UNIVERSITÀ
DEGLI STUDI
DI UDINE

Arduino - Introduzione

Nozioni di base, la scheda, ambiente di programmazione,
interfacciamento di sensori ed attuatori

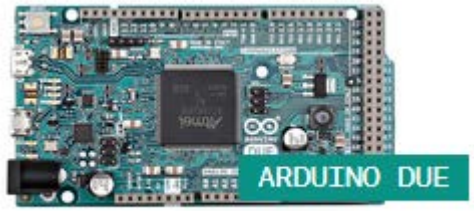
Parte I

Arduino

Cos'è Arduino

- Arduino (https://www.arduino.cc/) è una piattaforma **open-source** di **prototipazione elettronica** (nato ad opera di italiani ad Ivrea, nel **2005**).
- È stata ideata per rendere facile l'interazione di un sistema di calcolo con l'ambiente circostante utilizzando una grande varietà di sensori, motori ed altri attuatori.
- Il microprocessore sulla scheda si programma mediante un insieme di funzioni C/C++ usando un ambiente di sviluppo derivato da **Wiring** (a sua volta basato su **Processing**).
- I progetti sviluppati con Arduino
  - possono funzionare controllati direttamente dal software sulla scheda (modalità **stand-alone**),
  - oppure possono comunicare con software in esecuzione su un computer (per esempio **Processing**).
- Esistono molte versioni della scheda Arduino e molti prodotti basati su di essa:
  - https://www.arduino.cc/en/Main/Products

Tipi di schede Arduino



Programmare Arduino

- Arduino si può programmare tramite l'apposito IDE, scaricabile da:
  - https://www.arduino.cc/en/Main/Software
- Nei successivi esempi programmeremo la board con l'**IDE di Arduino**.

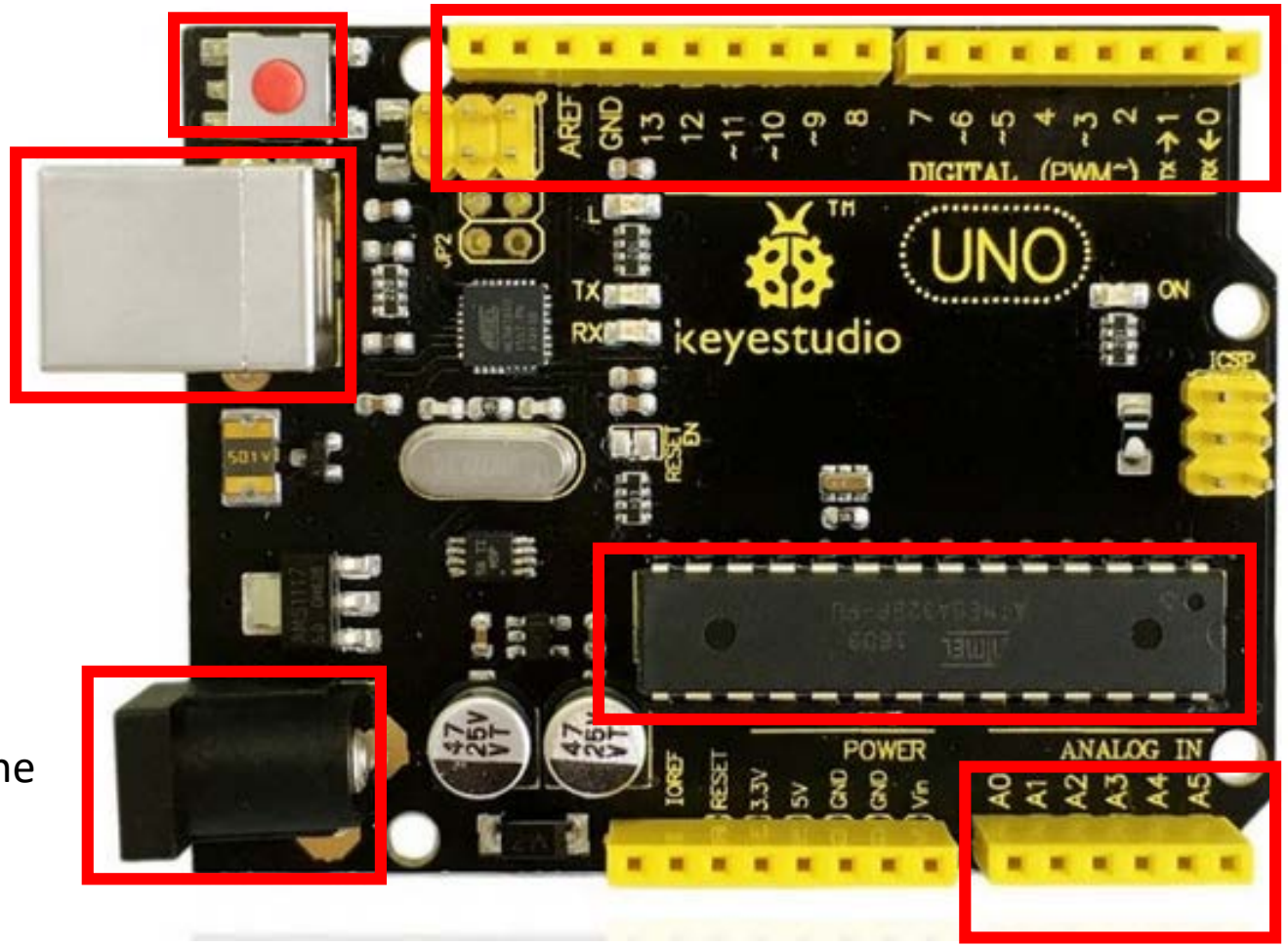
La scheda Arduino Uno

Prodotto a partire dal 2010
Microcontroller: ATmega328
14 digital input/output (I/O) pins
6 analog input pins

