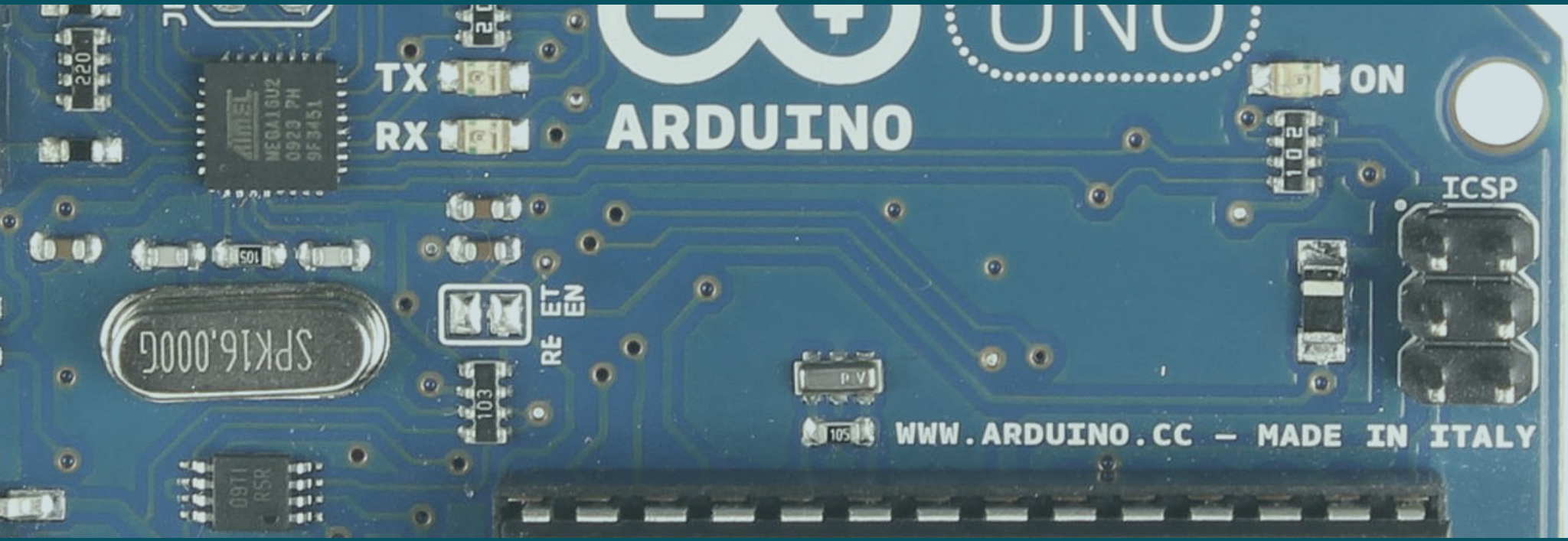
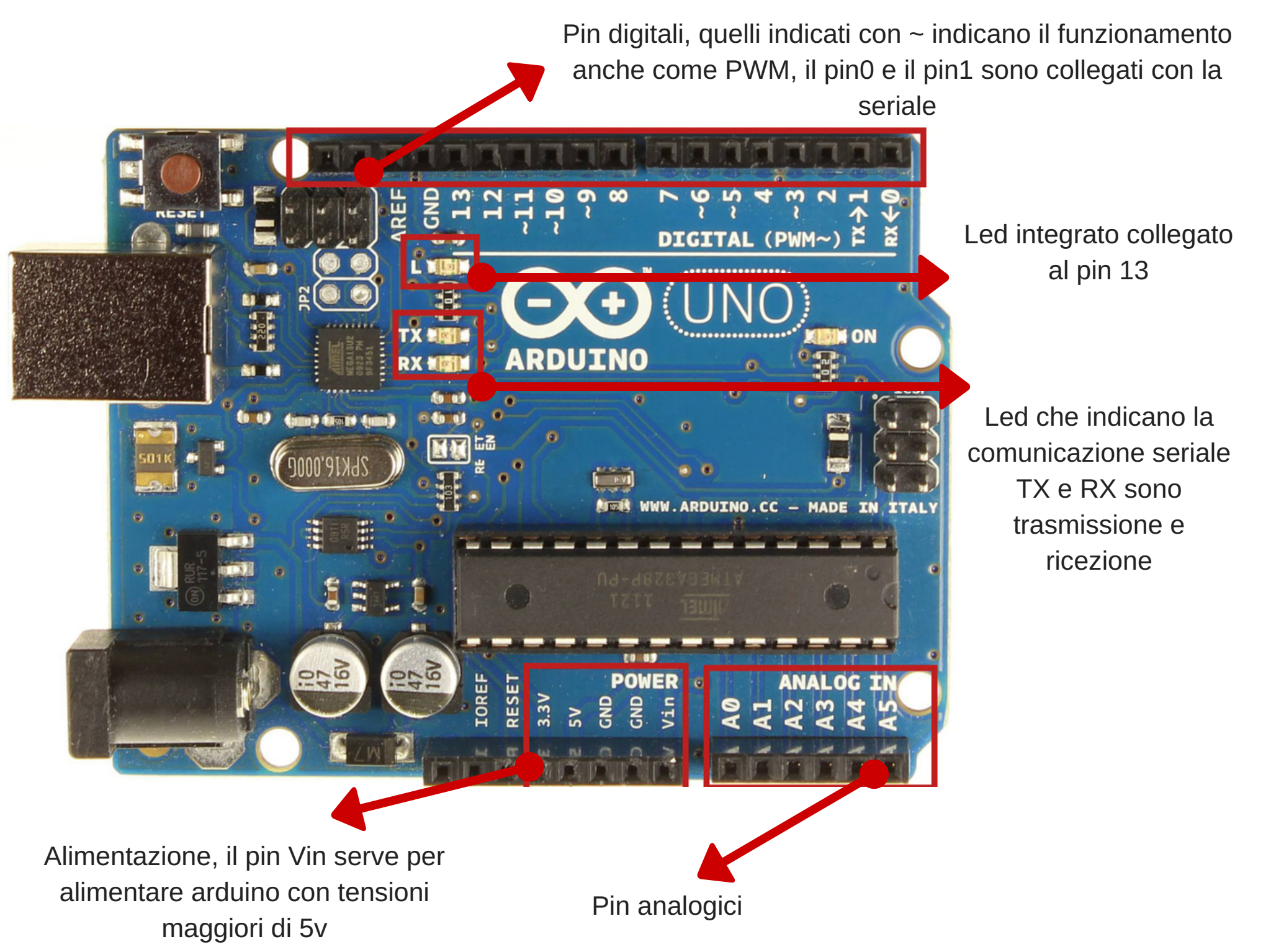
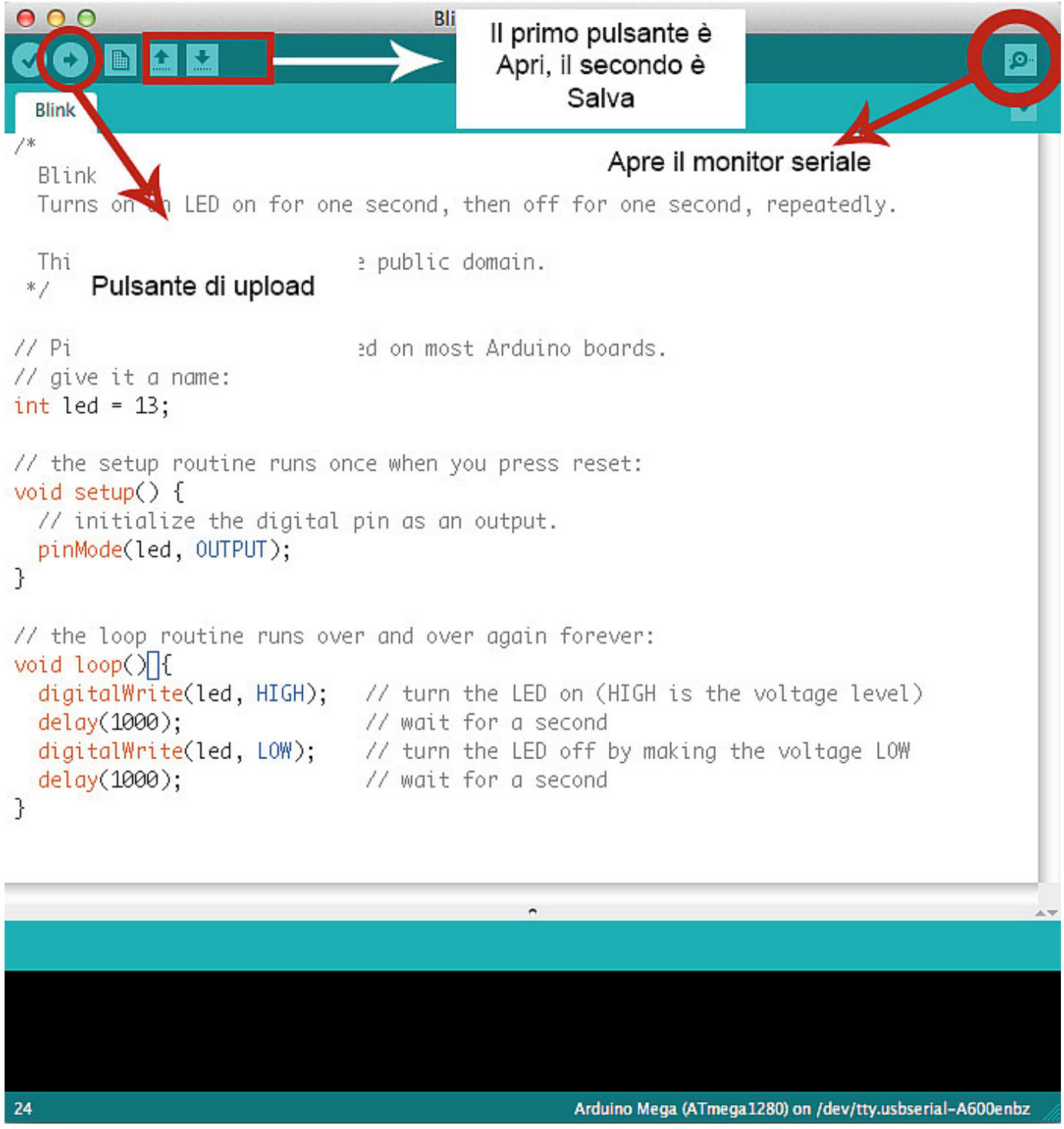
ARDUROBOT

