POLITECNICO DI MILANO

# Arduino

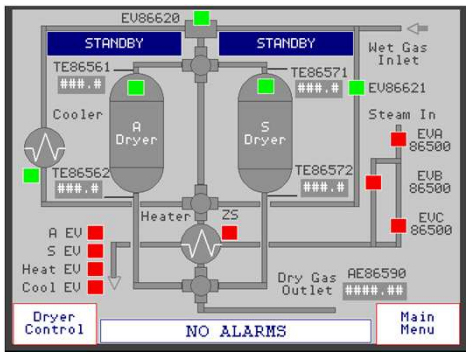
Sezione di Misure e Tecniche Sperimentali

# Finalità delle misure

- In ambito scientifico, studiare i fenomeni, verificare sperimentalmente modelli matematici e teorie scientifiche, monitorare lo stato per effettuare previsioni.

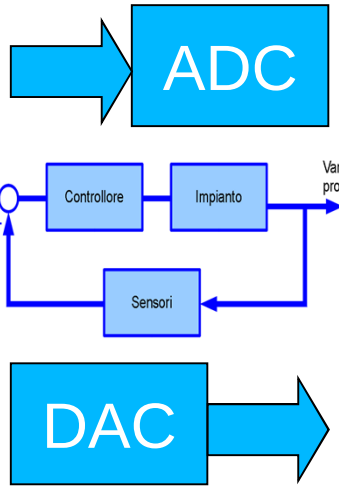

- In ambito commerciale quantificare parametri a cui è associato il valore delle merci (metrologia legale).


- In ambito industriale monitorare i processi, certificare conformità di prodotti, **controllare i processi.**

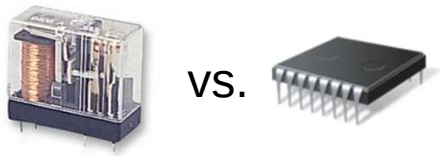


# Finalità dell'esercitazione

- Utilizzare un sistema che consente di:
  - acquisire dati da strumenti diversi anche con modalità di trasmissione del segnale diverse (segnale analogico o digitale),
  - "apprendere" una logica di controllo,
  - generare uscite tramite cui "regolare" un processo.




- Evidenziare le potenzialità di un sistema basato su "microcontrollore" rispetto a controllori "tradizionali".




- Imparare a usare un sistema semplice ed economico, eventualmente utilizzabile a casa per le proprie applicazioni.

# Definizione di “microcontrollore”


- È un **Circuito Integrato (IC)** che riunisce:
  - Central Processing Unit (CPU)
  - Memoria RAM
  - Memoria ROM, EEPROM o FLASH
  - Interfaccia di Input/Output (analogico, digitale, seriale)
  - Un circuito di clock


- Il microcontrollore riunisce tutti i componenti per operare come un computer standalone, ovvero è **programmabile.**
- Sono piccoli ed economici, hanno vasto impiego in molti settori (industriale, domotica, veicoli, giochi).

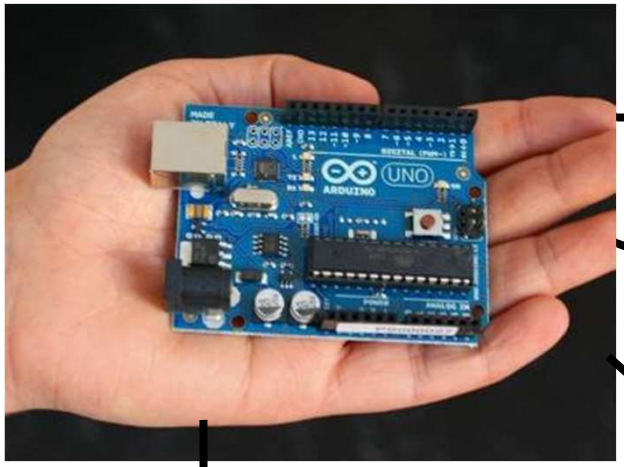
# Cosa è “Arduino”

Arduino è una **piattaforma di sviluppo** basata su microcontrollore

- Può interagire con il mondo mediante I/O analogici e digitali
- Sia il software sia l'hardware sono “Open Source”
- Il linguaggio di programmazione è molto simile al “C” e ha una IDE dedicata molto intuitiva
- E' sviluppato da una comunità molto vasta di programmatori, ingegneri e designer
- Può essere trovato su: http://www.arduino.cc

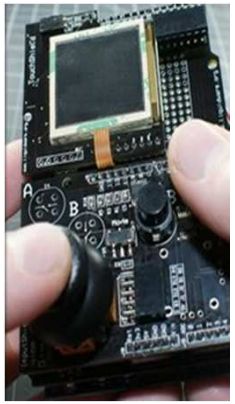
# Esempi di applicazioni

http://hacknmod.com/hack/top-40-arduino-projects-of-the-web/





Veicolo aereo senza pilota



Console portatile

Sistema automatico d'irrigazione

Fotografia di fenomeni veloci

# Caratteristiche tecniche

AVR Atmel Atmega328

Oscillatore quarzo 16 MHz

Alimentazione da 5 V a 12 V

6 Ingressi analogici 0-5 V

14 input/output digitali 0-5 V

6 uscite PWM 0-5 V

Pulsante Reset

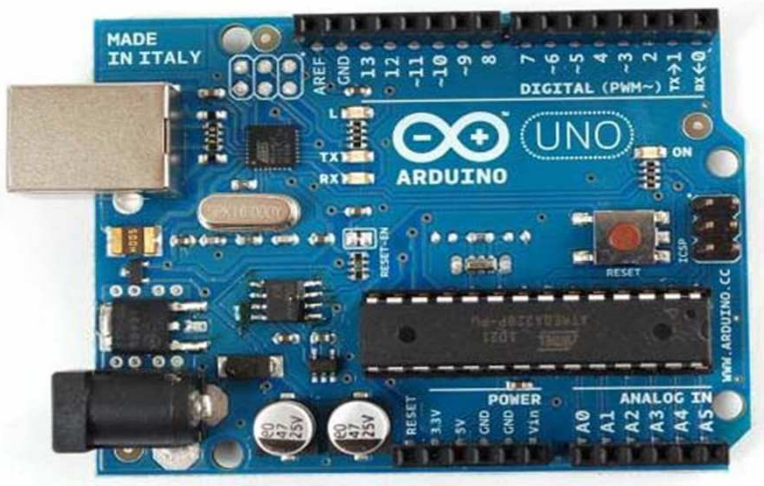
Flash Memory 32 KB (0.5 KB occupato dal bootloader)

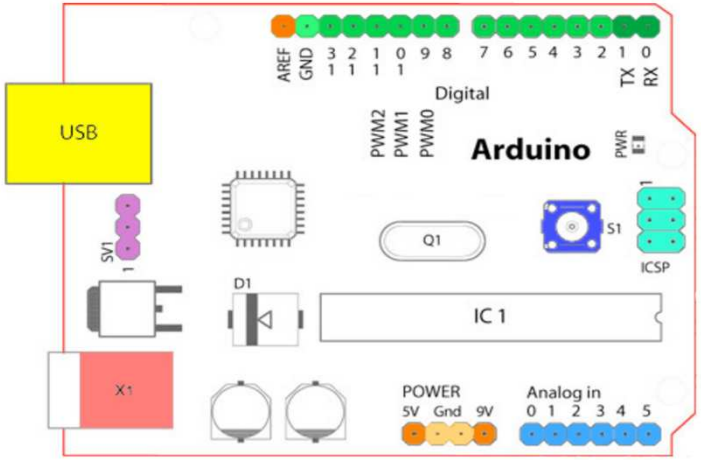
SRAM 2 KB

EEPROM 1 KB

TX/RX LED per la comunicazione seriale

Connettore USB con convertitore USB/seriale già integrato

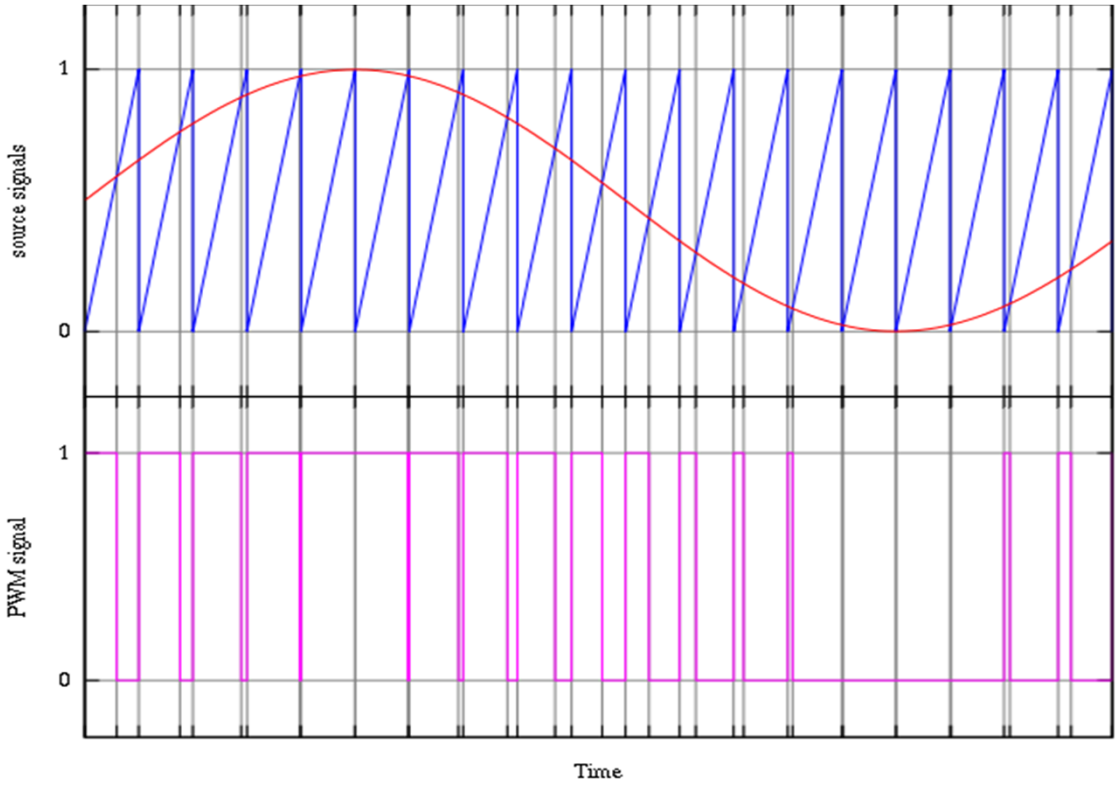


# Uscita PWM - Pulse-width modulation

E' un sistema per modificare una informazione analogica (es. tensione del segnale) utilizzando la modulazione di ampiezza di un impulso.

