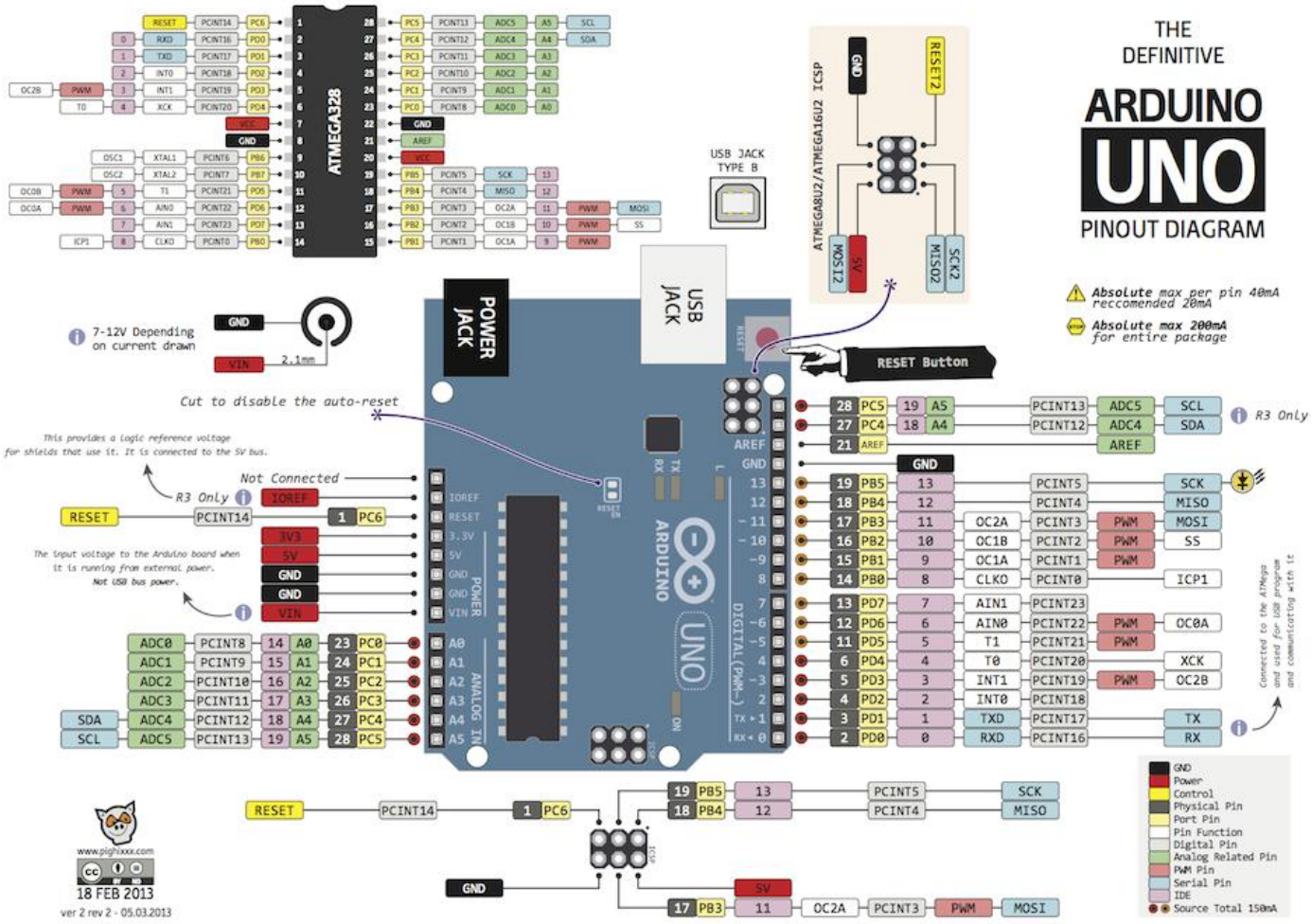
# Arduino uno

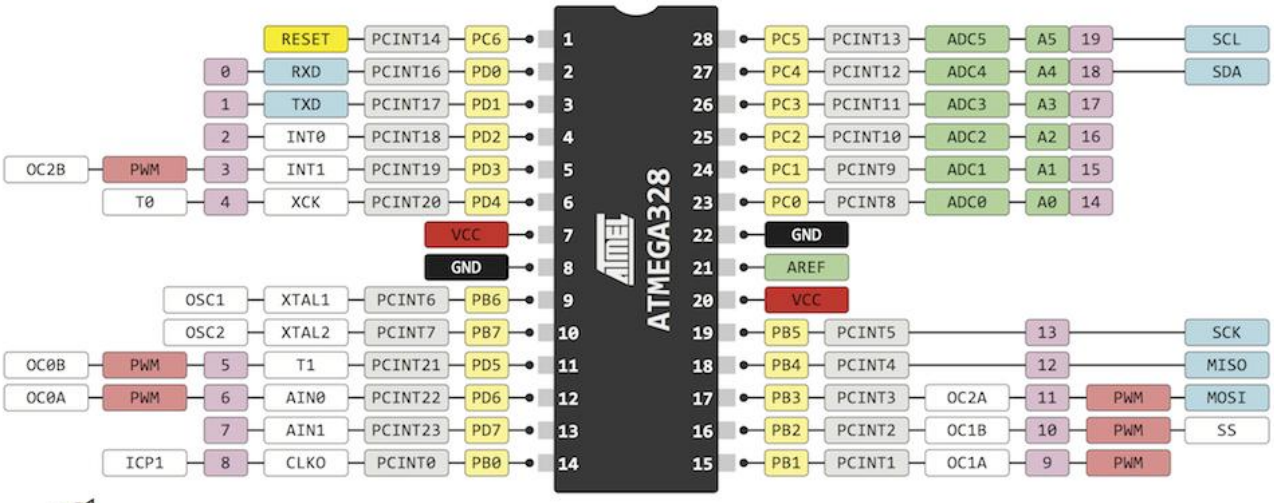
## *Caratteristiche generali*

- Piattaforma con microprocessore ATMEL-ATMEGA
- Idea Made in Italy, ideato nel 2005
- Funzionante su sistemi operativi Windows, Linux, Mac
- La scheda di sviluppo I/O è semplicissima e si basa su un ambiente di sviluppo che usa una libreria Wiring per semplificare la scrittura in C/C++. L'ambiente di programmazione integrato (IDE) è un ambiente multipiattaforma scritto in Java.
- Wiring è un ambiente di programmazione open-source per impieghi su schede elettroniche.
- La scheda Arduino è alimentata a 5 V con oscillatore a 16 MHz; la memoria programma del microcontrollore è di tipo flash di 32 KB.
- Arduino 1 è dotato di una memoria SRAM di 2 kB e una EEPROM di 1 kB.
- La programmazione della scheda è stata pensata via seriale RS-232 ma attualmente è programmata tramite USB.
- Il numero di ingressi uscite digitali è pari a 14 di cui 6 possono fornire il comando PWM.
- Il numero di ingressi analogici è pari a 6.
- Per programmare Arduino non è necessario conoscere tutto sui microcontrollori, l'importante è conoscere l'ambiente di sviluppo, la sintassi e…l'elettronica.
- Arduino ha la possibilità di avere degli Shield, delle espansioni con circuiti di acquisizione già opportunamente progettate.

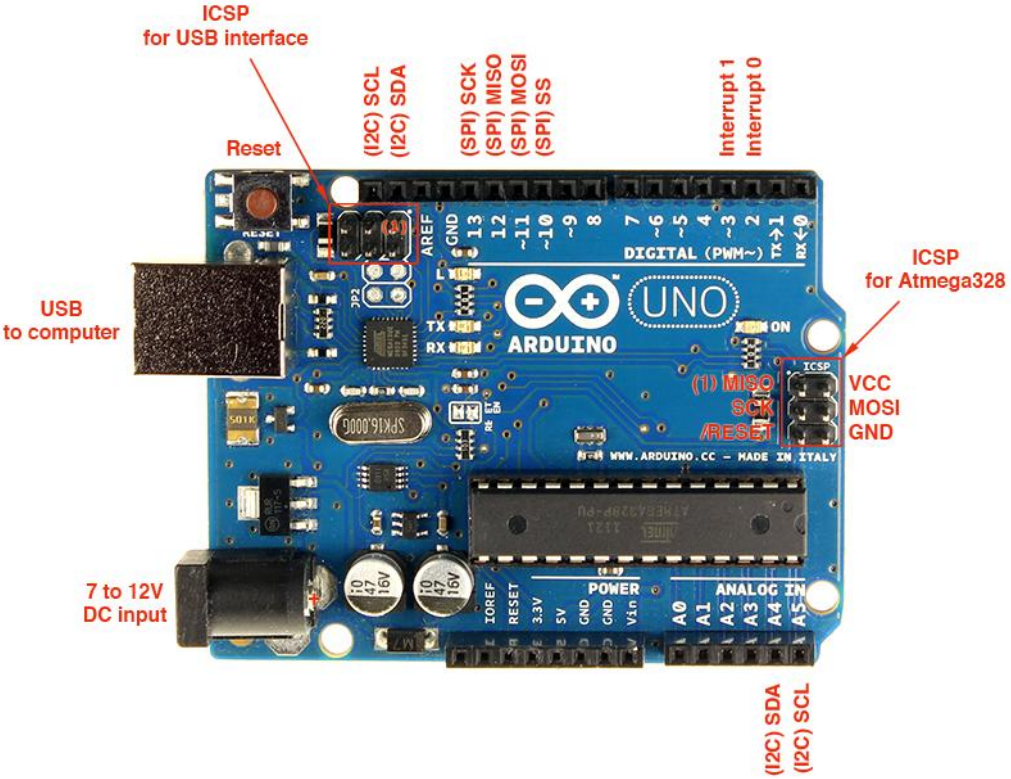
La piedinatura è la seguente:



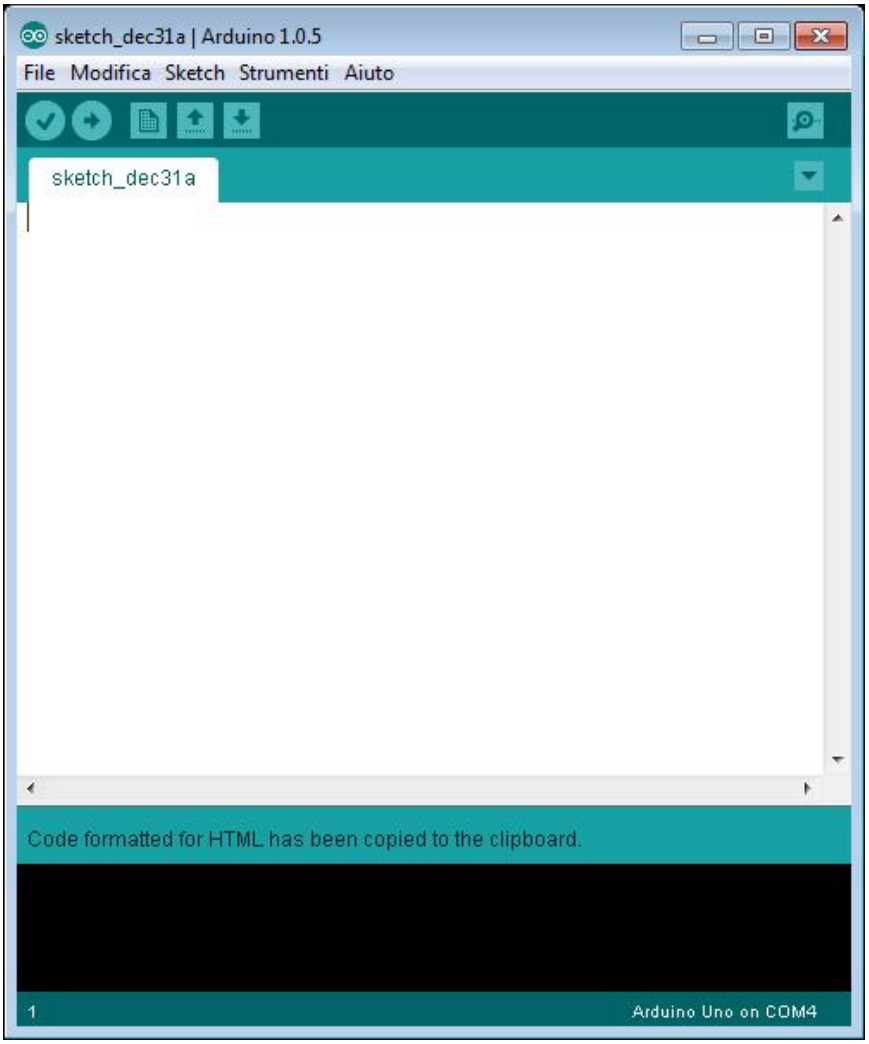
THE
DEFINITIVE
# ATMEGA328
# &Arduino
PINOUT DIAGRAM





Ambiente di sviluppo




I file scritti per Arduino si chiamano Sketch e sono salvati automaticamente con il nome dato dal software stesso con la data e l'ora di creazione con estensione .ino

# Sintassi

**Strutture**

- void setup()
- void loop()

**Controllo**

- if
- if…else
- while

- do…while
- for
- switch…case
- break
- continue
- return
- goto

**Sintassi**

- {}
- ;
- // commento per riga
- /*……*/ commento per più righe

**Operazioni aritmetiche**

- = di assegnazione
- + somma
- * prodotto
- - sottrazione
- / divisione
- % modulo

**Operazioni di confronto**

- == uguale
- != diverso
- < minore
- >maggiore
- <= minore o uguale
- >= maggiore o uguale

**Operazioni booleane**

- && AND
- || OR
- ! NOT

**Operatori**

- ++ incremento
- -- decremento
- += somma composta
- -=sottrazione composta

- *= prodotto composto
- /= divisione composto

**Costanti**

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

**Tipi di dati**

- Boolean
- char
- byte
- int
- float
- long
- double
- insigned char
- string
- array
- void
- insigned int

**Conversioni**

- int()
- long()
- float()

**Funzioni**

- INPUT/OUTPUT digitali
    - pinMode(n pin, modo input o output)
    - digitalWrite(n pin, modo input o output)
    - int digitalRead(n pin)

**Digital I/O**

• pinMode(pin, mode)

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

**Analog I/O**

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

**Advanced I/O**

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

**Time**

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

**Math**

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

**Trigonometry**

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

**Random Numbers**

- randomSeed(seed)
- long random(max)

• long random(min, max)

**Serial Communication**

Usate per comunicare tra schede arduino, oppure tra schede arduino ed il PC. Vengono usati i pin TX ed RX, facenti capo al modulo.

USART del microcontrollore.

• Serial.begin(speed)

• int Serial.available()

• int Serial.read()

• Serial.flush()

• Serial.print(data)

• Serial.println(data)

# Struttura generale di uno sketch

/*commento su più righe*/

//commento su una riga

Dichiarazioni di variabili;

void setup(){ stabilisce se i pin sono di ingresso o in uscita}

void loop(){}

es: lampeggio di un led sul pin 6

*int ledpin=6;*

*void setup()*[1]

*pinMode(ledpin, OUTPUT);*

»

*Void loop()*[1]

*digitalWrite(ledpin, HIGH);*

*delay(200);*

*digitalWrite(ledpin,LOW);*

*delay(200);*

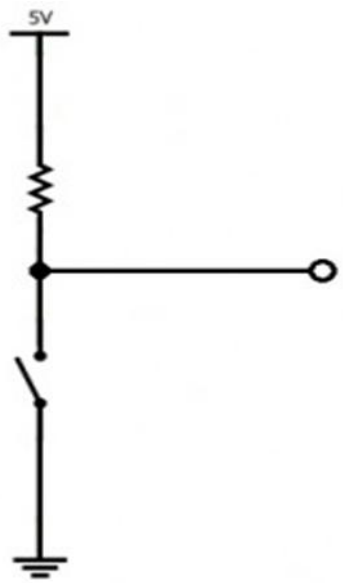
»

**La funzione map**

```
void loop()
{
  int val = analogRead(A0);
 val = map(val, 0, 1023, 0, 255);//legge valori da 0 a 1023 (10 bit) e li //mappa in valori da 0 a 255
(8 bit)
  ……
}
```

# Resistenze di pull up

Le resistenze sono delle resistenze interne che possono essere attivate o disattivate. Per essere attive una resistenze di pull up su un pin bisogna dichiarare digitalWrite(pin,HIGH); . La resistenza di pull up è di circa 20-50 K. Quando il pin è HIGH, l'interruttore è aperto e la corrente fluisce verso l'esterno. Se l'interruttore è chiuso la corrente fluisce verso la massa e il pin è LOW.