BASI DI PROTOTIPAZIONE

INTRODUZIONE E PRIMI ESERCIZI




Pin digitali, quelli indicati con ~ indicano il funzionamento anche come PWM, il pin0 e il pin1 sono collegati con la seriale
Led integrato collegato al pin 13
Led che indicano la comunicazione seriale TX e RX sono trasmissione e ricezione
Alimentazione, il pin Vin serve per alimentare arduino con tensioni maggiori di 5v
Pin analogici
DIGITAL (PWM~)
POWER
ANALOG IN
ARDUINO
UNO
WWW.ARDUINO.CC – MADE IN ITALY


Il primo pulsante è Apri, il secondo è Salva
Apre il monitor seriale
Pulsante di upload
Blink
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.
Thi e public domain.
*/
// Pi ed on most Arduino boards.
// give it a name:
int led = 13;
// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop(){
digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
delay(1000); // wait for a second
digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
delay(1000); // wait for a second
}
24
Arduino Mega (ATmega1280) on /dev/tty.usbserial-A600enbz

```
  // the setup routine runs once when you press reset:
  void setup() {
    // initialize the digital pin as an output.
    pinMode(led, OUTPUT);
  }

  // the loop routine runs over and over again forever:
  void loop() {
    digitalWrite(led, HIGH);   // turn the LED on (HIGH is the volt
    delay(1000);               // wait for a second
    digitalWrite(led, LOW);    // turn the LED off by making the vol
    delay(1000);               // wait for a second
  }
```

Funzioni principali: **setup() e loop()**

setup(): chiamata una sola volta, all'inizio del programma. Usata per definire gli elementi da usare e dove sono collegati i vari sensori o attuatori

loop(): è un ciclo, all'interno del quale implementare il vero e proprio programma. Quello che viene scritto qui viene ripetuto dalla scheda finchè non viene spenta

Blink

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

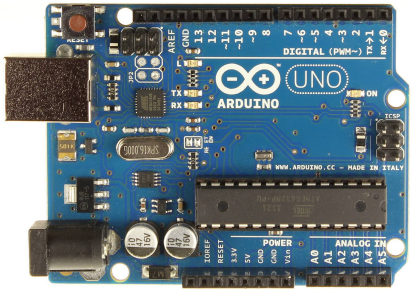
Ohm sweet Ohm

La corrente è la quntità di materia che viene trasportata in un intervallo di tempo.
La tensione elettrica è ciò che spinge le cariche lungo tutto il circuito.
Senza tensione le cariche non scorrono.
La resistenza è la capacita di un materiale di rallentare lo scorrere della corrente

Immaginiamo l'elettricita come una cascata.
La tensione è l'altezza della cascata.
L'intensita è la quantita di acqua che scorre.
La resistenza sono le rocce che rallentano il flusso d'acqua.

$$Ohm = \frac{Volt}{Ampere}$$

Utilizza 0,02 Ampere con circa 1,6 Volt



Eroga 5V

Resistenza = (5 - 1,6) / 0,02

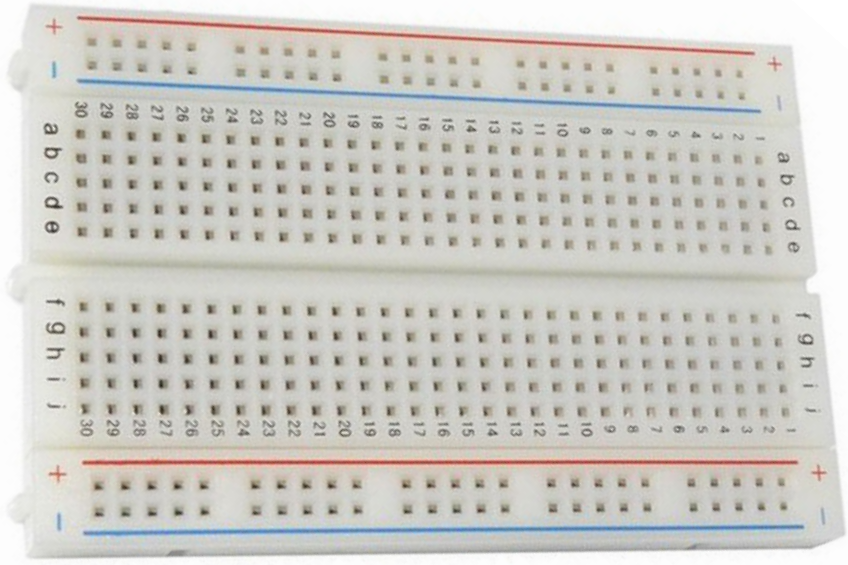
Resistenza = (3,4) / 0,02

Resistenza = 170 ohm
Si approssima a 180 ohm
Vanno bene anche 220 ohm
Queste approssimazioni sono necessarie per utilizzare valori di resistenze in commercio