Pulsante di reset

14 PIN per digital I/O,

Porta USB (tipo B)



ATmega328P, 8 bit, 16 MHz

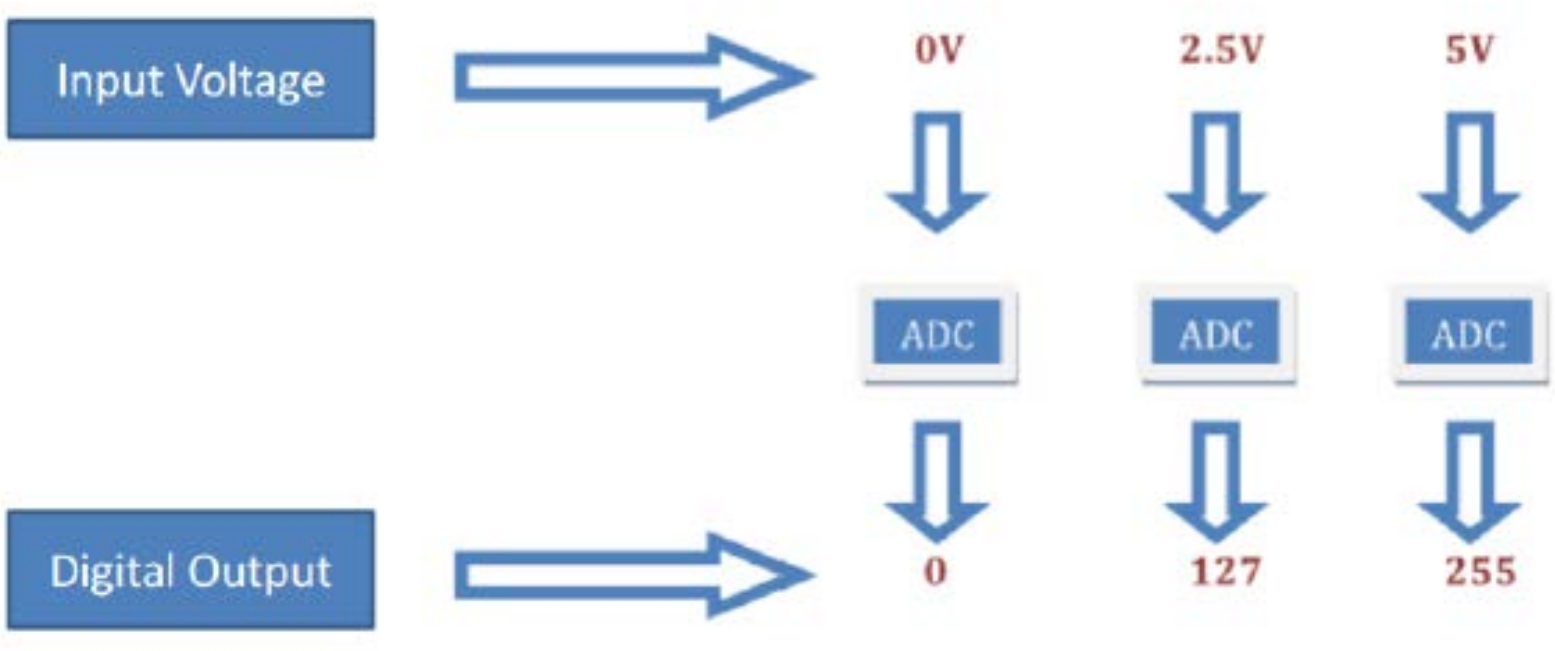
Jack di alimentazione

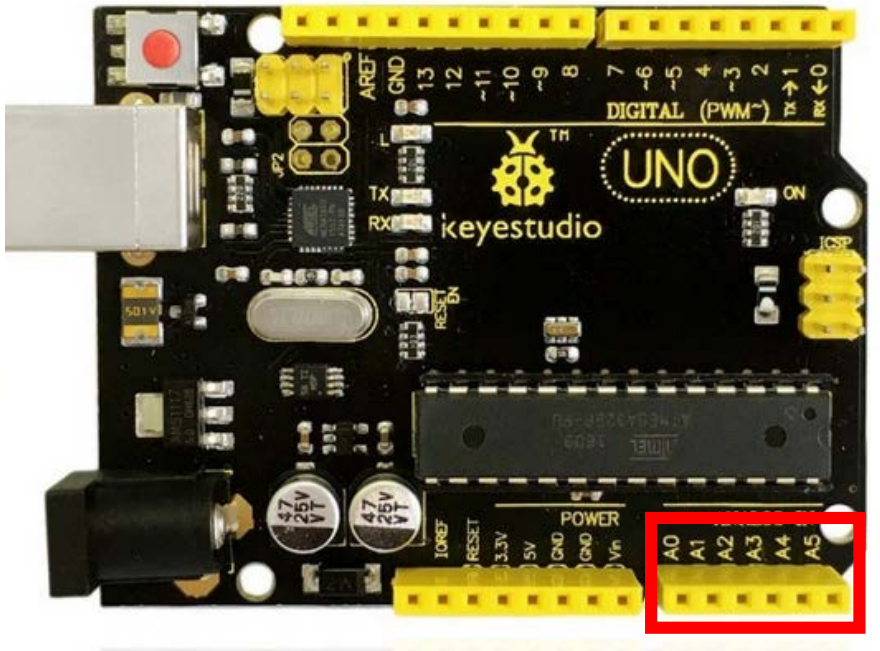
6 PIN per analog input

Segnali digitali e analogici

- Un **segnale analogico** è un **segnale continuo** e la sua caratteristica variabile è una rappresentazione di un'altra quantità variabile nel tempo.
- Un **segnale digitale** utilizza **valori discreti**. Sebbene le rappresentazioni digitali siano discrete, le informazioni rappresentate possono essere discrete o continue.
- Per l'utilizzo con Arduino bisogna **convertire i segnali continui in segnali digitali** (tramite l'uso di **ADC: Analog to Digital Converter**).