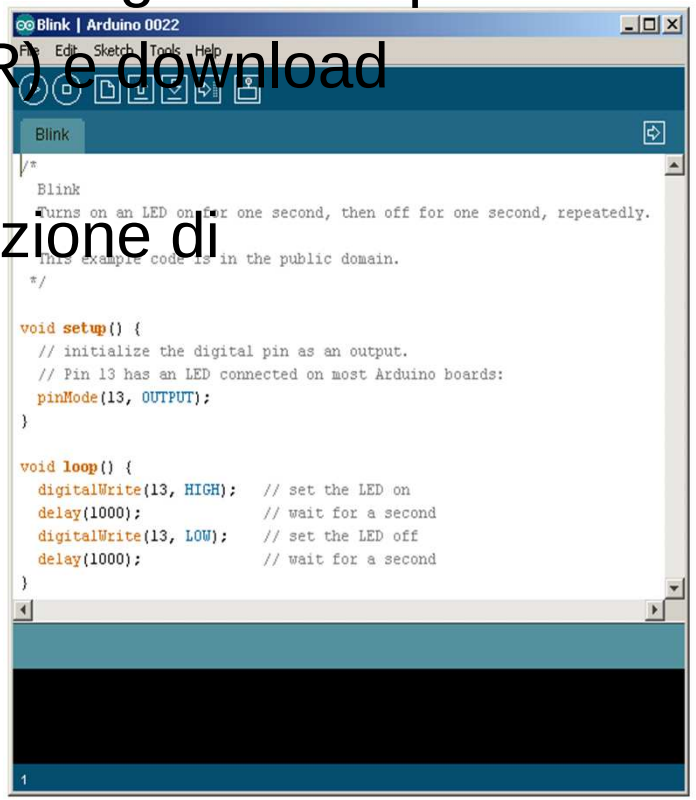
Esempio: "simulare" un segnale in tensione sinusoidale (con valore qualsiasi tra 0 V e 5 V) utilizzando una uscita digitale (che può assumere solo due valori, 0 V o 5 V).



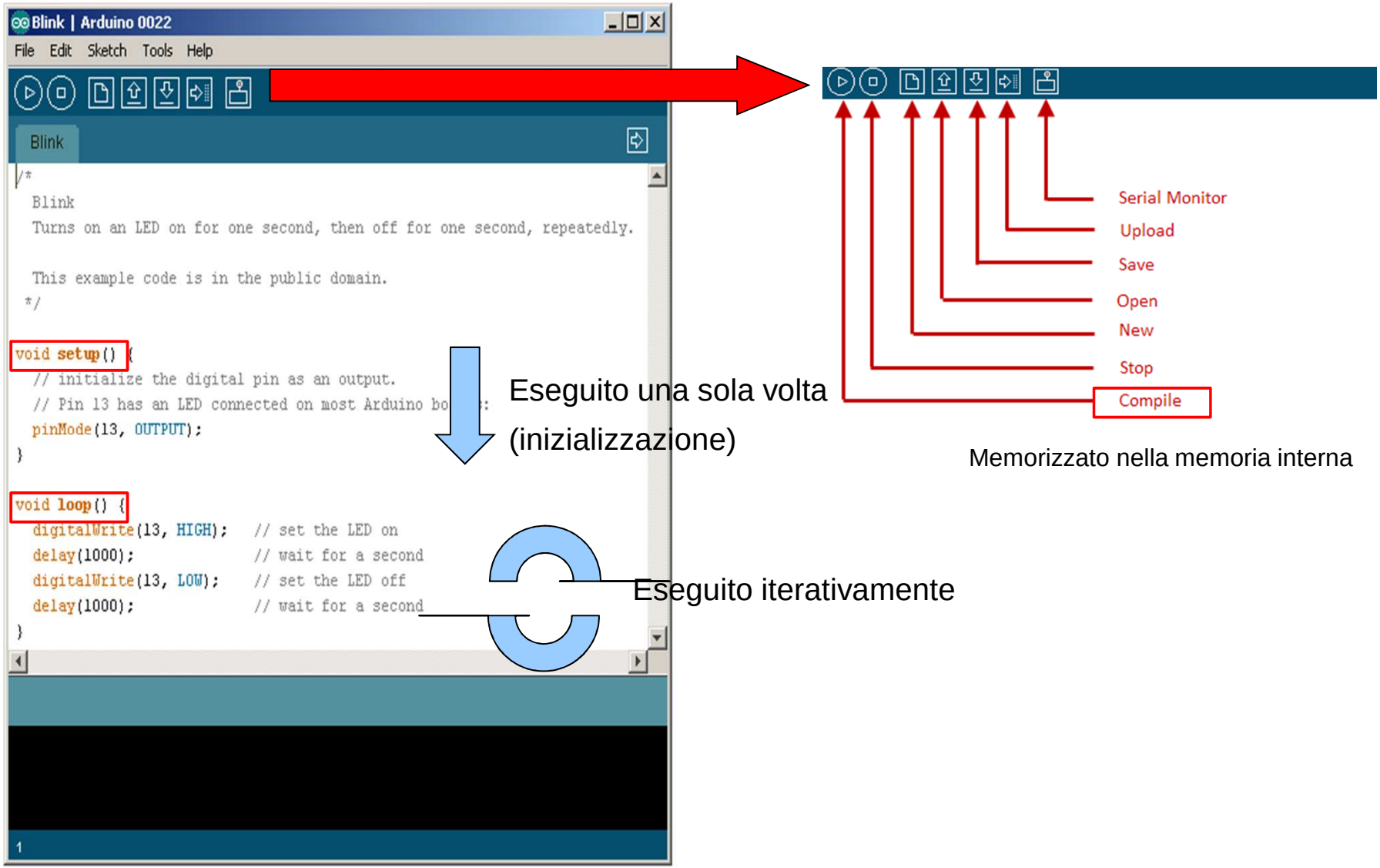

*(tratto da wikipedia)*

# Ambiente di sviluppo software

- Linguaggio programmazione in stile “C” semplificato
- Gestione integrata degli ingressi e delle uscite, sia analogici sia digitali
- Operazioni real time, reazione a watchdog e interrupt
- Compilazione del codice (librerie AVR) e download tramite porta seriale
- Comunicazione seriale integrata (funzione di debugging)
- Ampio set di librerie disponibili



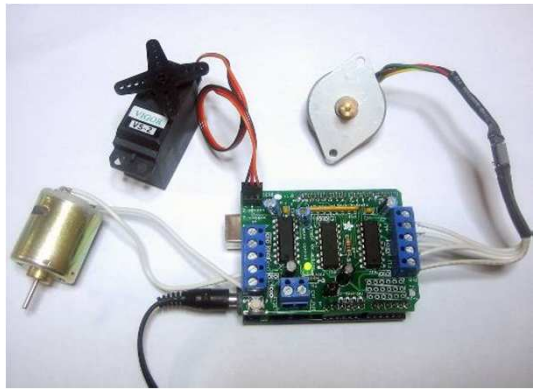
# Struttura del codice



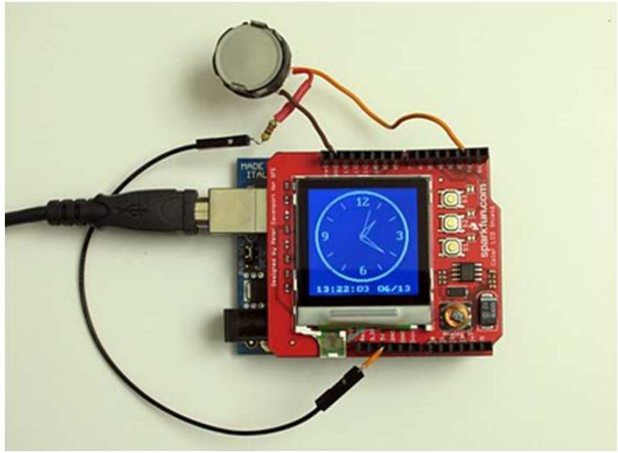
# Estensione dell'hardware: gli “shield”

