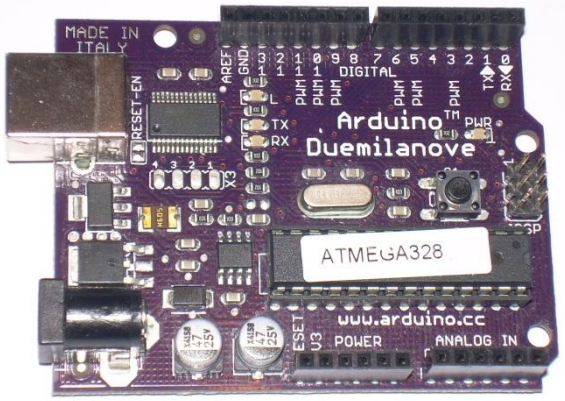
ARDUINO

Parte_1

Caratteristiche

Arduino è un progetto formato…



• da una parte hardware il cui cuore è un microcontrollore della ATMEL;

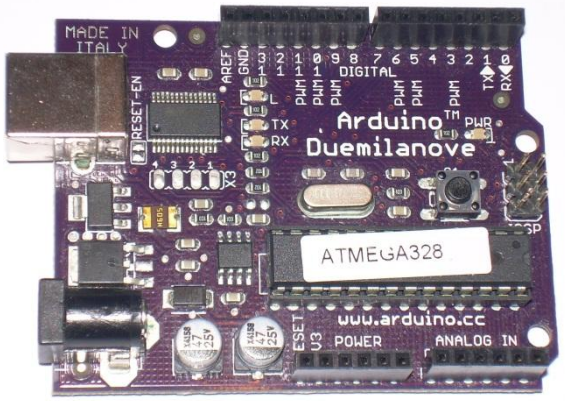
• da una parte software per la programmazione del dispositivo.

Offre un ampio ventaglio di interfacce:

Seriale, Bluetooth, Ethernet, SD, SPI, I2C, Wireless

Il sito di riferimento è www.arduino.cc

Arduino principali caratteristiche



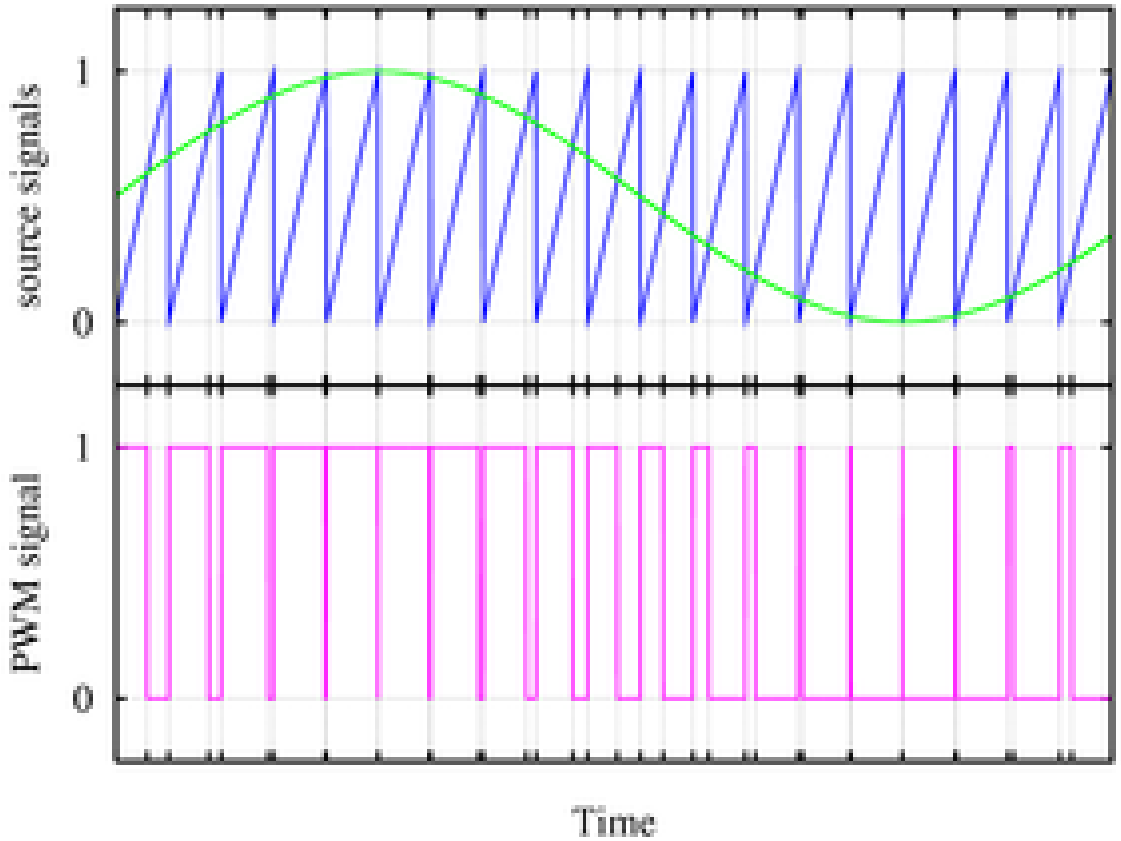
- 14 ingressi/uscite digitali di cui:
  - 6 utilizzabili come uscite di tipo **PWM**;
  - 4 utilizzabili per comunicazione **SPI**;
  - 2 utilizzabili per comunicazione **I^2C**;
  - 2 utilizzabili per i collegamenti **seriali TTL** level;
  - 2 utilizzabili per **interrupt** esterno;
- 6 ingressi analogici (risoluzione 10 bits).

PWM

La **modulazione a larghezza di impulso** è largamente utilizzata per regolare la potenza elettrica inviata ad un carico, per esempio negli inverter, per regolare la velocità dei motori in corrente continua e per variare la luminosità delle lampadine.

PWM

Come si può intuire, con un **duty cycle** pari a zero la potenza trasferita è nulla, mentre al 100% la potenza corrisponde al valore massimo trasferito nel caso non sia presente il circuito di modulazione. Ogni valore intermedio determina una corrispondente fornitura di potenza.



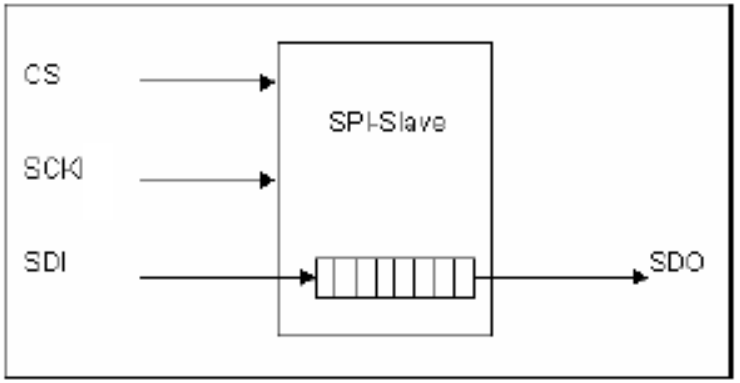
SPI

Il **Serial Peripheral Interface** è un sistema di comunicazione tra un microcontrollore e altri circuiti integrati o tra più microcontrollori.
La trasmissione avviene tra un dispositivo detto *master* e uno o più *slaves*. Il master controlla il bus, emette il segnale di clock, decide quando iniziare e terminare la comunicazione.
Il bus SPI si definisce:

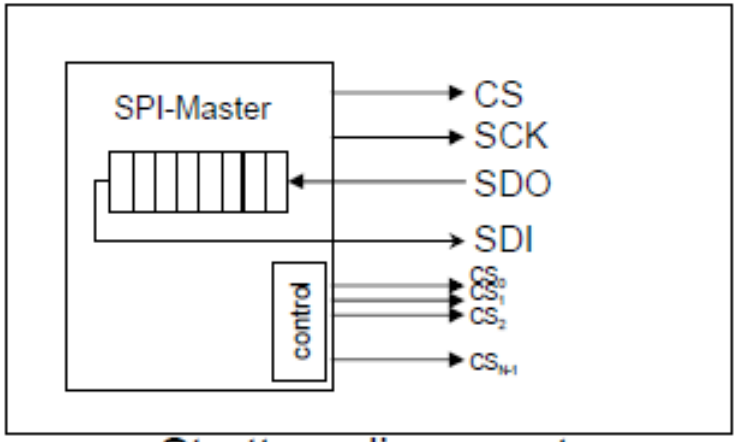
- di tipo **seriale**
- **sincrono** per la presenza di un clock che coordina la trasmissione e ricezione dei singoli bit e determina la velocità di trasmissione
- **full-duplex** in quanto il "colloquio" può avvenire contemporaneamente in trasmissione e ricezione.

SPI

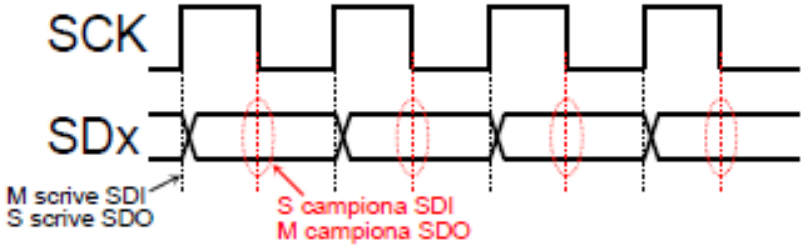
• Ogni device ha uno **shift register** contenente i dati. Il trasferimento prevede lo scambio del contenuto.
• In ogni trasferimeno avvengono gli scambi M->S e S<-M
• Il Master indirizza lo slave, e gestisce il trasferimento con il segnale SCK
• I dati in uscita vengono scritti in corrispondenza del fronte di salita [discesa] di SCK
• I dati vengono campionati sul fronte opposto



Struttura di uno slave



Struttura di un master



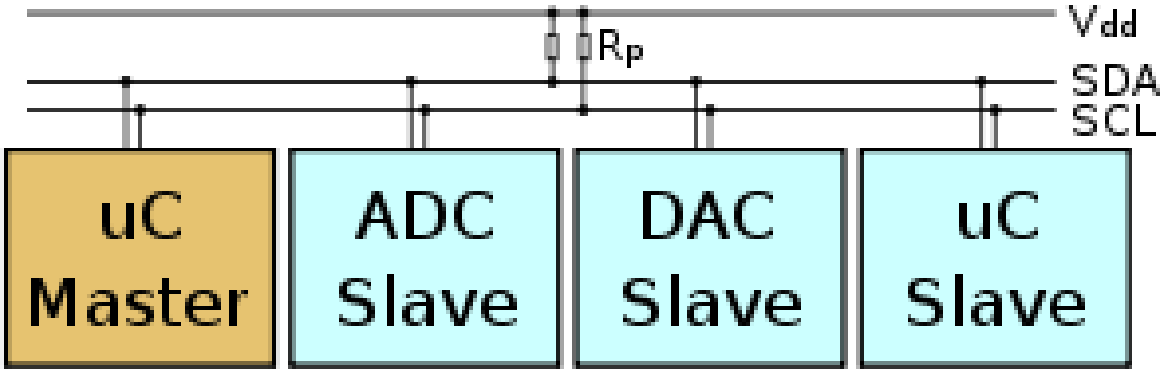
I²C

Acronimo di **Inter Integrated Circuit** , è un sistema di comunicazione seriale bifilare utilizzato tra circuiti integrati.
Il classico bus I²C è composto da almeno un *master* ed uno *slave*.
La situazione più frequente vede un singolo *master* e più *slave*; possono tuttavia essere usate architetture *multimaster* e *multislave* in sistemi più complessi.

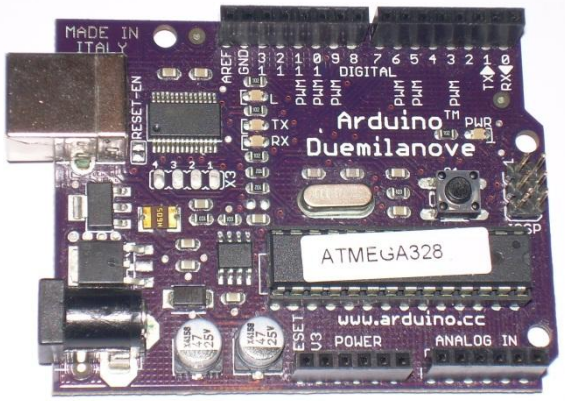
I^2C

Il protocollo hardware dell'I^2C richiede due linee seriali comunicazione:

- **SDA** (***S**erial **DA**ta line*) per i dati
- **SCL** (***S**erial **C**lock **L**ine*) per il clock (per la presenza di questo segnale l'I^2C è un bus sincrono).



Arduino principali caratteristiche



- Memoria **SRAM** 2KB;
- Memoria **FLASH** 32KB di cui 2 utilizzati per il bootloader;
- Memoria **EEPROM** 1KB
- Microcontrollore **ATMEGA328** a 16 MHz

SRAM

La **SRAM**, acronimo di *Static Random Access Memory*, è un tipo di RAM volatile che non necessita di refresh. I banchi di memorie SRAM consentono di mantenere le informazioni per un tempo teoricamente infinito, hanno bassi tempi di lettura e bassi consumi.