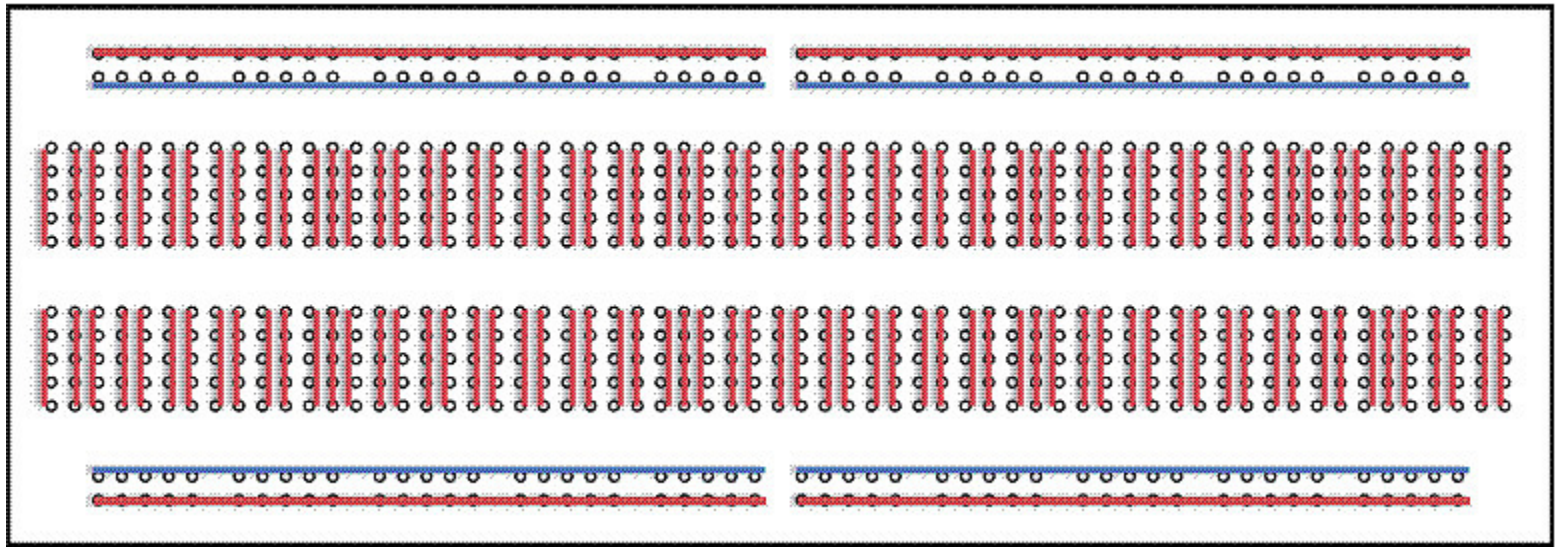
Quale resistenza applicare tra alimentazione e attuatore ?

