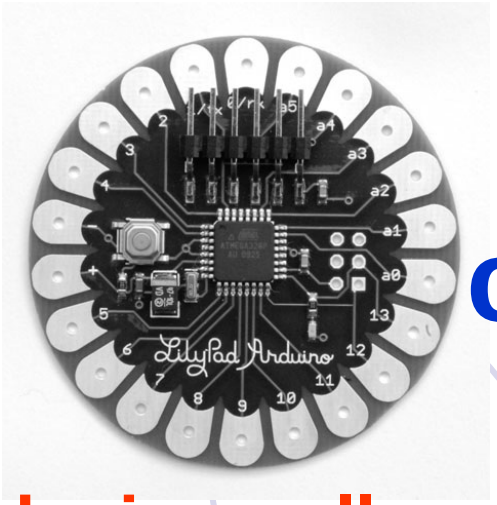


Progetti reali con ARDUINO

Introduzione alla scheda Arduino (parte 3ª)

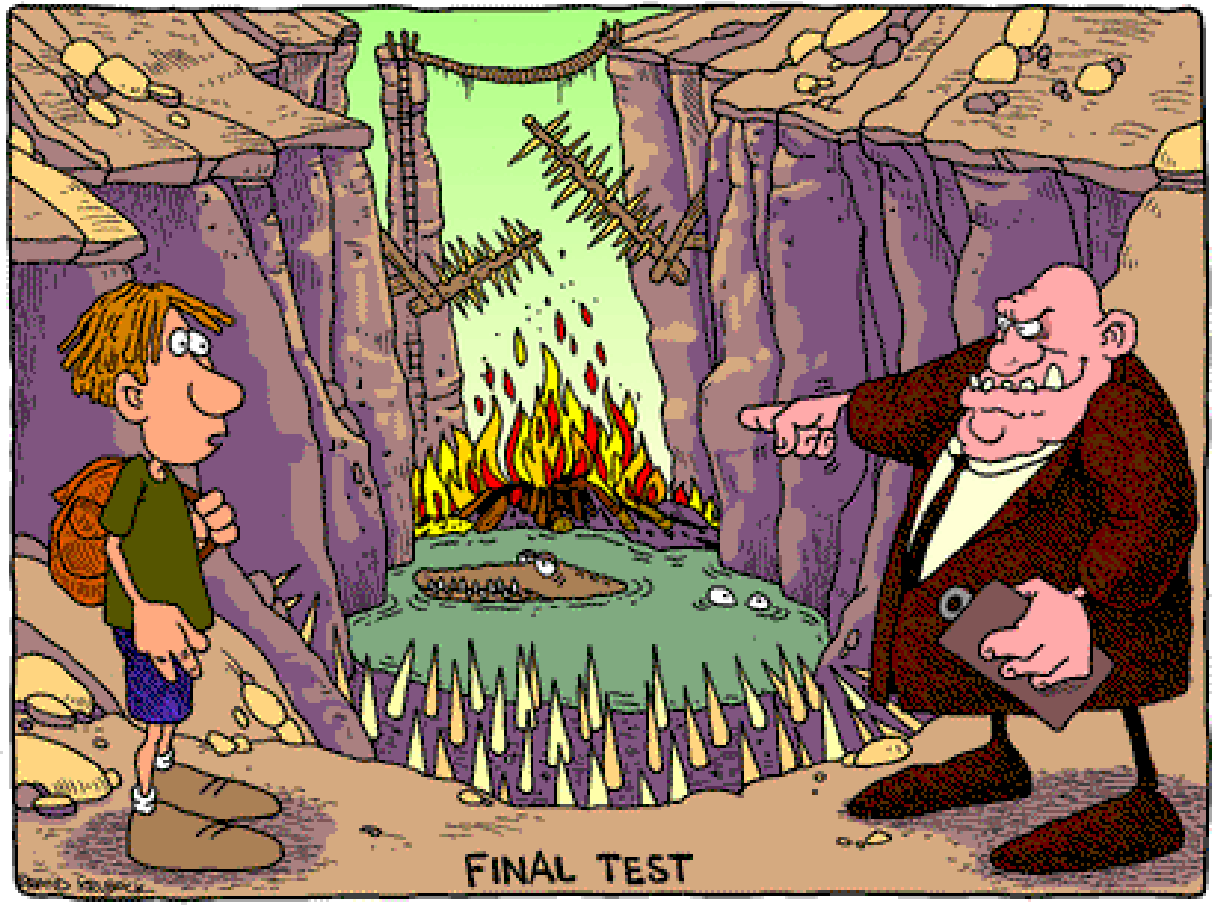
giugno 2013 – Giorgio Carpignano

I.I.S. PRIMO LEVI

C.so Unione Sovietica 490 (TO)

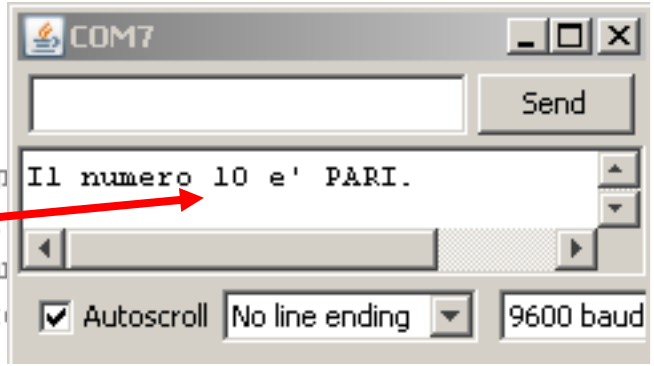
Materiale didattico: **www.iisprimolevi.it**

Esercizio da svolgere subito!



Scrivere un programma in modo tale che venga inserito un numero intero nella variabile denominata "valore" e stabilisca se il numero e' **pari** o **dispari**.

Esercizio pari o dispari (1°metodo)




```
/*  I.I.S. Primo LEVI - Torino    Data: 03/12/2010
  Esercizio N. 5    Progetto: pari_dispari_1     Autore: G. Carpig
  Descrizione: Scrivere un programma in modo tale che venga inseri
  intero nella variabile denominata "valore" e stabilisca se il nu
byte valore = 10; // variabile a 8 bit con il numero da controllar
void setup()
{// inizializza la seriale RS232 con 9600 baud, 8 bit dati, nessuna parità e 1 bit di stop
  Serial.begin(9600);
}
```

pari_dispari_1.ino

```
void loop() // programma principale
{ // la condizione viene verificata se e' uguale a 0, ma prima viene effettuata l'AND (&)
  // con la variabile "valore", cioe' si avra' 10 & 1 che vale in binario "00001010" & "00000001"
  // diventa, effettuando la logica AND su ogni singolo bit "00000000" che corrisponde a 0.
  if ((valore & 1) == 0)  // se il confronto vale 0
    {  // si e' in presenza di un numero PARI
      Serial.print("Il numero ");
      Serial.print(valore, DEC); // trasmissione sulla seriale del numero
      Serial.print(" e' PARI.");
     }
  else
    {  // altrimenti il numero e' DISPARI
      Serial.print("Il numero ");
      Serial.print(valore, DEC); // trasmissione sulla seriale del numero
      Serial.print(" e' DISPARI.");
     }
  while(1);  // loop infinito (blocca il micro con questa istruzione)
}
```

Esercizio pari o dispari (2°metodo)

```
/*  I.I.S. Primo LEVI - Torino    Data: 03/12/2010
  Esercizio N. 5    Progetto: pari_dispari_2     Autore: G. Carpignano
  Descrizione: Scrivere un programma in modo tale che venga inserito un numero
  intero nella variabile denominata "valore" e stabilisca se il numero e' pari o dispari.*/
byte valore = 10; // variabile con il numero da controllare se pari o dispari
void setup()
{// inizializza la seriale RS232 con 9600 baud, 8 bit di dati, nessun bit di parità, 1 bit di stop
  Serial.begin(9600);
}

void loop() // programma principale
{
  if ((valore % 2) == 0)  // se il resto della divisione per 2 vale 0
    {  // si e' in presenza di un numero PARI
      Serial.print("Il numero ");
      Serial.print(valore, DEC); // trasmissione sulla seriale del numero
      Serial.print(" e' PARI.");
     }
  else
    {  // altrimenti il numero e' DISPARI
      Serial.print("Il numero ");
      Serial.print(valore, DEC); // trasmissione sulla seriale del numero
      Serial.print(" e' DISPARI.");
     }
  while(1);  // loop infinito (blocca il micro con questa istruzione)
}
```

Il carattere " %" permette di calcolare il modulo, ovvero il resto della divisione.