- Estensione delle capacità hardware

Ethernet shield



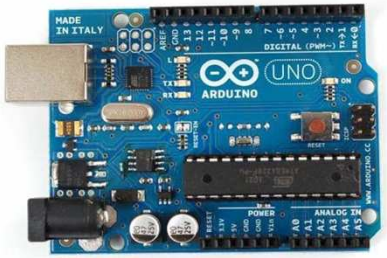
Motor shield

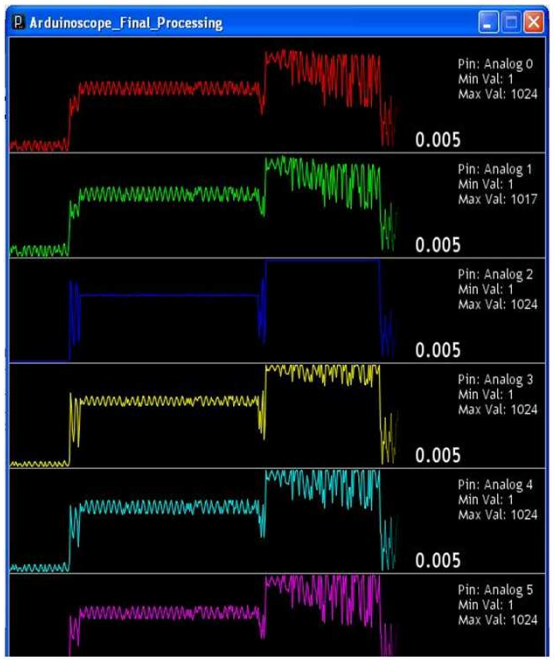


LCD shield

# Estensione del software: la porta seriale

- E' possibile comunicare dati in input e output con altri tutti gli altri linguaggi tramite comunicazione seriale (Matlab, C C++ C#, Java, Python, Processing)


•Esempio di comunicazione con Processing (www.processing.org)

oscilloscopio analogico/digitale

http://code.google.com/p/arduinoscope/

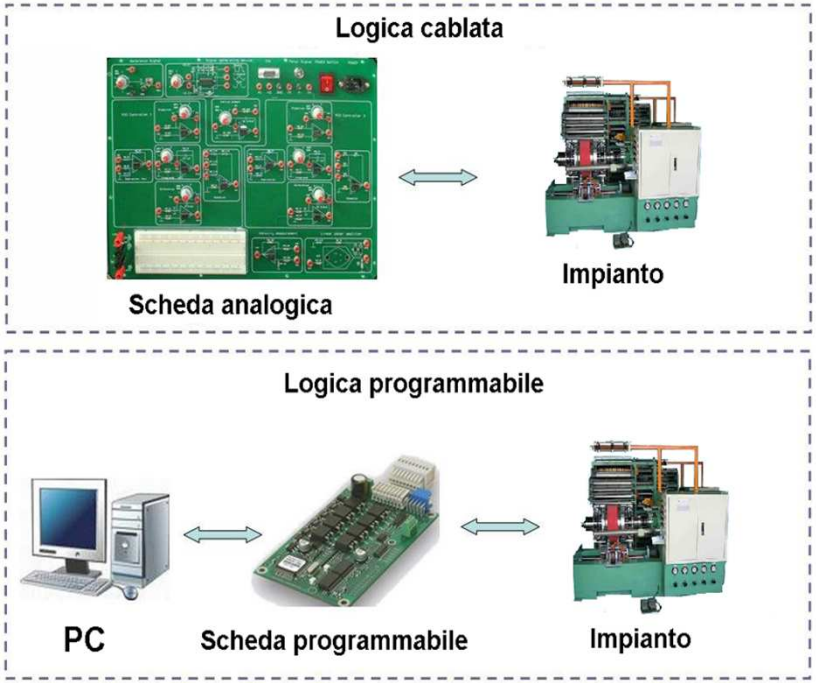
# Esperienza 1
# Logica programmabile e logica cablata

# Logica programmabile e logica cablata

Il controllo dei sistemi energetici era svolto tramite circuiti anologici (elettrici, pneumatici, oleodinamici)

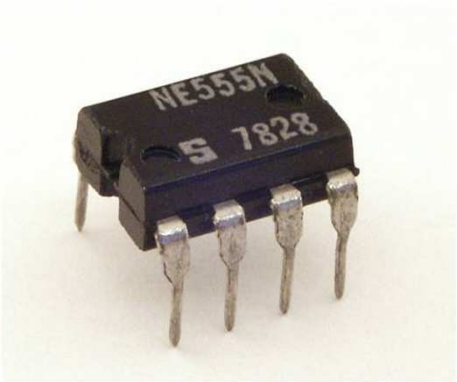
Si è passati a sistemi programmabili basati su microcontrollore in quanto offrono:

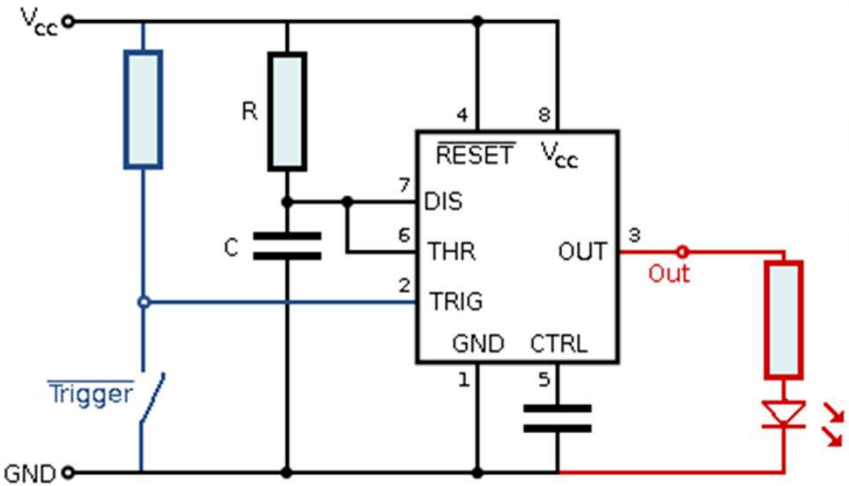
- Prestazioni migliori
- Integrazione delle funzioni (acquisizione dei segnali, elaborazione, attuazione)
- Approccio basato sulla scrittura di software
- Abbattimento dei costi per passaggio alla tecnologia digitale (riduzione componenti esterni e processo produttivo economicamente vantaggioso)
- Maggiore flessibilità ed orientamento al cliente

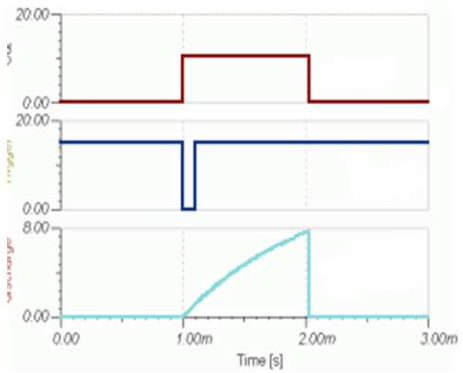


# Esempio: logica cablata integrata

Compito: accendere un LED per un tempo t stabilito al premere di un pulsante



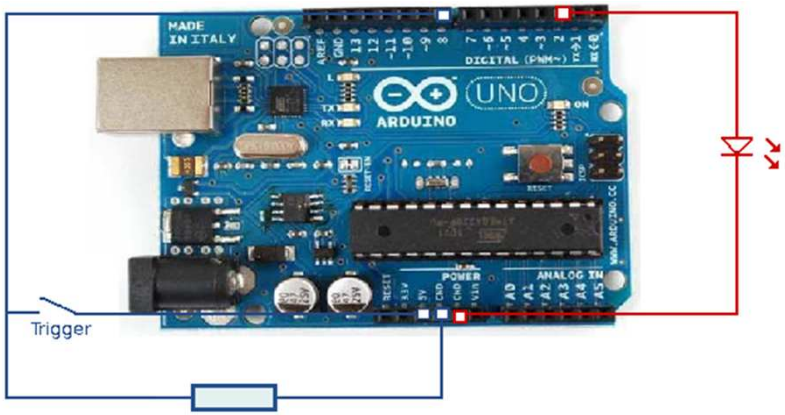

$$t = 1.1\ R\ C$$

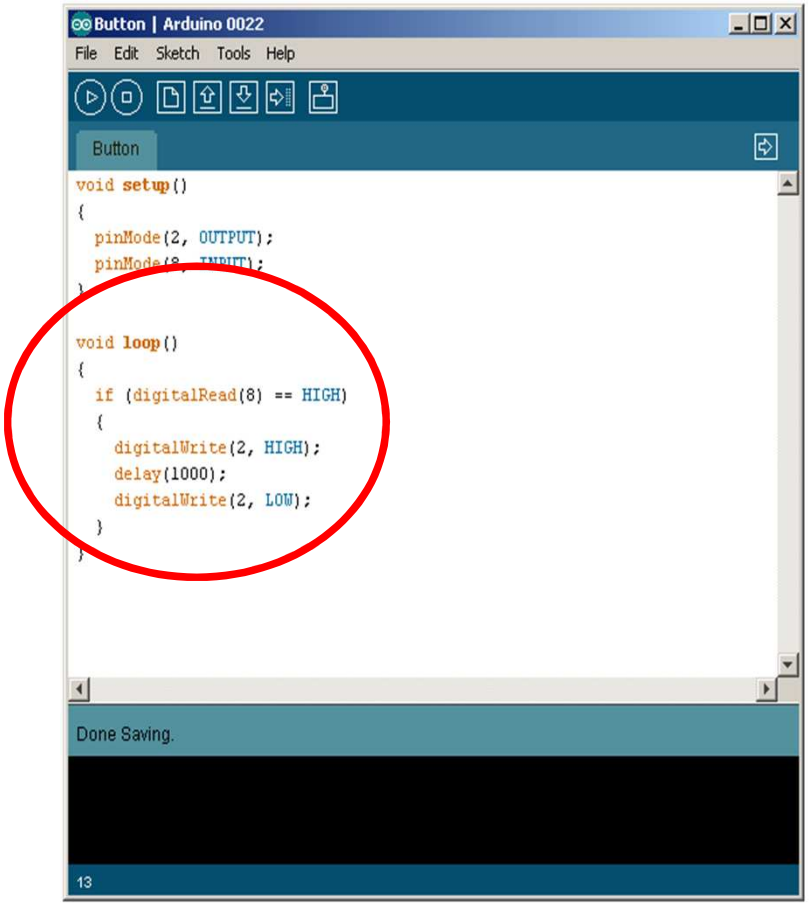
Soluzione analogica: circuito integrato NE555 (multivibratore) in configurazione monostabile (schema funzionale)