Pull up attiva

**Interruttore**

# LED, display e buzzer

**Semaforo**

Problema
Si vuole che il semaforo realizzato con tre led passi dal funzionamento normale a quello con giallo lampeggiante premendo un pulsante e viceversa.

```
const int verde=2;
const int giallo=3;
const int rosso=4;
const int ingresso=10;
boolean lampeggio=false;
void setup()
{
pinMode(verde,OUTPUT);
pinMode(giallo,OUTPUT);
pinMode(rosso,OUTPUT);
pinMode(ingresso,INPUT);
digitalWrite(ingresso,HIGH); // attivazione pull up
} // in alternativa pinMode(ingresso, INPUT_PULLUP);

void normale(){
digitalWrite(rosso,LOW);
digitalWrite(verde,HIGH);
digitalWrite(giallo,LOW);
delay(4000);
digitalWrite(giallo,HIGH);
digitalWrite(verde,LOW);
delay(1000);
digitalWrite(giallo,LOW);
digitalWrite(rosso,HIGH);
delay(4000);
}
void gialloL() {
digitalWrite(verde,LOW);
digitalWrite(rosso,LOW);
digitalWrite(giallo,HIGH);
delay(500);
digitalWrite(giallo,LOW);
delay(500);
}

void loop() {
if(lampeggio==false)
normale();
else
gialloL();
```

```
if (digitalRead(ingresso)==0)
lampeggio=!lampeggio;
}
```

## Suoni da arduino

```
int Buzzer = 3;
void setup() {
pinMode(Buzzer, OUTPUT);
}
void loop() {
// Suona una nota sul pin 3 alla frequenza di 1000Hz per 200msec:
tone(Buzzer, 1000, 200);
delay(1000);
}
```

Tone è una funzione di arduino

**Una piccola musichetta natalizia**

```
const int la4=440;
const int si4=440*pow(2,(2.0/12));
const int do5 =440*pow(2,(3.0/12));
const int re5=440*pow(2,(5.0/12));
const int sib4=440*pow(2,(1.0/12));
const int sol4=440*pow(2,-(2.0/12));
const int
tuScendi[]={do5,do5,re5,do5,sib4,sib4,la4,la4,sol4,la4,sib4,do5,do5,sib4,la4,sol4};
void setup() {
pinMode(3,OUTPUT);
}
void loop() {
for(int k=0;k<sizeof(tuScendi)/sizeof(int);k++){
tone(uscita,tuScendi[k],750);
delay(750);
}
delay(3000);
}
```

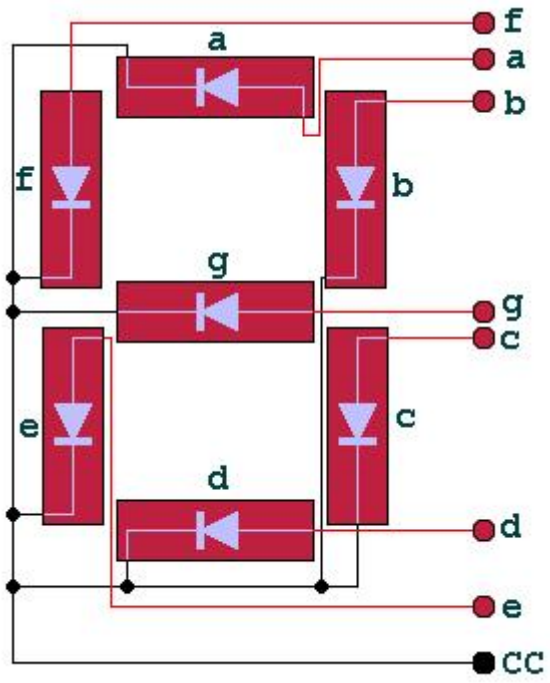
**Toni differenti al comando di un interruttore**

```
const int pulsante=3;

void setup(){

  pinMode(pulsante, INPUT);}

  void loop(){

int stato=0;

    stato=digitalRead(pulsante);

if(stato==HIGH)

    tone(8, 440, 30);

    else{

tone(8,500,30);

    delay(300);}

  }
```

## Contatore con display a 7 segs

```
const int segmentPins[7]={4,5,7,8,9,3,2}; // a b c d e f g
const int catodo =10; // segmentPin[0] a
const int attivazione[10]={0b00111111,0b00000110,0b01011011,
0b01001111,0b01100100,0b01101101,0b01111101,0b00000111,0b01111
111,0b01101111}; // 0gfedcba
void setup() {
for (int i=0;i<8;i++)
pinMode(segmentPins[i],OUTPUT);
pinMode(catodo,OUTPUT);
digitalWrite(catodo,LOW); // abilitazione del display
}
void loop() { // cifre da 0 a 9 in sequenza
for (int numero=0; numero<10;numero++)
{
showDigit(numero);
delay(1000);
}
}

void showDigit(int number) {
boolean isBitSet;
// abilitazione digit
```

```
for (int segment=0; segment<7; segment++)
{
isBitSet=bitRead(attivazione[number],segment);
digitalWrite(segmentPins[segment],isBitSet);
}
}
```




**PWM**

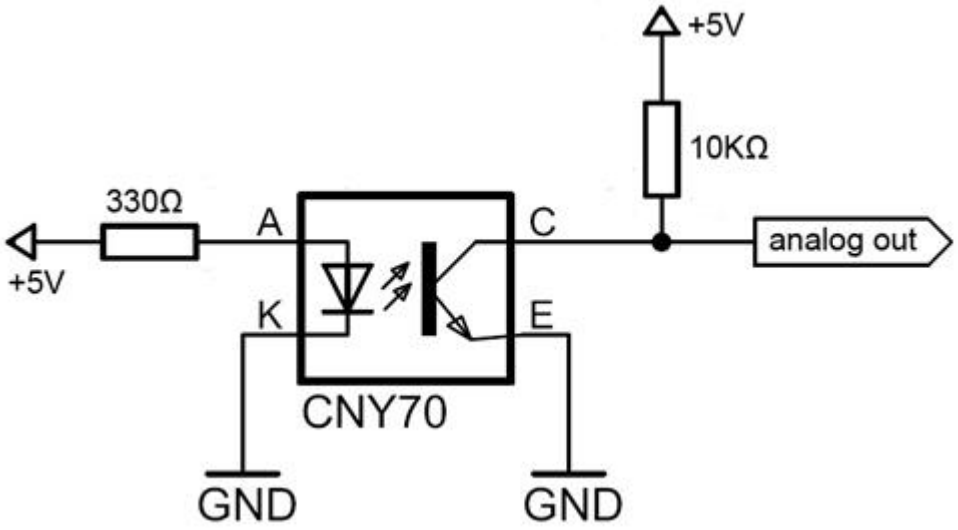
Pulse Width Modulation è un tipo di modulazione digitale che permette di ottenere una tensione media variabile dipendente dal rapporto tra la durata dell'impulso positivo e di quello negativo (duty-cycle)

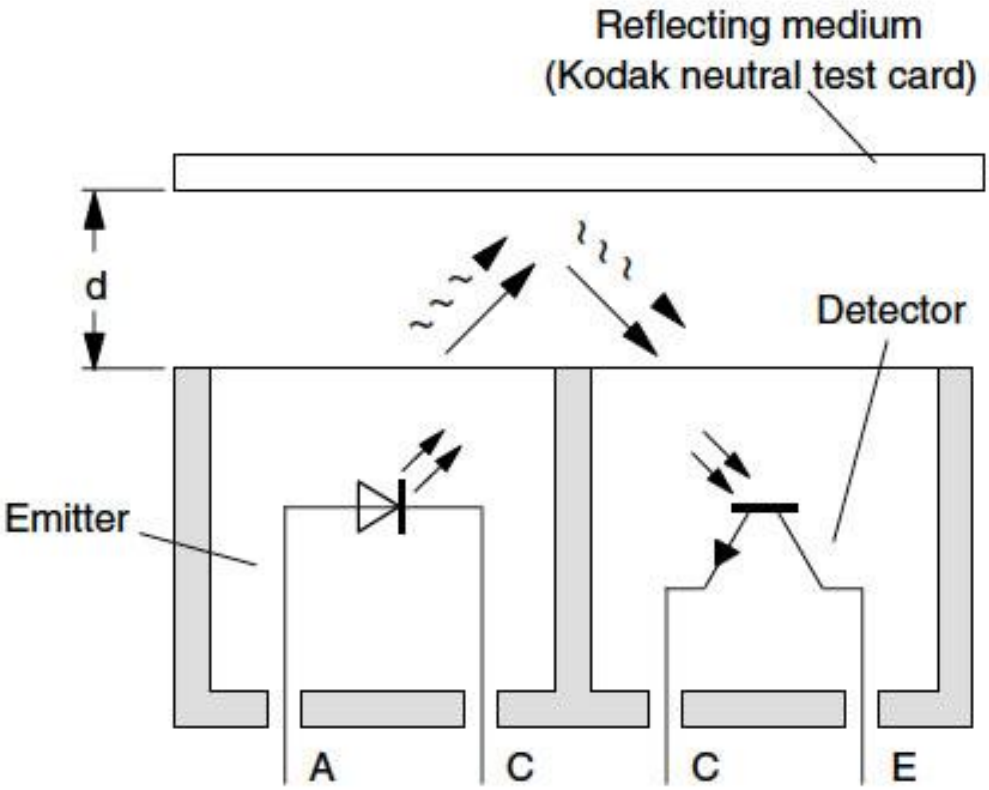
```
#define LED 11            // LED collegato al pin digitale 11
  int valoreFade = 0;        // variabile usata per contare in avanti e indietro
void setup()  {
  pinMode(LED, OUTPUT);     // imposta il pin digitale come output
}
  void loop()  {
  // procede ciclicamente da 0 a 254 (fade in -> aumento luminosità)
  for (valoreFade = 0 ; valoreFade < 255; valoreFade++) {
    analogWrite(LED, valoreFade);      //impostiamo la luminosità del LED
```