In questo esempio il resto della divisione per 2 può valere solo "0" oppure "1".

pari_dispari_2.ino

Istruzione **while()**

- L’espressione presente all'interno della parentesi tonda (**condizione di ripetizione**) viene valutata all’inizio di ogni ciclo.
- Se la condizione risulta **VERA** si eseguono tutte le istruzioni presenti tra le parentesi graffe.
- Se la condizione risulta **FALSA** (cioè se è uguale a zero) il programma salta all'esecuzione della prima istruzione dopo la parentesi graffa chiusa.
- Se inizialmente la condizione ha valore zero, il corpo del ciclo non viene mai eseguito.
- In generale, *non è noto* ***quante volte*** l’istruzione sarà ripetuta.
- (Attenzione che qualsiasi valore memorizzato in una variabile purchè sia diverso da zero è VERO).

Istruzione **while()** con esempio

while_1.ino

```
/*  I.I.S. Primo LEVI - Torino    Data: 03/12/2010
  Esercizio N. 7    Progetto: while_1     Autore: G. Carpignano
  Descrizione: Controllare se un input digitale (pulsante collegato al pin 7)
   e' premuto, e per tutto il tempo che rimane tale accendere un led
   (collegato al pin 13), mentre se il pulsante viene rilasciato spegnere il led.*/
int led = 13; // definizione della variabile "led" utilizzata per scrivere sul pin 13
int pulsante = 7; // definizione della variabile "pulsante" utilizzata per leggere sul pin 7
void setup() // funzione di inizializzazione dei INPUT/OUTPUT
{
  pinMode(led, OUTPUT); // inizializza il pin 13 come OUTPUT collegato al led
  pinMode(pulsante, INPUT); // inizializza il pin 7 come INPUT collegato al pulsante n.a.
  digitalWrite(pulsante, HIGH); // utilizza la R=10K di pull-up interna al microcontrollore
}
void loop()  // programma principale (main) --> ciclo infinito (loop)
{
  while(digitalRead(pulsante) == 1) // acquisisci il valore del pulsante pin 7 se il pulsante
    {  // NON E' PREMUTO si avra' un livello ALTO quindi si deve spegnere il led
     digitalWrite(led, LOW); // spegni il LED collegato al pin 13 della scheda Arduino
    }
  digitalWrite(led, HIGH); // accendi il LED collegato al pin 13 della scheda Arduino
}
```

Istruzione **do … while()**

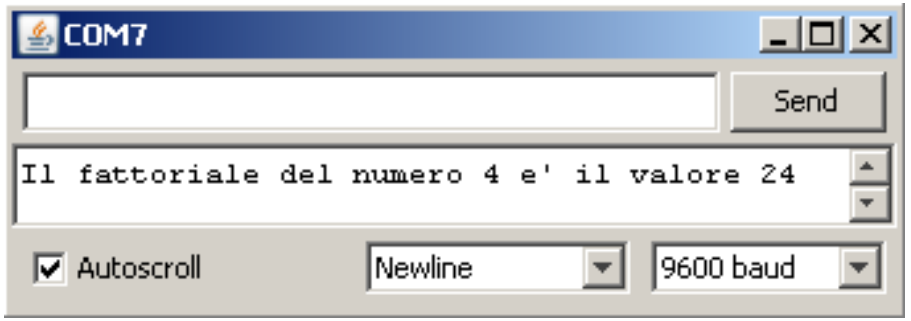
- La condizione di ripetizione viene verificata
  - **alla fine di ogni ciclo**

- Le istruzioni presenti tra le parentesi graffe **vengono sempre eseguite almeno una volta.**

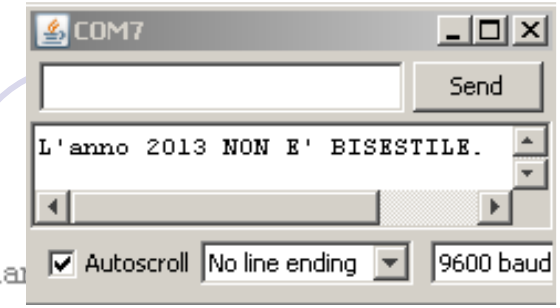
Istruzione do … while() con esempio

fattoriale_do_while.ino

```
/*  I.I.S. Primo LEVI - Torino
  Esercizio N. 9            Data: 03/02/2012
  Progetto: fattoriale_do_while       Autore: G. carpignano
  Descrizione: calcola e stampa il fattoriale con l'istruzione do .... while  */
void setup()   // funzione di inizializzazione della seriale RS232
{ // inizializza la seriale RS232 con 9600 baud, 8 bit dati, nessuna parità e 1 bit di stop
  Serial.begin(9600);
}
void loop()    // programma principale (main) --> ciclo infinito (loop)
{
  int fattoriale = 1; /* inizializzazione del fattoriale*/
  int numero = 4; // valore massimo del fattoriale da calcolare
  int i=0; /* inizializzazione del contatore*/
  do
  {  // calcolo del numero fattoriale (ad esempio per il num. 4 si avra' 1*2*3*4 = 24
    fattoriale = (i + 1) * fattoriale;
    i = i + 1;
  } while (i < numero);
  Serial.print("Il fattoriale del numero ");
  Serial.print(numero, DEC);
  Serial.print(" e' il valore ");
  Serial.print(fattoriale, DEC);
  while (1);  // blocca il programma (loop infinito)
}
```



Esempio



anno_bisestile_1.ino

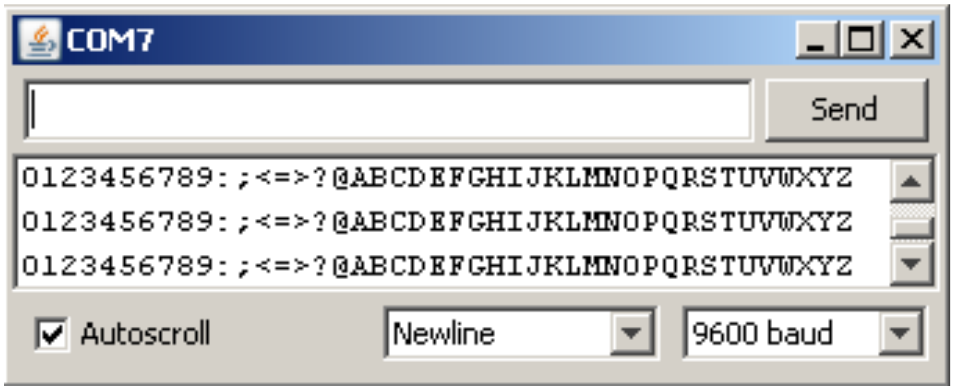
```
/*  I.I.S. Primo LEVI - Torino    Data: 03/12/2010
  Esercizio N. 8    Progetto: anno_bisestile_1     Autore: G. Carpignar
  Descrizione: Scrivere un programma in modo tale che venga inserito un anno
  nella variabile denominata "anno" e stabilisca se e' bisestile.*/
int anno = 2013; // variabile a 16 bit con l'anno da controllare se bisestile
void setup()
{// inizializza la seriale RS232 con 9600 baud, 8 bit dati, nessuna parità e 1 bit di stop
  Serial.begin(9600);
}
void loop() // programma principale
{ // la condizione viene verificata se e' uguale a 0, ma prima viene effettuata l'AND (&)
  // con la variabile "anno", cioe' si avra' 2012 & 3 che vale in binario
  // "1111101111101" & "0000000000011" diventa, effettuando la logica AND su ogni singolo bit
  // "0000000000001" che corrisponde a 1. Si ricorda che solo gli anni divisibili per 4, cioe'
  // solo quegli anni che divisi per 4 danno come resto 0 sono anni bisestili.
  if ((anno & 0x03) == 0)  // se il confronto vale 0
    {  // si e' in presenza di un ANNO BISESTILE
     Serial.print("Il numero "); Serial.print(anno, DEC); // trasmissione sulla seriale dell'anno
     Serial.print(" E' BISESTILE.");
    }
  else
    {  // altrimenti di un ANNO NON BISESTILE
     Serial.print("L'anno "); Serial.print(anno, DEC); // trasmissione sulla seriale dell'anno
     Serial.print(" NON E' BISESTILE.");
    }
  while(1);  // loop infinito (blocca il micro con questa istruzione)
}
```

Istruzione **for**

```
for (x = 0; x < 10; x++)
{
  Serial.print(x, HEX);
}
```

- È una istruzione di ripetizione particolarmente adatta per realizzare un numero predefinito di *cicli tramite un contatore*.
- La prima espressione è di inizializzazione (**x=0;**) viene eseguita una volta sola, prima di entrare nel ciclo.
- La seconda espressione (**x<10;**) rappresenta la condizione di permanenza nel ciclo (viene valutata all'inizio di ogni iterazione).
- La terza espressione (**x++**) rappresenta l'incremento o il decremento (**x--**) per il passaggio al ciclo successivo (valutata alla fine di ogni iterazione).
- Per forzare l'uscita da un ciclo "**for**" si utilizza l'istruzione "**break**".