FLASH

La **memoria flash**, anche chiamata **flash memory**, è una tipologia di EEPROM, quindi di memoria non volatile, che per le sue prestazioni può anche essere usata come memoria a lettura-scrittura.
La memoria flash, trattandosi di memoria a stato solido, non presenta alcuna parte mobile quindi è piuttosto resistente alle sollecitazioni e agli urti, inoltre è estremamente leggera e di dimensioni ridotte.
La memoria flash è particolarmente indicata per la *trasportabilità*, proprio in virtù del fatto che non richiede alimentazione elettrica per mantenere i dati e che occupa poco spazio.
Molto usata nei lettori di musica portatili, nelle pendrive (chiavette), ecc….

Bootloader 1/2

Il microcontrollore è fornito con un **bootloader**, che è un software che permette il caricamento dei programmi in memoria senza l'ausilio di programmatori esterni (occupa 2 KB di memoria flash).

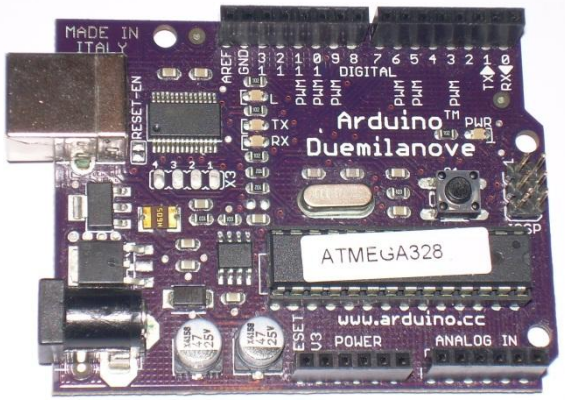
Quando si resetta la scheda viene fatto girare il bootloader (se presente) che fa lampeggiare il led collegato al pin 13.

Bootloader 2/2

Il bootloader si mette in ascolto di comandi o dati in arrivo dal computer (che generalmente sono i programmi scritti dall'utente) e li carica nella memoria flash del microcontrollore; dopodiché viene lanciato il programma in memoria. Se non ci sono programmi in arrivo dal computer viene lanciato l'ultimo **sketch** caricato. Se invece, il microcontrollore è vuoto viene lanciato in continuazione il bootloader.

sketch: programma scritto dall'utente

Alimentazione



Arduino può essere alimentato:

• direttamente dalla porta USB (5V – 500mA max);

• con un alimentatore esterno con tensione limite 6-20V, consigliata 7-12V, attraverso l'apposita presa jack;

• tramite una batteria da 5V attraverso gli appositi piedini.

Arduino fornisce due tensione per alimentare i dispositivi esterni: una a 5V ed un'altra a 3,3V.

Comunicazione

Arduino comunica con il PC attraverso la porta USB.

Nella scheda trova posto l'integrato FTDI FT232RL che consente di avere un collegamento **seriale virtuale** sopra un collegamento USB.

I driver del chip FTDI FT232RL devono essere caricati nel PC e comunque sono forniti insieme al software per la programmazione di Arduino.

Il sito di riferimento del chip è www.ftdichip.com

Software

Il software per la programmazione di tutti i tipi di schede Arduino, si chiama semplicemente **Arduino**.

Ne esistono diverse versioni sia per Windows (quella testata in questo lavoro), sia per Linux sia per Mac.

L'ultima versione disponibile è la **0018**.

Software

La prima volta che si carica il programma si deve selezionare il tipo di scheda che l'utente ha a disposizione.

Si seleziona in seguito la porta seriale corrispondente, ad esempio la COM4.

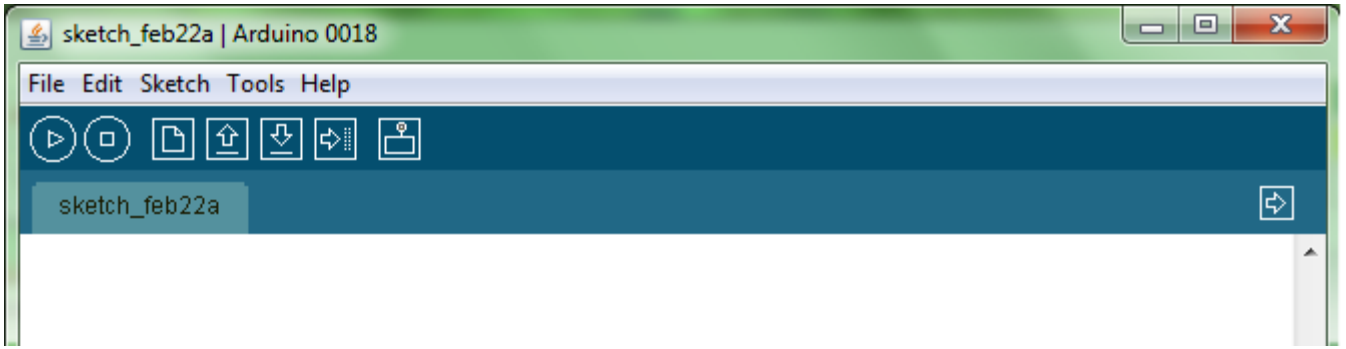
Si procede con la scrittura del programma, chiamato sketch.

Con l'operazione di Upload si carica lo sketch nel dispositivo e si verifica il suo funzionamento.

IDE

I principali comandi del software sono:

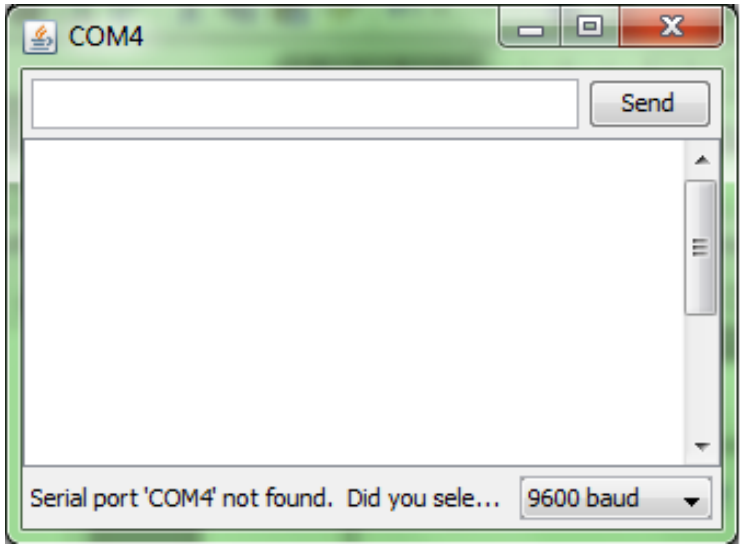
- **Verify**: per compilare il programma;
- **Stop**: per interrompere la verifica del programma;
- **Save**: per salvare lo sketch;
- **Open**: per aprire uno sketch salvato in precedenza;
- **New**: per creare un nuovo sketch;
- **Upload**: per caricare lo sketch nel microcontrollore.



IDE

Uno strumento utilissimo nella fase di programmazione e di debug è il **Serial Monitor** che permette di inviare e di ricevere dati testuali dalle Arduino board.

Il Serial Monitor durante il suo funzionamento tiene occupati i due pin del collegamento seriale: pin 0 (Rx) e pin 1 (Tx); per tale motivo questi due pin non possono essere utilizzati come ingressi o uscite.



Struttura degli sketch

Uno sketch si compone di due funzioni principali che non accettano nessun parametro e non restituiscono alcun valore:

• void **setup**() , tutte le istruzioni contenute all'interno di questa funzione vengono eseguite una sola volta al lancio dello sketch da parte del microcontrollore;

• void **loop**() , tutte le istruzioni contenute in questa funzione sono eseguite in continuazione.

E' sempre possibile inserire parti di programma all'interno di altre funzioni che vengono richiamate dal programma principale all'occorrenza.

Struttura degli sketch

Di solito uno sketch utilizza delle librerie di comandi che consentono di controllare dei particolari dispositivi.

Le librerie possono essere scritte anche dagli utenti a proprio uso e consumo.

Le librerie vengono inserite con il comando

#include <SoftwareSerial.h>

Il discorso sulle librerie sarà ripreso più avanti

Linguaggio di programmazione

Il linguaggio di programmazione può essere diviso in tre parti:

- **Strutture**;
- **Variabili e costanti**;
- **Funzioni**.

Linguaggio di programmazione

Strutture

- setup()
- loop()
- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return

Linguaggio di programmazione

Variabili e costanti

Le variabili sono dei contenitori che possono essere utilizzati nei programmi per memorizzare dei valori;

Le costanti predefinite sono:

- HIGH | LOW
- INPUT | OUTPUT
- true | false
- Integer Constants

Linguaggio di programmazione

Funzioni

Digital I/O

- pinMode(pin, mode)
- digitalWrite(pin, value)
- int digitalRead(pin)

Analog I/O

- int analogRead(pin)
- analogWrite(pin, value)

Advanced I/O

- tone()
- noTone()
- shiftOut(dataPin, clockPin, bitOrder, value)
- unsigned long pulseIn(pin, value)

Time

- unsigned long millis()
- unsigned long micros()
- delay(ms)
- delayMicroseconds(us)

Linguaggio di programmazione: funzioni

Funzioni

Math

- min(x, y)
- max(x, y)
- abs(x)
- constrain(x, a, b)
- pow(base, exponent)
- sq(x)
- sqrt(x)
- map(value, fromLow, fromHigh, toLow, toHigh)

Trigonometry

- sin(rad)
- cos(rad)
- tan(rad)

Random Numbers

- randomSeed(seed)
- long random(max)
- long random(min, max)

Communication

- Serial

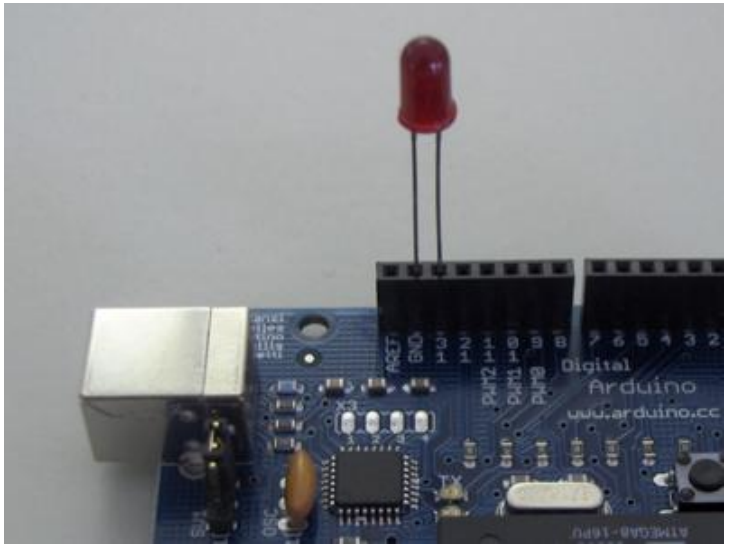
ARDUINO

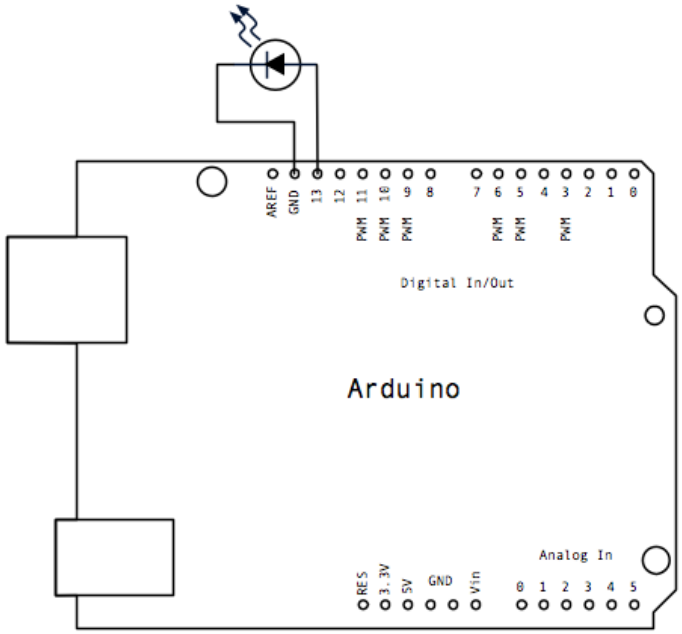
Parte_2

Laboratorio

Esempio 1

```
int ledPin = 13;

void setup() {
  pinMode(ledPin, OUTPUT); }

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```



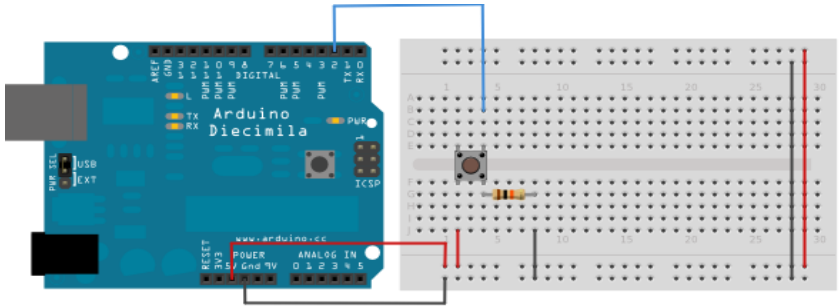


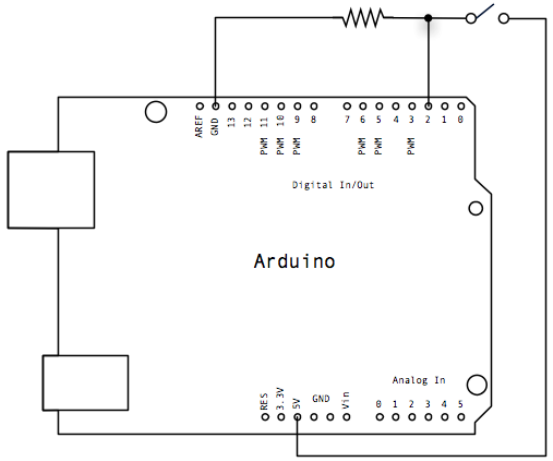
"Il led collegato al pin 13 si accende e si spegne alternativamente una volta al secondo."

Esempio 2