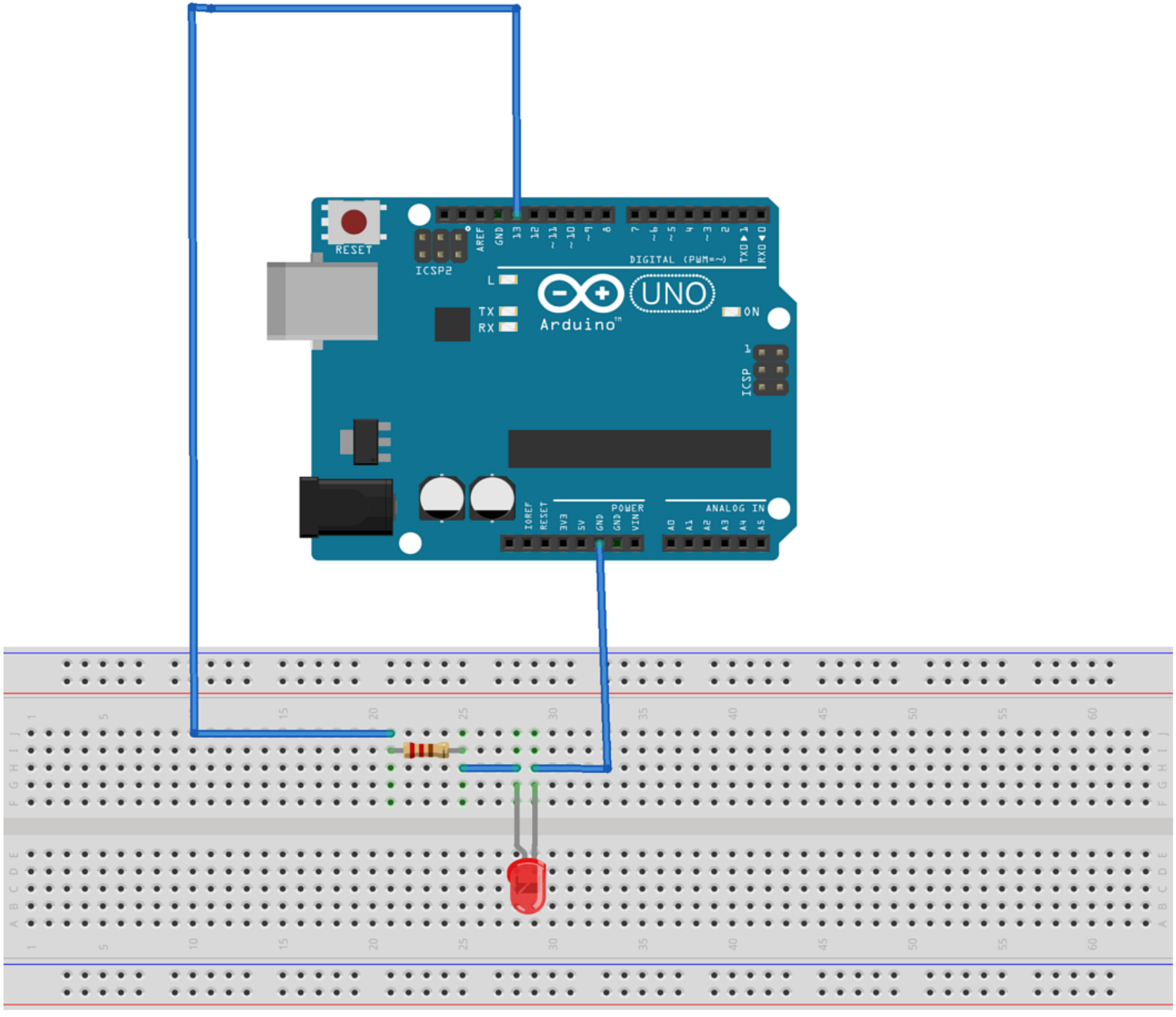
Resistenza = Tensione / Corrente

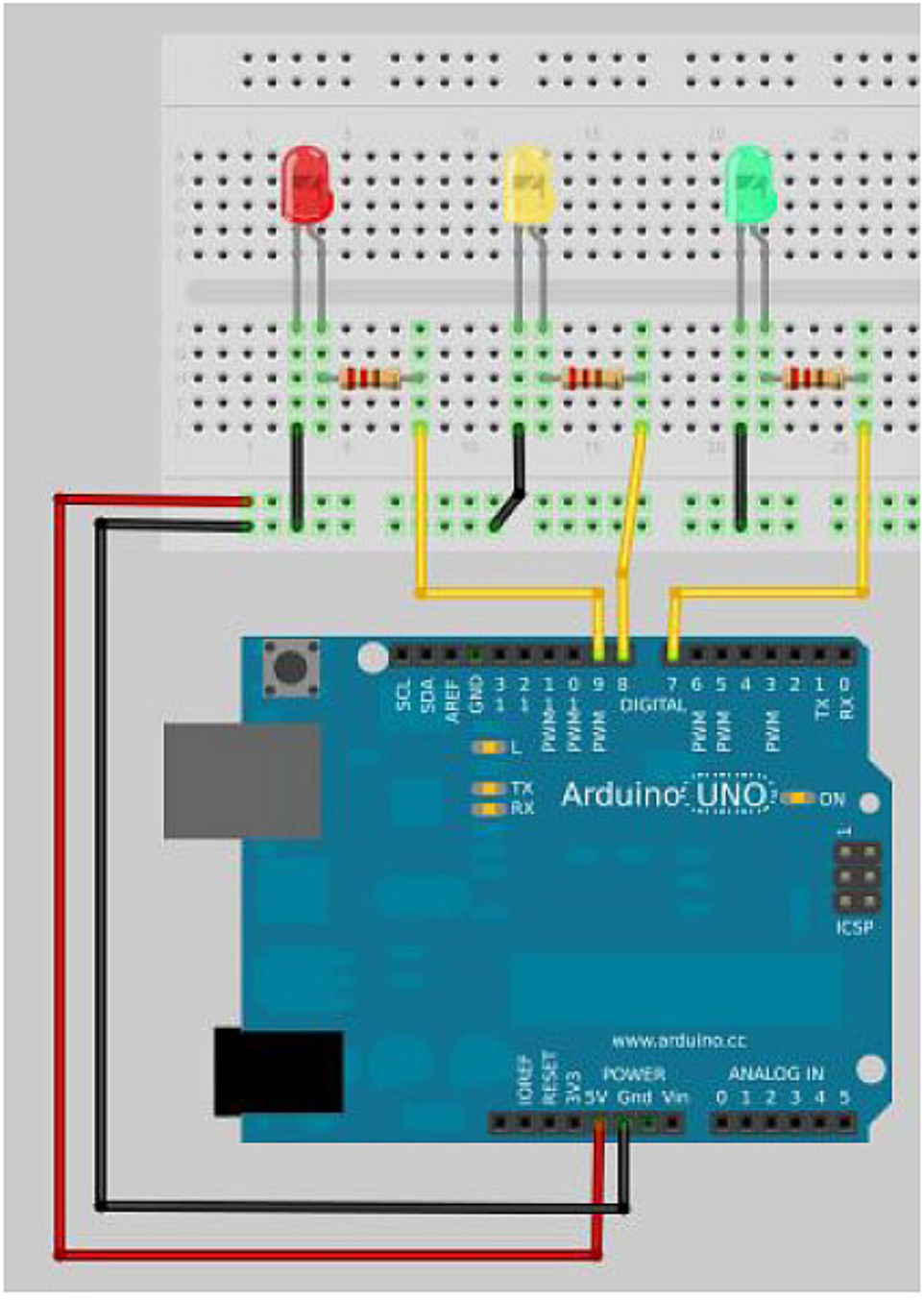
Resistenza =
(Tensione alimentatore - Tensione Led)
/
Corrente utile per far funzionare il Led



Le line esterne sono connesse in ORIZZONTALE
Le line centrali sono connesse in VERTICALI




RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
Arduino™
UNO
ON
ICSP
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5
fritzing



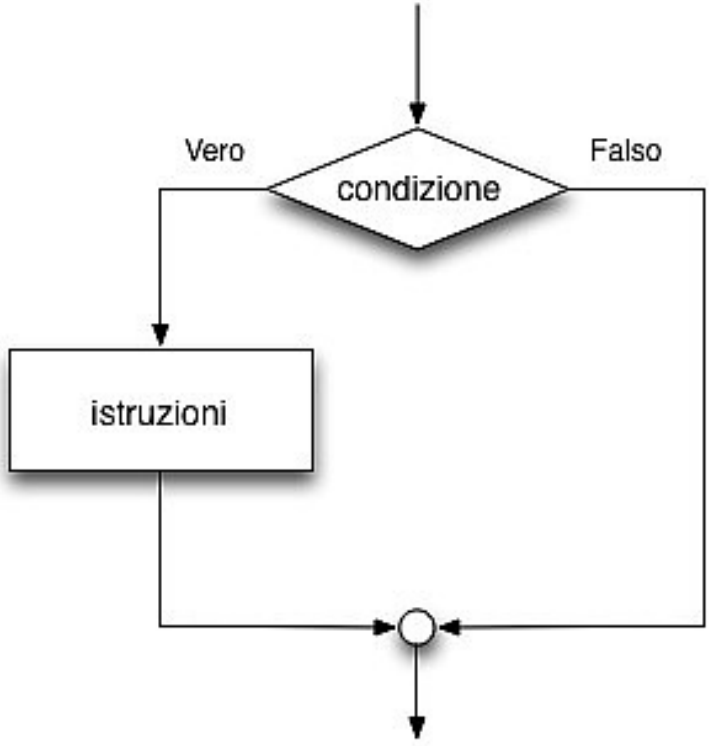
Il led rosso collegato al pin 9 attraverso la resistenza
Il led giallo collegato al pin 8 attraverso la resistenza
Il led verde collegato al pin 7 attraverso la resistenza

La 5V collegata lla linea rossa della breadboard
la GND collegata alla linea blu della breadboard

sketch_jan13a §

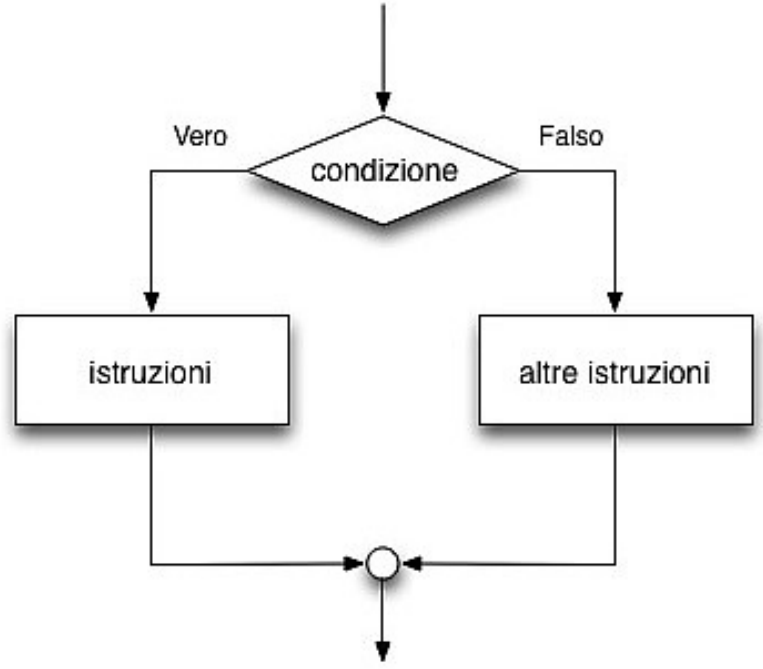
```
int verde = 7; //definisco a quali pin connettere i LED
int giallo = 8;
int rosso = 9;

void setup()
{
pinMode(verde, OUTPUT); // i LED reagiscono ad un impulso quindi sono un OUTPUT
pinMode(giallo, OUTPUT);
pinMode(rosso, OUTPUT);
}

void loop()
{
digitalWrite(verde, HIGH); // Accendo il verde per 5 secondi
delay(5000);
digitalWrite(verde, LOW); //Spengo il verde e accendo il giallo per 2 secondi
digitalWrite(giallo, HIGH);
delay(2000);
digitalWrite(giallo, LOW); //Sepngo il giallo e accendo il rosso per 2 secondi
digitalWrite(rosso, HIGH);
delay(5000);
digitalWrite(giallo, HIGH); //Giallo e Rosso insieme per 2 secondi
delay(2000);
digitalWrite(rosso, LOW); //Spengo sia il giallo che il rosso
digitalWrite(giallo, LOW);
}
```

```
if (variabile  valore )
        {
    fa qualcosa;
        }
```




```
if (valorePin  > 0)
        {
digitalWrite (ledPin, HIGH);
    delay(1000);
        }
```