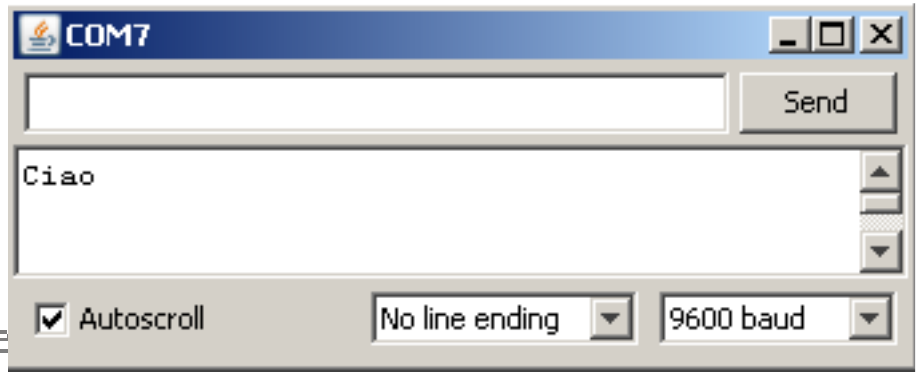
Istruzione **for** con esempio



Alfabeto.ino

```
/*  I.I.S. Primo LEVI - Torino  Esercizio N. 4    Progetto: Alfabeto   Autore: G. Carpignano
  Descrizione: effettuare la trasmissione dei caratteri ASCII compresi tra lo "0" (zero)
               e il carattere "Z" sull'interfaccia seriale RS232. */
char carattere; // variabile per memorizzare un carattere ASCII
long i; // variabile per memorizzare loop di ritardo di circa 2 secondi
void setup()    // funzione di inizializzazione della seriale RS232
{ // inizializza la seriale RS232 con 9600 baud, 8 bit dati, nessuna parità e 1 bit di stop
  Serial.begin(9600);
}
void loop()     // programma principale (main) --> ciclo infinito (loop)
{
  // ciclo con inizio dal valore 30 Hex (coincidente con il carattere "0" zero)
  // al valore 5A Hex = 5B Hex - 1 (coincidente con il carattere "Z")
  for (carattere = 0x30; carattere < 0x5B; carattere++)
  {
    Serial.print(carattere, BYTE); // trasmissione sulla seriale del carattere in codice ASCII
  }
  delay(1000);  // ciclo di ritardo di 1 secondo
  Serial.print("\n"); // trasmissione sulla seriale del carattere "new line" (nuova linea)
}
```

Esempio con le stringhe



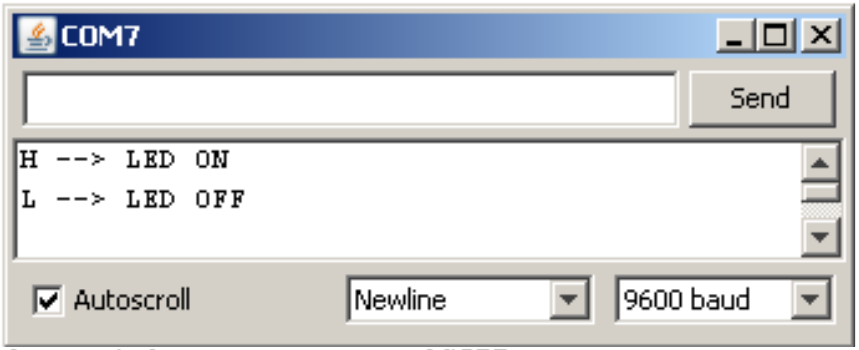

```
/*   I.I.S. Primo LEVI - Torino
  Esercizio N. 11   Progetto: stringhe
  Descrizione: Stampa la stringa "Ciao" sulla seriale.
  Il carattere "\0" e' corrispondente alla fine della stringa.
  Data: 03/02/2012  */
  byte stringa_1[] = { 'C', 'i', 'a', 'o', '\0'};
void setup() // funzione di configurazione dei Input/Output
{ // inizializza la seriale RS232 con 9600 baud
  Serial.begin(9600);
}
void loop() // programma principale (main) --> ciclo infinito (loop)
{
  for(int x=0; x<5; x++) // ciclo con il numero di caratteri da stampare
   {
    // Serial.print(stringa_1[x], BYTE); // vecchia stampa della stringa
    Serial.write(stringa_1[x]); // stampa della stringa
   }
  while (1); // loop infinito (blocca il micro)
}
```

stringhe_1.ino

Inserimento dati da tastiera del Computer alla scheda Arduino. Come fare?

Nel software se si digita "**H**“ il led si **accende**, mentre se si digita “**L**” si **spegne**. Qualsiasi altro carattere viene ignorato.

```
int led = 13; // il led e' collegato con l'Anodo sul pin 13 e il Catodo a GND.
int leggi_byte;
void setup()
{
 pinMode(led, OUTPUT);  // configura il pin 13 come output
 Serial.begin(9600);  // inizializza la seriale a 9600 baud
}
void loop()
{
 if (Serial.available() > 0) // se e' presente sul buffer della seriale un carattere ASCII
 {
   leggi_byte = Serial.read(); // acquisisci il carattere dalla seriale e memorizzalo
   if ((leggi_byte == 'H') || (leggi_byte == '1'))
     { // se il byte letto dalla seriale e' coincidente con il carattere maiuscolo "H" (0x48)
       // oppure con "1" (0x31) accendi il led
       digitalWrite(led, HIGH); // accendi il led
       Serial.write(leggi_byte); // ritrasmetti il carattere sulla seriale
       Serial.println(" --> LED ON");
     }
   if ((leggi_byte == 'L') || (leggi_byte == '0'))
     { // se il byte letto dalla seriale e' coincidente con il carattere maiuscolo "L" (0x4C)
       // oppure con "0" (0x30) spegni il led
       digitalWrite(led, LOW); // spegni il led
       Serial.write(leggi_byte); // ritrasmetti il carattere sulla seriale
       Serial.println(" --> LED OFF");
     }
 }
}
```

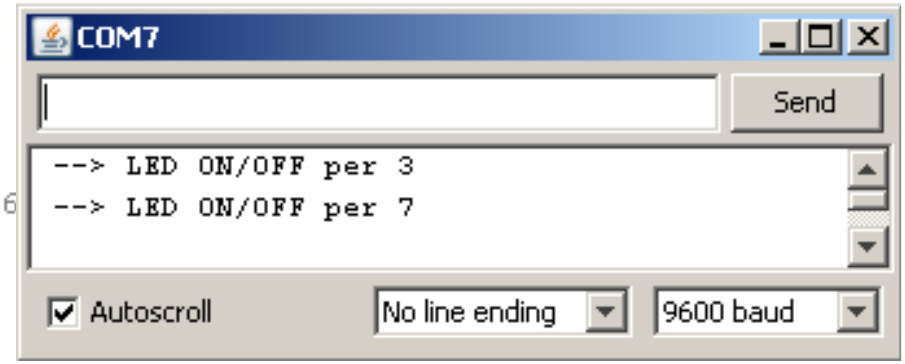


Led_controllato_da_tastiera_PC.ino

Inserimento dati da tastiera del Computer

Digitare un numero da **1** a **9** e il LED deve lampeggiare per il numero di volte digitato sulla tastiera del Personal Computer

```
int led = 13; // il led e' collegato con l'Anodo sul pin 13 e il Catodo a GND.
int leggi_byte;
void setup()
{
  pinMode(led, OUTPUT);  // configura il pin 13 come
  Serial.begin(9600);  // inizializza la seriale a 96
}
void loop()
{
  if (Serial.available() > 0) // se e' presente sul buffer della seriale un carattere ASCII
  {
    leggi_byte = Serial.read(); // acquisisci il carattere dalla seriale e memorizzalo
    if ((leggi_byte > '0') && (leggi_byte <= '9')) // se il valore letto e' > 0 e <= 9
      {
        leggi_byte = leggi_byte - '0'; // converti valore da ASCII a valore numerico
        Serial.print(" --> LED ON/OFF per ");
        Serial.println(leggi_byte, DEC); // ritrasmetti il numero sulla seriale
        // ripeti la sequenza di accensione/spegnimento per il numero inserito da tastiera del PC
        for (int i = 0; i < leggi_byte; i++)
           {
             digitalWrite(led, HIGH); // accendi il led
             delay (100); // ritardo di 0,1 secondi
             digitalWrite(led, LOW); // spegni il led
             delay (100); // ritardo di 0,1 secondi
           }
      }
  }
}
```



Led_con_n_flash_controllato_da_tastiera_PC.ino

Istruzione **switch ….. case**

Consente di selezionare l’esecuzione tra gli N blocchi di istruzioni componenti, in base al valore di una espressione (solo con variabili intere, cioè senza virgola).

Per terminare ogni "**case**" si utilizza l'istruzione "**break**" (che provoca l'uscita forzata dallo switch).

È possibile specificare un’etichetta "**default**". Essa viene eseguita per qualunque valore diverso dai valori specificati in precedenza nei vari "case".