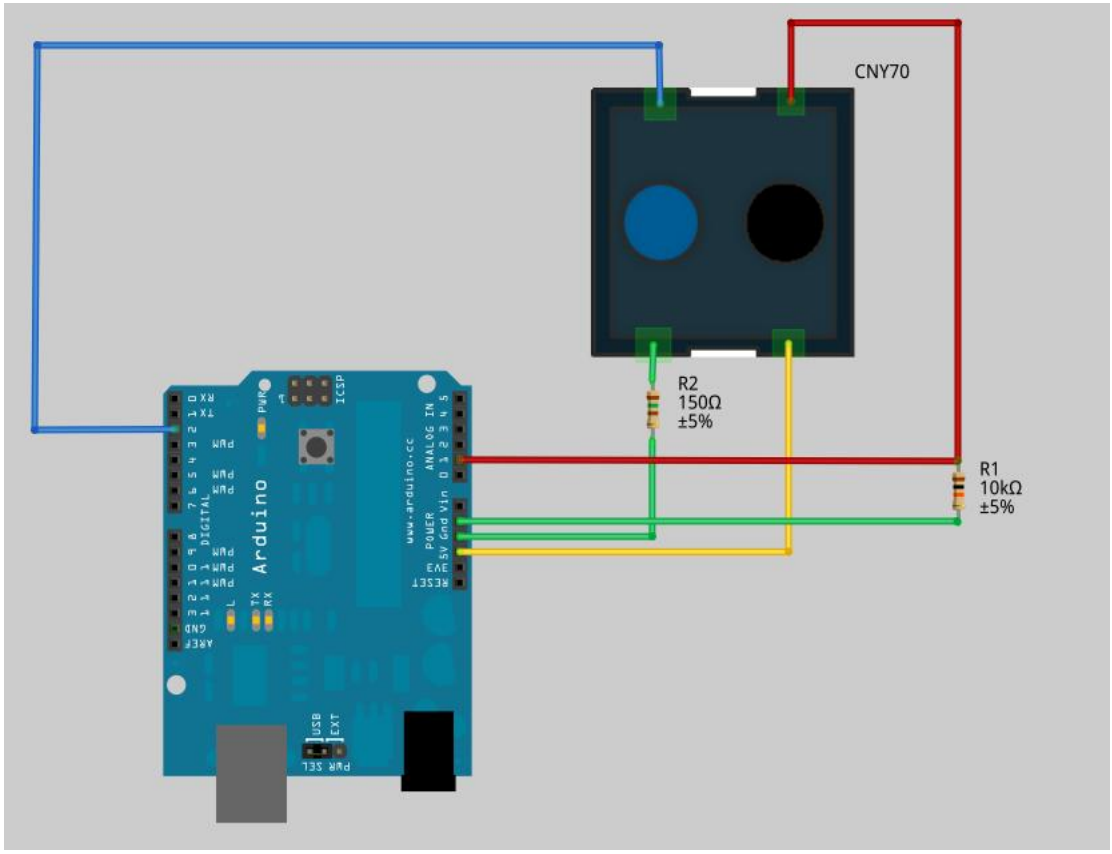
```
  delay(10);
  // aspettiamo 10ms per percepire la viariazione di luminosità,
  //perché analogWrite è istantaneo
}
// procede ciclicamente da 255 a 1 (fade out -> diminuzione della luminosità)
for(valoreFade = 255 ; valoreFade > 0; valoreFade--) {
  analogWrite(LED, valoreFade);      //impostiamo la luminosità del LED
  delay(10);
  // aspettiamo 10ms per percepire la viariazione di luminosità,
  //perché analogWrite è istantaneo
}
```

# Un Pò di sensori

## Sensore di prossimità a infrarosso

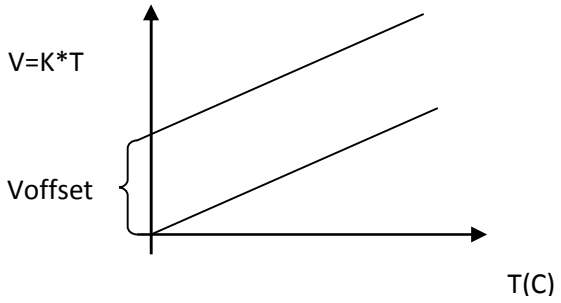



Reflecting medium
(Kodak neutral test card)
d
Detector
Emitter
A
C
C
E


CNY70
R2
150Ω
±5%
R1
10kΩ
±5%
Arduino

```
int value = 0;

void setup(){

  Serial.begin(9600);

  pinMode(A0,INPUT);

}

void loop(){

  value = analogRead(A0);

  Serial.println(value);

delay(2000);

}
```

## LM35 con Arduino




La relazione tra tensione e temperatura è V=kT dove k=10 mV/C. La tensione in uscita da un ADC di Arduino viene convertito secondo la regola V=Vref*V10/1024. Per ottenere il valore di temperatura basta dividere per K=10/1000 V/C. Il nucleo del programma è in grassetto.

```
/*

 Test_lettura_sensore_LM35.pde

 Il programma misura la temperatura tramite il sensore LM35
```