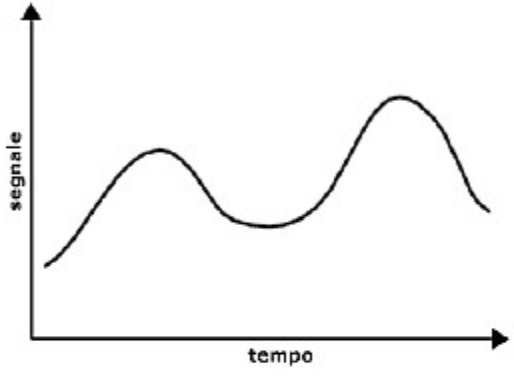
```
if (variabile  valore )
{
fa qualcosa;
}
else
{ fa un'altra cosa }
```



```
if (valorePin  > 100)
{
digitalWrite (ledPin, HIGH);
delay(1000);
}
else
{
digitalWrite (ledPin, LOW);
}
```

Un segnale analogico può assumere qualsiasi valore (all'interno di un range noto). Con notevole semplificazione, si può pensare che siano "analogiche" tutte le grandezze fisiche misurabili nell'ambiente.

Un segnale digitale può assumere due soli stati (High/Low, 1/0), corrispondenti a due livelli di tensione convenzionali (ad esempio, 5-0V). Con simile semplificazione, si può pensare che siano "digitali" tutte le informazione scambiate tra componenti logici (microprocessori, memorie, interfaccie di rete, display...).

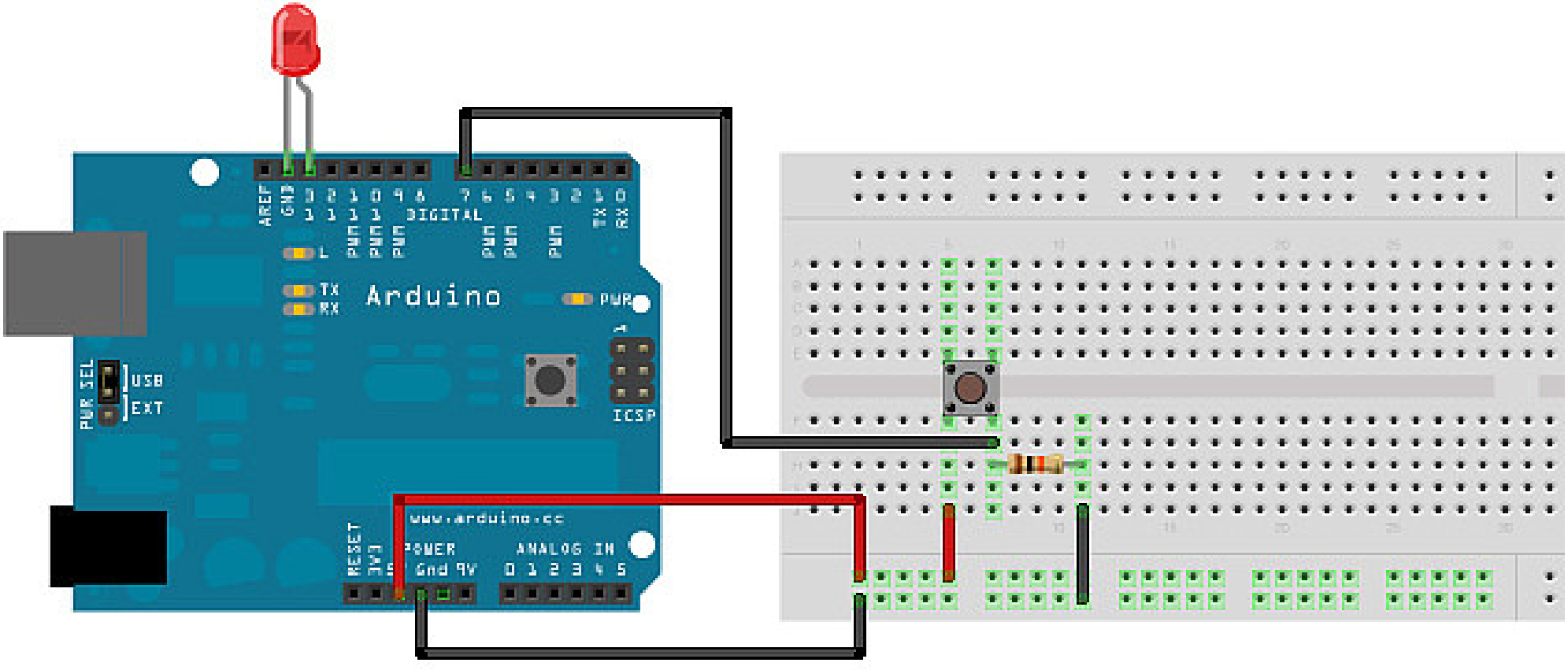


Per essere compreso da Arduino, il segnale analogico deve essere campionato, ovvero convertito in una sequenza di bit che ne esprime l'ampiezza.

Il segnale digitale è immediatamente "leggibile" non appena ne è stato discriminato il livello (Hight/Low).

Gestione dei segnali digitali

digitalRead()
digitalWrite()



led_pulsante1

```
// Accendi il led appena si preme il pulsante

#define LED 13                  // LED collegato al pin digitale 13
#define BUTTON 7                // Pin dove è collegato il pulsante
int  val = 0;                   // valore per conservare lo stato del pulsante

void setup() {
  pinMode(LED, OUTPUT);         // il led è un output
  pinMode(BUTTON, INPUT);       // il bottone è un input
}

void loop() {
  val = digitalRead(BUTTON);  // Leggo il valore del bottone e lo conservo nella variabile

  // Se il pulsante è premuto il segnale è alto HIGH e allora accendo il led
  if (val == HIGH) {
    digitalWrite(LED, HIGH);  //accendo il led
  }
  else {                        // altrimenti
    digitalWrite(LED, LOW);   //spengo il led
  }
}
```