```
const int buttonPin = 2;
const int ledPin = 13;
int buttonState = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
            digitalWrite(ledPin, HIGH);
  } else {
            digitalWrite(ledPin, LOW);
  }
}
```

"Se il pulsante collegato al pin 2 è premuto, si accende il led collegato al pin 13 che altrimenti rimane spento."

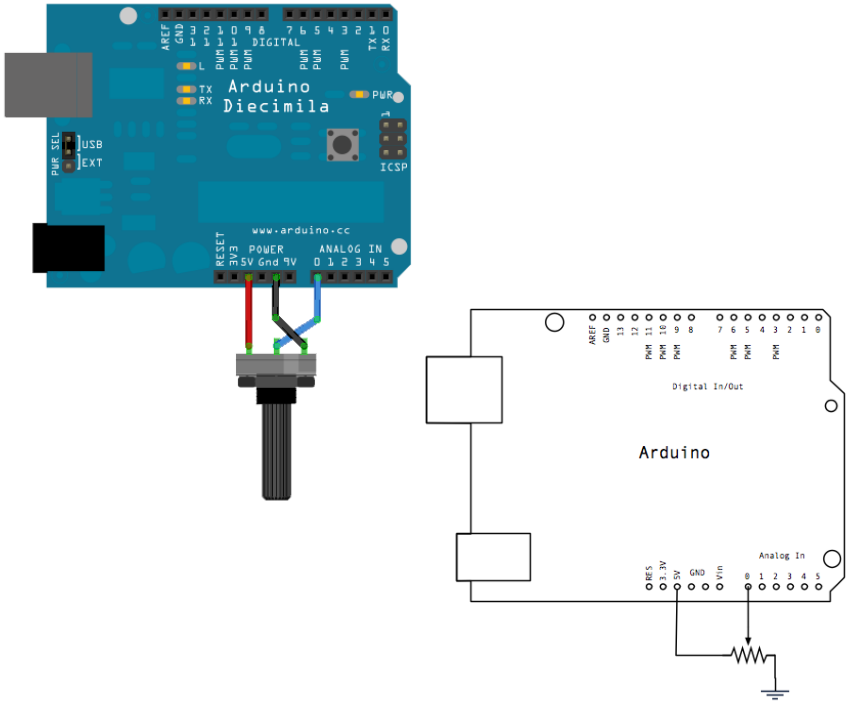




Esempio 3

```
int sensorPin = 0;
int ledPin = 13;
int sensorValue = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  sensorValue = analogRead(sensorPin);
  digitalWrite(ledPin, HIGH);
  delay(sensorValue);
  digitalWrite(ledPin, LOW);
  delay(sensorValue);
}
```

"Il led collegato al pin 13 lampeggia con una frequenza proporzionale al valore di tensione letto sull'ingresso analogico 0 regolato da un potenziometro."



Comandi della seriale

I comandi che controllano il collegamento seriale sono molto importanti perché permettono ad Arduino di comunicare con una miriade di dispositivi seriali: gps, modem, ecc…

- begin()
- available()
- read()
- flush()
- print()
- println()
- write()

Esempio 4

“Utilizzo dei comandi della seriale e del Serial Monitor – parte 1”

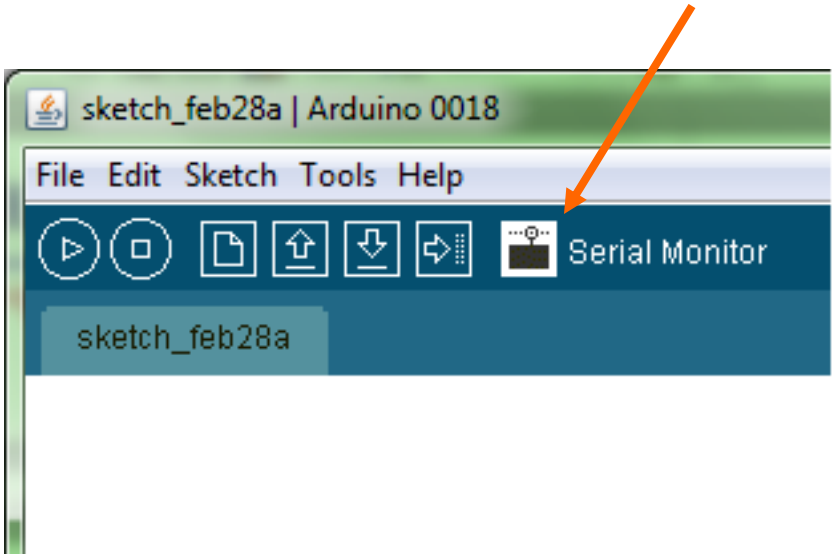
```
const int buttonPin = 2;
const int ledPin = 13;
int buttonState = 0;
void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
            digitalWrite(ledPin, HIGH);
            Serial.println(“On”);
 } else {
            digitalWrite(ledPin, LOW);
            Serial.println(“Off”);}
 }
```

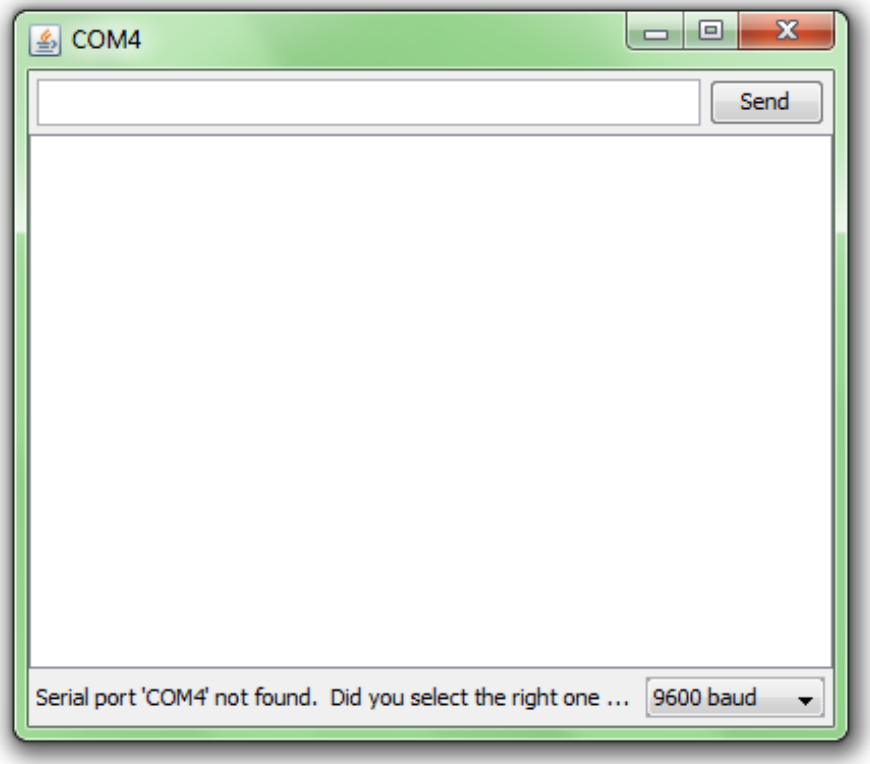
“Se il pulsante collegato al pin 2 è premuto, si accende il led collegato al pin 13 che altrimenti rimane spento; inoltre viene inviata sul collegamento seriale la stringa “On” quando il led è acceso, oppure la stringa “Off” quando è spento.”

Esempio 4

“Utilizzo dei comandi della seriale e del Serial Monitor – parte 2”

Con il **Serial Monitor** (strumento dell’ambiente di sviluppo Arduino) è possibile analizzare il trasferimento dei dati che avviene attraverso il collegamento seriale.





Esempio 5

"Utilizzando la tecnica PWM, il led connesso al pin 9 attraverso un resistore di 1KΩ, si accende e si spegne in modo graduale."

```
int ledPin = 9;
void setup()  {
}

void loop()  {
 for(int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) {
            analogWrite(ledPin, fadeValue);
            delay(30);}

 for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) {
            analogWrite(ledPin, fadeValue);
            delay(30);}
}
```

Display LCD

Principali collegamenti

I seguenti pin servono per il funzionamento del dispaly LCD (vedere data sheet per il loro significato).

- **RS**
- **R/W**
- **Enable**
- **D0-D3 D4-D7**
- **display contrast pin (Vo)**
- **power supply pins (+5V and Gnd)**
- **LED Backlight (Bklt+ and BKlt-)**

Libreria per il controllo dei display LCD

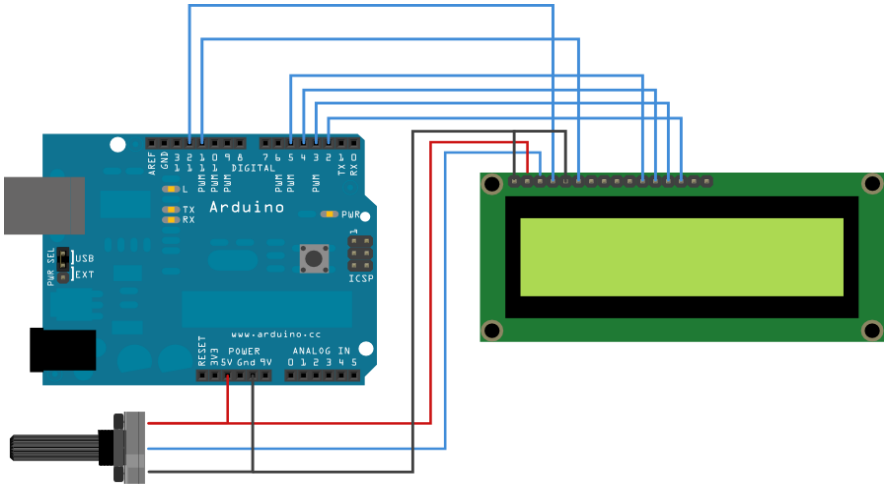
<LiquidCrystal.h>

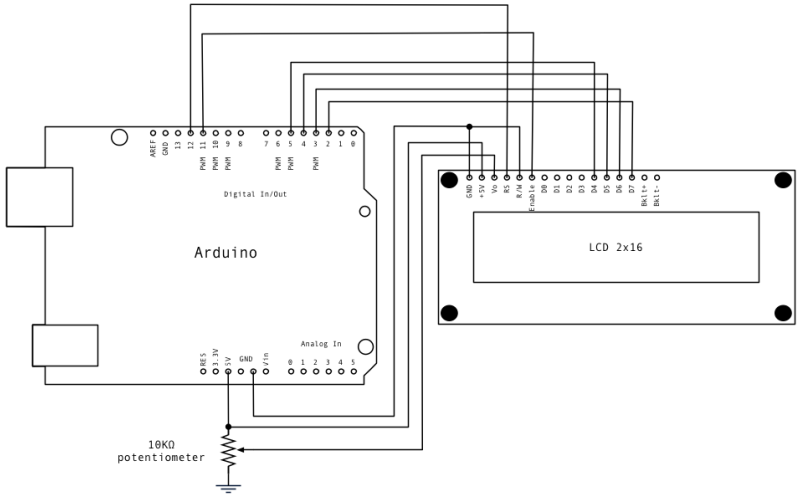
Utilizzando le seguenti funzioni della libreria LiquidCrystal è possibile scrivere sul display nella posizione desiderata, e altro ancora……

- LiquidCrystal()
- begin()
- clear()
- home()
- setCursor()
- write()
- print()
- cursor()
- noCursor()
- blink()
- noBlink()
- display()
- noDisplay()
- scrollDisplayLeft()
- scrollDisplayRight()
- autoscroll()
- noAutoscroll()
- leftToRight()
- rightToLeft()
- createChar()

Esempio 6

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  lcd.begin(16, 2);
  lcd.print("hello, world!");
}

void loop() {
  lcd.setCursor(0, 1);
  lcd.print(millis()/1000);
}
```





"Lo sketch stampa sulla prima linea del display la striga "hello,world!" e sulla seconda riga i secondi trascorsi."