Istruzione **switch ….. case** con esempio

Digitare un numero da **1** a **4** e il LED deve lampeggiare per il numero di volte digitato sulla tastiera del Personal Computer

Led_con_n_flash_controllato_da_tastiera_PC_con_switch.ino

```
int led = 13; // il led e' collegato con l'Anodo sul pin 13 e il Catodo a GND.
int leggi_byte, num_flash;
void setup()
{
  pinMode(led, OUTPUT);  // configura il pin 13 come output
  Serial.begin(9600);  // inizializza la seriale a 9600 baud
  Serial.flush();  // azzera buffer seriale
}
void loop()
{ // controlla se e' presente nel buffer della seriale un carattere ASCII
  while (!Serial.available());
  leggi_byte = Serial.read(); // acquisisci il carattere e memorizzalo
  switch (leggi_byte)  // confronta con i possibili valori
  {
    case '1': // caso relativo alla ricezione
     num_flash = 1; // numero di volte che dev
    break;
    case '2': // caso relativo alla ricezione
      num_flash = 2; // numero di volte che de
    break;
    case '3': // caso relativo alla ricezione
      num_flash = 3; // numero di volte che de
    break;
    case '4': // caso relativo alla ricezione
      num_flash = 4; // numero di volte che de
    break;
    default: // caso relativo alla ricezione diverso da un numero valido
      num_flash = 0; // numero di volte che deve lampeggiare
    break;
  }  // fine switch
  Serial.print(" --> LED ON/OFF per ");
  Serial.println(num_flash, DEC); // ritrasmetti il numero
// ripeti la sequenza di accensione/spegnimento per il nume
  for (int i = 0; i < num_flash; i++)
     {
        digitalWrite(led, HIGH); // accendi il led
        delay (1000); // ritardo di 1 secondo
        digitalWrite(led, LOW); // spegni il led
        delay (1000); // ritardo di 1 secondo
     }
}
```

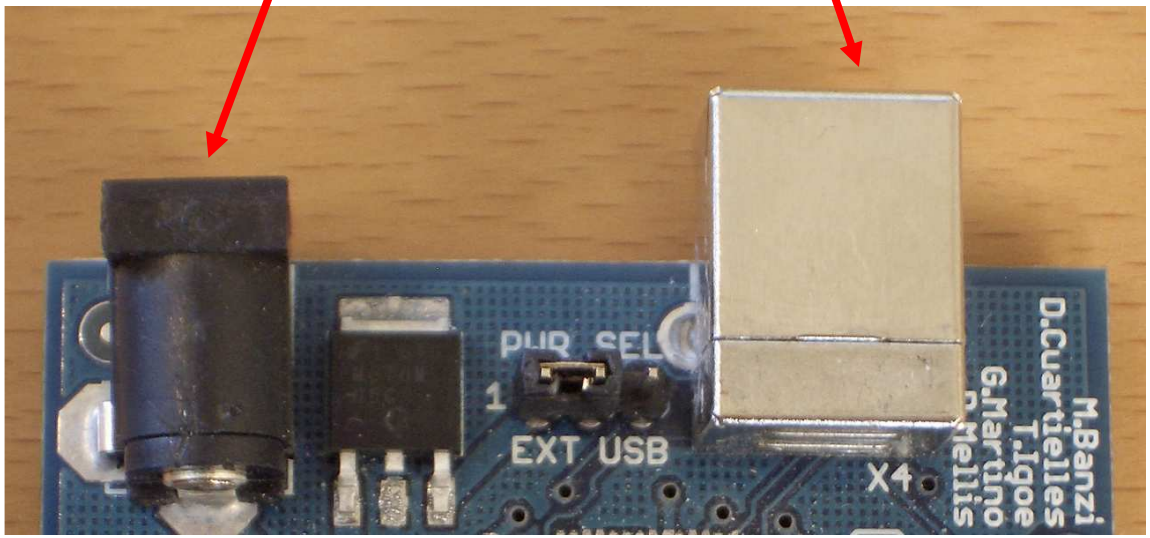
Scheda Arduino in modalità “Stand-alone”

- ***Stand-alone*** è un termine inglese che può essere tradotto letteralmente come "**a sé stante**", e significa quindi "**indipendente**".
- In informatica, l'espressione **stand-alone** indica che un oggetto o un software è capace di funzionare da solo o in maniera indipendente da altri oggetti o software, con cui potrebbe altrimenti interagire.
- È ovvio che la completa indipendenza si ottiene solo con una alimentazione esterna di tipo trasportabile.

Alimentazione della scheda Arduino

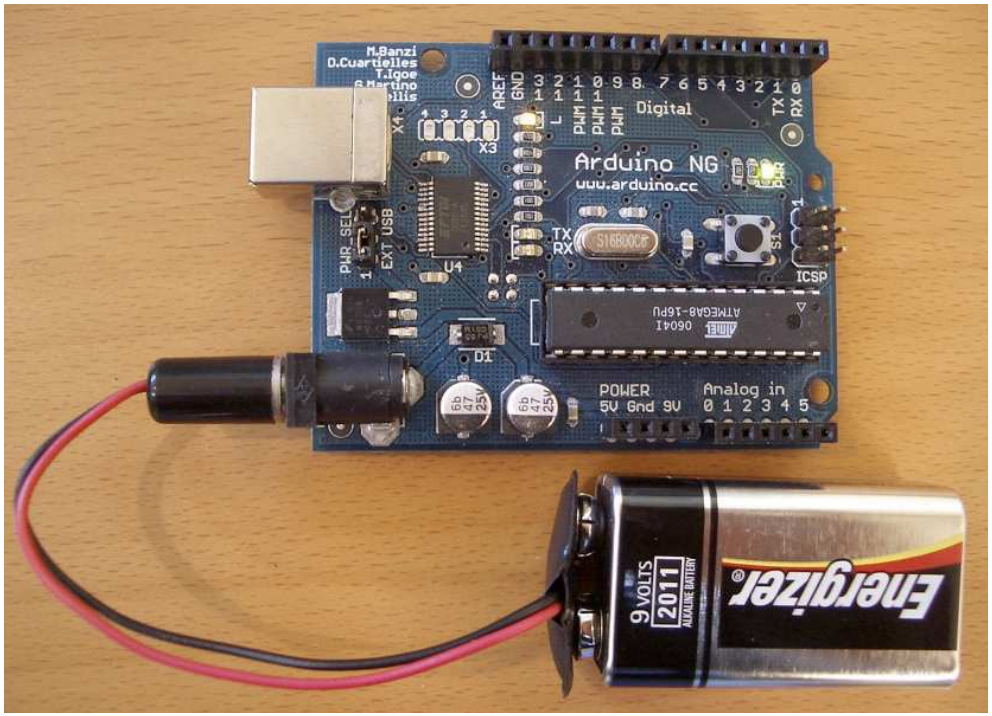
Arduino può essere alimentato tramite:

- Interfaccia USB (+5V)
- Alimentatore esterno (+9÷15V con contatto centrale collegato al positivo e corrente > 300 mA)



Alimentazione esterna da batteria

- Un metodo veloce e semplice per alimentare la scheda Arduino
- L’ingresso è protetto contro la polarità invertita da un diodo



Durata della batteria?

- La sola scheda Arduino richiede una corrente di circa 40 mA
- Ogni led aggiunto assorbe circa 20 mA quando viene acceso
- Ogni servo motore richiede una corrente media di circa 100÷150 mA
- le resistenze di pull-up dei pulsanti, interruttori e dei potenziometri assorbono quasi 0 mA
- La batteria da 9V possiede una capacità media espressa in milliampere all'ora (mA/h) di **400 mA/h**
- Quindi alimentando la sola scheda Arduino si avrà:

  400 mA/h / 40 mA = 10 ore di ininterrotto funzionamento.
- Ovviamente dovendo alimentare altri circuito il tempo si riduce ulteriormente in funzione del loro assorbimento medio richiesto.
- Nel caso si richieda un tempo maggiore di corretto funzionamento si ha a disposizione due tecniche di funzionamento:
  - 1) disporre il microcontrollore in modalità "sleep" (max assorbimento di pochi μA.
  - 2) disporre di una batteria di capacità superiore magari collegando due batterie in parallelo.