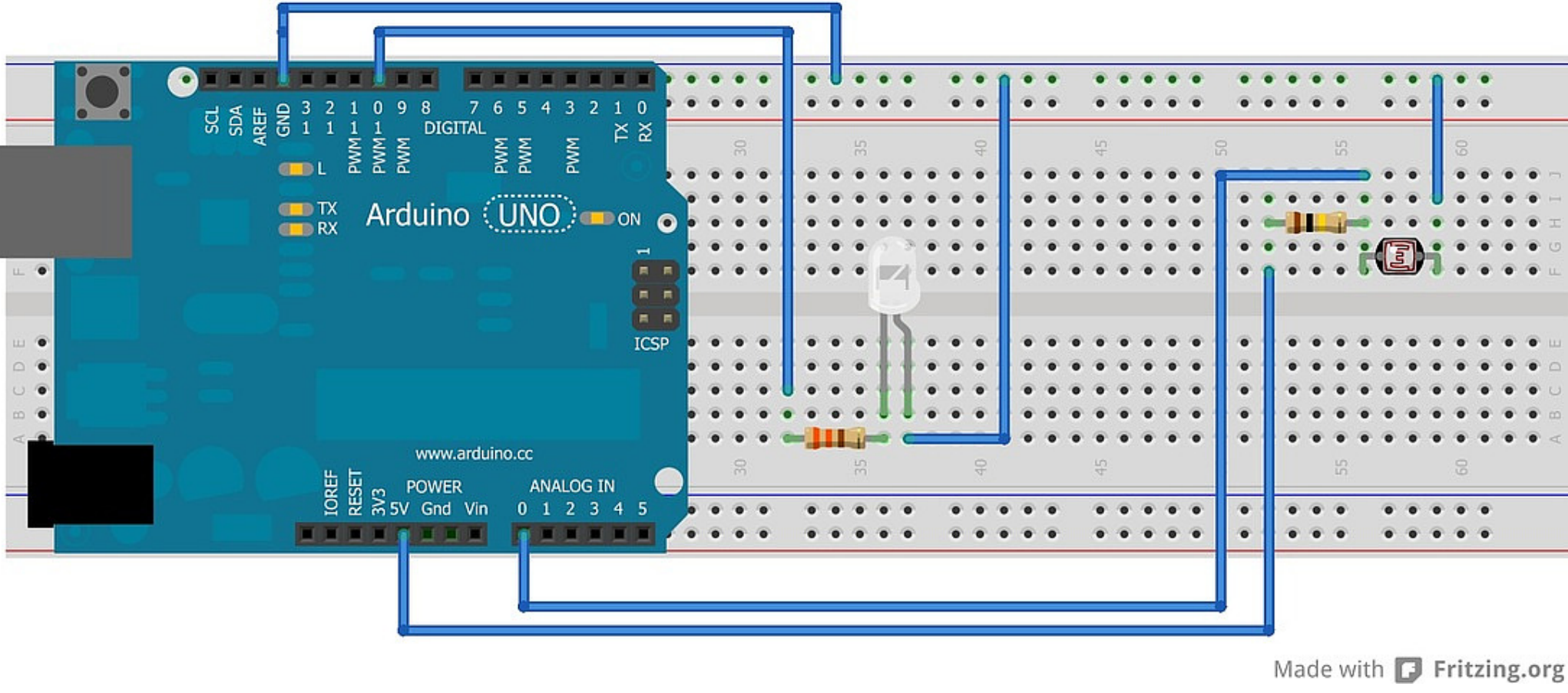
Migliorabile?

Che problema abbiamo?

```
// Soluzione alla permanenza del pulsante. Bisogna ricordare il vecchio stato

#define LED 13                // LED collegato al pin digitale 13
#define BUTTON 7              // Pin dove è collegato il pulsante
int  val = 0;                 // valore per conservare lo stato del pulsante
int vecchio_val = 0;          // Lo useremo per ricordare il vecchio stato del pulsante
int stato = 0;                // ricorda lo stato in cui si trova il led, stato = 0 led spento, stato = 1 led acceso

void setup() {
  pinMode(LED, OUTPUT);       // il led è un output
  pinMode(BUTTON, INPUT);     // il bottone è un input
}

void loop() {
  val = digitalRead(BUTTON);  // Leggo il valore del bottone e lo conservo nella variabile

  // Se il pulsante è acceso e se il vecchio stato del pulsante è spento
  if ((val == HIGH) && (vecchio_val == LOW)){
    stato = 1 - stato;        // lo stato è uguale a 1 a cui sottraggo anche il valore di stato che di base è 0
    delay(15);                // aspetto 15 millisecondi per non sovraccaricare il controllore
  }

  vecchio_val = val;            // ricordiamo il vecchio valore del bottone

    if (stato == 1) {           // se lo stato è uguale a 1 ( significa che il pulsante prima era spento e ora è acceso 1-0)
    digitalWrite(LED, HIGH);    // accende il led
  }
  else {                        // se lo stato è diverso da 1 quindi 0
    digitalWrite(LED, LOW);     //allora spengo il led
  }
}
```

Gestione dei segnali analogici

analogRead()
analogWrite()



```
int led = 10;  //Il led è collegato al pin 10
int fotoresistenza = A0;  //La fotoresistenza è collegata al pin analogico 0 quello indicato come A0
int valore=0; // imposto a 0 il valore iniziale della fotoresistenza


void setup() {
 pinMode (led, OUTPUT); //Il pin a cui è connesso il led è impostato come uscita
 pinMode (fotoresistenza, INPUT);
 Serial.begin(9600); // Inizializzo la comunicazione seriale
  }

void loop () {

 valore = analogRead (fotoresistenza);   // Leggo il valore sul pin 2 che è quello della fotoresistenza
                                         // e lo immagazzino nella variabile "valore"
 Serial.print("La luce e' uguale a:");   // Scrivo una riga che mi mostra esattamente il testo fra le virgolette
 Serial.println (valore);                // Scrivo una riga a capo in cui segno il valore rilevato dalla fotoresistenza
 delay(5);


  if (valore > 300) {   // se il valore è sopra 300 allora
    digitalWrite(led, HIGH); // accendo il led
   }
   else {  // altrimenti
    digitalWrite(led, LOW); // spengo il led
    }
 delay(5);
}
```

```
paolo@discienza.org

*/

int led = 10;  //Il led è collegato al pin 10
int fotoresistenza = 0;  //La fotoresistenza è collegata al pin analogico 0 quello indicato come A0
int valore=0; // imposto a 0 il valore iniziale della fotoresistenza


void setup() {
 pinMode (led, OUTPUT);  //Il pin a cui è connesso il led è impostato come uscita
 Serial.begin(9600);
  }

void loop () {
 valore = analogRead (fotoresistenza);   // Leggo il valore sul pin 2 che è quello della fotoresistenza e lo immagazzir
 Serial.print("La luce è uguale a:");   // Scrivo una riga che mi mostra esattamente il testo fra le virgolette
 Serial.println (valore);                // Scrivo una riga a capo in cui segno il valore rilevato dalla fotoresistenza

 int luminosita = map(valore, 250, 700, 0, 255);

analogWrite(led, luminosita);

delay(100);
}
```

Arduino e i servomotori

I servomotori sono dei particolari motori che è possibile controllare in modo che rispettino la rotazione in una posizione specifica.
La maggior parte dei servmotori ruota di 180° ma esistono anche da 45°, 90° e 360°.

Per comandarli attraverso Arduino è necessaria una libreria chiamata Servo.h

La connessione con la scheda è generalmente semplice, un cavo di alimentazione alla 5V, il cavo di massa alla GND e il cavo dati al pin

Lato software di usano questi comandi:

Servo myservo; --> per indicare l'esistenza del servomotore
myservo.attach(10) --> per indicare che il servomotore è connesso al pin 10
myservo.write (valore) --> per dare il comando al servomotore