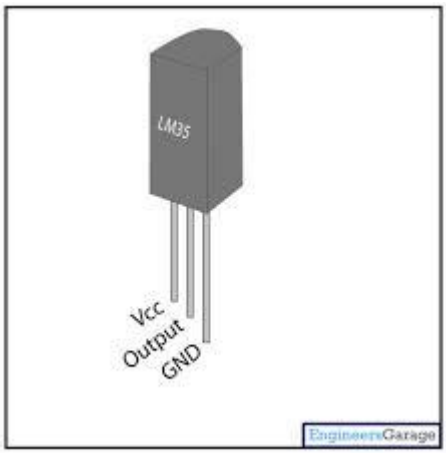
Pin +5V -> Alimentazione

Pin GND -> Alimentazione

Pin Analogico 1 -> lettura LM35

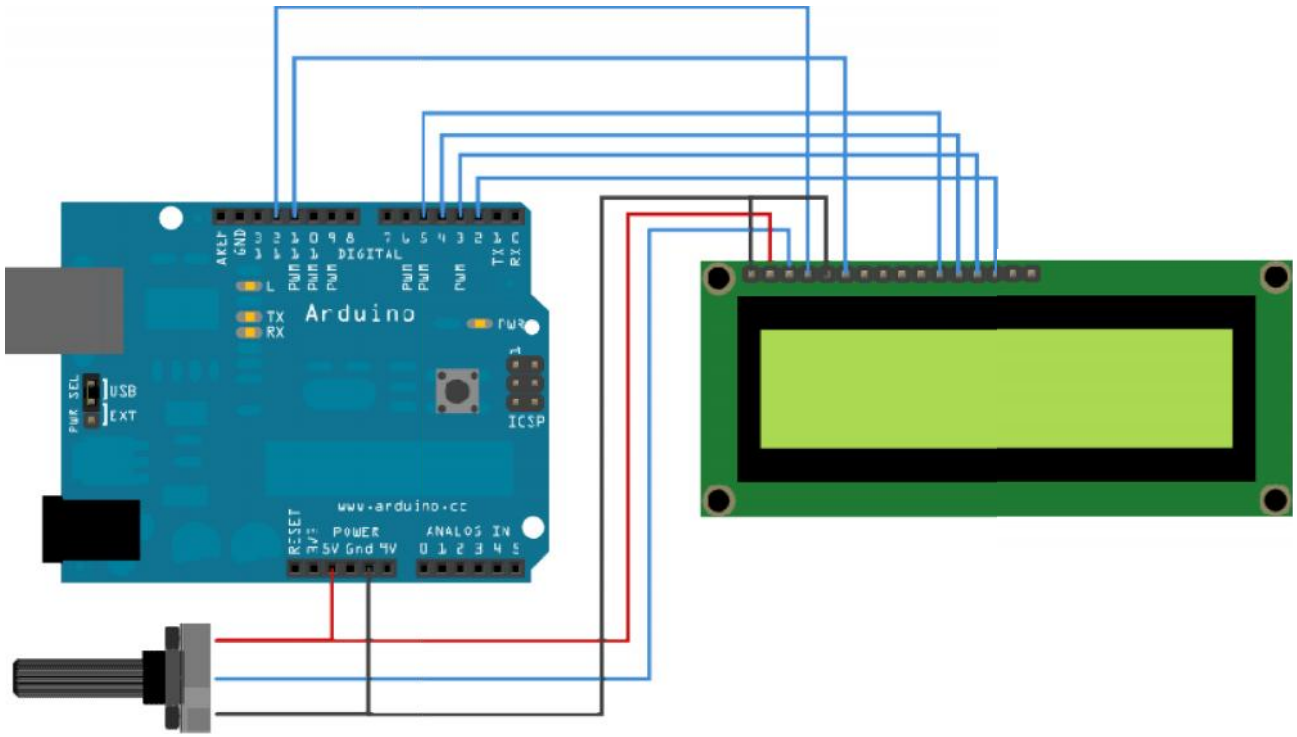
```
#include <LiquidCrystal.h>

/* Corrispondenza pin LCD <-> pin digitali di Arduino */

#define RS  8

#define EN  9

#define D7  7

#define D6  6

#define D5  5

#define D4  4

/* Numero del pin analogico sul quale è collegato l'LM35  */

#define LM35_pin 1

/* Definizioni globali */

float vref = 1.1;                    // Vref dell'ADC (quell'interno è di 1,1V)

LiquidCrystal lcd( RS, EN, D4, D5, D6, D7 ); // 'lcd' è una variabile di tipo LiquidCrystal */

/* Impostazione dell'hardware */

void setup()

{

  analogReference( INTERNAL );  // per l'ADC usiamo il Vref interno da 1,1V (migliore precisione)

  analogRead( LM35_pin );      // Prima lettura "a vuoto" (serve per l'assestamento dell'ADC)

  lcd.begin( 2, 16 );          // Impostazione per l' LCD (2x16)

}

void loop()

{

  sendTempToLCD( temp );    // invia il valore al LCD

float temp = 0.0;      // valore convertito in temperatura (°C)
```

```
  int val = 0;          // valore quantizzato dall'ADC [0..1023]
  int nread = 5;        // numero di letture (da 5 a 8)
  float somma = 0.0;    // somma delle letture
  for (int i=0; i<nread; i++)
  {
    val = analogRead( LM35_pin );           // legge il dato della tensione sul pin 'LM35_pin'
    temp = ( 100.0 *  vref * val ) / 1024.0;   // lo converte in °C
    somma += temp;                          // aggiunge alla somma delle temperature lette
  }
  return ( somma / nread );                 // ne calcola il valore medio
}
```




```
/* Invia la temperatura su un LCD (modo 4bit) */
void sendTempToLCD( float temp )
{
  lcd.clear();            // Pulisce lo schermo
  lcd.setCursor( 0, 0 );    // Va in posizione di home: colonna 1, riga 1
  lcd.print( "Temperatura di: ");
  lcd.setCursor( 0, 1 );    // Va in posizione di home: colonna 1, riga 2
```

```
  lcd.print( temp );        // Stampa solo la parte intera della temp.

  lcd.print( ' ' );         // Stampa uno spazio

lcd.print( (char) 223 );   // Stampa '°'

lcd.print( 'C' );

  delay( 250 );

}
```



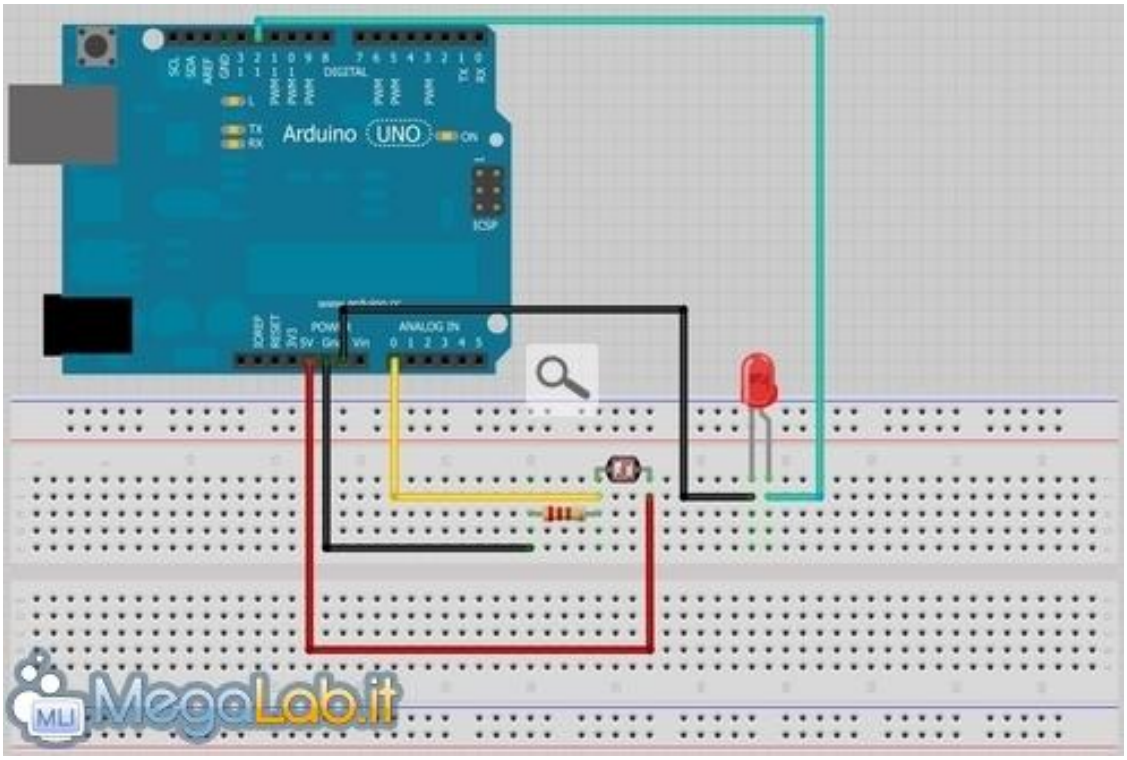

**Fotoresistenza**

```
#define fotoresistenza A0

#define led 12

void setup()  {  pinMode(A0,INPUT);

pinMode(12,OUTPUT);

 Serial.begin(9600); // Inizializzo la comunicazione seriale }

void loop()  {

 int val = analogRead(fotoresistenza); // salvo il valore fotoresistenza dentro alla variabile val

 Serial.println(val, DEC); // Scrivo il valore della fotoresistenza, espresso in numeri decimali
```

*if(val<800) //se il valore letto dalla fotoresistenza (luminosità) è basso, accendo il led*