ADC





6 PIN per ADC

La scheda Arduino Uno contiene 6 pin per ADC.
La risoluzione della conversione da analogico a digitale è di 10 bit.
Ciò significa che le tensioni di ingresso tra 0V e 5V sono mappate in valori interi compresi tra 0 e 1023.

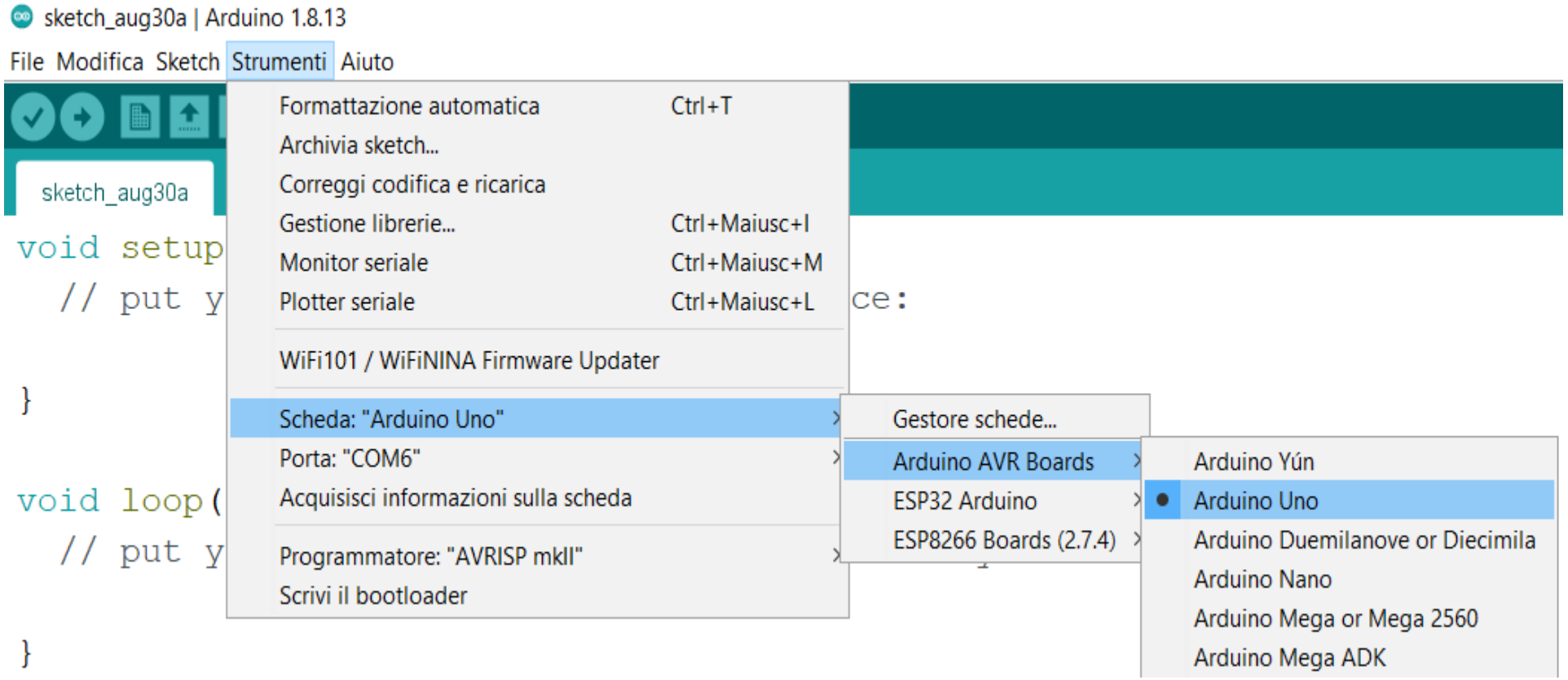
Libreria di Arduino

- La documentazione di riferimento del linguaggio è reperibile all'indirizzo https://www.arduino.cc/reference/en/
- Le due funzioni che devono essere implementate sono **`setup()`** e **`loop()`**.
- Le funzioni di libreria principali usate negli esempi sono:
  - **`Serial.begin()`**
  - **`Serial.print()`**/**`Serial.println()`**
  - **`pinMode()`**
  - **`digitalRead()`**/**`digitalWrite()`**
  - **`analogRead()`**/**`analogWrite()`**
  - **`Servo motor`** → **`motor.attach()`** e **`motor.write()`**
  - **`map()`**
  - **`delay()`**