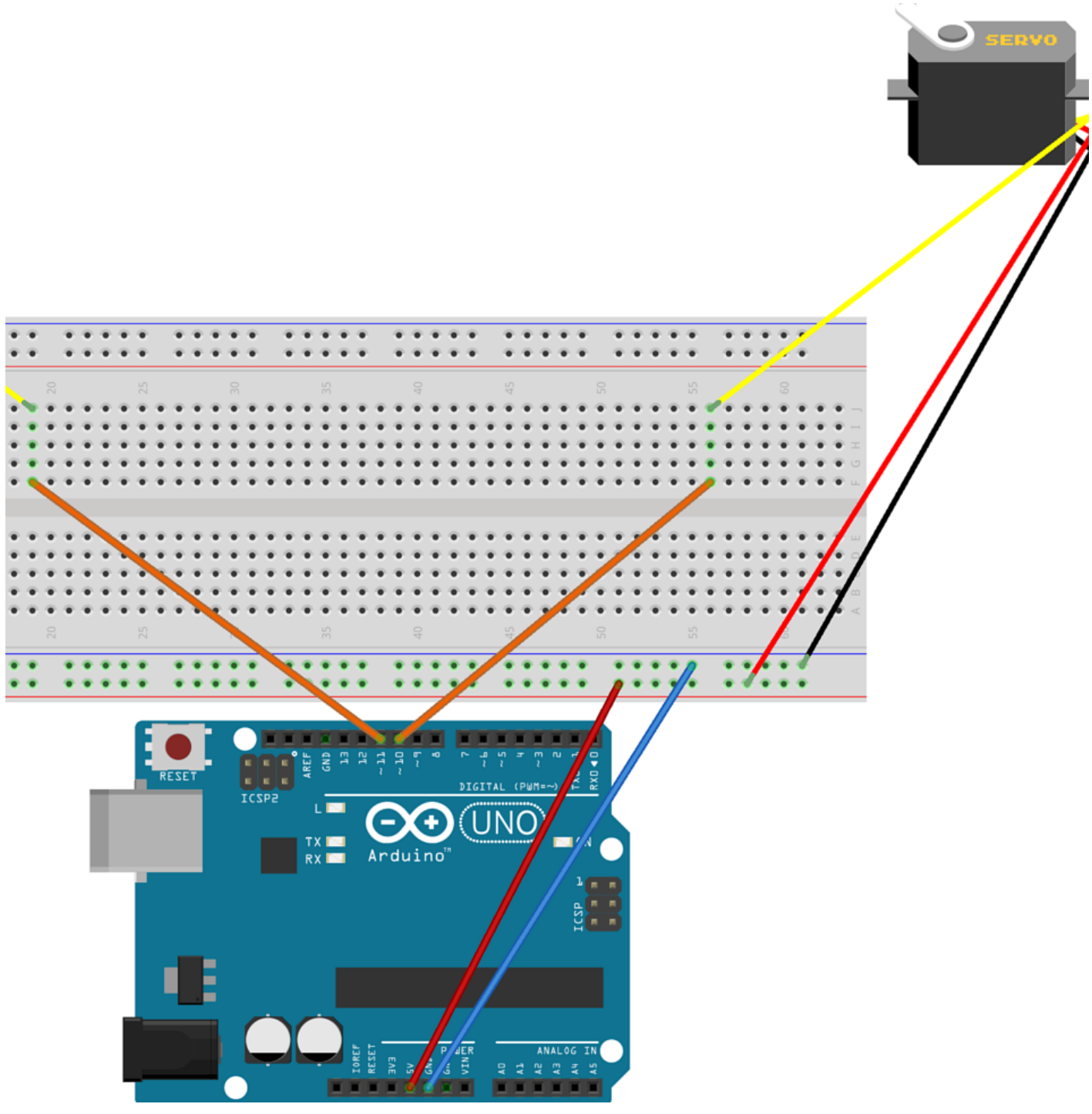
Qui viene usato un servomotore 360° questo tipo di servomotore da 0 a 90° gira in un verso mentre da 90° a 180° gira nel verso opposto. La velocità di rotazione cresce aumentando il valore numerico nei rispettivi versi.

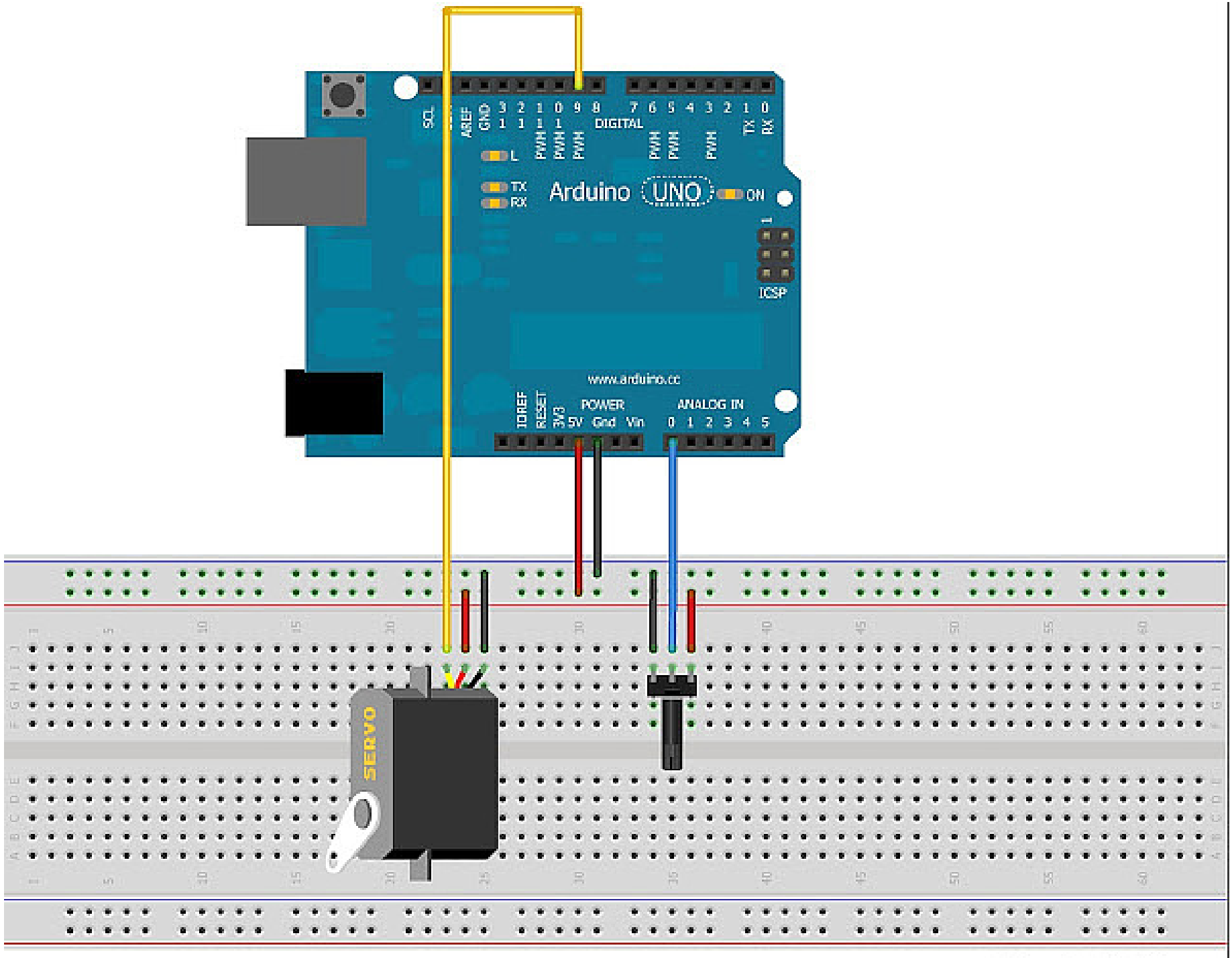
Esempio: da 0 a 50 va veloce e da 50 a 80 va piano in un verso

da 180 a 120 va veloce e da 120 a 100 va piano nel verso opposto

a 90 si ferma

```
#include <Servo.h> // carico la libreria Servo che contiene le istruzioni per Arduino necessarie a gestire i servomotori

// Creo i due motori con la funzione Servo
Servo motoreSinistro;
Servo motoreDestro;

void setup()
{
  Serial.begin(9600);     // Inizializza la comunicazione con la seriale a 9600 bit per secondo
  motoreSinistro.attach(11); // attacco il motore sinistro al pin 11
  motoreDestro.attach(10);  // attacco il motore destro al pin 10
}


void loop()
{
  motoreDestro.write(180);  // il motore destro gira in una direzione
  motoreSinistro.write(0);  // il motore sinistro gira in quella opposta ma poichè sono messi di fronte girano nella stessa direzione
  Serial.print("vado");    // faccio scrivere la parola vado così dal seriale capisco che sta andando avanti

}
```


SERVO
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
ON
UNO
Arduino™
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN



Qui viene usato un servomotore 180° questo tipo di servomotore gira da 0 a 180° mantenendo l'angolo assegnato.