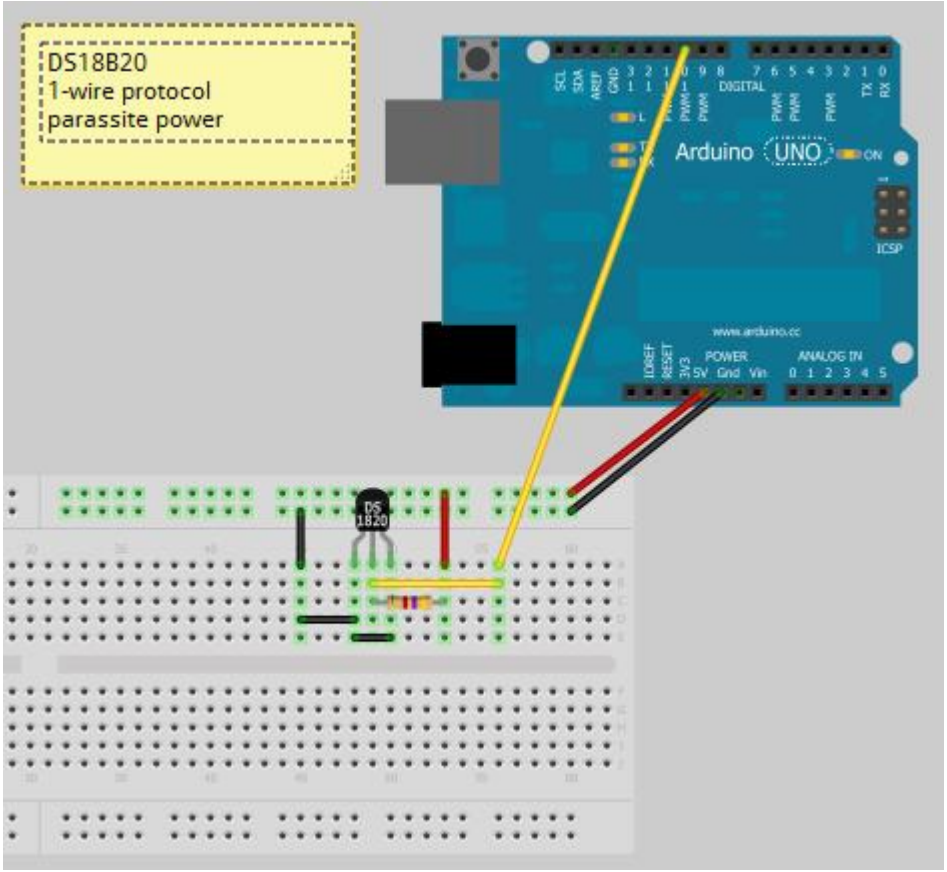
*digitalWrite(led,HIGH);  else*

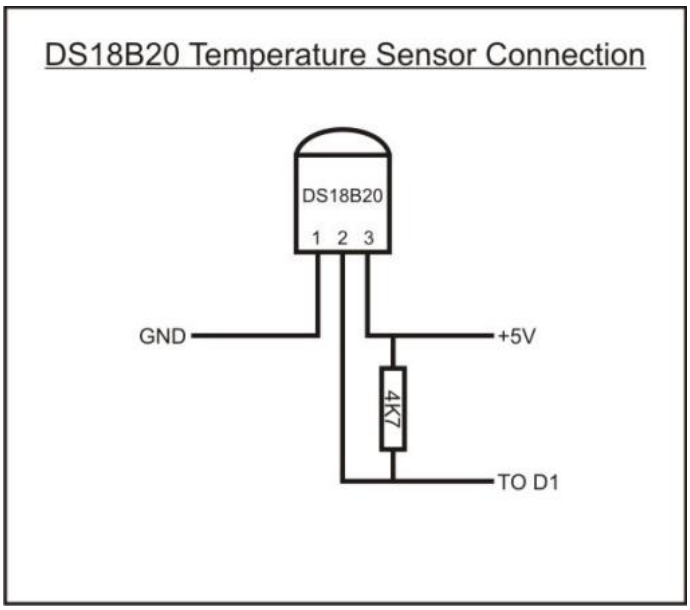
*digitalWrite(led,LOW); //altrimenti lo spengo (o lo lascio spento)  }*




**DS18B20**

È una sonda in grado di rilevare temperature comprese tra -55 °C e 125 °C con accuratezza di 0.5 °C.




La sonda è digitale, è formata da tre pin, quello centrale trasmette i dati al microcontrollore e i due laterali sono per Vcc e GND. Se si utilizza la sola libreria OneWire si può far riferimento al seguente codice anche un po' complesso:

```
#include <OneWire.h>
OneWire  ds(10);  // on pin 10
void setup(void) {
  Serial.begin(9600);
}
```

```
void loop(void) {
  byte i;
  byte present = 0;
  byte data[12];
  byte addr[8];
  if ( !ds.search(addr)) {
    Serial.print("No more addresses.\n");
    ds.reset_search();
    delay(250);
    return;
  }

  Serial.print("R=");
  for( i = 0; i < 8; i++) {
    Serial.print(addr[i], HEX);
    Serial.print(" ");
  }

  if ( OneWire::crc8( addr, 7) != addr[7]) {
      Serial.print("Il CRC non è valido\n");
      return;
  }

  if ( addr[0] != 0x10) {
      Serial.print("Errore\n");
      return;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44,1);
  delay(1000);
  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);

  Serial.print("P=");
  Serial.print(present,HEX);
  Serial.print(" ");
  for ( i = 0; i < 9; i++) {
    data[i] = ds.read();
    Serial.print(data[i], HEX);
    Serial.print(" ");
  }
  Serial.print(" CRC=");
  Serial.print( OneWire::crc8( data, 8), HEX);
  Serial.println();
}
```

---

Se si utilizza anche la libreria del sensore, il codice è più semplice

```
#include <OneWire.h>

#include <DallasTemperature.h>

#define ONE_WIRE_BUS_1 2

const int led_rosso=3;

const int led_verde=4;

const int led_blu=5;

OneWire oneWire_in(ONE_WIRE_BUS_1);

DallasTemperature sensor_inhouse(&oneWire_in);

void setup(void)

{

   Serial.begin(9600);

   Serial.println("Dallas Temperature Control Library Demo - TwoPin_DS18B20");

   sensor_inhouse.begin();

   pinMode(led_rosso,OUTPUT);

   pinMode(led_verde,OUTPUT);

   pinMode(led_blu,OUTPUT);

}

void loop(void)

{float temperatura;

   Serial.print("temperatura interna");

  sensor_inhouse.requestTemperatures();

  temperatura=sensor_inhouse.getTempCByIndex(0);

   Serial.println(" done");

   Serial.print("Inhouse: ");

   Serial.println(sensor_inhouse.getTempCByIndex(0));
```

```
  if(temperatura<15 && temperatura>8){digitalWrite(led_blu,HIGH);

  digitalWrite(led_rosso,LOW);

  digitalWrite(led_verde,LOW);}

  if(temperatura<25 && temperatura>15){digitalWrite(led_blu,LOW);

  digitalWrite(led_rosso,LOW);

  digitalWrite(led_verde,HIGH);}

  if(temperatura<30 && temperatura>25){digitalWrite(led_blu,LOW);

  digitalWrite(led_rosso,HIGH);

  digitalWrite(led_verde,LOW);}

}
```

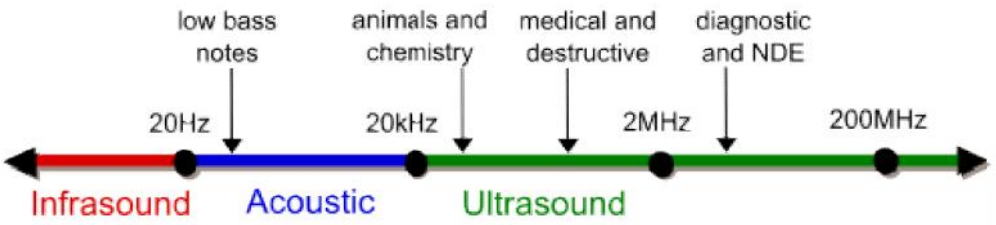
Il codice precedente fa accendere dei led a seconda della temperatura rilevata. Con il sensore DS18B20 si possono effettuare più misure di temperature ponendo diverse sonde su uno stesso bus. Il codice diventa:

```
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 10
OneWireoneWire(ONE_WIRE_BUS);
DallasTemperaturesensors(&oneWire);
DeviceAddresstmp_address;
intnumberOfDevices;



voidsetup(void)
{
  Serial.begin(9600);
  Serial.println("Sto cercando i sensori...");
  sensors.begin();
  delay(6000);
  numberOfDevices = sensors.getDeviceCount();
  Serial.print("Trovati ");
Serial.print(numberOfDevices);
Serial.println(" sensori!");
Serial.println("Inizio la misurazione...");
Serial.println();
}

voidloop(void)
{
  sensors.requestTemperatures(); // Comando per misurare la temp.
```

```
for(inti=0;i<numberOfDevices; i++)
  {
    Serial.print("Sensore ");
    Serial.print(i);
    Serial.print(": ");
    Serial.print(sensors.getTempCByIndex(i));
    Serial.print(" gradi C");
    delay(100);
   Serial.println();
  }
  Serial.println();
  delay(1000);
}
```