Rilevamento di tempi con traguardi meccanici o ottici (microswitches o sensori IR)

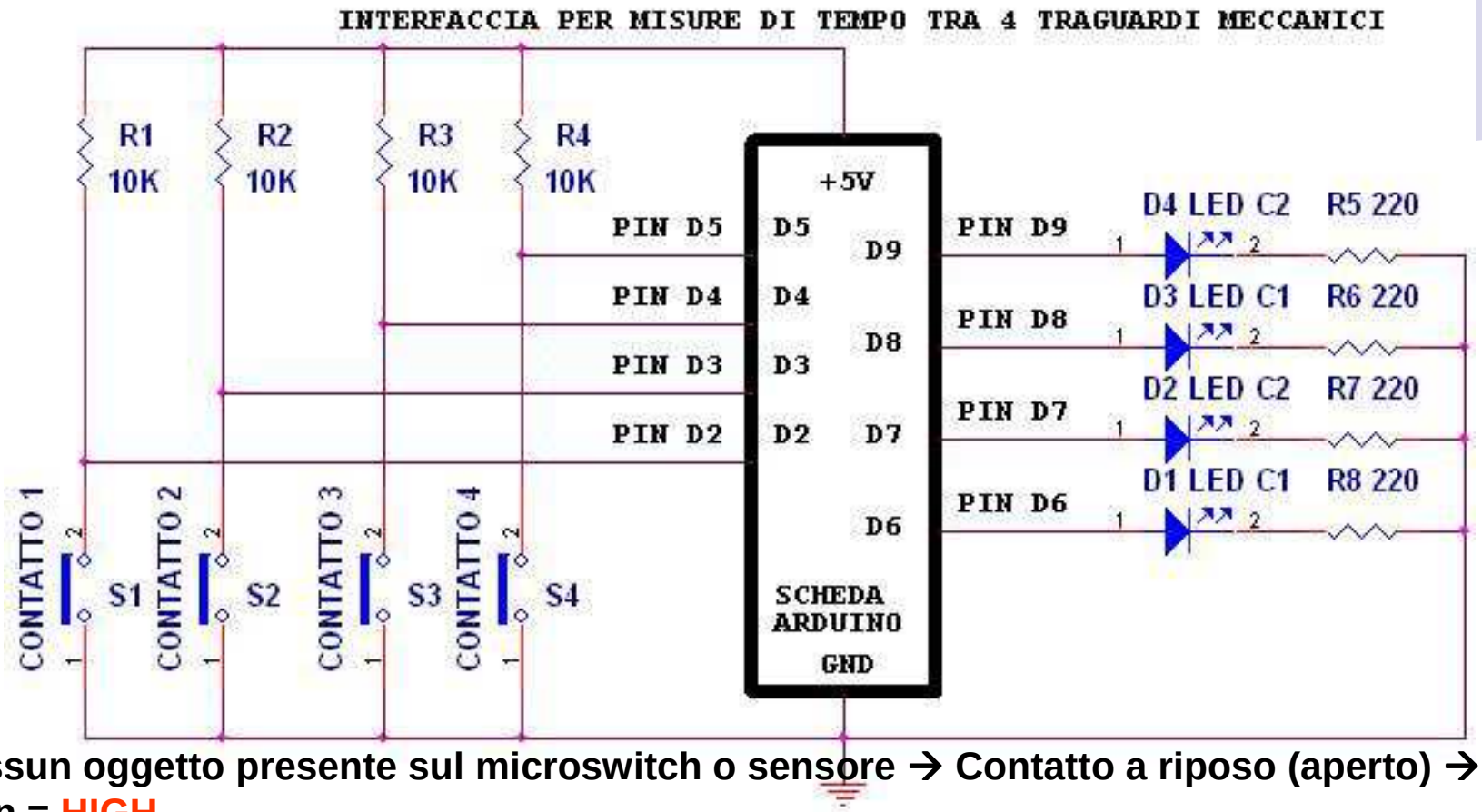
I microswitches

- Richiedono un circuito hardware o software di antirimbalzo dei contatti
- Sono più costosi dei sensori ottici
- Hanno una sensibilità limitata
- Richiedono un contatto “fisico” con l’oggetto in movimento
- Funzionano anche in presenza di forte illuminazione

I sensori ottici

- Non ci sono rimbalzi in fase di commutazione
- Il prezzo è di tipo “low cost”
- Hanno un basso campo di copertura (fino a 10÷20 cm di distanza)
- Non richiedono un contatto “fisico” con l’oggetto in movimento
- Vengono facilmente “accecati” da una forte sorgente luminosa

Rilevamento di tempi con traguardi meccanici o ottici (microswitches o sensori IR)



Nessun oggetto presente sul microswitch o sensore → Contatto a riposo (aperto) → +5V → Vpin = HIGH

Oggetto presente sul microswitch o sensore → Contatto attivato (chiuso) → GND → Vpin = LOW

ATTENZIONE! il SW proposto funziona solo se la sequenza di attivazione dei sensori è:

1) **attivazione del CONTATTO 1 (pin 2)**
2) **attivazione del CONTATTO 2 (pin 3)**
3) **attivazione del CONTATTO 3 (pin 4)**
4) **attivazione del CONTATTO 4 (pin 5)**

Rilevamento di tempi con traguardi meccanici o ottici (microswitches o sensori IR)

misura_tempo_4_traguardi.ino

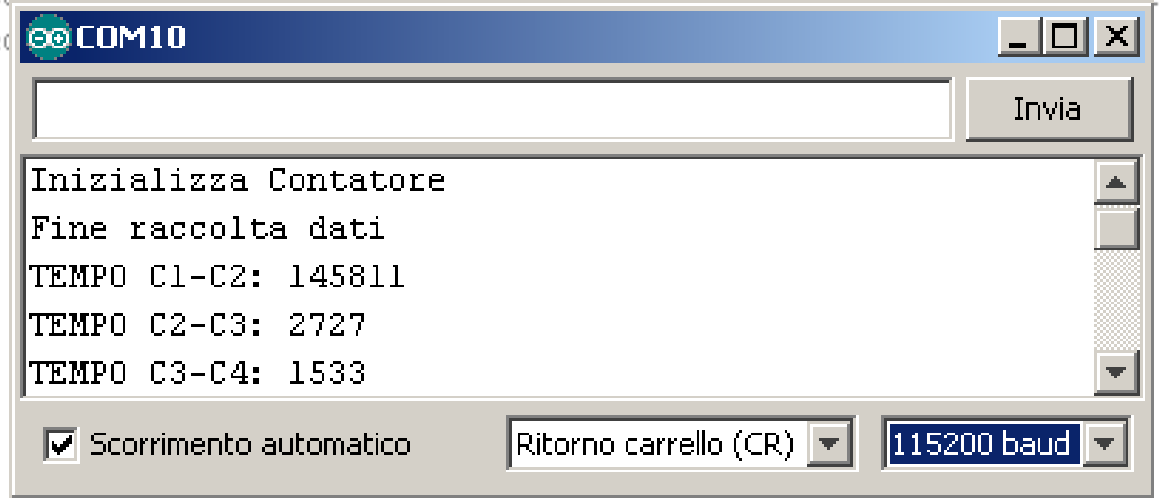
Parte 1ª

```
/* Software per un "Rilevamento di tempi con traguardi meccanici o ottici (microswitches o sensori IR)".
  Data: 25/04/2013 by Giorgio Carpignano  */
int pin_led_c1 = 6; // led colore ROSSO collegato al pin 6
int pin_led_c2 = 7; // led colore ROSSO collegato al pin 7
int pin_led_c3 = 8; // led colore ROSSO collegato al pin 8
int pin_led_c4 = 9; // led colore ROSSO collegato al pin 9
int pin_c1 = 2; // microswitch o sensore collegato al pin 2 come INPUT
int pin_c2 = 3; // microswitch o sensore collegato al pin 3 come INPUT
int pin_c3 = 4; // microswitch o sensore collegato al pin 4 come INPUT
int pin_c4 = 5; // microswitch o sensore collegato al pin 5 come INPUT
long valore_iniziale = 0;
long tempo_c1_c2, tempo_c2_c3, tempo_c3_c4;
void setup() // funzione di inizializzazione dei INPUT/OUTPUT
{
 pinMode(pin_c1, INPUT); // inizializza il pin 2 come INPUT collegato al microswitch n.a. o sensore 1
 pinMode(pin_c2, INPUT); // inizializza il pin 3 come INPUT collegato al microswitch n.a. o sensore 2
 pinMode(pin_c3, INPUT); // inizializza il pin 4 come INPUT collegato al microswitch n.a. o sensore 3
 pinMode(pin_c4, INPUT); // inizializza il pin 5 come INPUT collegato al microswitch n.a. o sensore 4
 pinMode(pin_led_c1, OUTPUT); // inizializza il pin 6 come OUTPUT collegato al led CONTATTO 1
 pinMode(pin_led_c2, OUTPUT); // inizializza il pin 7 come OUTPUT collegato al led CONTATTO 2
 pinMode(pin_led_c3, OUTPUT); // inizializza il pin 8 come OUTPUT collegato al led CONTATTO 3
 pinMode(pin_led_c4, OUTPUT); // inizializza il pin 9 come OUTPUT collegato al led CONTATTO 4
 digitalWrite(pin_led_c1, LOW);  // spegni led contatto 1
 digitalWrite(pin_led_c2, LOW);  // spegni led contatto 2
 digitalWrite(pin_led_c3, LOW);  // spegni led contatto 3
 digitalWrite(pin_led_c4, LOW);  // spegni led contatto 4
 Serial.begin(115200);  // inizializza la seriale RS232 con 115200 baud, 8 bit dati, nessuna parità e 1 bit di stop
}
```