La costante di tempo è funzione dei componenti del sistema

# Esempio: logica programmabile

Compito: accendere un LED per un tempo t stabilito, al premere di un pulsante

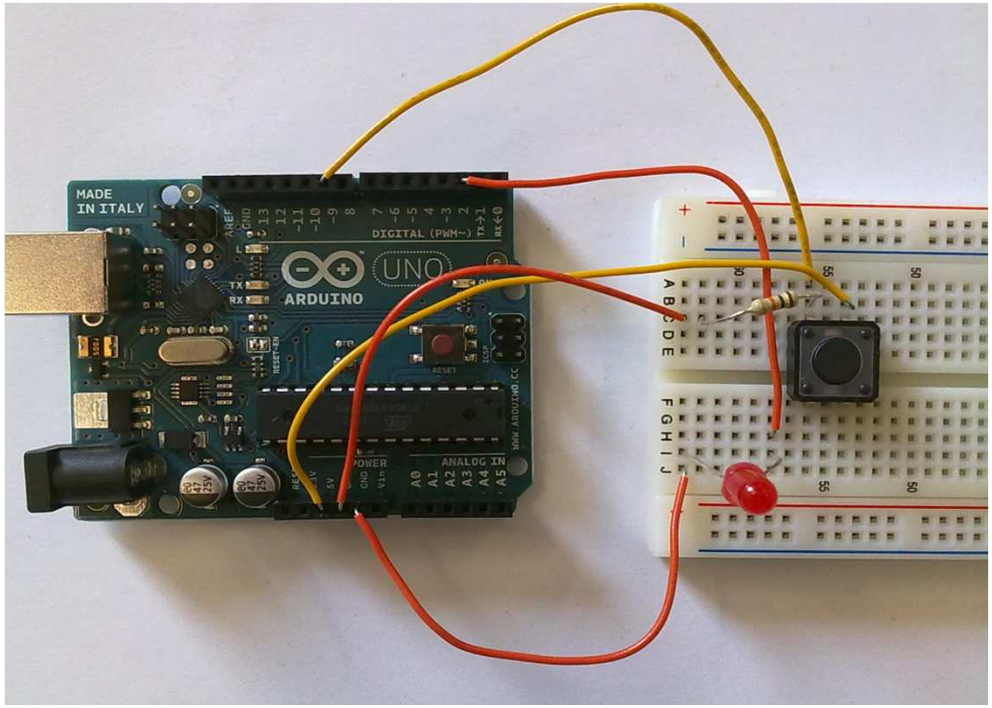


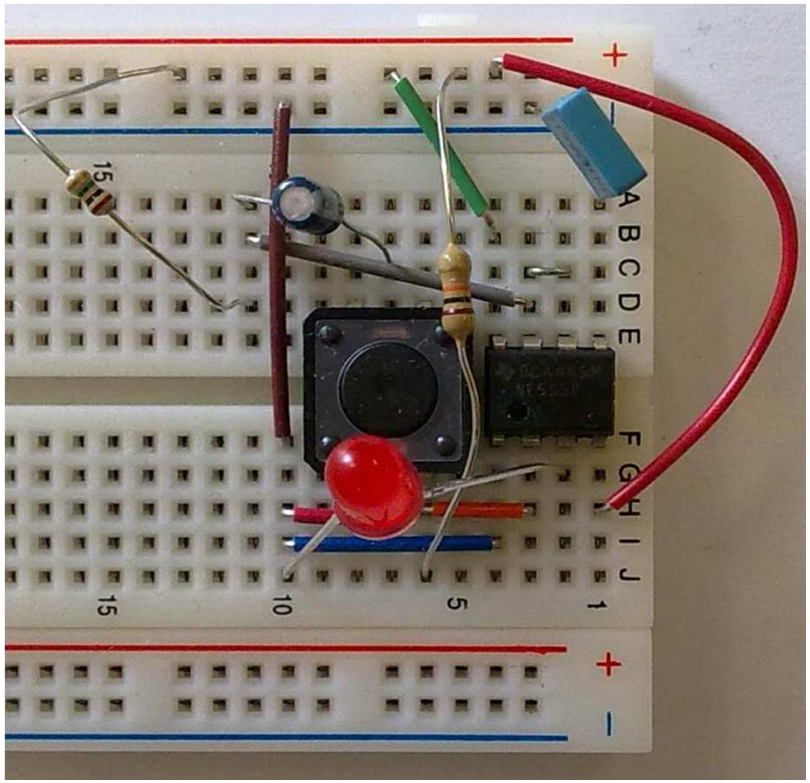

La costante di tempo è una variabile del software delay(t)

- Maggior precisione (incertezza nei valori di R e C)
- Riprogrammabile con facilità
- Maggior flessibilità

# Realizzazione pratica

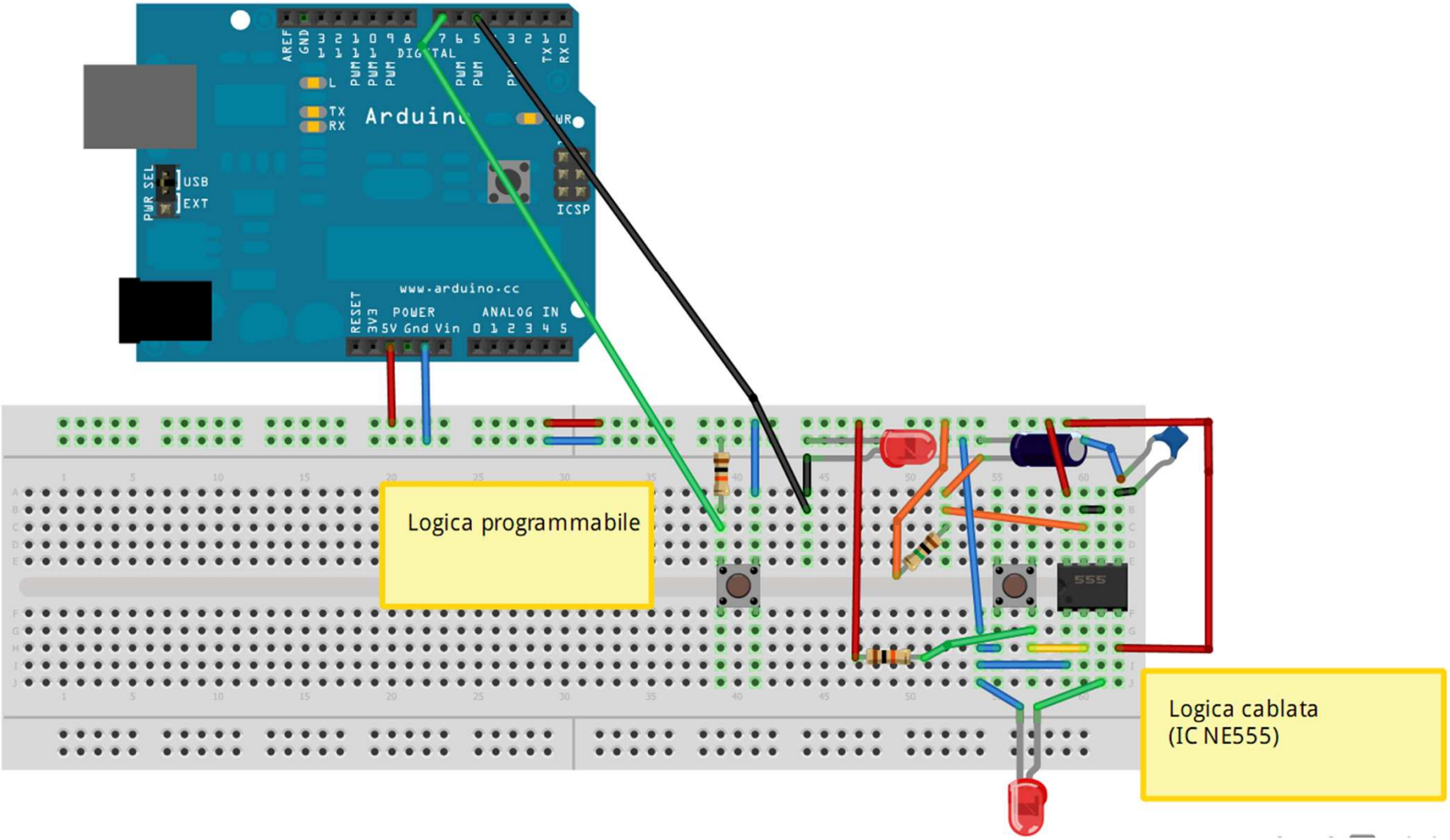



Logica programmabile



Logica cablata NE555

# Schema elettrico



# Software da caricare su Arduino: TimedButton.pde

```
#define BUTTON 7
#define LED 9
int brightness = 255;
int durata = 2000; //[ms]
int val = 0;

void setup()
{
 pinMode(LED,OUTPUT);
 pinMode(BUTTON,INPUT);
}

void loop()
{
  val = digitalRead(BUTTON);
  if (val == HIGH)
  {
      delay(10); //debouncing
      digitalWrite(LED,HIGH);
      delay(durata); //tempo in cui il LED resta acceso
      digitalWrite(LED,LOW);
  }
}
```