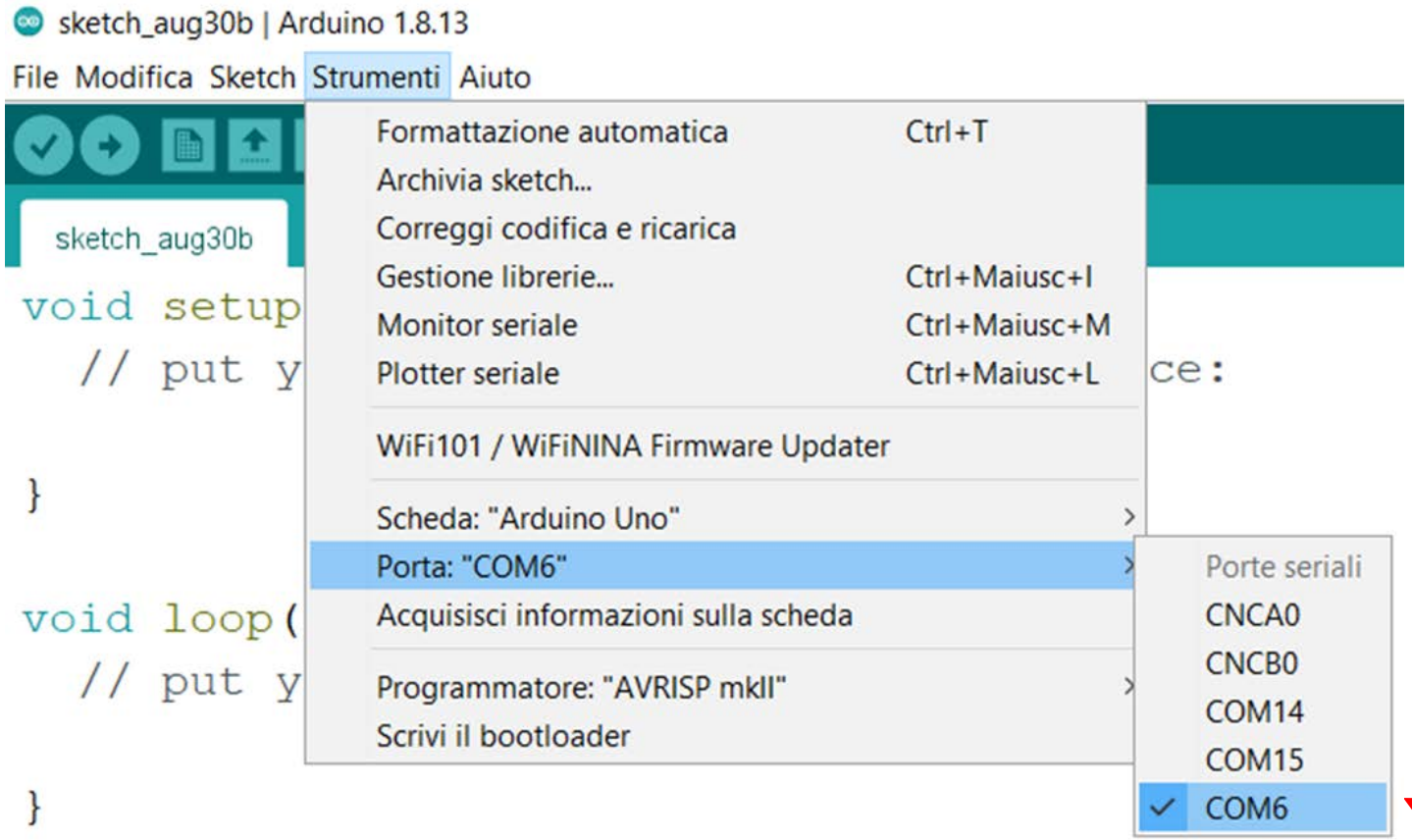
Setup e Loop

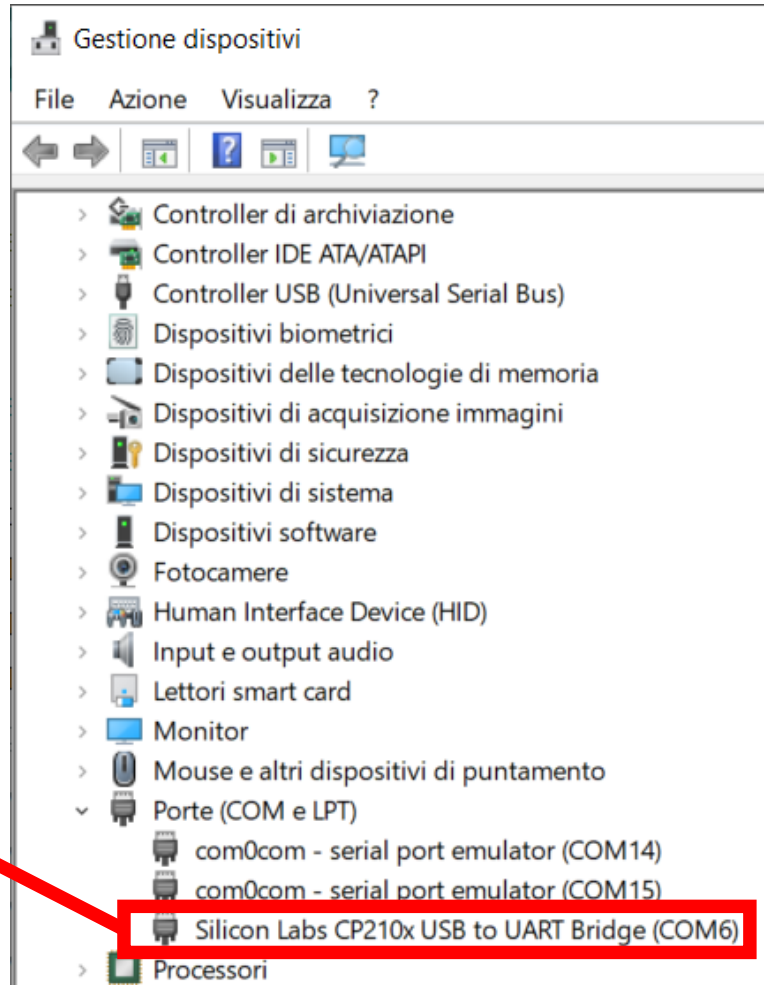
```
void setup()
{
// initialization of variables, pin modes, libraries
// run once after each power up or reset
}
void loop()
{
// loops the content consecutively
// allowing the program to change and respond
}
```