Creazione di una libreria

Una libreria è composta da una **interfaccia** e da un file in cui sono **implementate le funzioni** che la libreria svolge.

Morse.h (interfaccia)

```
#ifndef Morse_h
#define Morse_h
#include "WProgram.h“
class Morse {
  public:
            Morse(int pin);
            void dot();
            void dash();
  private:
            int _pin; };
#endif
```

Morse.cpp (implementazione delle funzioni)

```
#include "WProgram.h“
#include "Morse.h“
Morse::Morse(int pin) {
   pinMode(pin, OUTPUT);
   _pin = pin;
}
void Morse::dot() {
  digitalWrite(_pin, HIGH);
  delay(250);
  digitalWrite(_pin, LOW);
  delay(250);
}
void Morse::dash() {
  digitalWrite(_pin, HIGH);
  delay(1000);
  digitalWrite(_pin, LOW);
  delay(250);
}
```

Creazione di una libreria

Le librerie si trovano nella cartella ***arduino-0018/libraries***.

Ogni libreria è completamente posizionata in una cartella che porta il suo nome e contiene oltre all'interfaccia (xxx.h) e al file di implementazione delle funzioni (xxx.cpp), la cartella **Examples** dove sono inseriti gli esempi di utilizzo della libreria stessa. Inoltre deve anche essere sempre presente il file **keywords.txt** perché serve ad indicare i colori con cui appariranno a video le funzioni (aprirlo per vederne il contenuto).

Creazione di una libreria - Esempio 7

“Visualizza S.O.S. – esempio di utilizzo della libreria Morse”

L’esempio seguente serve per mostrare l’utilizzo della libreria creata in precedenza; le funzioni dot() e dash() contenute nella libreria **Morse** realizzano rispettivamente il punto e la linea del “codice Morse”.

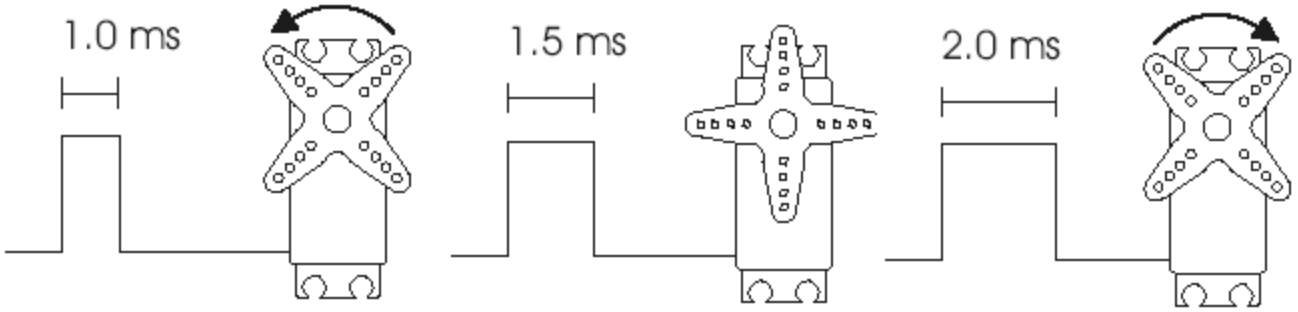
```
#include <Morse.h>

Morse morse(13);

void setup() {
}

void loop() {
  morse.dot();   morse.dot();   morse.dot();
  morse.dash(); morse.dash(); morse.dash();
  morse.dot();   morse.dot();   morse.dot();
  delay(3000);
}
```

“Lo sketch accende il led collegato al pin 13, in modo tale da realizzare un S.O.S. luminoso.”

…---…

Servomotori

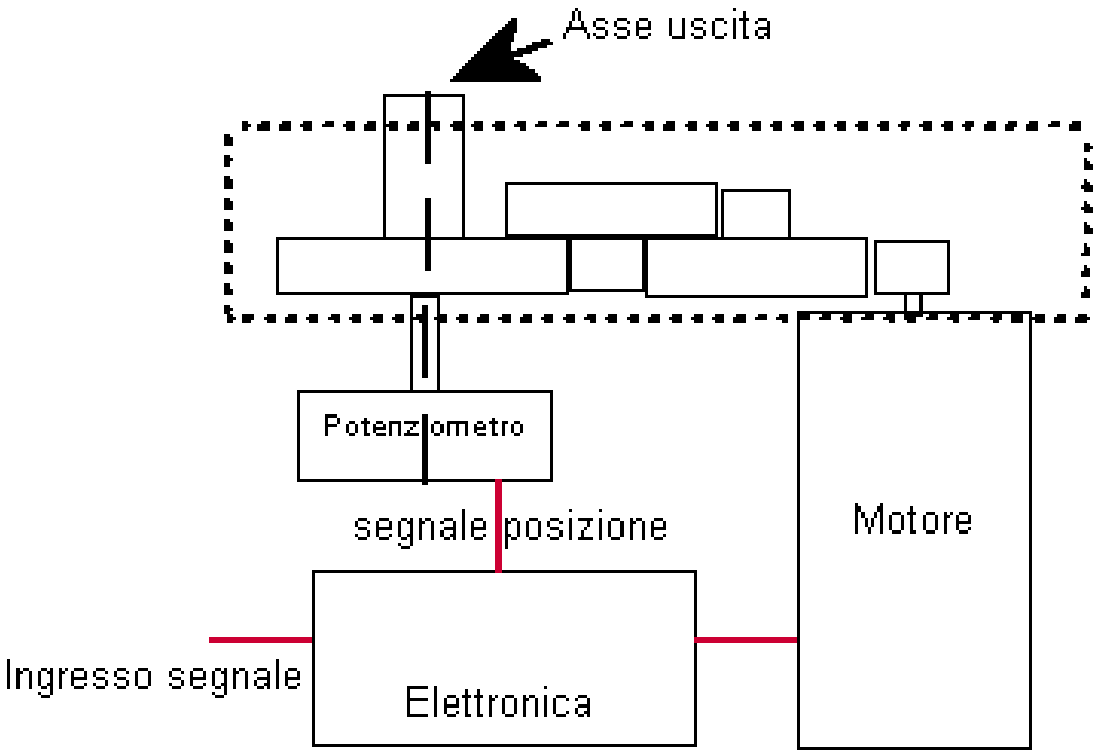
I servomotori sono controllabili mediante un segnale di tipo PWM **la cui durata dell'impulso indica la posizione dell'albero**. Per cui se l'impulso avrà durata pari a 1 msec l'albero del servomotore si porterà in posizione 0°, mentre se l'impulso avrà durata 2 msec l'albero si porterà in posizione di massima apertura (l'angolo reale raggiunto dipende dal modello si servomotore utilizzato). Generalmente con un impulso di durata pari a 1.5ms il perno del servomotore si posiziona esattamente al centro del suo intervallo di rotazione.



Servomotori

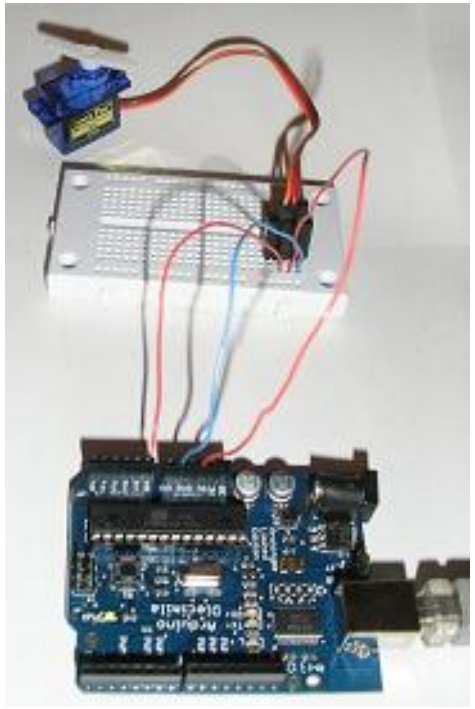
Funzionamento di massima

Il segnale d'ingresso è confrontato con il segnale di posizione; la risultante è il segnale inviato al motore.



Esempio 8

```
#include <Servo.h>

Servo myservo;
int potpin = 0;
int val;

void setup() {
 myservo.attach(9);
}

void loop() {
  val = analogRead(potpin);
  val = map(val, 0, 1023, 0, 179);
  myservo.write(val);
  delay(15);
}
```

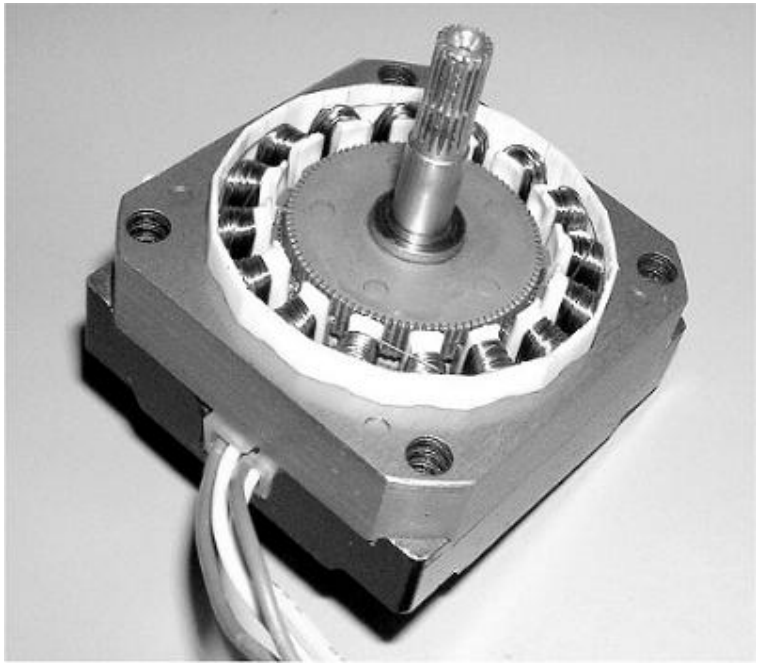


“Facendo ruotare il potenziometro collegato all’ingresso analogico 0, si determina la posizione del servomotore collegato al pin 9, compresa tra 0° e 180°.”

Motori passo-passo

Descrizione

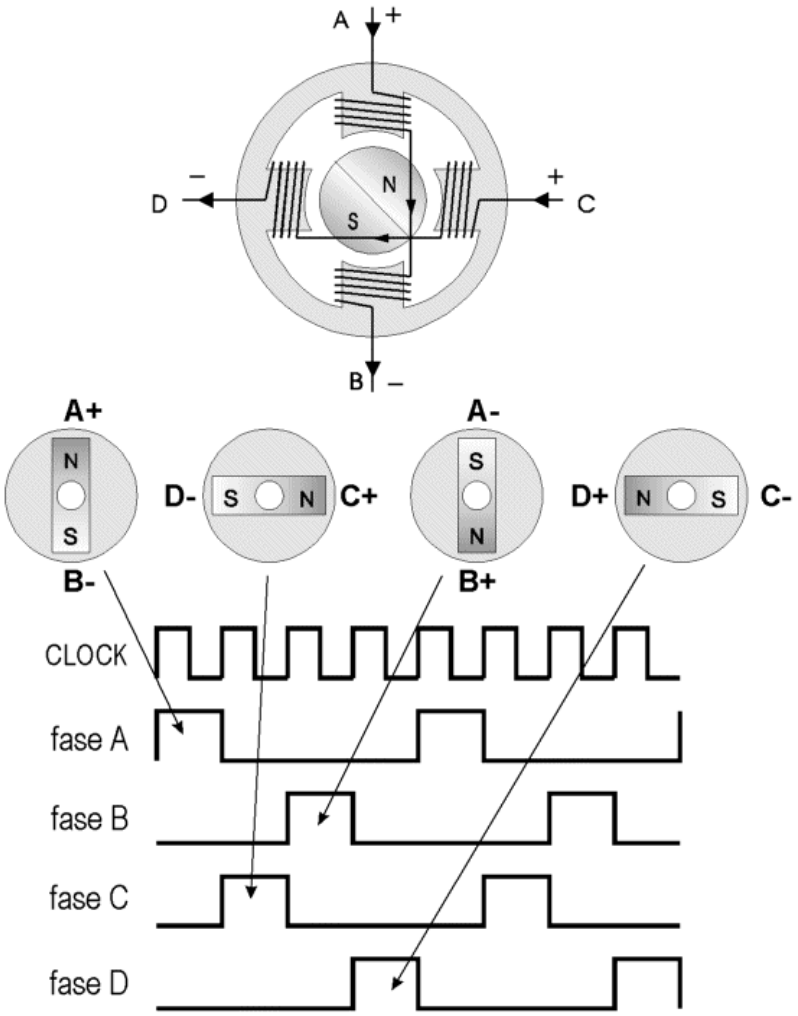
La tipica costituzione di un **motore passo-passo** prevede l'impiego di un rotore a magnete permanente (e quindi privo di avvolgimenti) dotato di un certo numero di "denti", con lo statore costituito da numerose espansioni polari, alimentate da altrettanti avvolgimenti



Motori passo-passo

Descrizione

Questi avvolgimenti sono collegati in modo da portare esternamente un numero fisso di fili, che può essere di 4, 5 o 6 conduttori. A questi conduttori (fra di loro non intercambiabili) viene applicata la corretta sequenza di impulsi per l'avanzamento del rotore. I "passi" ottenibili – che dipendono dalle modalità costruttive del motore – possono andare da 40 a 200, a seconda dei modelli.