Rilevamento di tempi con traguardi meccanici o ottici (microswitches o sensori IR)

misura_tempo_4_traguardi.ino **Parte 2ª**

```
void loop()
{
 Serial.println("Inizializza Contatore");
 while(digitalRead(pin_c1));  // aspetta finche' avviene attivazione del microswitch CONTATTO 1 a livello LOW
 valore_iniziale = millis();  // funzione per leggere i millisecondi di accensione della scheda
 digitalWrite(pin_led_c1, HIGH);  // accendi led relativo al contatto 1
 while(digitalRead(pin_c2));  // aspetta finche' avviene attivazione del microswitch CONTATTO 2 a livello LOW
 tempo_c1_c2 = millis() - valore_iniziale; // calcola tempo trascorso per percorrere il tratto compreso tra il CONTATTO 1 e 2
 valore_iniziale = millis();  // funzione per leggere i millisecondi di accensione della scheda
 digitalWrite(pin_led_c2, HIGH);  // accendi led relativo al contatto 2
 while(digitalRead(pin_c3));  // aspetta finche' avviene attivazione del microswitch CONTATTO 3 a livello LOW
 tempo_c2_c3 = millis() - valore_iniziale; // calcola tempo trascorso per percorrere il tratto compreso tra il CONTATTO 2 e 3
 valore_iniziale = millis();  // funzione per leggere i millisecondi di accensione della scheda
 digitalWrite(pin_led_c3, HIGH);  // accendi led relativo al contatto 3
 while(digitalRead(pin_c4));  // aspetta finche' avviene attivazione del microswitch CONTATTO 4 a livello LOW
 tempo_c3_c4 = millis() - valore_iniziale; // calcola tempo trascorso per percorrere il tratto compreso tra il CONTATTO 3 e 4
 digitalWrite(pin_led_c4, HIGH);  // accendi le
 Serial.println("Fine raccolta dati");
 Serial.print("TEMPO C1-C2: ");
 Serial.println(tempo_c1_c2);
 Serial.print("TEMPO C2-C3: ");
 Serial.println(tempo_c2_c3);
 Serial.print("TEMPO C3-C4: ");
 Serial.println(tempo_c3_c4);
}
```



Rilevamento di tempi con 2 traguardi ottici senza conoscere la sequenza di attivazione con indicazione della direzione del movimento. (esempio: misura del tempo, velocità e direzione delle auto, persone, oggetti)

- Nel precedente esempio la sequenza di attivazione dei sensori è nota.
- Cosa succede se non lo è?
- La risposta è che il software non funziona correttamente e quindi come si può rimediare?
- Gestendo l'**interrupt** del microcontrollore.

Interrupt

- Un microcontrollore normalmente esegue le istruzioni nella sequenza definita in fase di programmazione. Comunque, il microcontrollore può essere programmato per trattare eventi non schedulati, ovvero eventi che non si verificano con scadenze fisse di temporizzazione, e che necessitano di un livello di priorità differente a seconda delle esigenze.
- La risposta da parte del microcontrollore a questi eventi deve essere pianificata a priori dal programmatore, anche se non si conosce quando gli stessi eventi si verificheranno.
- Quando un **interrupt** (interruzione) viene generato da una periferica, il microcontrollore completa l'istruzione in corso (il tempo di risposta è inferiore al microsecondo) e poi salta al programma specifico della gestione dell'interrupt (ISR = *Interrupt Service Routine*) associato alla periferica che ha richiesto l'attenzione del microcontrollore. Ogni differente interrupt viene gestito da un programma specifico dove è possibile avere differenti risposte alla stessa periferica.
- Quando il software di gestione dell'interrupt è terminato il microcontrollore riprende ad eseguire il normale programma da dove era stato interrotto prima del verificarsi dell'evento generato dalla periferica.

Interrupt

- Il microcontrollore della scheda Arduino UNO (ATMEGA328P) possiede una potente e flessibile gestione di 26 differenti sorgenti dell'interrupt.
- Due interrupt sono generati da un segnale esterno mentre i rimanenti 24 interrupt supportano in modo efficiente le periferiche disponibili all'interno del chip del microcontrollore.

- Nel compilatore Arduino 1.0.4 esistono **4** funzioni predefinite per supportare gli interrupt esterni alla scheda:
- a) la funzione denominata "**interrupts();**" serve per abilitare l'interrupt globale.
- b) la funzione denominata "**noInterrupts();**" serve per disabilitare l'interrupt globale.
- c) la funzione denominata "**attachInterrupt(interrupt, function, mode);**" serve per collegare l'interrupt alla tabella dei vettori dell'interrupt.
- d) la funzione denominata "**deattachInterrupt(interrupt);**" serve per disabilitare l'interrupt specificato

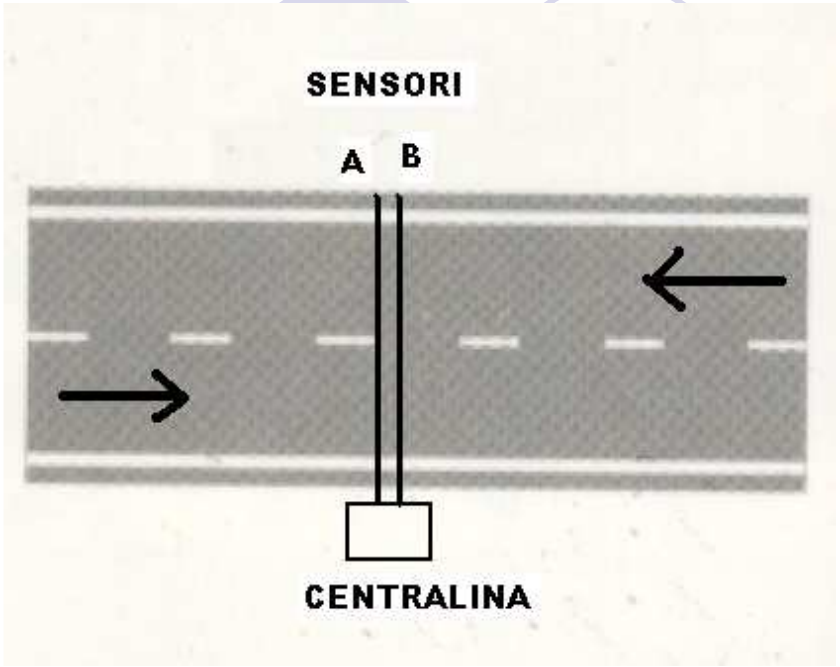
Interrupt

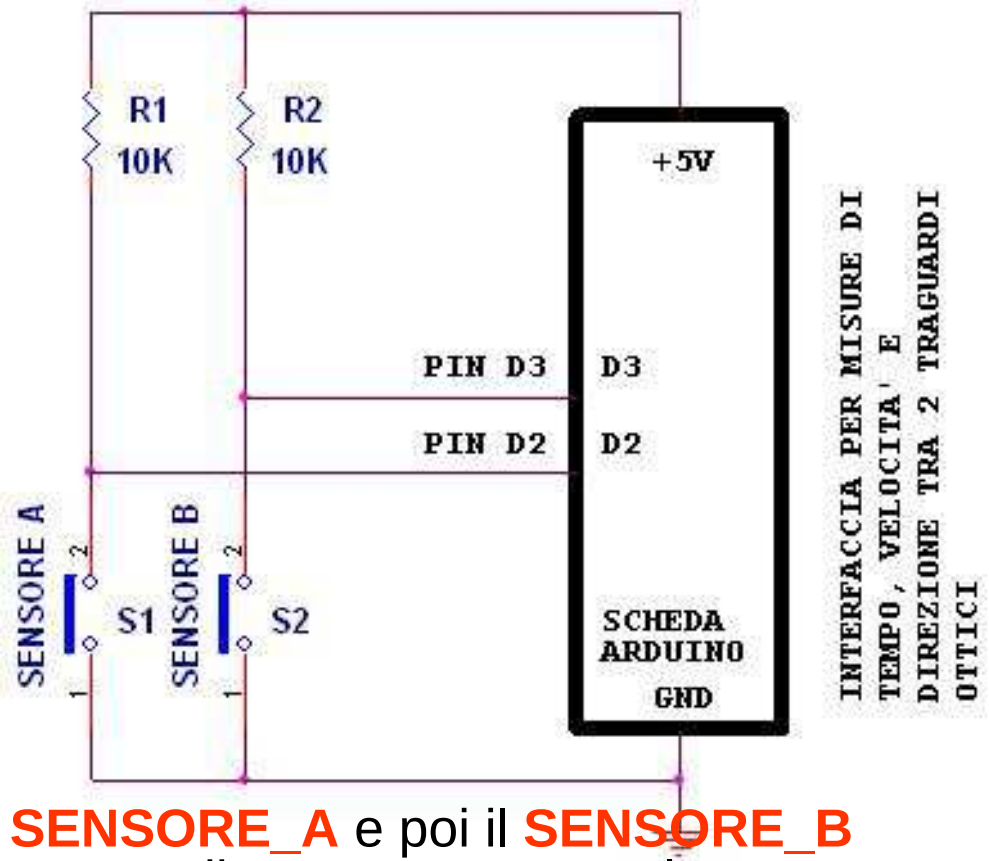
- La funzione "**attachInterrupt(interrupt, function, mode)**" è utilizzata per collegare il pin hardware all'appropriata gestione del software di interrupt. I tre argomenti della funzione devono essere configurati nel seguente modo:
- ***interrupt.*** --> Specifica il numero dell'interrupt. Valori possibili sono **0** oppure **1**.
- ***function*.** specifica il nome della routine di gestione dell'interrupt.
- ***mode*.** Specifica quale tipo di attività deve essere valutata quando si verifica l'evento che genera l'interrupt.
  - Nella modalità denominata "**LOW**" si genera un interrupt quando il pin è a livello basso.
  - Nella modalità denominata "**CHANGE**" si genera un interrupt quando il pin passa da un livello all'altro, cioè quando si passa da HIGH a LOW e viceversa.
  - Nella modalità denominata "**RISING**" si genera un interrupt solo quando il pin passa dal livello LOW al livello HIGH.
  - Nella modalità denominata "**FALLING**" si genera un interrupt solo quando il pin passa dal livello HIGH al livello LOW.