Selezionare la scheda in Arduino IDE



Selezionare la porta in Arduino IDE





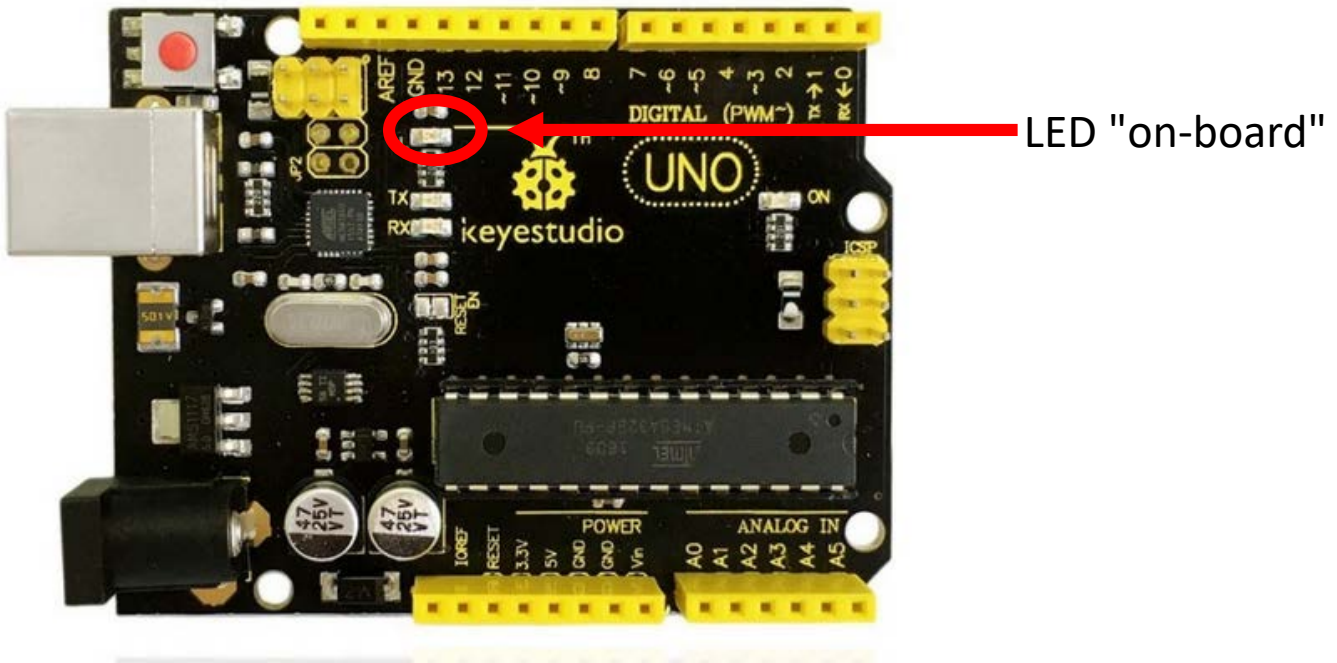
Parte II

Shield, sensori ed attuatori

Il LED on-board di Arduino (sketch: LED_13.ino)

- La **porta digitale 13** (PIN 13) permette anche di interagire con il vicino **LED interno** alla scheda.
- Il codice seguente **alterna** per il LED interno dei cicli di accensione (per 5 secondi) e di spegnimento (per 5 secondi):

```
#define LED_PIN 13
#define TIME_ON 5000
#define TIME_OFF 5000

void setup() {
  pinMode(LED_PIN, OUTPUT);
}
void loop() {
  digitalWrite(LED_PIN, HIGH);
  delay(TIME_ON);
  digitalWrite(LED_PIN, LOW);
  delay(TIME_OFF);
}
```