Esempio 9

```
#include <Stepper.h>

#define STEPS 100

Stepper stepper(STEPS, 8, 9, 10, 11);
int previous = 0;

void setup() {
  stepper.setSpeed(30);
}

void loop() {
  int val = analogRead(0);
  stepper.step(val - previous);
  previous = val;
}
```

"Al motore passo-passo collegato sui pin 8,9,10 e 11 viene per prima cosa impostata la velocità di rotazione pari a 30 RPM; di seguito il valore letto dall'ingresso 0, opportunamente elaborato, determina il numero di passi che il motore deve compiere."

GSM/GPRS Telit GM862

Utilizzo del dispositivo GM862

Il modem della **Telit GM862** è utilizzato nei due esempi seguenti per inviare un SMS e per aprire una collegamento internet sfruttando il servizio di trasmissione dati GPRS offerto dalla rete GSM.

Una volta inserita la SIM di un qualsiasi gestore di telefonia mobile il dispositivo ricerca la rete e, nel caso sia presente, si collega ad essa.

Per comandare il modem si utilizzano i comandi AT.

Comandi AT

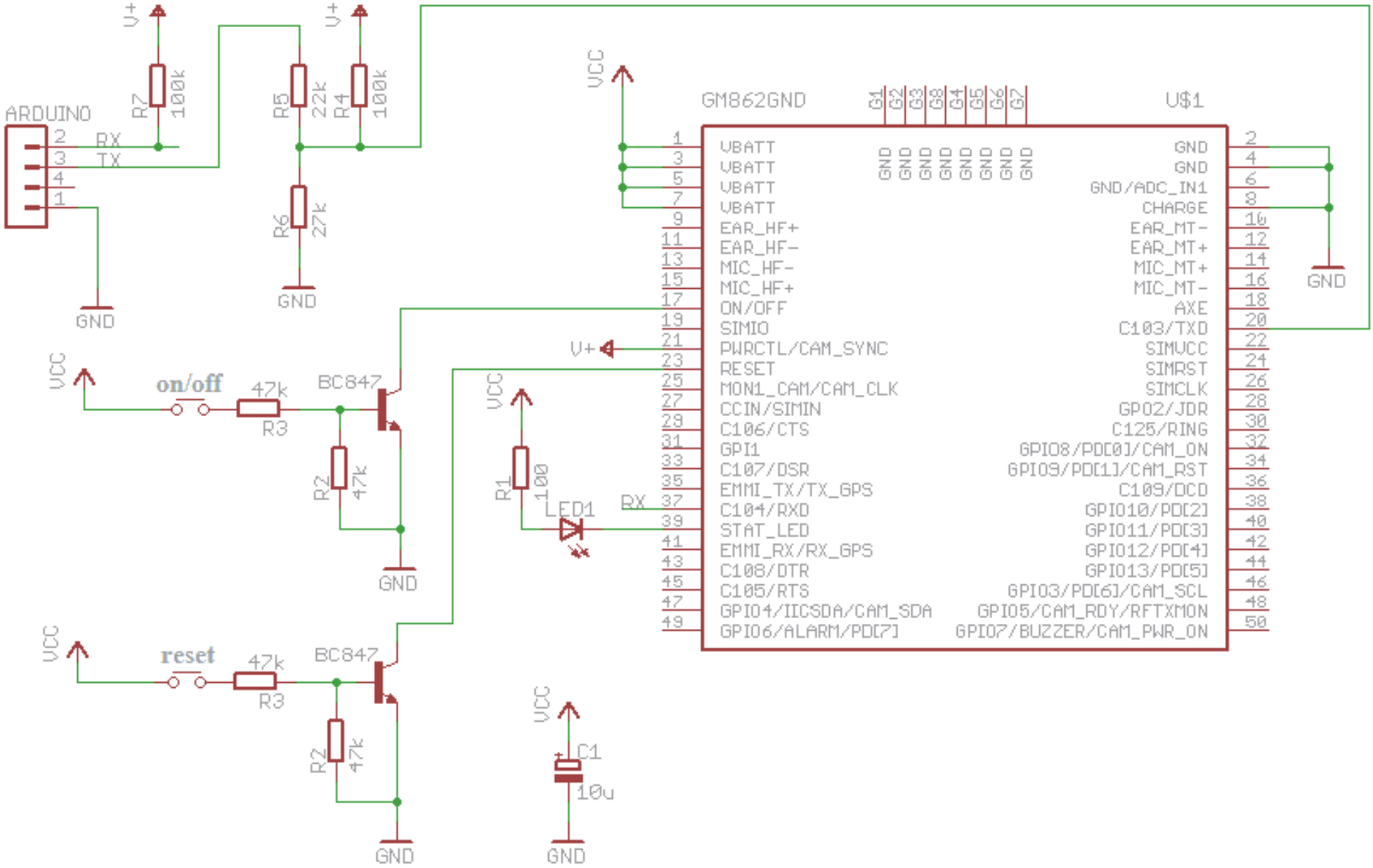
Descrizione

La maggior parte degli attuali modem utilizza i **comandi AT Hayes**, uno specifico insieme di comandi originalmente sviluppato per il modem *Hayes Smartmodem* da 300 baud.

Ogni funzione del modem è governata dal relativo comando AT (che sta per ***ATtention***, attenzione). Per inviare un comando occorre trasmettere sulla porta seriale del modem una stringa **ASCII** formata da AT seguito da uno o più comandi e da un carattere di ritorno a capo (CR).

GSM/GPRS

Schema elettrico



Esempio 10

Programma per l'invio di un SMS

"Dopo aver caricato lo sketch nell'ATmega328, viene inviato un SMS attraverso il modem ad esso collegato tramite una seriale software."

```
01 #include <SoftwareSerial.h>
02
03 int rxPin = 0;
04 int txPin = 1;
05
06 // set up a new serial port
07 SoftwareSerial serial=SoftwareSerial(rxPin,txPin);
08
09 void setup()  {
10   pinMode(rxPin, INPUT);
11   pinMode(txPin, OUTPUT);
12
13   // set the data rate for the SoftwareSerial port
14   serial.begin(9600);
15
16   // Set SMS to text mode. Note it is critical
17   // to use \r\n to end each line
18   // The delays are also critical, without them,
19   // you may lose some of the
20   // characters of your message
21
22   serial.print("AT+CMGF=1\r\n");
23   delay(300);
24   serial.print("AT+CMGS=");
25   delay(300);
26   // Replace with a valid phone number
27   serial.print("+14081234567\r\n");
28   delay(300);
29   serial.print("Hello from Arduino.");
30   delay(300);
31
32   // End the SMS with a control-z
33   serial.print(0x1A,BYTE);
34 }
35
36 void loop() {
37 }
```

…per divertirsi un po' con i comandi AT

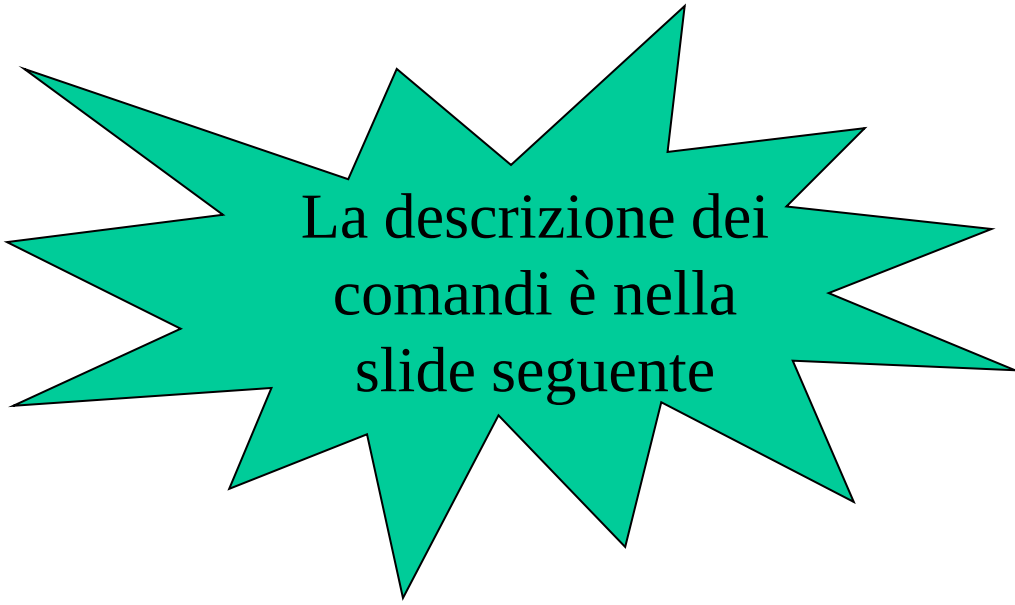
Inviamo un SMS dal nostro telefonino utilizzando i comandi AT:

a) collegare il proprio cellulare al PC (seguire le istruzioni del produttore);

b) aprire un programma che simula un terminale (HyperTerminal);

c) digitare i seguenti comandi:

1: AT
2: OK
3: AT+CMGF=1
4: OK
5: AT+CMGW="+85291234567"
6: > **Un semplice messaggio :-)**
7: +CMGW: 1
9: OK
10: AT+CMSS=1
11: +CMSS: 20
13: OK



…per divertirsi un po’ con i comandi AT

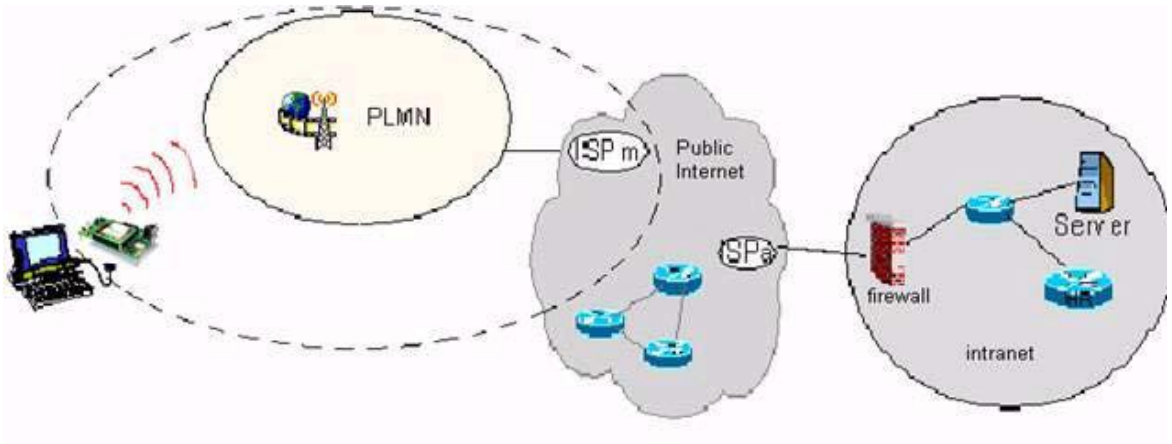
• **Linea 1**: "AT" viene spedito al modem GSM/GPRS per testare la connessione. Il modem risponde con un codice di risposta (OK alla linea 2)
• **Linea 3**: Il comando AT+CMGF viene usato per istruire il modem ad operare in modalità testuale per gli SMS. Il codice di risposta OK alla linea seguente indica che linea di comando AT+CMGF=1 è stata eseguita con successo;
• **Linee 5-6**: Il comando AT+CMGW viene usato per memorizzare nel modem il messaggio, mentre +85291234567 è il numero telefonico del destinatario. Dopo aver digitato tale numero, premete il tasto Enter. Il modem a questo punto dovrebbe ritornare il prompt ">" e a questo punto potete incominciare a scrivere il vostro SMS (in questo caso "Un semplice messaggio"). Quando avete finito, premete Ctrl+Z;
• **Linea 7**: +CMGW: 1 ci dice che l'indice assegnato al messaggio è 1. Questo indica la locazione dell'SMS nella memoria del modem;
• **Linea 9**: Il risultato OK indica che l'esecuzione del comando +CMGW ha avuto successo;
• **Linea 10**: il comando +CMSS viene usato per spedire i messaggi dalla memoria del modem. 1 indica l'indice dell'SMS ottenuto alla linea 7;
• **Linea 11**: +CMSS: 20 ci dice che il numero di riferimento assegnato all'SMS è 20;
• **Linea 13**: Il risultato OK indica che l'esecuzione del comando +CMSS ha avuto successo.

GSM/GPRS

Collegamento GPRS

Il **General Packet Radio Service (GPRS)** è una delle tecnologie di telefonia mobile progettata per realizzare il trasferimento di dati a media velocità.

Per attivare una connessione GPRS, non c'è bisogno di comporre un numero di telefono, ma solo dei parametri per connettersi ad un internet point della rete GPRS , forniti da un Internet Service Provider (ISP).