AUTOVELOX

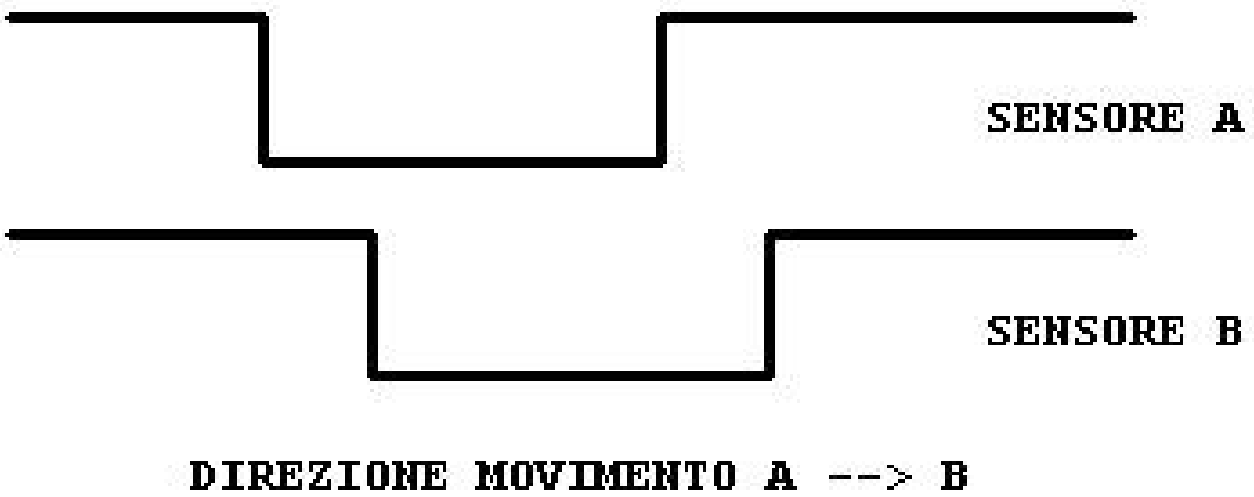
- Per semplicità si consideri la possibilità di utilizzare due fotocellule per cui il programma deve eseguire il calcolo della velocità di una auto che passi davanti ai due sensori (fotocellule) posti parallelamente alla distanza di **100 cm**. La velocità viene visualizzata in **Km/h** oppure in **m/s**.
- Le due fotocellule con circuito antirimbalzo sono collegate al pin 2 e al pin 3 della scheda Arduino.
- Quando l'auto **NON VIENE RILEVATA** davanti alle fotocellule il segnale in uscita è a livello logico **ALTO**, e diviene **BASSO quando l'auto è in transito**.
- Si presuppone che la distanza tra le due fotocellule sia esattamente di **100 cm** tra il loro centro.
- La formula della velocità = spazio / tempo

AUTOVELOX



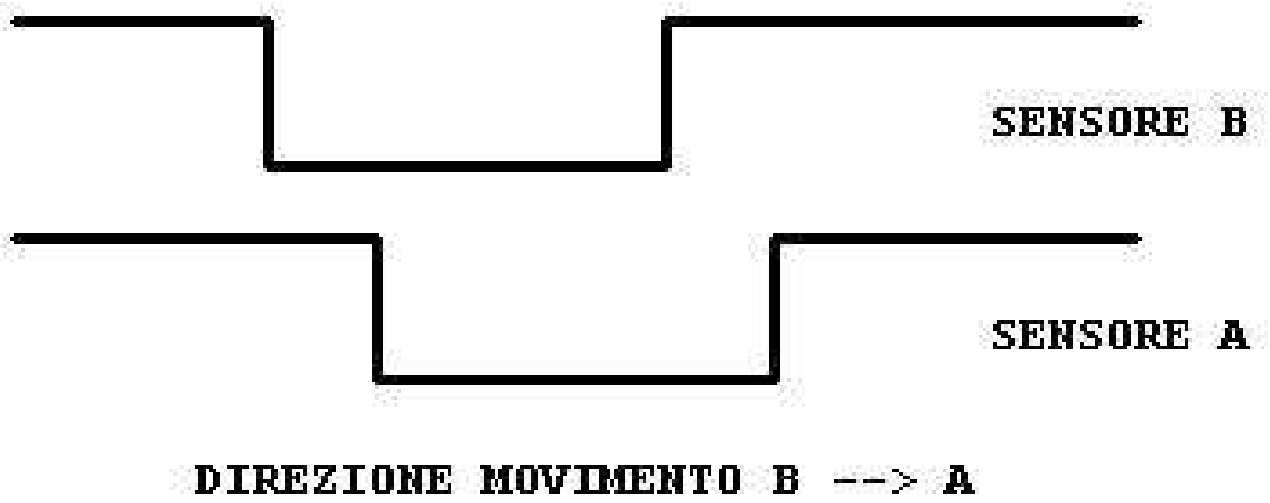


Se l'AUTO transita attivando prima il **SENSORE_A** e poi il **SENSORE_B** (**DIREZIONE_A_B**) si ottiene il seguente diagramma temporale:



AUTOVELOX

Se l'AUTO transita attivando prima il **SENSORE_B** e poi il **SENSORE_A** (**DIREZIONE_B_A**) si ottiene il seguente diagramma temporale:



- Si utilizzi per il calcolo del tempo trascorso la funzione "**millis()**" che restituisce in una variabile di tipo "**unsigned long**" il numero di millisecondi di tempo trascorsi dall'esecuzione del software stesso. Il numero restituito si azzera ogni ogni 50 giorni di ininterrotto funzionamento del software con una risoluzione di un millisecondo.
- Il software è denominato: **Autovelox.ino**

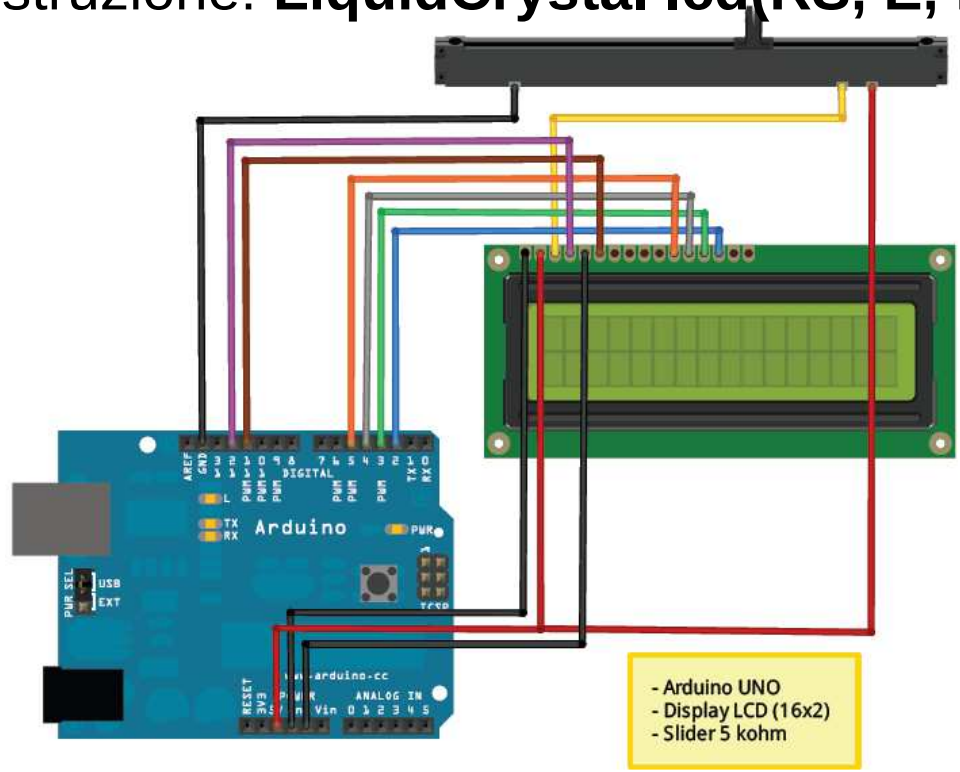
Pilotaggio di un display LCD con 16 x 2 caratteri