**Controllo di un parcheggio**

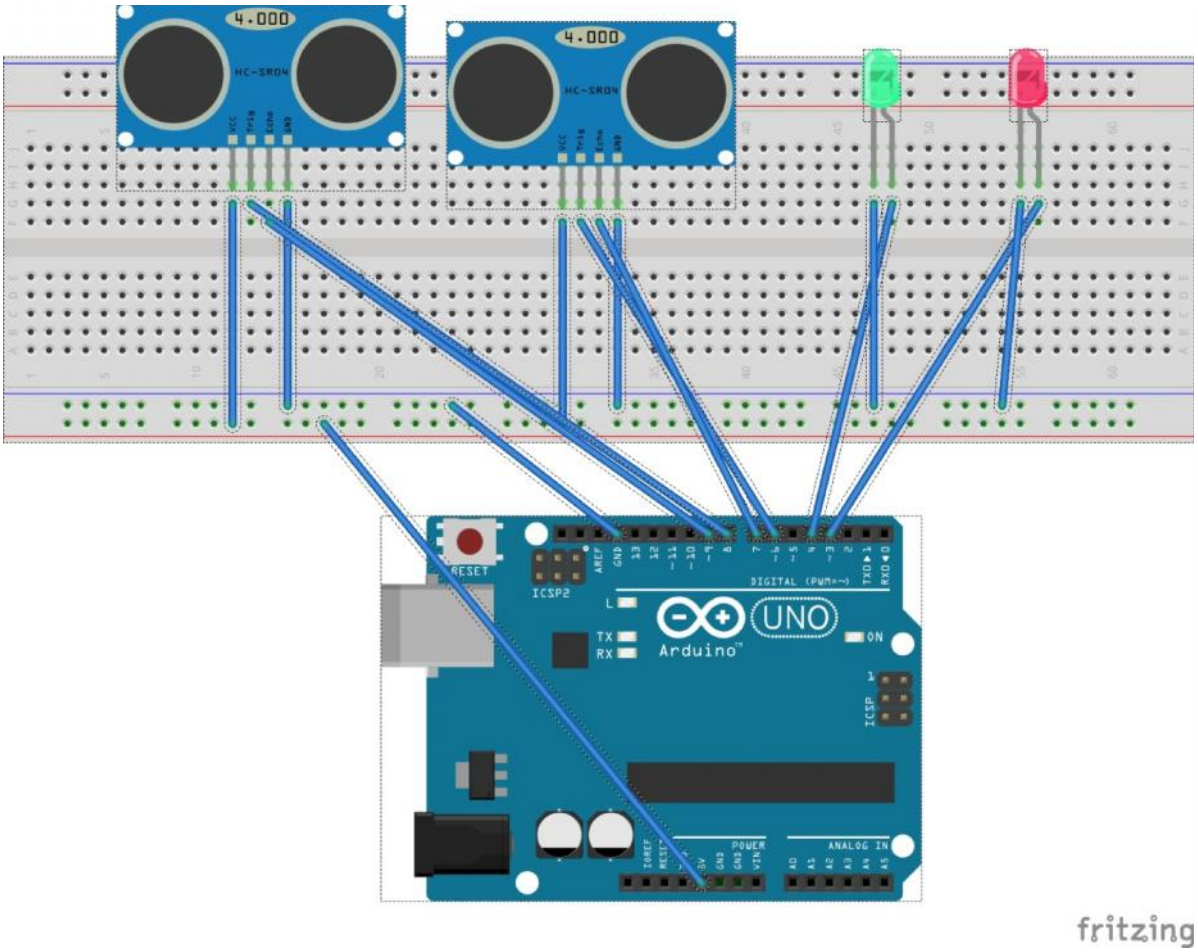
Questo controllo può essere fatto con il sensore di prossimità che si basa sugli ultrasuoni




La velocità del suono nell'aria  a temperatura di 20 C e pressione di 1 atm è pari a 340 m/s

```
const int TRIG = 8;
const int ECHO= 9;
 int rosso=3;
 int verde=4;
const int TRIG1 = 6;
const int ECHO1= 7;
 int n=0, m=0, k=0;
void setup() {

  Serial.begin(9600);
  pinMode(TRIG,OUTPUT);
  pinMode(ECHO,INPUT);
  pinMode(TRIG1,OUTPUT);
  pinMode(ECHO1,INPUT);
  pinMode(verde,OUTPUT);
  pinMode(rosso,OUTPUT);
}

void loop()
{
   long durata, distanza;
   long durata1, distanza1;
digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
delayMicroseconds(10);
  digitalWrite(TRIG, LOW);
  durata = pulseIn(ECHO,HIGH); //misura il tempo in cui il livello logico è alto
distanza = durata / 29.1 / 2 ;
```

```
  if (distanza <= 0){
    Serial.println("Out of range");
  }
  else {
    Serial.print(distanza);
    Serial.println("cm");
    Serial.println();
          if(distanza<50){
               if(distanza>20)m=m+1;
                         }
   digitalWrite(TRIG1, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG1, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG1, LOW);
  durata1 = pulseIn(ECHO1,HIGH);
  distanza1 = durata1 / 29.1 / 2 ;
  if (distanza1 <= 0){
    Serial.println("Out of range");
  }
  else {
    Serial.print(distanza1);
    Serial.println("cm");
Serial.println();
          if(distanza1<50){
               if(distanza1>20)n=n+1;
                         }

        }
        k=m-n;
        if(k>10){digitalWrite(rosso,HIGH);
digitalWrite(verde,LOW);}
        else{digitalWrite(verde,HIGH);
        digitalWrite(rosso,LOW);}
}}
```

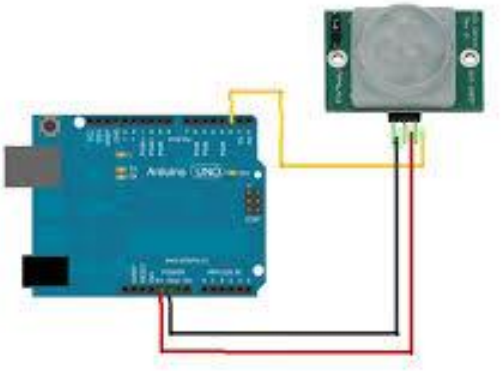


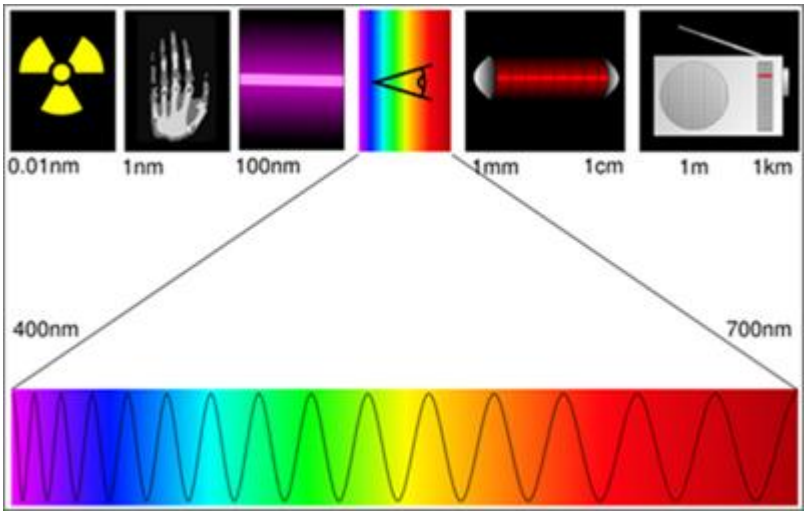

Motorino in CC pilotato da tastiera

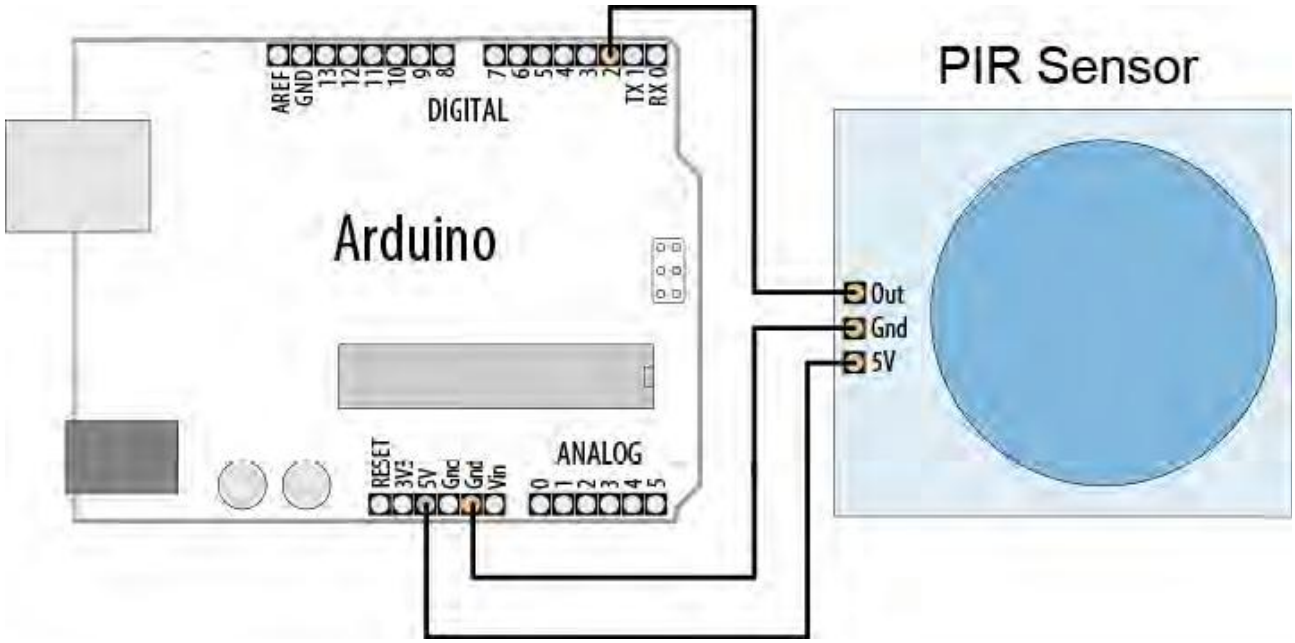
```
const int motorPin = 3;
void setup() {
Serial.begin(9600);
}
void loop() {
if (Serial.available()) {
char ch= Serial.read();
if(isDigit(ch))
{
int speed=map(ch,'0','9',0,255);
analogWrite(motorPin,speed);
Serial.println(speed);
}
else
{
Serial.println("Carattere inaspettato ");
Serial.println(ch);
}
}}
```