Definizioni

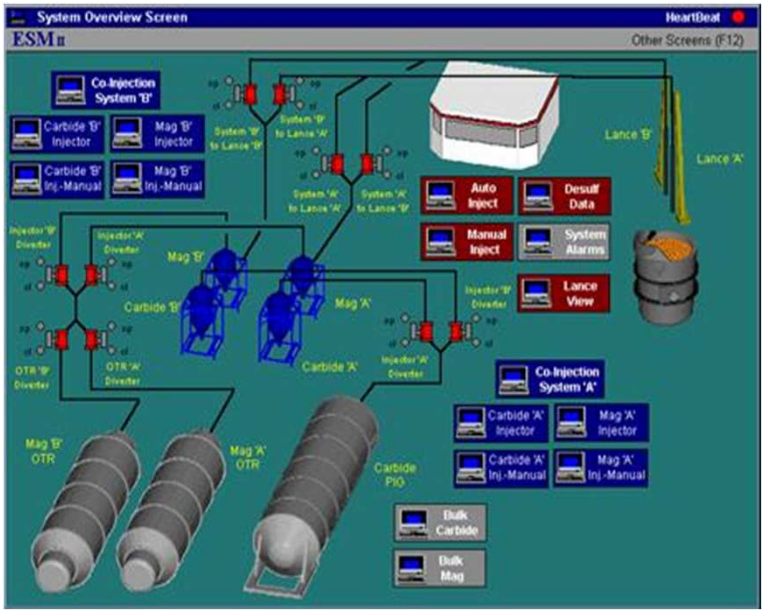
Inizializzazione I/O

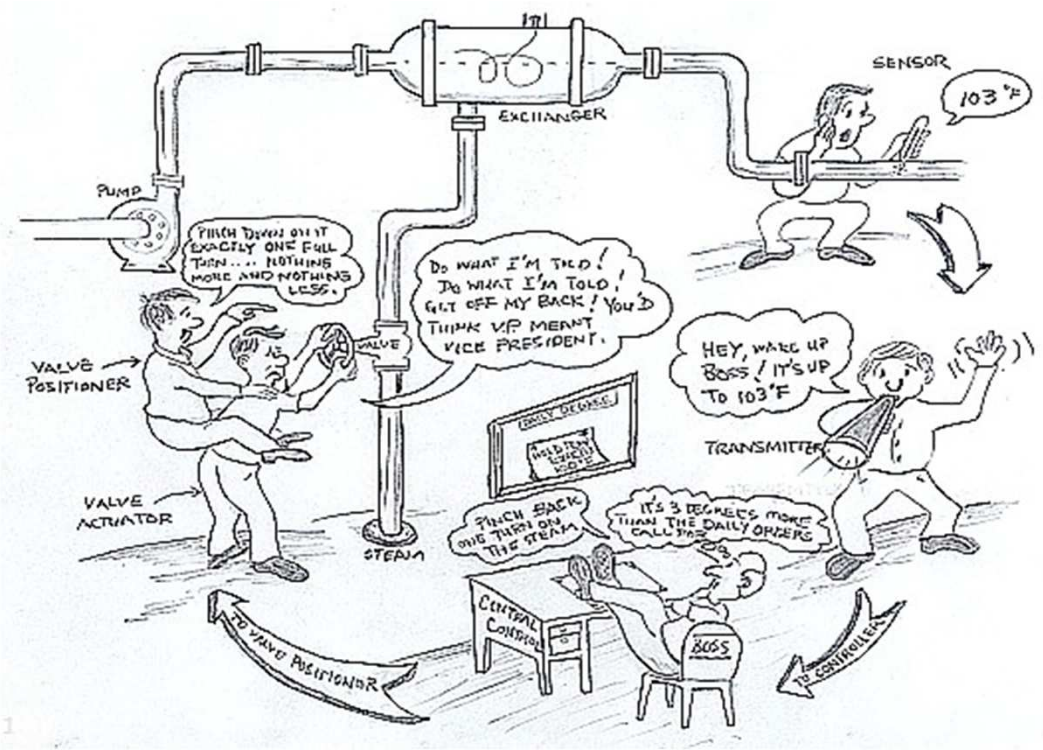
Ciclo

# Esperienza 2
# Controllo di processo

# Controllo di processo

# PLC (programmable logic controller)

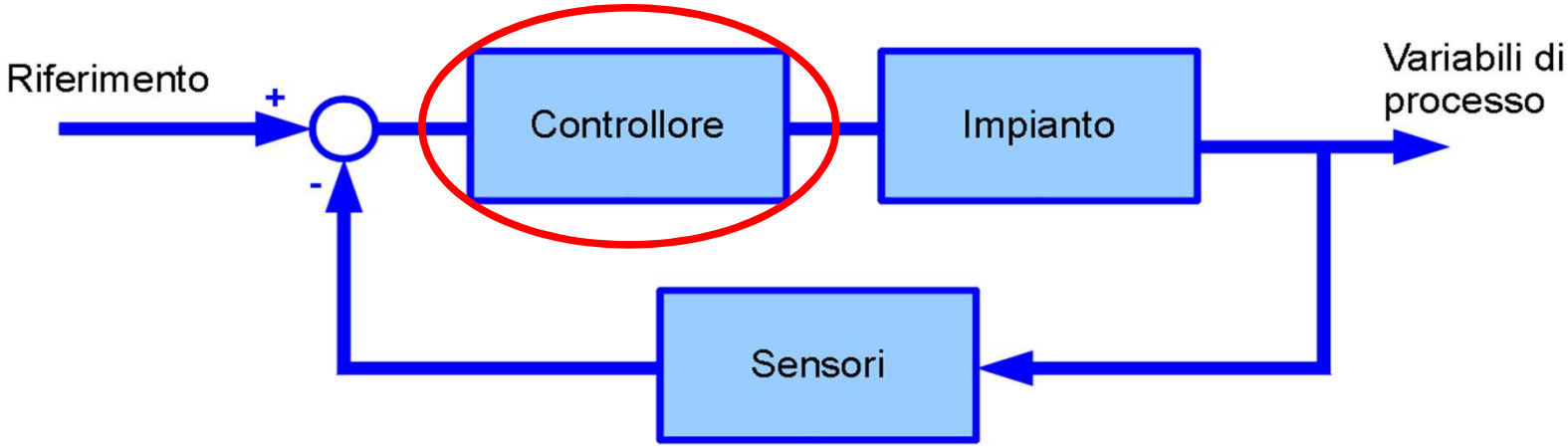


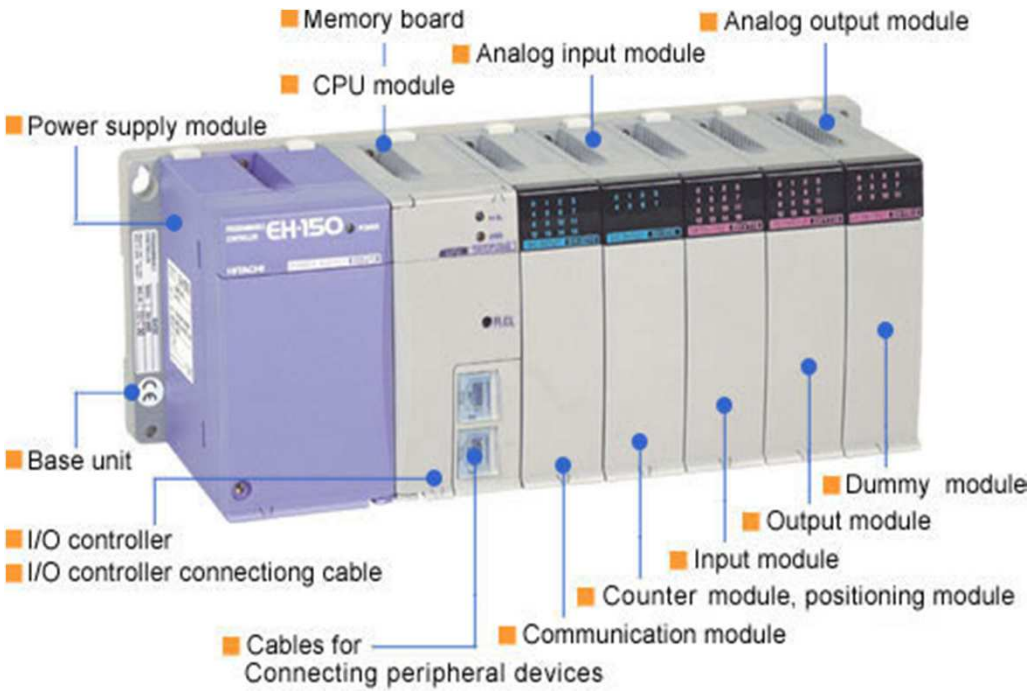




Esempio di controllo grandezze:

- Di processo
  - Temperatura
  - Pressione
- Tecnologiche
  - Finitura superficiale