Assegnazione dei segnali sul connettore

Pin-N.	Segnali	Funzione
1	BL+	Terminale di alimentazione LED (+)
2	BL-	Terminale di alimentazione LED (-)
3	GND	Alimentazione (0V)
4	V_{DD}	Alimentazione (5V)
5	Vo	Drive LCD (0V rispetto V_{DD})
6	RS	(Alto) ingresso codici di istruzione (Basso) ingresso dati
7	R/W	(Alto) lettura dati (Basso) scrittura dati
8	E	Seganle di abilitazione
9	DB_0	Linea di bus dati
10	DB_1	
11	DB_2	
12	DB_3	
13	DB_4	
14	DB_5	
15	DB_6	
16	DB_7	

display_2_seriale_74HC595.inor

• **VDD** e **GND** sono collegati rispettivamente a **+5V** (rosso) e a massa (**GND** = nero)
• **R/W** è collegato a massa
• **Vo** è collegato ad un potenziometro da 10 KΩ per il contrasto
• **RS** è collegato al PIN2 della scheda Arduino
• **E** è collegato al PIN3 della scheda Arduino
• i 4 bit dato **DB4**, **DB5**, **DB6**, **DB7** sono collegati rispettivamente ai PIN 5, 6, 7, 8
• utilizza l'istruzione: **LiquidCrystal lcd(RS, E, DB4, DB5, DB6, DB7);**



Pilotaggio di un display LCD con 16 x 2 caratteri

```
/*      nome progetto:  LCD.ino
Visualizzazione a display dei secondi passati dall'accensione. Regolazione
contrasto con slider o trimmer da 10Kohm.
creato da: G. Carpignano  data: 27/01/2011  compilatore: Arduino 1.0.4 */
#include <LiquidCrystal.h>
//Inizializza i PIN per il controllo del display
 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup() // funzione di inizializzazione dei INPUT/OUTPUT
{  // Definisce il tipo di display, ovvero 16 caratteri per 2 righe
   lcd.begin(16, 2);
   //Scrittura messaggio di avvio
   lcd.print("HELLO WORD!");
 }

void loop() // programma principale (main) --> ciclo infinito (loop)
{  // Imposta colonna 0 e riga 1 (0: Prima riga - 1: Seconda riga)
   lcd.setCursor(0, 1);
   // Visualizza il numero di secondi trascorsi dall'accensione della scheda
   lcd.print(millis()/1000);
   lcd.print(" sec.");
 }
```

LCD.ino

Comportamento di una bobina e un condensatore

Un **condensatore** è in grado di assorbire inizialmente la corrente continua fino a quando non è completamente carico, in seguito blocca il flusso di corrente.
Esiste un altro fenomeno che è l'esatto opposto della capacità ed è conosciuto come **autoinduttanza**, e lo si trova in qualsiasi spira di conduttore. Inizialmente la **bobina** blocca la corrente continua (reagisce opponendosi contro il passaggio di corrente in continua), ma poi gradualmente riduce questa opposizione fino ad annullarla.

Procedura

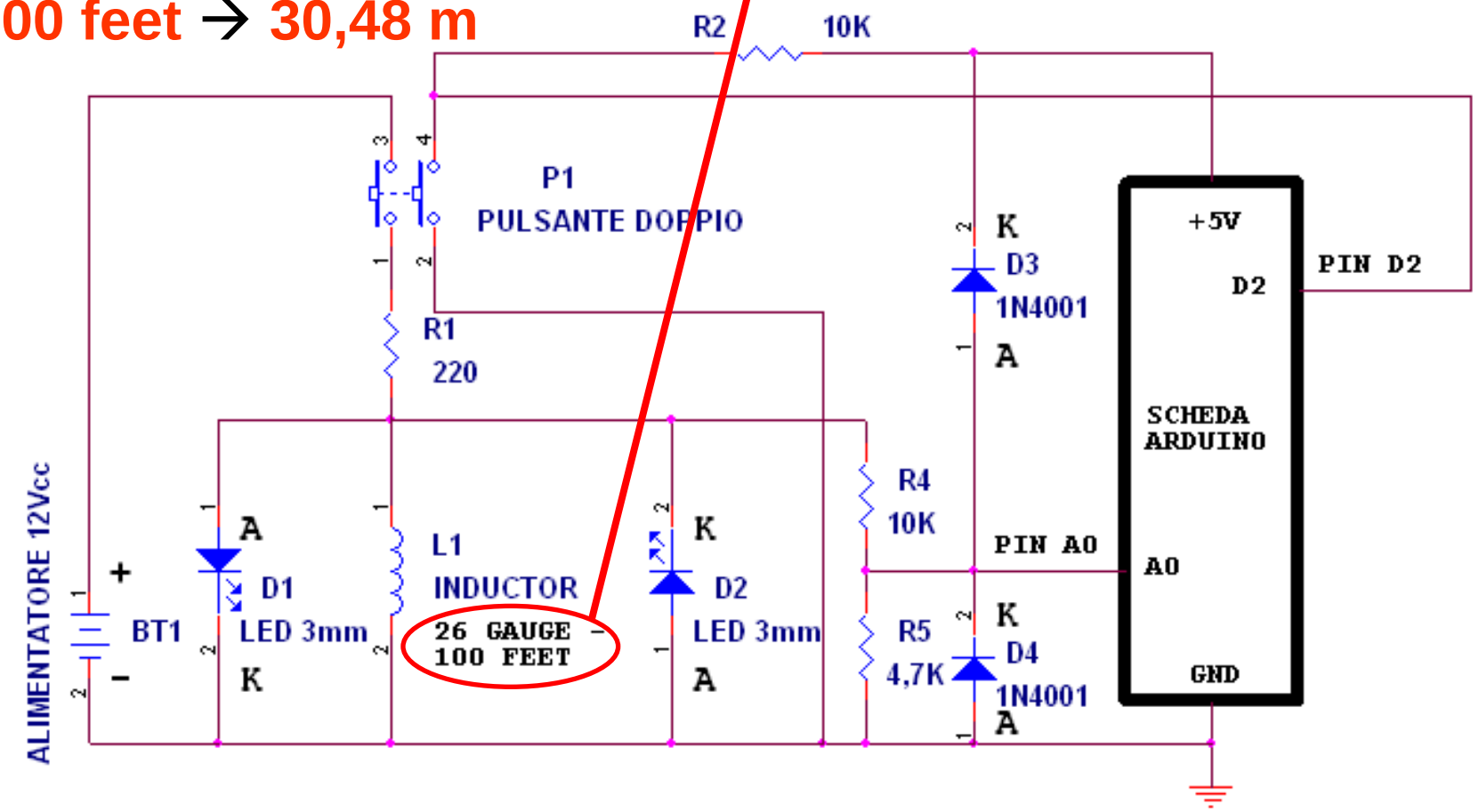
Collegare i componenti seguendo lo schema in Figura anche se può sembrare che alcuni collegamenti non abbiano molto senso. Così apparentemente il passaggio di corrente avviene attraverso la resistenza di 220 Ω, e poi attraverso la bobina, ignorando i due LED perché la bobina ha ovviamente una resistenza molto più bassa di quella offerta dai diodi led che richiedono una tensione di almeno 1,8V per accenderli.

Comportamento di una bobina

ATTENZIONE → i diodi led devono essere ad alta luminosità e del diametro di 3 mm, altrimenti non sarà visibile nulla!

American Wire Gauge (AWG): **26** → Diametro: **0,0159 inches** → Diametro: **0,40 mm** → Sezione: **0,13 mm²**

100 feet → 30,48 m



Comportamento di una bobina

Modificare il circuito sostituendo la bobina con un condensatore molto grande del valore di 4700 µF (prestando la massima attenzione al rispetto delle polarità perché è un condensatore elettrolitico quindi collegando il terminale "**-**" alla **massa** (GND) ed il terminale "**+**" alla resistenza **R1**).

A quale fenomeno potrete assistere?

Ricordate: il comportamento della capacità è l'opposto dell'induttanza!

Bobina_autocostruita.ino