**Sensore PIR (Pyroelectric InfraRed sensor)**

Il sensore PIR misura i **raggi infrarossi** emessi nel suo campo visivo e quindi è un sensore capace di stabilire se qualcosa si muove nelle vicinanze. Infatti, tale sensore individua tutto ciò che si trova a temperatura superiore allo 0 assoluto ed emette energia sottoforma di radiazioni luminose.
Davanti al sensore è presente una lente (**lente di Fresnel**) che scompone l'ambiente in fasci in modo da creare le zone necessarie a percepire la differenza di temperatura generata da un corpo caldo che le attraversa.
Schema per **collegare** il sensore PIR alla scheda arduino:

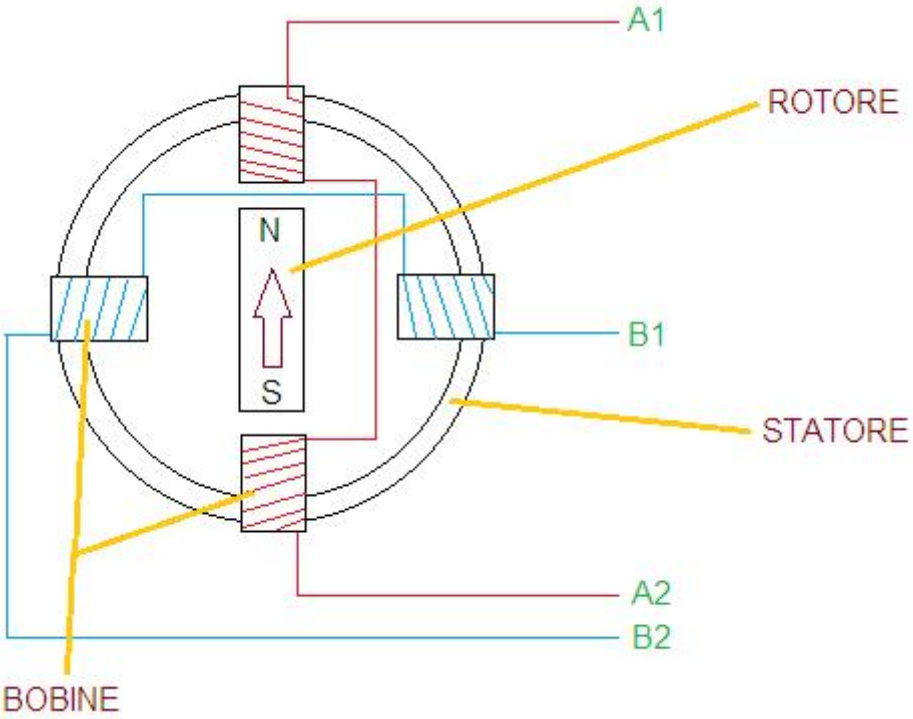




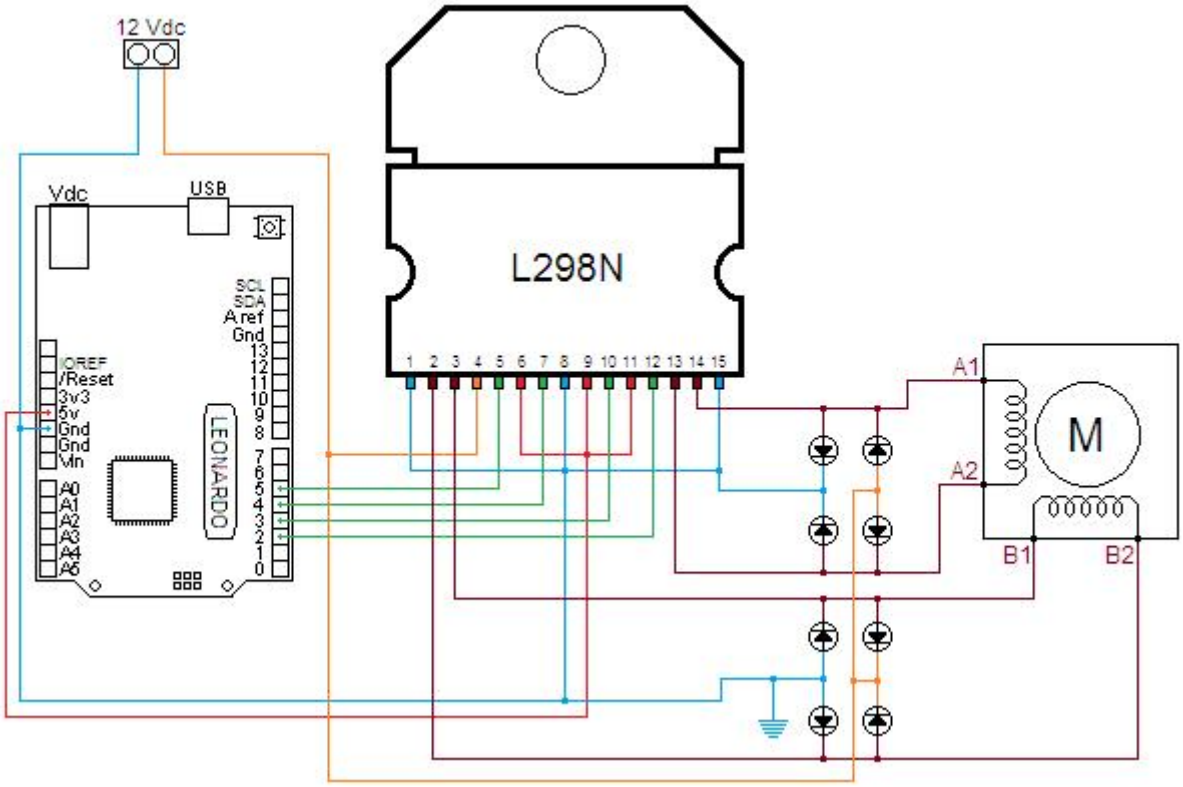



```
const int ledPin = 12; //pin per il LED
const int inputPin = 2; //pin per il sensore
void setup() {
pinMode(ledPin, OUTPUT);
pinMode(inputPin, INPUT);
}
void loop(){
int val = digitalRead(inputPin);
if (val == HIGH)
{
digitalWrite(ledPin, HIGH); // se c'è movimento il led si accende
delay(500);
digitalWrite(ledPin, LOW); }
}
```

Motore Passo passo




| Fasi di Alimentazione | | | | |
|---|---|---|---|---|
| | A1 | A2 | B1 | B2 |
| Fase 1 | + | - | - | - |
| Fase 2 | - | - | + | - |
| Fase 3 | - | + | - | - |
| Fase 4 | - | - | - | |




```
Void setup()
{
  //i pin 2-3-4-5 sono
  //configurati come uscite
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);

  //forzo le uscite a livello logico basso
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
}

voidloop()
{
  //FASE 1
  //Alimento solo la prima bobina
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  delay(10);

  //FASE 2
```

```
  //Alimento solo la seconda bobina
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
  delay(10);

  //FASE 3
  //Alimento solo la terza bobina
  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  delay(10);

  //fase 4
  //Alimento solo la quarta bobina
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
  delay(10);

}
```

Oppure

```
const int fase[4]= {0b0001,0b0010,0b0100,0b1000};
const int pin[4]= {2,3,4,5};
void setup() {
pinMode(2,OUTPUT);
pinMode(3,OUTPUT);
pinMode(4,OUTPUT);
pinMode(5,OUTPUT);
}
void loop() {
giraOr(100);
delay(500);
giraAn(100);
delay(2000);
}

void giraOr(int numPassi) { // numPassix4
for(int passi=0;passi< numPassi;passi++) {
for (int j=0;j<4;j++) { // per le 4 fasi
for(int k=0;k<4;k++){ // per i 4 bit
```

```
digitalWrite(pin[k],bitRead(fase[j],k));
}
delay(3); // ritardo di passo
}
}
for(int k=0;k<4;k++) // togli alimentazione
digitalWrite(pin[k],LOW);
}
void giraAn(int numPassi) { // numPassix4
for(int passi=0;passi< numPassi;passi++) {
for (int j=3;j>=0;j--) {
for(int k=0;k<4;k++){
digitalWrite(pin[k],bitRead(fase[j],k));
}
delay(3);
}
}
for(int k=0;k<4;k++) // togli alimentazione
digitalWrite(pin[k],LOW);}
```