modalità punto a punto. Ciò significa che un modulo comunica direttamente con un altro modulo. I dati seriali inviati da un XBee, si trasmettono in modalità wireless, all'altro e viceversa.

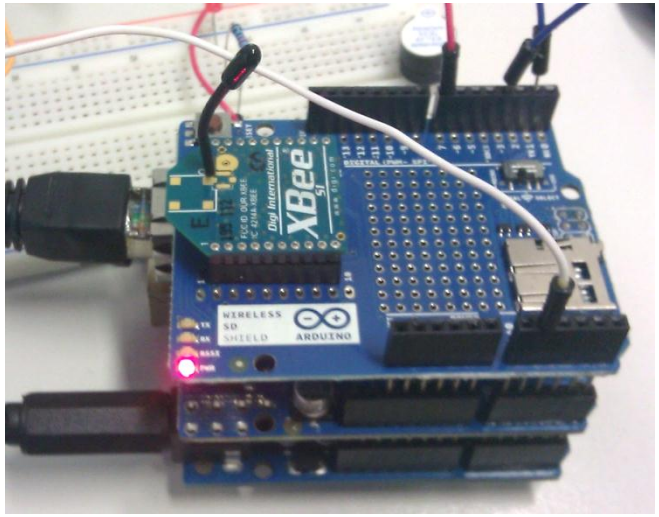
Un esempio potrebbe essere quello di mandare via wireless i valori letti da un potenziometro e l'altro modulo, ricevendo i dati, modifica la posizione di un servo motore.

->PotToMotor

XBEE – Esempio

Monitoraggio della presenza di persone all'interno di un ambiente. Comunicano tra di loro con i moduli XBee e inviano i dati al server attraverso la porta Ethernet.





->XbeeAdvanced

XBEE – Esercizio

1. Con Processing creare 4 bottoni:
   Su - Giu - Dx - Sx
   Con Arduino ricevere via seriale i valori dei bottoni premuti e inviarli ad un modulo Xbee che si preoccuperà di far accendere 4 led che simulano la pressione dei tasti

2. Pseudo-Controller per automobile radiocomandata

3. Integrare i due esercizi precedenti

RIFERIMENTI

www.mancio90.it

mirkomancin90@gmail.com