GSM/GPRS

Collegamento GPRS

Lo schema elettrico per collegare Arduino al modem è, evidentemente, lo stesso visto nell'esempio 10.
I comandi AT per collegarsi ad internet e ritrovare una pagina HTML sono i seguenti (parametri Vodafone):

AT+CGDCONT=1,"IP",“web.omnitel.it","0.0.0.0",0,0

Con questo comando si vanno ad impostare tutte le informazioni che identificano un punto d'accesso alla rete internet

Nota: per maggiori informazioni vedere “Easy GPRS User Guide” - Telit

GSM/GPRS

Collegamento GPRS

AT#SCFG=1,1,300,90,600,50

Comportamento dello stack TCP/IP. E' possibile impostare tutti i valori di timeout e le dimensioni dei pacchetti.

AT#SGACT=1,1

Richiesta di attivazione di un contesto GPRS.

AT#SD=1,0,80,"www.telit.com"

Richiesta TCP/UDP di apertura della connessione con un internet host.

Collegamento GPRS

```
GET / HTTP/1.1
Host: www.telit.com
Connection: keep-alive
```

In risposta dall'host interrogato si ottiene la pagina desiderata!

GPS

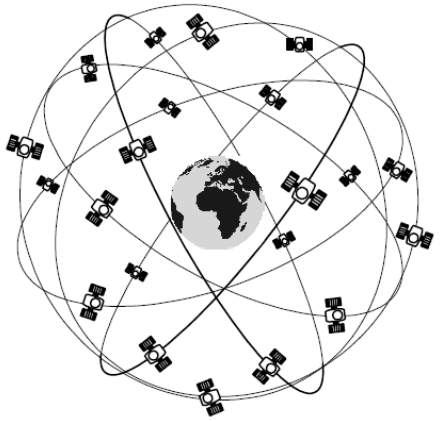
Descrizione

Il sistema **GPS** (**Global Positioning System**) è un sistema satellitare basato su una costellazione di **24 satelliti** orbitanti intorno alla terra.

Esso è in grado di fornire, con estrema precisione, le coordinate geografiche, la quota e la velocità di qualsiasi mezzo mobile in ogni punto della Terra e per l'intero arco delle ventiquattro ore.

I satelliti emettono con continuità opportuni segnali di **tempo** e **orbitali** che servono ai ricevitori per calcolare la posizione del satellite.

I dati elaborati dai ricevitori sono disponibili agli utenti sotto forma di stringhe chiamate ***sentence*** (**sentenze**).



NMEAO183

Sentenza GPRMC

I GPS emettono molti tipi di sentenze, secondo lo standard **NMEA0183**, che contengono dati utili ad una moltitudine di applicazioni.

La **sentenza GPRMC** è molto importante perché fornisce informazioni circa l'orario, la posizione al suolo e altitudine, velocità verticale e al suolo e via di seguito.

Esempio di sentenza NMEA0183:

$GPRMC,l61229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598,,*10

NMEAO183

Sentenza GPRMC

$GPRMC,l61229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598,,*10

- 161229.487 é l'informazione UTC (ore, minuti, secondi, millesimi di secondo) ossia indica che l'ora di sistema è 16 ore, 12 minuti, 29 secondi e 487 millesimi;
- la lettera che segue indica se i dati sono validi o meno (A = dati validi, V = dati non validi);
- 3723.2475 è la latitudine e la lettera che segue il suo riferimento (N = nord, S = sud);
- 12158.3416 è, invece, la longitudine, riferita ad est o ad ovest a seconda che la lettera seguente sia E o W; in questo caso é W (west = ovest);

NMEAO183

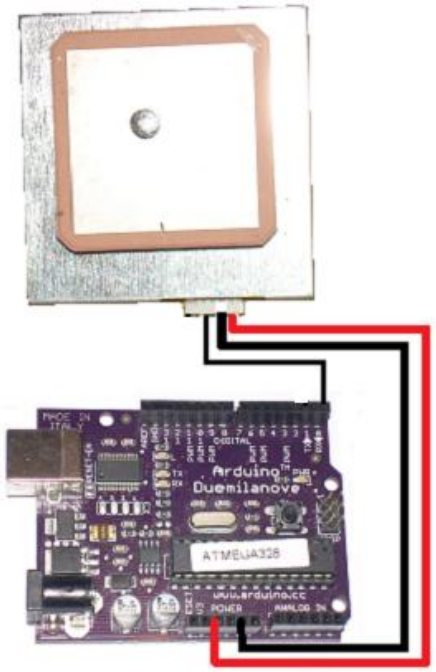
Sentenza GPRMC

$GPRMC,l61229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598,,*10

- 0.13 è la velocità al suolo, espressa in knot;
- 309.62 corrisponde allo spostamento orizzontale espresso in gradi sessagesimali (normalmente rispetto al meridiano, di Greenwich);
- 120598 é la data UTC, espressa nel formato ggmmaa;
- il campo seguente (tra le virgole che seguono) è riservato alla variazione magnetica che può affliggere la posizione rilevata orizzontalmente (spostamento orizzontale) e può non contenere dati; vale E quando la variazione è uno spostamento a est e W quando, invece, la variazione corrisponde a uno spostamento ad ovest.
- la stringa termina con il checksum, che in questo caso è *10.

Esempio 11

```
byte val;

void setup() {
  Serial.begin(4800);
}

void loop() {
  while (Serial.available()){
     val = Serial.read();
     Serial.write(val);
  }
}
```



"I dati trasmessi sulla seriale dal GPS vengono intercettati e inviati tramite i comandi seriali al Personal Computer dove possono essere visualizzati ed interpretati."

ARDUINO

Parte_3

Xbee, Bluetooth, SD, Ethernet

Arduino Ethernet shield

Descrizione

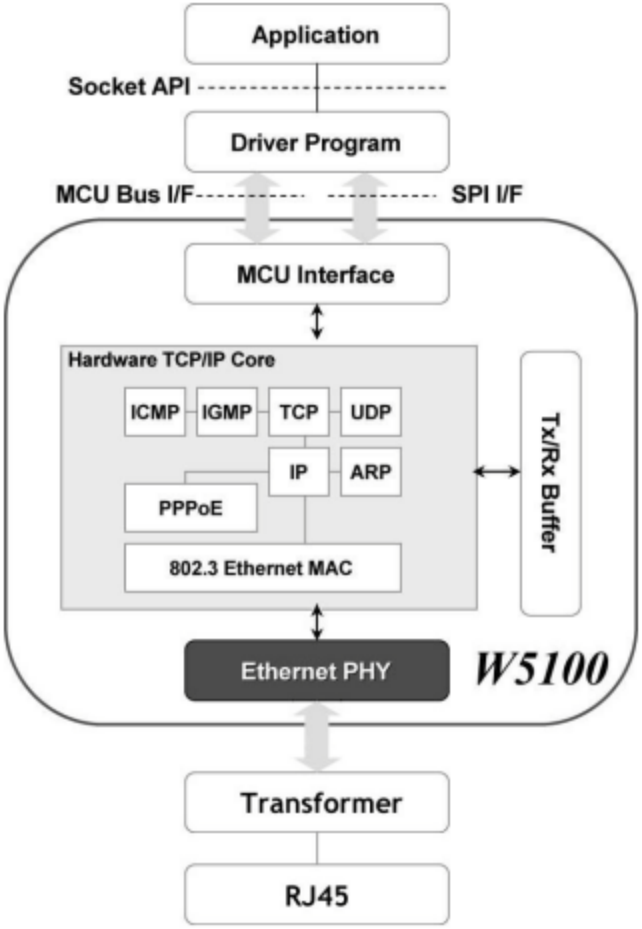
Arduino Ethernet shield permette ad una Arduino board di collegarsi ad Internet usando la libreria Ethernet.

Arduino Ethernet shield

Descrizione

Arduino Ethernet shield è basato sull'integrato **Wiznet W5100** che è un 10/100 ethernet controller, progettato per applicazioni embedded.

Questo dispositivo fornisce lo stack **TCP/IP** dal livello fisico fino al livello di trasporto.



Arduino Ethernet shield

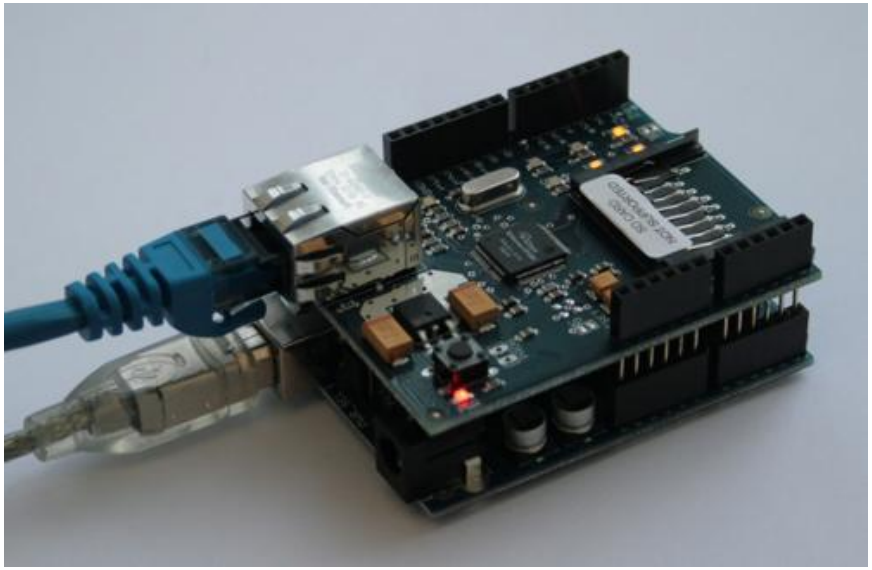
Descrizione

• Arduino usa i pin digitali 10, 11, 12, e 13 (SPI) per comunicare con il W5100 della ethernet shield.

• Questi pin non possono essere più usati per collegare altri dispositivi.

• La shield è fornita di un jack standard RJ45.

• Il bottone di reset presente sulla shield resetta sia la Arduino board sia la Ethernet shield.



Arduino Ethernet shield

Descrizione

La Ethernet shield contiene dei LEDs che indicano lo stato del dispositivo:

- **PWR**: indica che la Arduino board e la shield sono alimentati;
- **LINK**: indica la presenza di un collegamento di rete e lampeggia quando la shield trasmette o riceve i dati;
- **FULLD**: indica che il collegamento di rete è full-duplex;
- **100M**: indica la presenza di una rete a 100 Mb/s;
- **RX**: lampeggia quando la shield riceve i dati;
- **TX**: lampeggia quando la shield spedisce i dati;
- **COLL**: lampeggia quando viene individuata una collisione.

Arduino Ethernet shield

Ethernet library

Ethernet class

Inizializza la libreria Ethernet e setta i parametri della rete (**mac** , **IP** , **gateway** , **subnet**).

- begin()

Server class

Crea il server che spedisce e riceve i dati dai clients collegati.

- Server()
- begin()
- available()
- write()
- print()
- println()

Arduino Ethernet shield

Ethernet library

Client class

Crea i clients che possono connettersi al server e spedire o ricevere dei dati.

- Client()
- connected()
- connect()
- write()
- print()
- println()
- available()
- read()
- flush()
- stop()

Arduino BT (Bluetooth)

Descrizione

La **Arduino BT** è una Arduino board con un modulo Bluetooth integrato che permette la comunicazione wireless.
Il modulo bluetooth usato è il **Bluegiga WT11** e può essere configurato con appositi comandi (si veda manuale iWRAP) sopra un collegamento seriale.

Arduino BT (Bluetooth)

Descrizione

La Arduino BT è provvista del microcontrollore **ATmega168** precaricato con un bootloader che permette di caricare gli sketch nella board attraverso il collegamento bluetooth.

Nota: evidentemente il Personal Computer contenente lo sketch deve essere anch'esso provvisto di collegamento bluetooth.