# PLC: componenti base

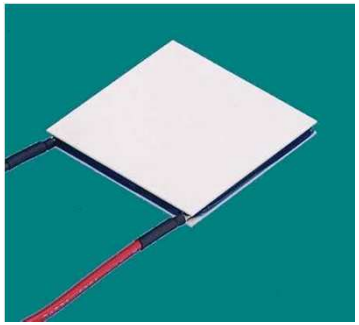


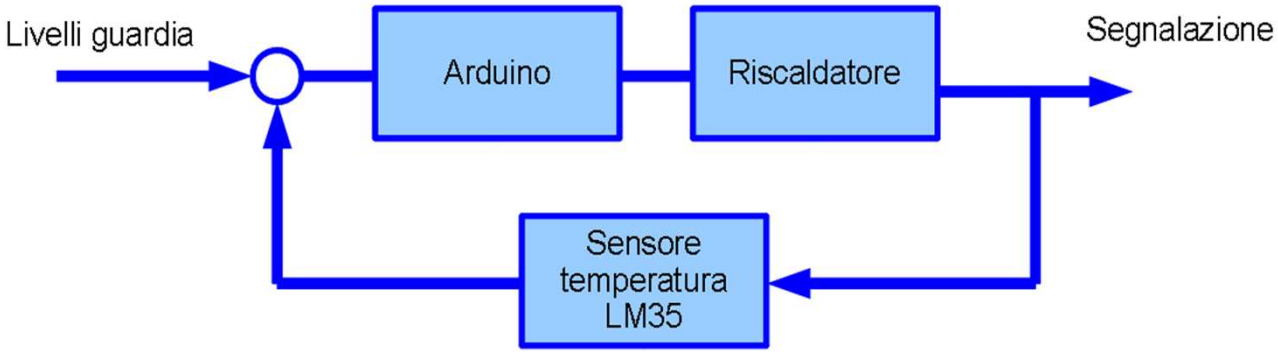

Componenti di un PLC


RACK industriale

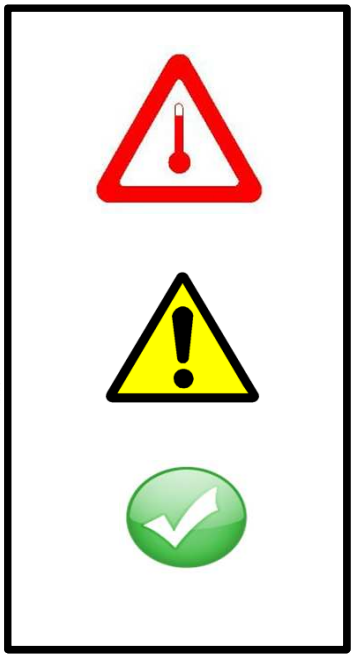
# PLC Monitoraggio della temperatura



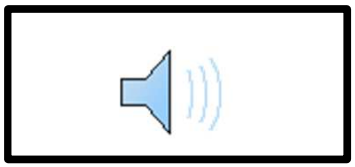
Cella di peltier

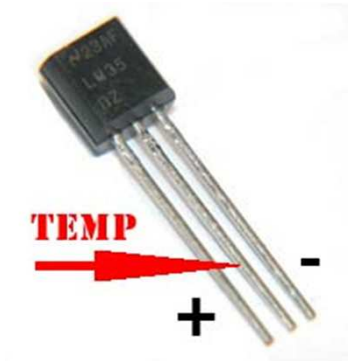



Visiva

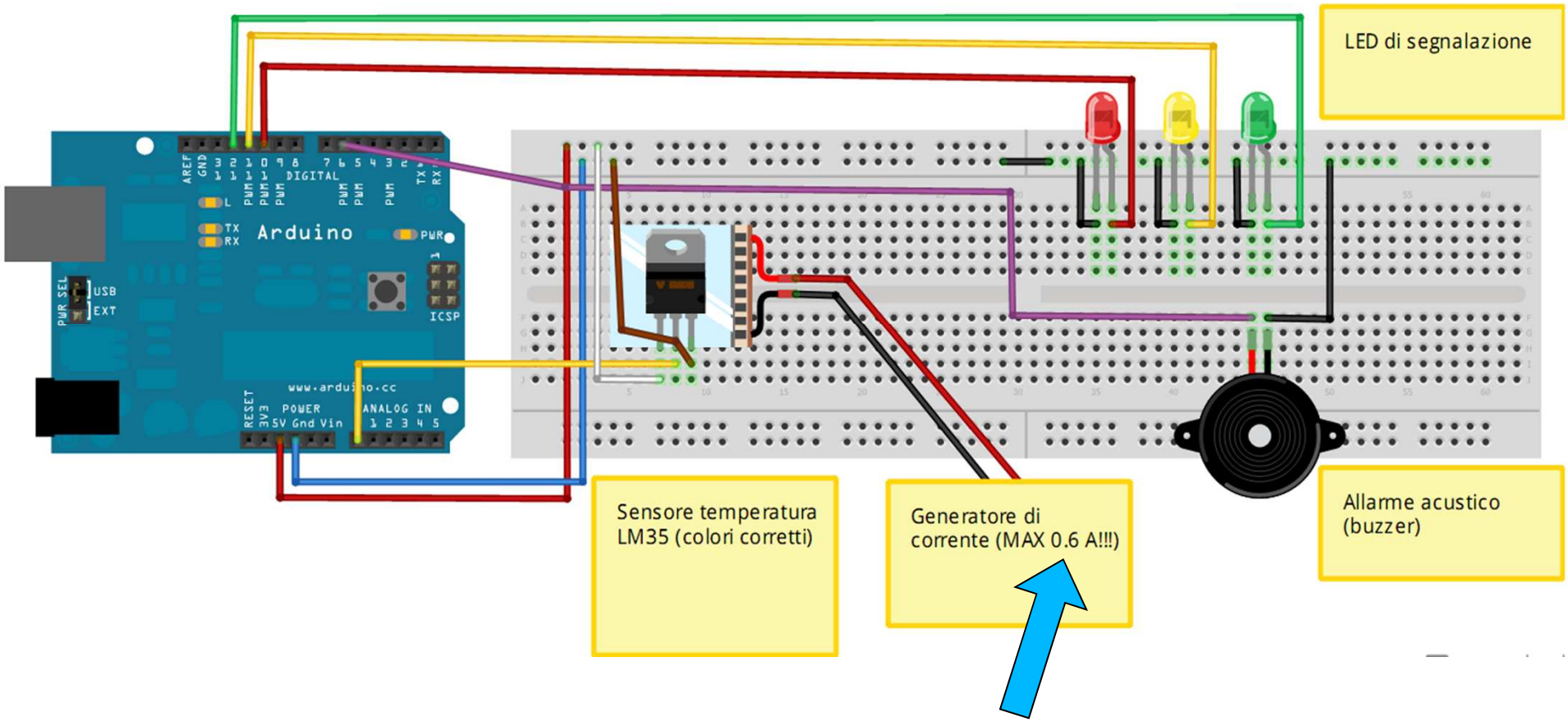


Acustica





# PLC schema elettrico




fare attenzione!

# Software da usare

Programmi residenti e su PC:

- programma residente su Arduino: gira indipendentemente, ovvero posso staccare il PC
- programma su PC (processing): sfrutta Arduino per leggere informazioni e per controllare dispositivi; se si stacca il collegamento da PC, il programma smette di funzionare.

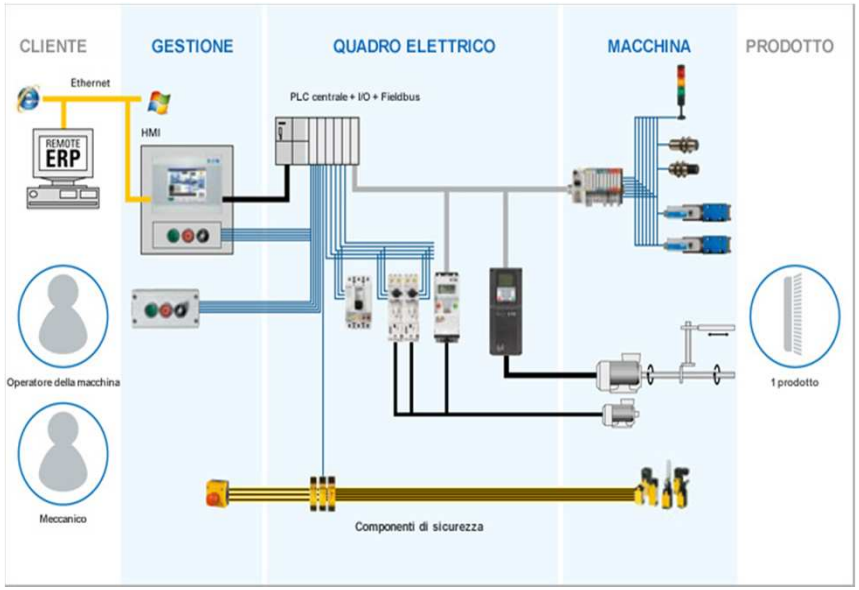
Questa esperienza richiede due software:

- residente su Arduino, serve solo per comunicare
- su PC, che visualizza e attua la logica