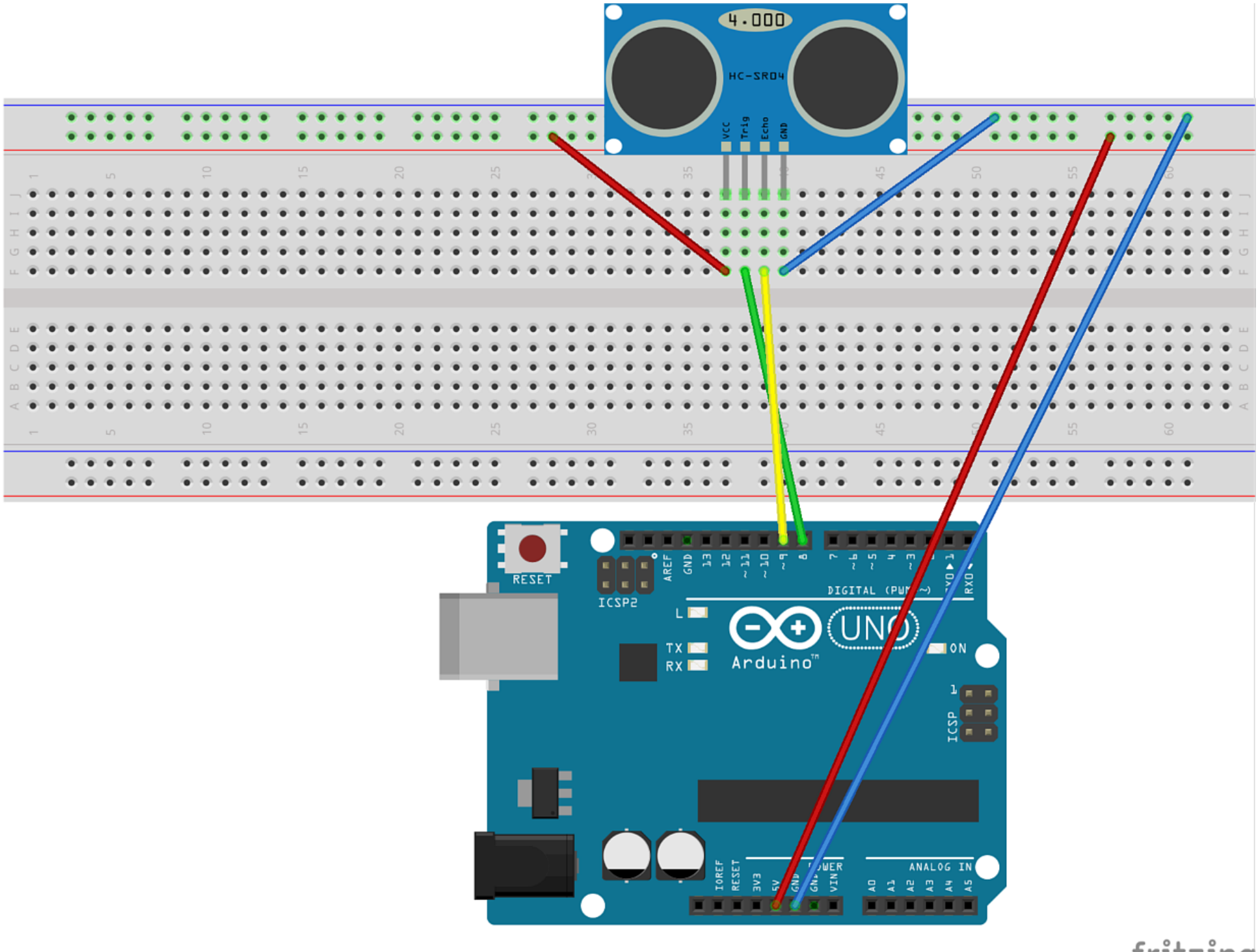
servo_potenziometro §

```
#include <Servo.h> // include la Libreria Servo.h

Servo servoMotor;  // Crea l'oggetto di tipo Servo, servoMotor sarà l'oggetto su cui opererai.
int valore;        // Inizializza una variabile di tipo intero "valore" il cui valore sarà la posizione da impartire al servo.

void setup() {
   servoMotor.attach(10);  // Lega l'oggetto servoMotor al pin a cui abbiamo collegato il nostro servo, in questo caso il pin 8.
}

void loop()
 {
   valore = analogRead(A0);  // Legge il valore analogico del potenziometro sul pin A0
   valore = map(valore, 0, 1023, 0, 180);  // "Mappa" i valori di una lettura analogica (che vanno quindi da 0 a 1023)
                                           // a valori che vanno da 0 a 180.

   servoMotor.write(valore);  // con il metodo write() passi all'oggetto servoMotor la posizione che deve raggiungere.
   delay(15);
  }
```

Misuriamo la distanza

Utilizzeremo un sensore ad ultrasuoni.
Ci sono 4 pin
1-Vcc -> è l'alimentazione a 5V
2-Trig -> è il pin che emette il segnale
3-Echo -> è il pin che riceve il segnale
4-GND -> è il pin che chiude il circuito

Nel nostro programma controlleremo quanto tempo impiega il segnale emesso dal sensore a tornare indietro sul ricevitore.
Il segnale torna indietro quando rimbalza su un ostacolo.
Dal tempo trascorso possiamo rilevare la distanza conoscendo la velocità del suono 343,8 m/s a 20°


4.000
HC-SR04
VCC
Trig
Echo
GND
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
UNO
Arduino™
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
fritzing

```
// specifica i pin di trig e echo usati per il sensore ad ultrasuoni
const int TrigPin = 8; // allaccio sul pin 8 della Arduino il pin che attiva lo sparo del sensore ultrasuoni
const int EchoPin = 9; // allaccio al pin 9 della Arduino il pin che legge il rilevatore del segnale ultrasuoni inviato
unsigned int distanza; // distanza rilevata dal sensore unsigned significa che non avrà mai valori negativi.
                       //Il sensore a volte ci resitutisce valori negativi quando non ha nulla davanti
int durata;            // durata del segnale di ritorno rilevato dal sensore a ultrasuoni


void setup()
{
  Serial.begin(9600);     // Inizializza la comunicazione con la seriale a 9600 bit per secondo
  pinMode(TrigPin, OUTPUT);  // Il pin allacciato al Trig del sensore spara il segnale quindi è un OUTPUT
  pinMode(EchoPin, INPUT);   // Il pin allacciato al pin Echo del sensore riceve il segnale di ritorno quindi è un INPUT

}


void loop() // il loop è il ciclo di azioni che faranno le componenti
{

  {
  digitalWrite(TrigPin, LOW);   // Sparo del segnale spento
  delayMicroseconds(2);          // aspetto 2 microsecondi
  digitalWrite(TrigPin, HIGH);  // accendo lo sparo del segnale, ora il sensore sta inviando ultrasuoni
  delayMicroseconds(10);         // aspetto 10 microsecondi
  digitalWrite(TrigPin, LOW);  // spengo l'invio di ultrasuoni

  durata = pulseIn(EchoPin, HIGH);  // accendo il rilevatore del sensore e chiedo quando tempo ha impiegato il segnale a tornare
  distanza = (durata/2)/29;         // il valore inviato è la distanza che divido per 2 (andata e ritorno) e poi per 29 che è
                                   // frutto della conversione della velocità del suono in cm/s diviso per due
  Serial.print("distanza: ");    // faccio scrivere la parola distanza
  Serial.print(distanza);    // mi faccio indicare la distanza
  Serial.print("cm: ");       // faccio scrivere la parola cm
  Serial.println();          // vado una riga sotto
  delay(500); // aspetto 500 millisecondi prima di riniziare
  }

}
```