Arduino BT (Bluetooth)

Descrizione

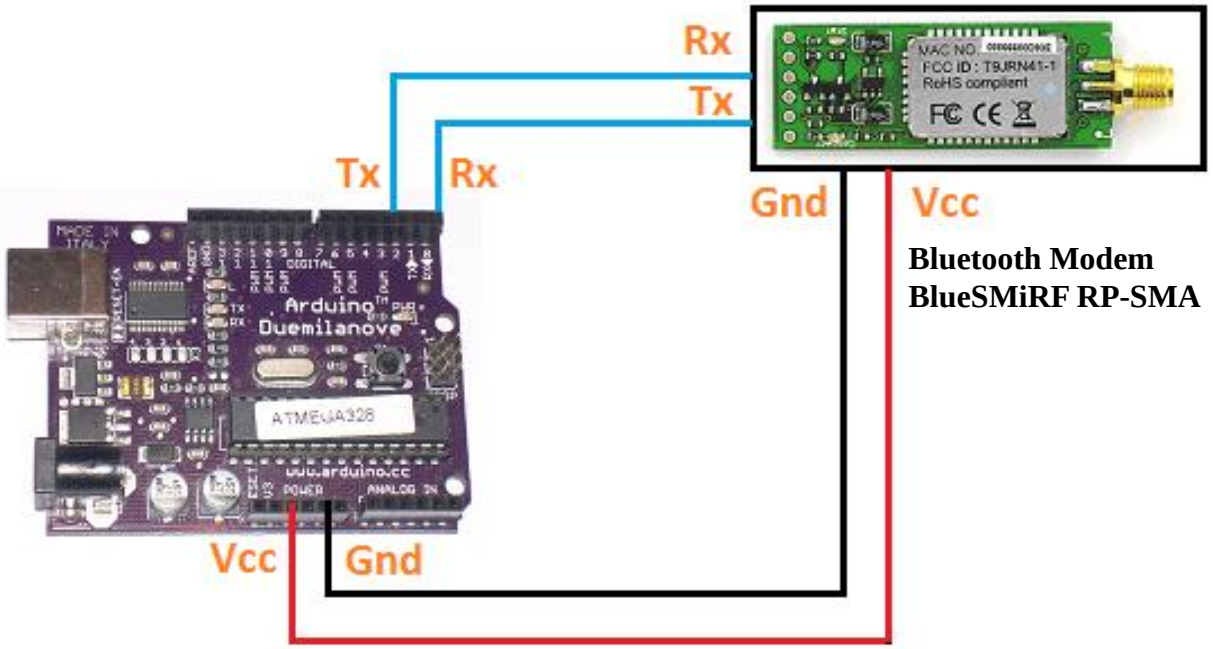
La comunicazione tra Arduino BT e il PC (quando necessaria) può avvenire esattamente come per le altre Arduino board (non bluetooth) utilizzando i semplici comandi seriali per inviare e ricevere dati; infatti il modulo bluetooth realizza un collegamento seriale virtuale.

Per tale motivo non è necessario l'utilizzo di particolari librerie per la gestione del dispositivo, ma sono sufficienti i normali comandi seriali come print() o println().

Arduino BT (Bluetooth)

… per i più appassionati

Il collegamento bluetooth può anche essere realizzato con una Arduino Duemilanove board e un modem bluetooth, collegandoli come in figura.
Anche in questo caso, dopo aver configurato il modem (vedere manuale del dispositivo), è possibile comunicare con altri moduli bluetooth semplicemente utilizzando i normali comandi seriali come print() o println().



Secure Digital SD

Descrizione

Secure Digital (chiamate più brevemente **SD**) è il più diffuso formato di schede di memoria.
Sono dispositivi elettronici utilizzati per memorizzare in formato digitale grandi quantità di informazioni all'interno di memorie flash.

Arduino SD

SD library

SD class

La classe SD fornisce funzioni per accedere alla scheda SD e manipolare i suoi file e directory.

- begin()
- exists()
- mkdir()
- open()
- remove()
- rmdir()

Arduino SD

SD library

File class

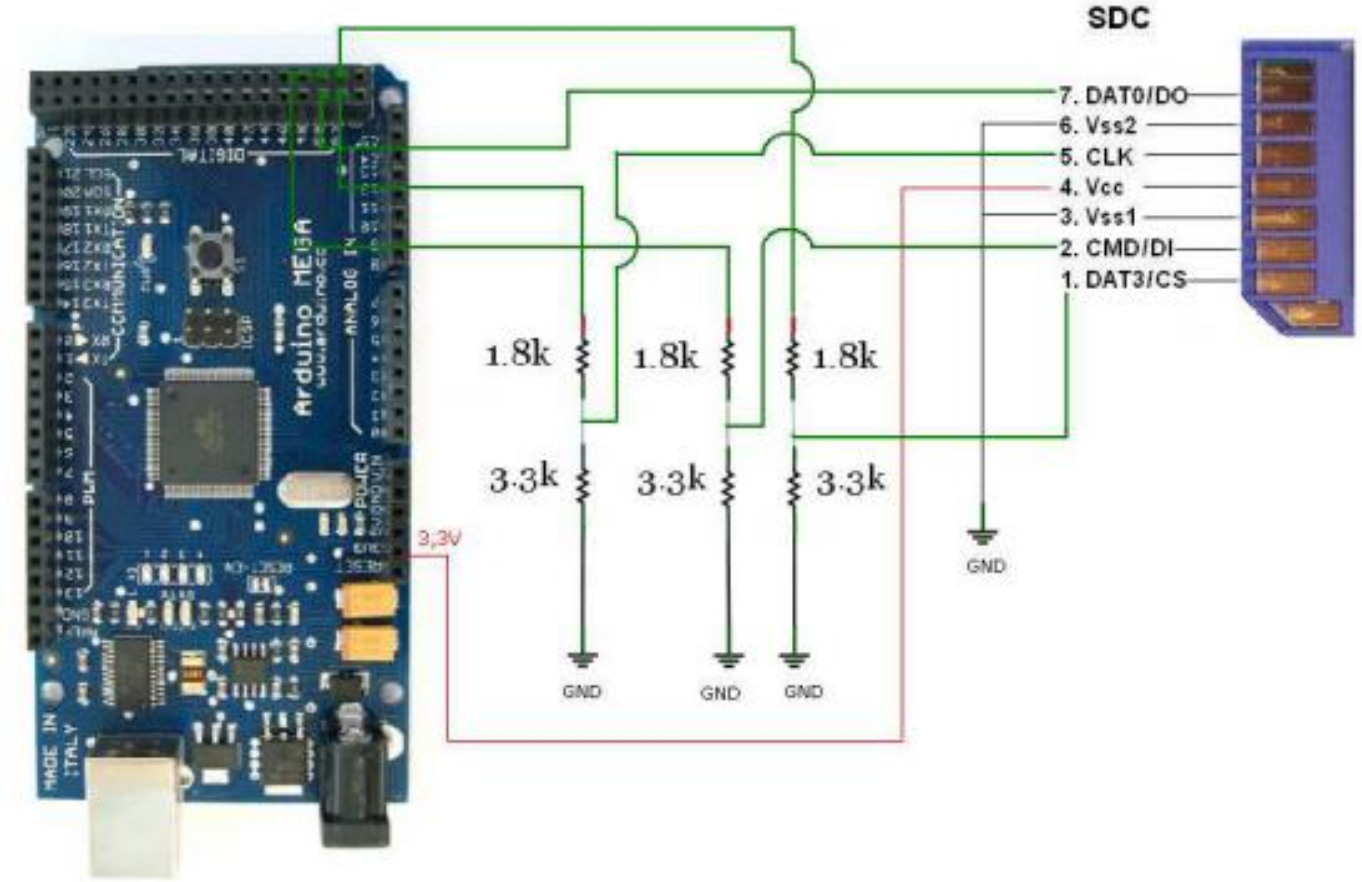
La classe File permette la lettura e la scrittura di singoli file sulla scheda SD.

- available()
- close()
- flush()
- peek()
- position()
- print()
- println()
- seek()
- size()
- read()
- write()

Secure Digital SD e Arduino

Schema elettrico

Il collegamento tra Arduino (in figura nella versione Arduino Mega) e la SD, può essere realizzato utilizzando lo schema seguente.



Secure Digital SD e Arduino

Libreria per SD

La scrittura su files di testo dei dati rilevati è resa possibile attraverso l'uso della libreria **Fat16lib**, di grandissima utilità in questo ambito, poiché rende immediate tutte le operazioni da eseguire a corredo della memorizzazione.

La bibreria FAT16lib implementa una versione minimale del file system FAT16 sulle memory cards SD.

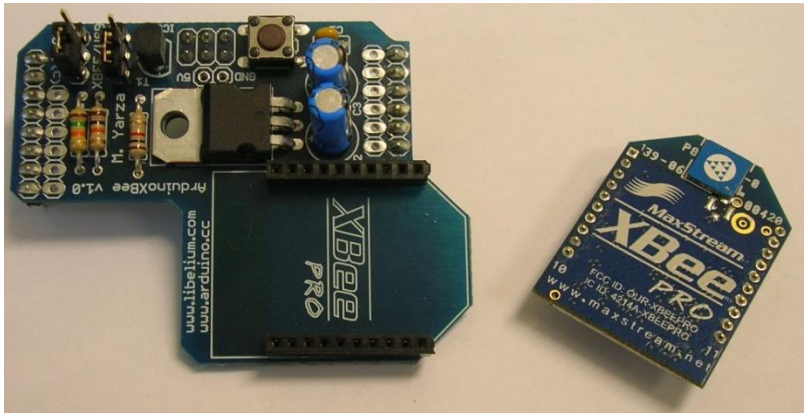
Essa supporta:

la lettura, scrittura, creazione, cancellazione e troncamento di file.

XBee shield

Descrizione

La shield permette di collegare semplicemente un modulo Xbee alla Arduino board, al fine di realizzare una **WPAN** (**Wireless Personal Area Network**).



Il modulo XBee

Descrizione

Il modulo **XBee** è una soluzione compatibile con lo standard **ZigBee**/**IEEE 802.15.4** che soddisfa la necessità di una rete a basso costo e a basso consumo, pensata soprattutto per l'utilizzo con sensori.

CARATTERISTICHE TECNICHE:

- Frequenza operativa 2.4 GHz
- Potenza RF 1 mW (fino a 100m di portata)
- Possibilità di antenna filo, Chip oppure connettore U.FL. RF
- Range di Temperatura Industriale (-40 °C 85°C)

Il modulo XBee

Vantaggi

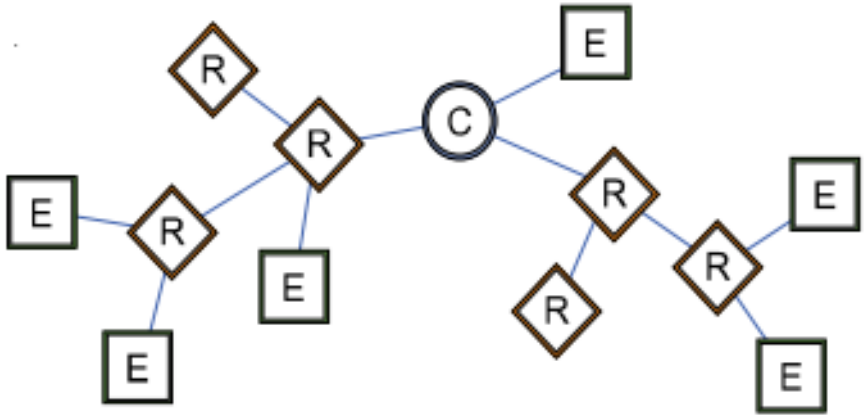
• **bidirezionale** , in questo modo e' possibile testare facilmente (da entrambe i lati) se il sistema sta funzionando correttamente.

• **indirizzamento univoco** di questi moduli. Ogni XBee ha un numero seriale univoco. Questo significa che due o più unita possono essere settate per parlare esclusivamente tra loro, ignorando tutti i segnali di altri moduli.

• il protocollo XBee consente **diversi numeri di canali**; settando differenti unità in differenti canali, possono essere minimizzate eventuali interferenze.

XBee shield

ZigBee Network



Il dispositivo **Coordinator** in genere è unico per ogni rete ed è il dispositivo che forma la rete, allocando gli indirizzi di rete e tenendo memoria della tabella di allocazione con l'associazione degli indirizzi dei vari dispositivi di rete.

Coordinator C

Router R

End Device E

Il dispositivo **Router** è opzionale e consente di estendere il range della rete consentendo a più nodi di comunicare tra loro. Questo può eseguire anche funzioni di monitoraggio e/o controllo come il dispositivo **End Device**.

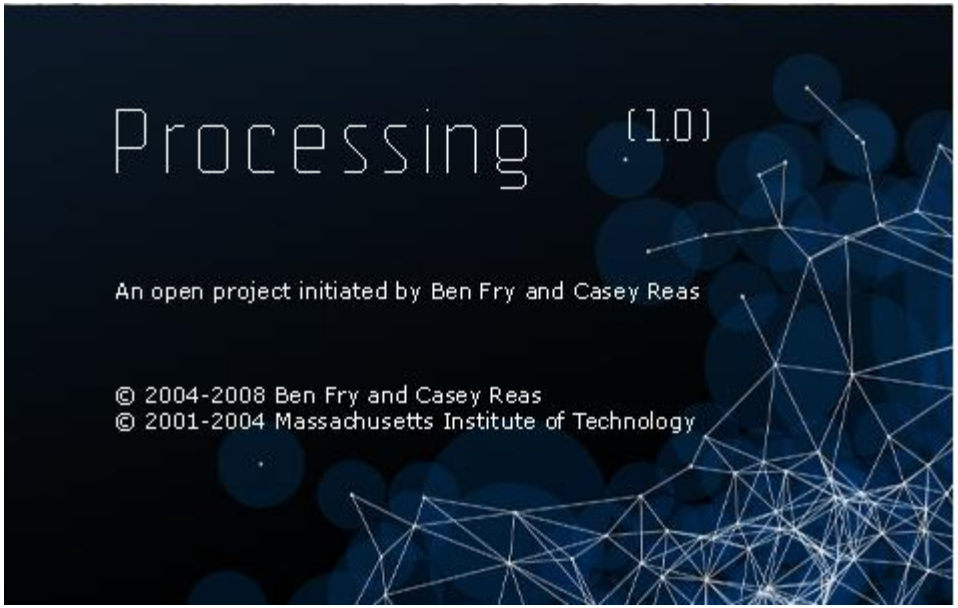
ARDUINO

Parte_4

Arduino e Processing

Processing

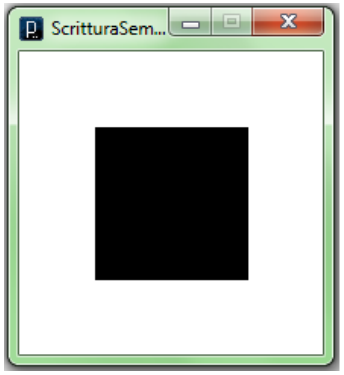
E’ un linguaggio di programmazione che permette anche ad utenti meno esperti di realizzare lavori di grafica accattivanti.
Ha delle librerie per gestire degli oggetti di tipo Arduino.
Gestisce il collegamento seriale in modo semplice ed intuitivo.