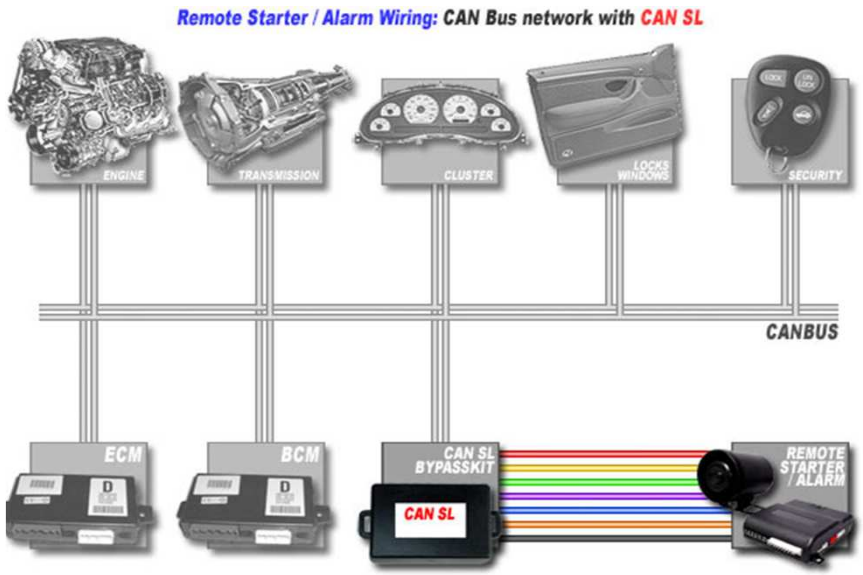
Si trova nella cartella: Graph2D_adv

# Esperienza 3
# Bus di campo

# Bus di campo: introduzione




Bus industriale



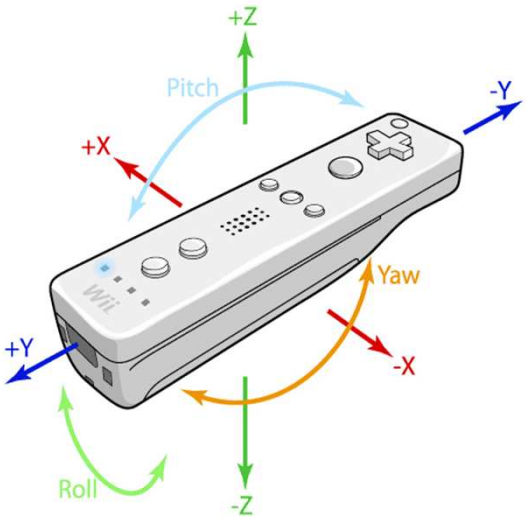

CAN (Automotive)

# Bus di campo: introduzione I²C

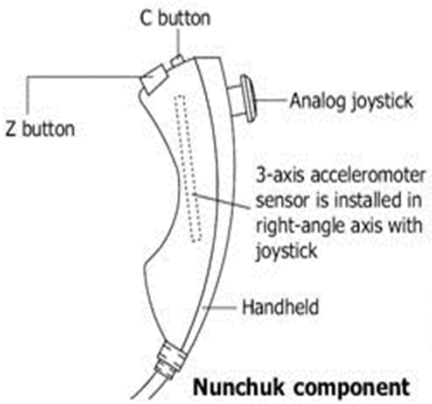
Il protollo I²C è usato per comunicare con dispositivi in cui semplicità e basso costo sono prioritari rispetto alla velocità di trasmissione. Trattandosi di un protocollo **seriale** i vantaggi offerti sono l'impegno di sole due linee (e quindi due pin dei dispositivi che lo usano), oltre a quelle di alimentazione (+5V o +3,3V) ed il riferimento (GND).

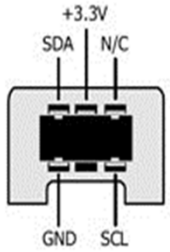
Applicazioni comuni sono:

- Accesso a Memorie flash ed EEPROM
- Accesso a DAC e ADC a bassa velocità
- Cambiamento dei settaggi nei monitor
- Controllo di display come nei telefoni cellulari
- Accesso sensori MEMS (esempio controller nintendo WII)








**Pin assignment of Nunchuk connector. See from front view.**

# Bus di campo: funzionamento I²C

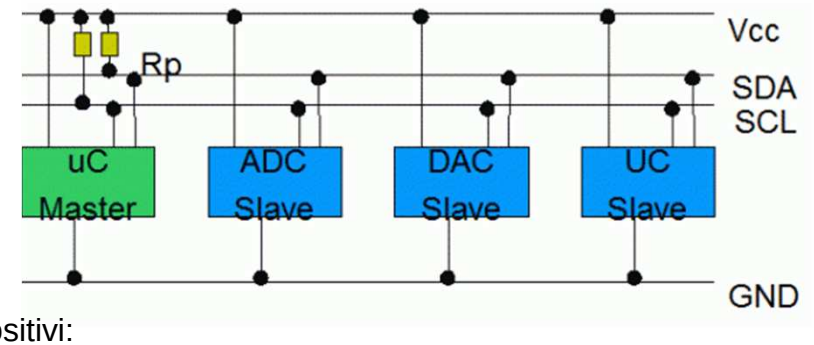
Il protocollo hardware dell'I2C richiede due linee seriali di comunicazione:

SDA (Serial DAta line) per i dati

SCL (Serial Clock Line) per il clock (per la presenza di questo segnale l'I2C è un bus sincrono)

Alimentazione (Vcc: +5V o +3,3V)

Riferimento di zero (GND)




Esistono due tipologia di dispositivi:

nodo master – il dispositivo che emette il segnale di clock (generalmente il microcontrollore)

nodo slave – il nodo che si sincronizza sul segnale di clock senza poterlo controllare (generalmente i sensori)

Il tipo di trasferimento dati può essere:

un master trasmette – controlla il clock e invia dati agli slave

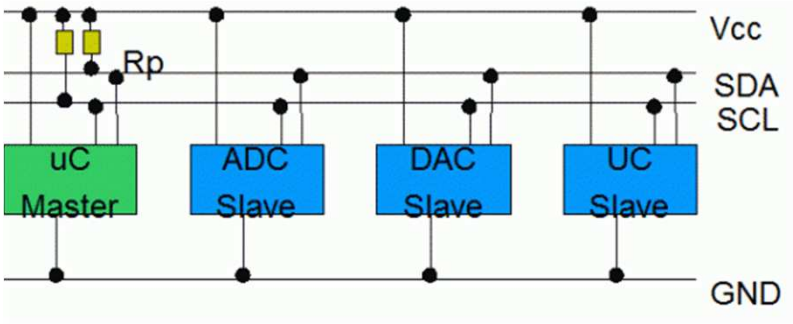
un master riceve - controlla il clock ma riceve dati dallo slave

lo slave trasmette – il dispositivo non controlla il clock ma invia dati al master

lo slave riceve – il dispositivo non controlla il clock e riceve dati dal master

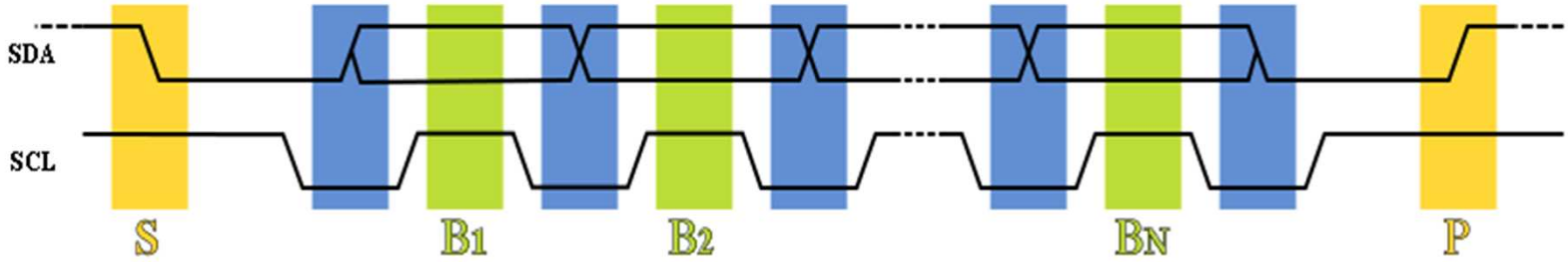
# Bus di campo: funzionamento I²C

Ogni dispositivo della rete ha un proprio indirizzo. Il corretto indirizzamento dei dati è garantito dalle resistenze di pull-up Rp che mantengono la linea normalmente “alta”




Trasmissione dei dati

S è lo START bit (la linea SDA viene forzata bassa dal master mentre il clock SCL è a livello logico alto). Segue, quando SCL è basso il settaggio del primo bit B1 (in blu) la commutazione di SCL indica che il dato è stabile e può essere letto (verde). La stessa procedura prosegue fino all'ultimo bit BN. La transazione termina con lo STOP bit (P) in giallo in cui SDA viene commutato da basso ad alto quando SCL è alto



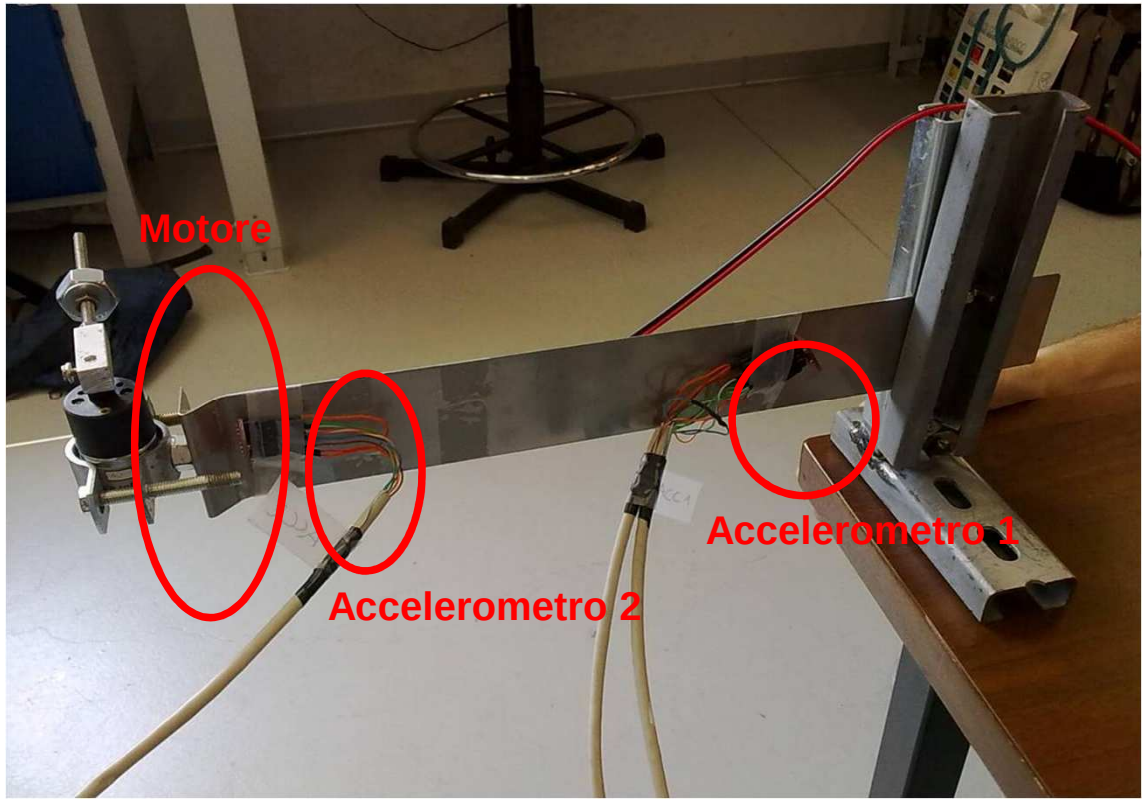

*(tratto da wikipedia)*

Altre informazioni: http://en.wikipedia.org/wiki/I2c

# I²C: misura di vibrazioni

Misura delle vibrazioni su un sistema meccanico mediante accelerometro MEMS BMA180 prodotto dalla BOSCH per il settore automotive

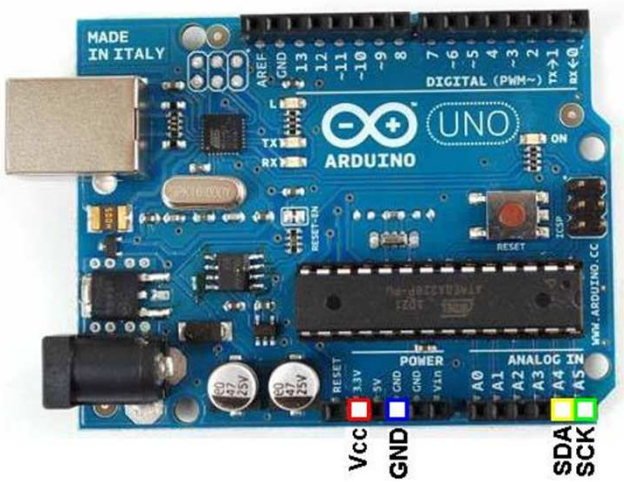
Accelerometro BMA180




Sistema vibrante

Il motore movimenta una massa eccentrica che forza la mensola ad oscillare. Le vibrazioni sono misurate dai due accelerometri. La velocità di rotazione del motore è regolabile via software mediante un'uscita PWM (Pulse Width Modulation: onda quadra a duty cicle variabile) che simula un'uscita analogica a valore variabile.

# I²C: misura di vibrazioni




Collegamenti utilizzati

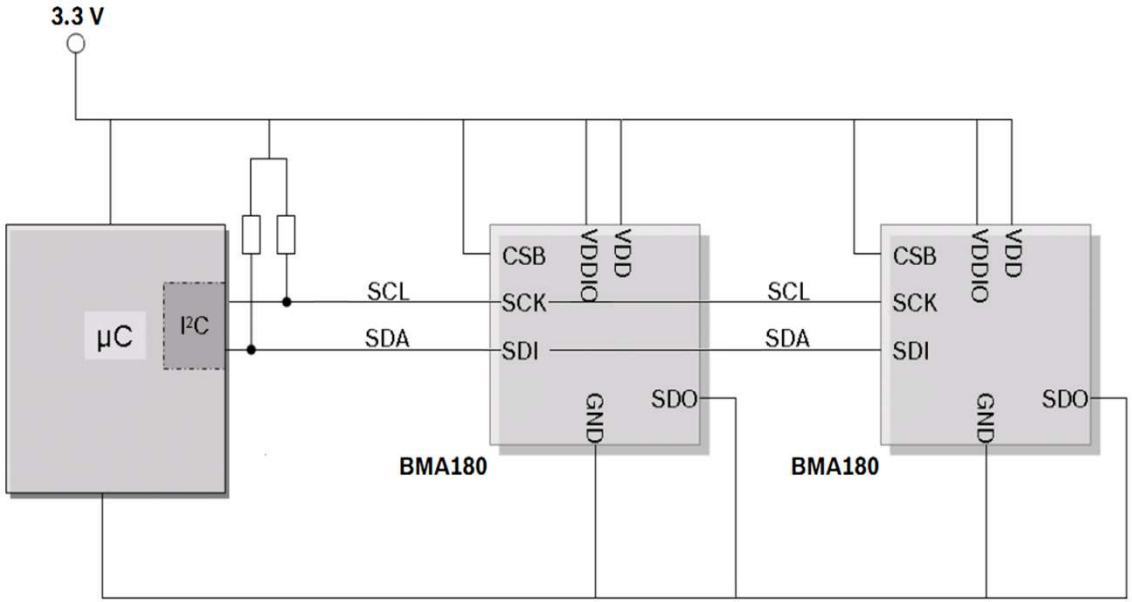
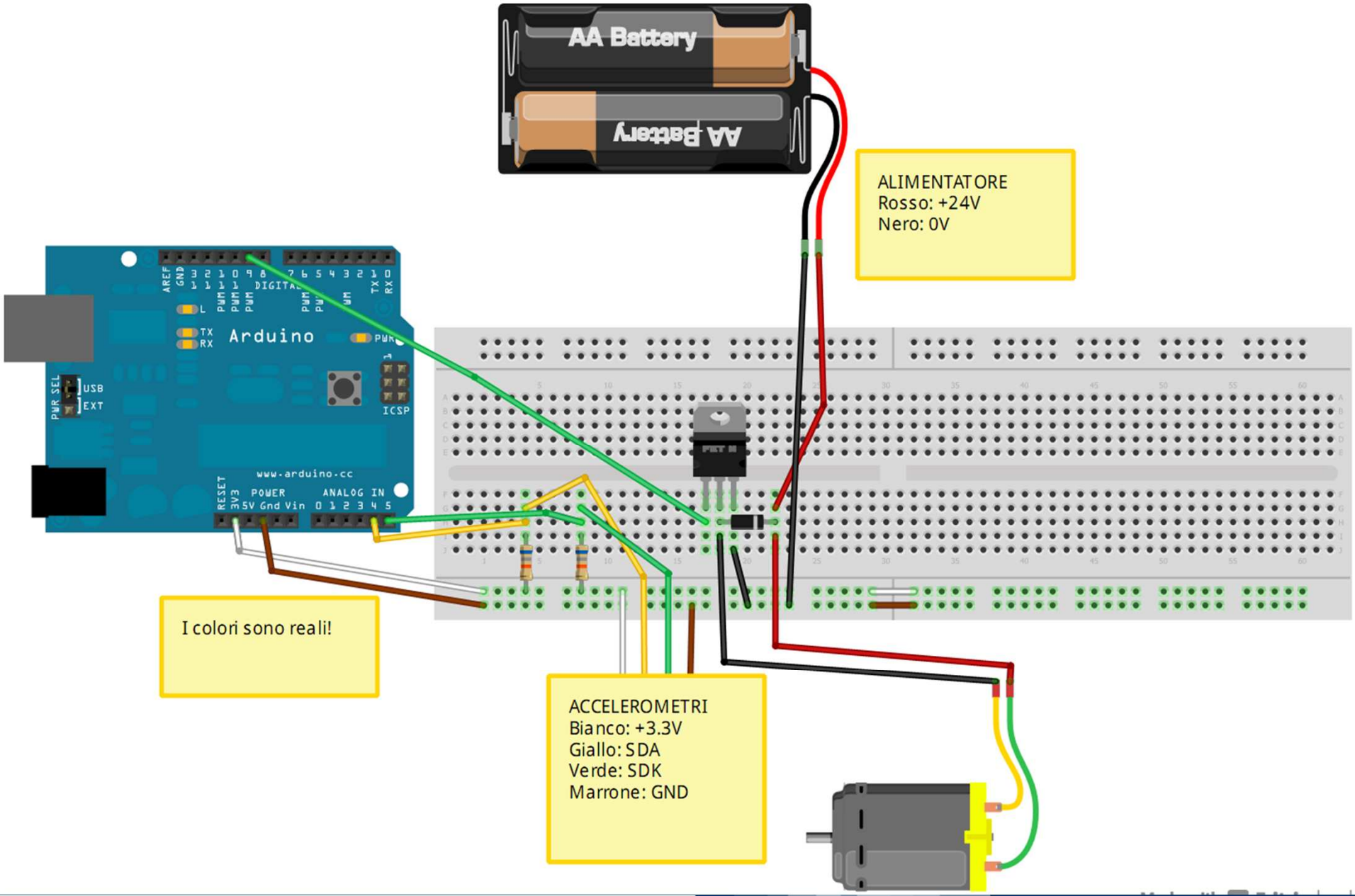



Diagramma di connessione dei componenti

# I²C: schema elettrico



# Software da usare

Analogo alla esperienza 2: software residente su Arduino usato per comunicare con il PC (processing).

Si trova nella cartella: EsrcitazioneMotoreArduino