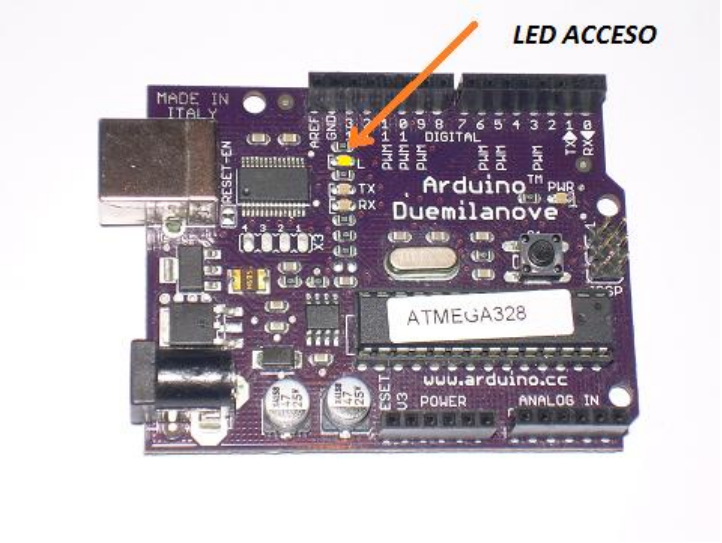


Processing

Nelle due slide seguenti possiamo apprezzare come sia semplice ed intuitivo lavorare con Processing per comandare Arduino.

"Il programma in Processing crea una applicazione Windows con al suo interno un quadrato; passando sopra il quadrato con il muose viene inviato ad Arduino il comando di accendere il led collegato sul pin 13."





Processing

Programma da inserire in Processing

```
import processing.serial.*;
Serial myPort;
int val;

void setup(){
  size(200, 200);
  String portName = Serial.list()[0];
  myPort = new Serial(this, portName, 9600);}

void draw(){
  background(255);
  if (mouseOverRect() == true) {
                fill(204);
                myPort.write('H');
   }else {  fill(0);
                myPort.write('L');}
   rect(50, 50, 100, 100);}

boolean mouseOverRect() {
  return ((mouseX >= 50) && (mouseX <= 150) && (mouseY >= 50) && (mouseY <= 150));}
```

Processing

Programma da inserire in Arduino

```
char val;
int ledPin = 13;

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  if (Serial.available()) {
    val = Serial.read();
  }
  if (val == 'H') {
            digitalWrite(ledPin, HIGH);
  } else {
            digitalWrite(ledPin, LOW);
 }}
```

ARDUINO

Parte_5

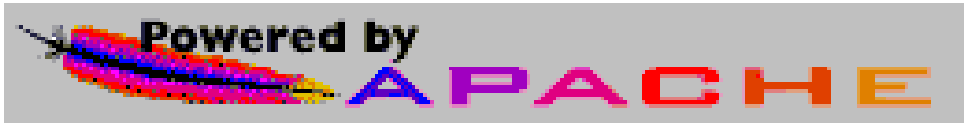
Arduino e PHP

Ambiente di lavoro

In questa parte del corso vedremo come sia possibile utilizzare il **PHP** per inviare dei comandi ad una Arduino board.

Per fare questo dovremo prima predisporre l'ambiente di lavoro adatto, installando il **Server Web Apache** e il **PHP** (per farlo seguire una delle numerosissime guide presenti su Internet).

Immagine PHP

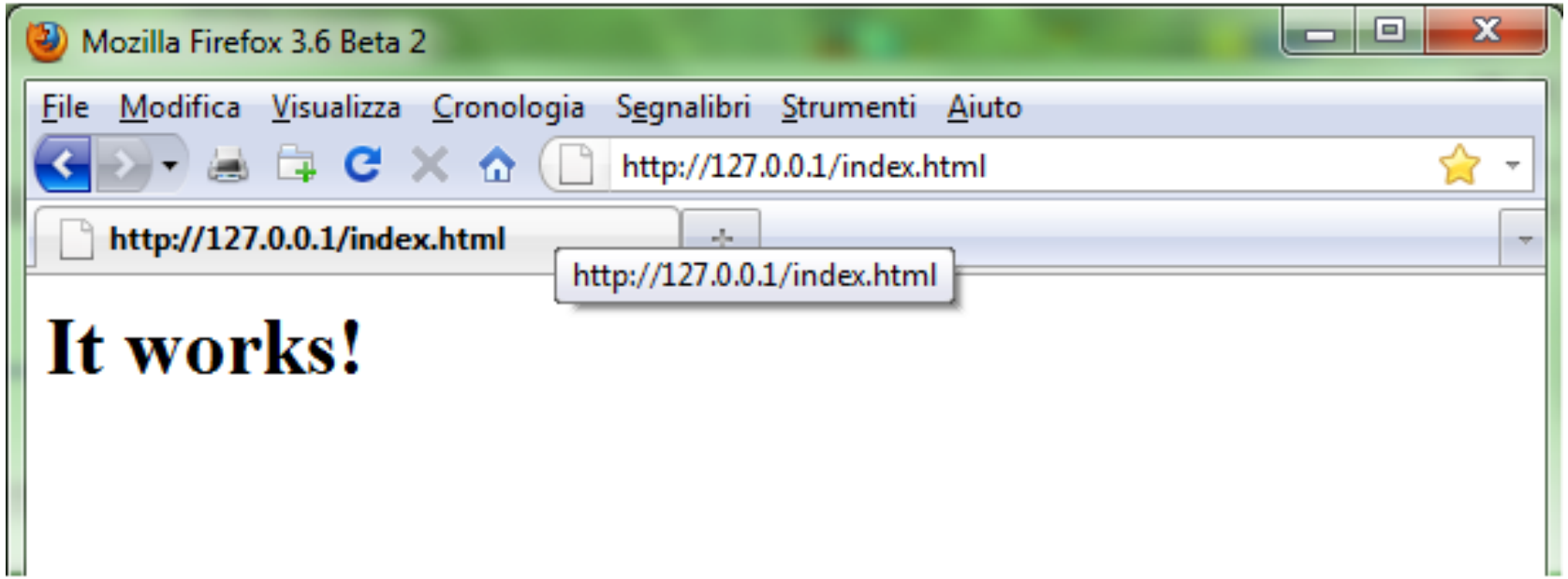


Ambiente di lavoro

Assicurarsi che il Server Web funziona correttamente: con un browser collegandosi a

http://localhost/index.html

si dovrà vedere sullo schermo la scritta:
It works!

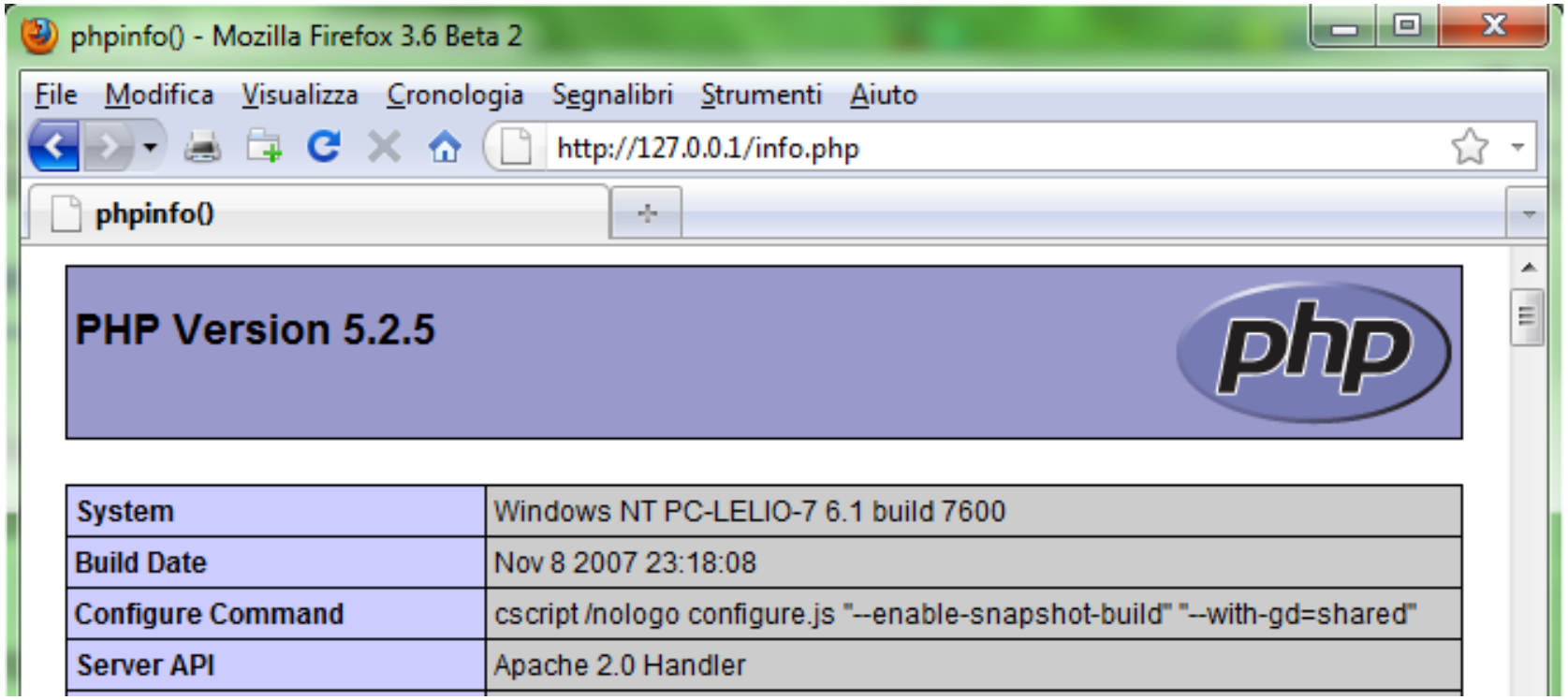


Ambiente di lavoro

Assicurarsi che anche il PHP lavori correttamente:
con un browser collegandosi a

http://localhost/info.php

si dovrà vedere sullo schermo una tabella con le caratteristiche del PHP



PHP

Programma da inserire in PHP

Il PHP è in grado di inviare dati attraverso la porta seriale. Alla riga 2 viene aperta la porta seriale in modalità scrittura e nella riga 3 si invia sul canale seriale il carattere “1”; la riga 4 contiene l’istruzione per chiudere la porta seriale.

```
1:  <?php
2:  $fp = fopen("com4", "w");
3:  fwrite($fp, chr(1));
4:  fclose($fp);
5:  echo "<html>";
6:  echo "<body>";
7:  echo "<h1>Il led collegato al pin 13 della Arduino board si accende!</h1>";
8:  echo "</body>";
9:  echo "</html>";
10: ?>
```

Arduino

Programma da inserire in Arduino 1/2

Lo sketch da inserire in Arduino contiene le istruzioni che permettono di leggere i dati presenti sulla seriale;
nel caso che il dato letto è il carattere “1” allora il led collegato al pin 13 lampeggia due volte, mentre nel caso che il dato letto sia il carattere “2” (o più in generale un numero pari) allora il led collegato al pin 13 lampeggia una sola volta.

```
int ledPin = 13;
int usbnumber = 0;

void setup() {
   pinMode(ledPin, OUTPUT);
   Serial.begin(9600);
}

void loop() {
   if (Serial.available() > 0) {
      usbnumber = Serial.read();
   }
```

Arduino

Programma da inserire in Arduino 2/2

```
if (usbnumber > 0) {
        if (usbnumber % 2 == 0){
            digitalWrite(ledPin, HIGH);
            delay(300);
            digitalWrite(ledPin, LOW);
            delay(300);
        }else{
            digitalWrite(ledPin, HIGH);
            delay(300);
            digitalWrite(ledPin, LOW);
            delay(300);
            digitalWrite(ledPin, HIGH);
            delay(300);
            digitalWrite(ledPin, LOW);
            delay(300);
        }
        usbnumber = 0;
    }
}
```

Arduino

Procedimento per avviare il processo

Copiare il file php_arduino.php nella cartella seguente:
C:\programmi\Apache Software Foundation\Apache2.2\htdocs.

Lanciare il browser (ad es. Firefox) e digitare sulla barra degli indirizzi il seguente URL:

http://127.0.0.1/php_arduino.php

Sullo schermo comparirà la seguente scritta e contemporaneamente il led collegato sul pin 13 della Arduino board lampeggerà due volte.