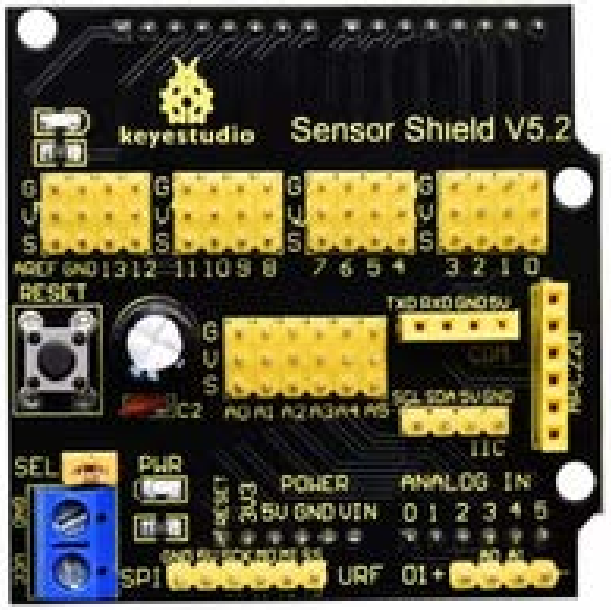
Il LED

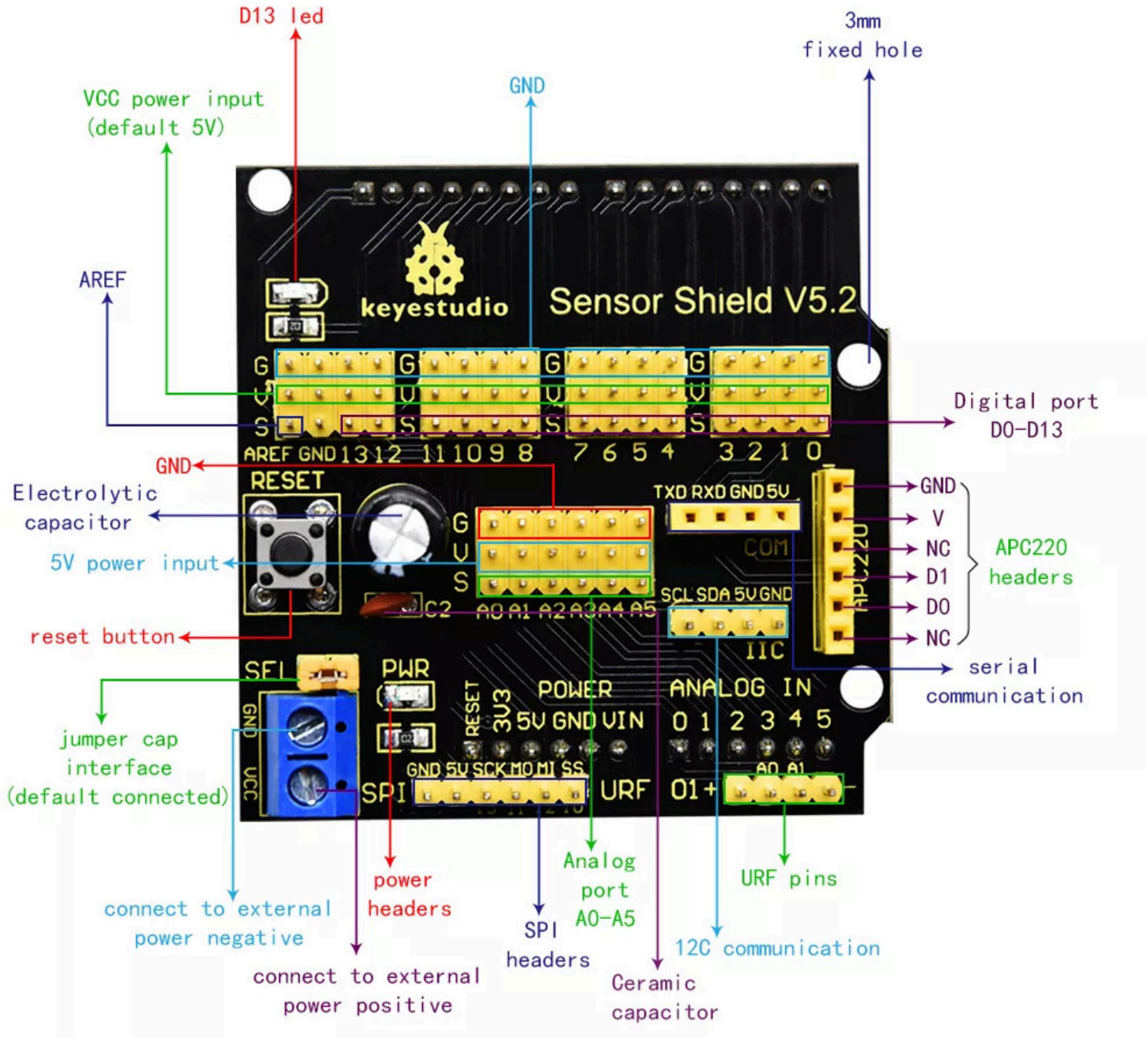
- Il **LED** (Light Emitting Diode) è un dispositivo che si serve della capacità di alcuni materiali semiconduttori di **produrre fotoni** attraverso un fenomeno di emissione spontanea.
- Nel kit è montato su una scheda con le opportune resistenze, un transistor e tre pin per facilitarne la connessione:

S – Signal (segnale)
V – Voltage (tensione)
G – Ground (terra)

Sensor Shield

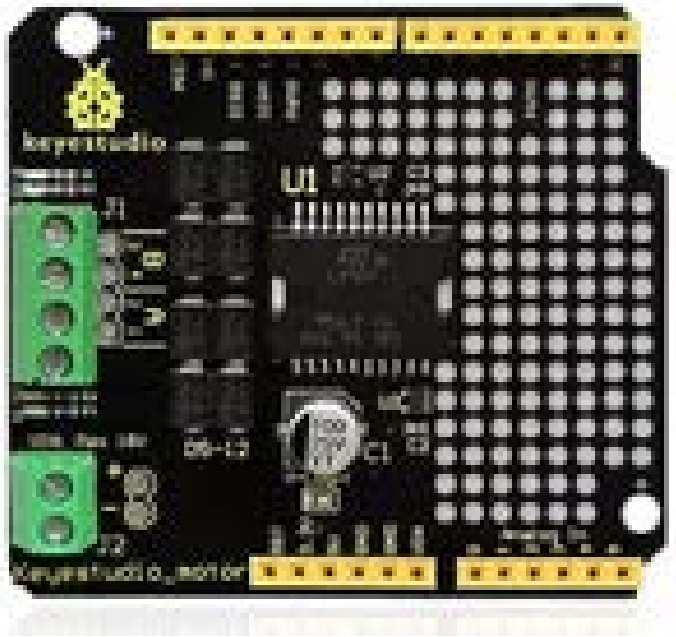
- Per facilitare le connessioni alla scheda Arduino e/o per espanderne le capacità, spesso si ricorre ai cosiddetti «shield», ovvero, delle schede elettroniche che si innestano sui PIN di Arduino.
- Nel kit è presente il Sensor Shield:




D13 led
3mm
fixed hole
VCC power input
(default 5V)
GND
AREF
keyestudio
Sensor Shield V5.2
Digital port
D0-D13
AREF GND 13 12 11 10 9 8 7 6 5 4 3 2 1 0
GND
Electrolytic
capacitor
RESET
TXD RXD GND 5V
GND
V
NC
D1
D0
NC
APC220
headers
5V power input
COM
SCL SDA 5V GND
IIC
reset button
A0 A1 A2 A3 A4 A5
SEL
PWR
POWER
ANALOG IN
0 1 2 3 4 5
serial
communication
jumper cap
interface
(default connected)
3V3
5V GND VIN
GND 5V SCK MO MI SS
SPI
URF
A0 A1
connect to external
power negative
power
headers
Analog
port
A0-A5
URF pins
SPI
headers
12C communication
connect to external
power positive
Ceramic
capacitor

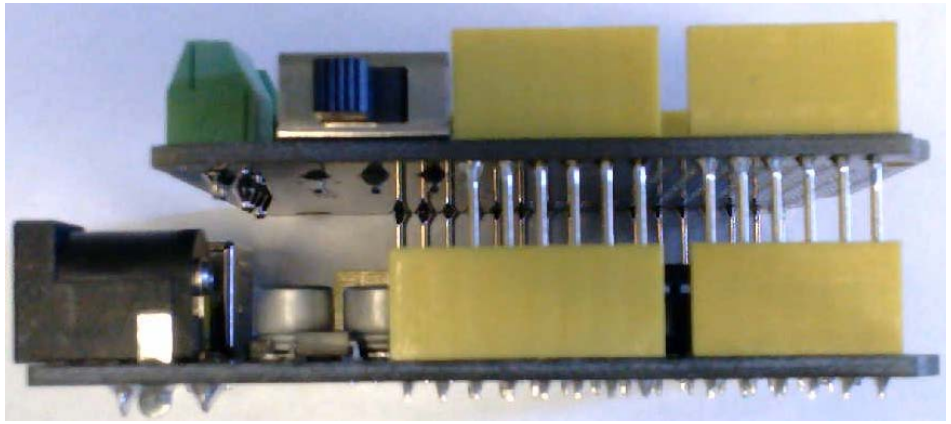
L298P Shield

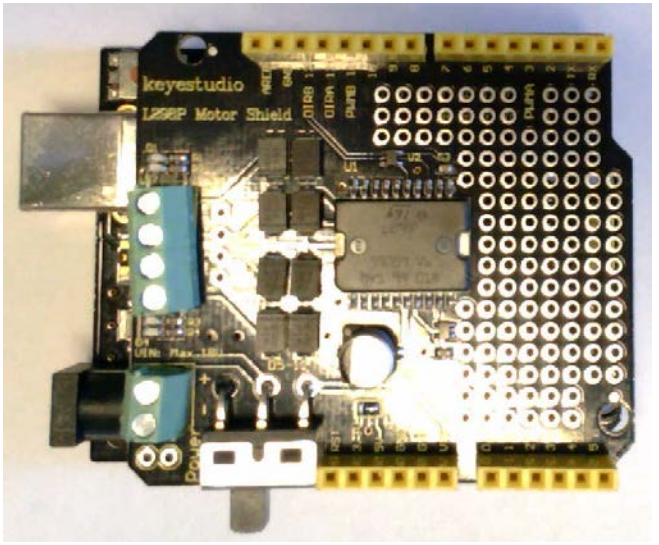
- L’altro shield presente nel kit serve a pilotare tramite la scheda Arduino Uno dei motori a corrente continua alimentati da sorgenti esterne (batterie) erogando un’intensità di corrente fino a 2A.



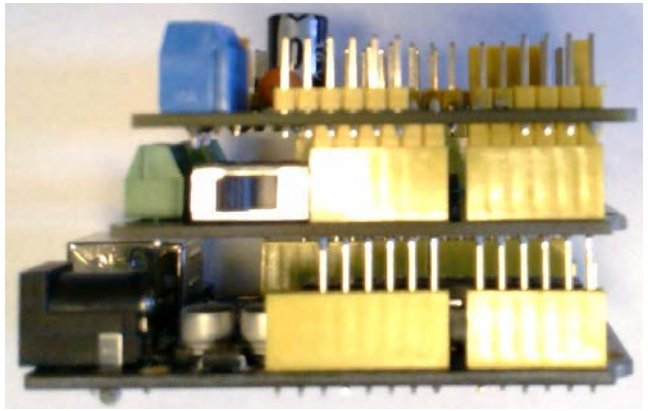
Montaggio dei due shield

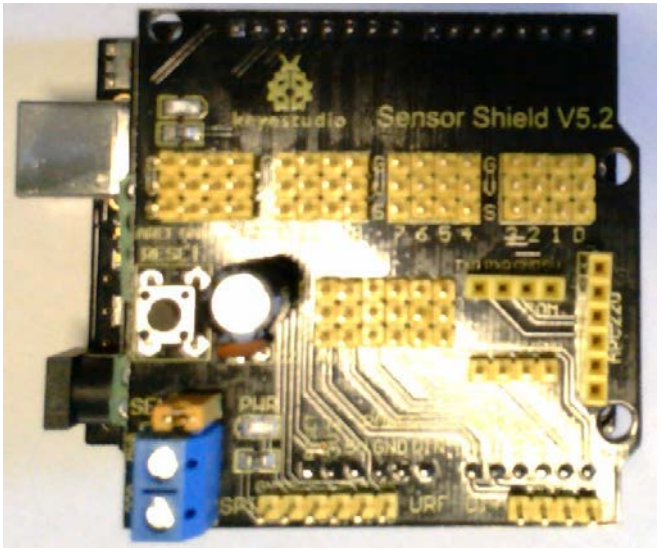
- Connettere prima lo shield L298P ad Arduino:





- In seguito connettere il Sensor Shield sopra lo shield L298P:

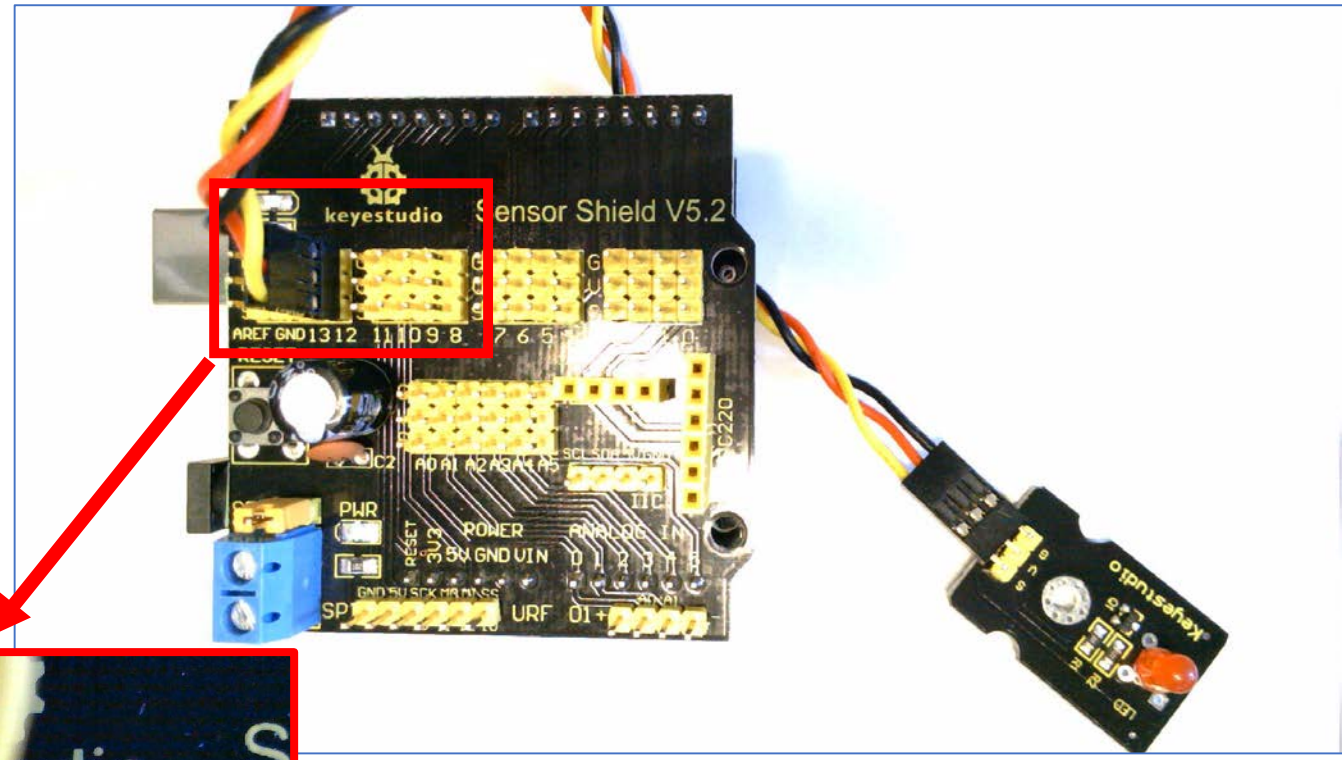


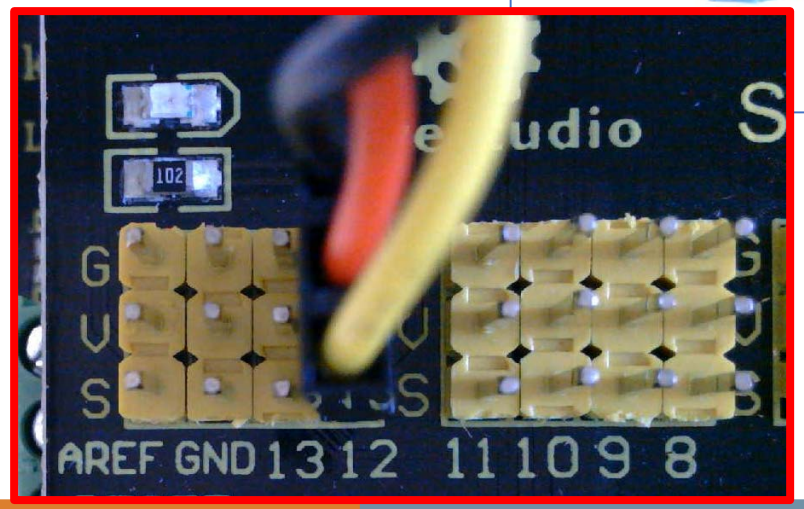


Accendere un LED esterno (sketch: LED_On.ino)

Codice:

```
#define LED_PIN 12
#define TIME_ON 7000
#define TIME_OFF 3000

void setup() {
  pinMode(LED_PIN, OUTPUT); // Configuro il pin come OUTPUT
}

void loop() {
  // Accendo il LED
  digitalWrite(LED_PIN, HIGH);
  // Pausa (7 sec.)
  delay(TIME_ON);
  // Spengo il LED
  digitalWrite(LED_PIN, LOW);
  // Pausa (3 sec.)
  delay(TIME_OFF);
}
```





Far lampeggiare un LED (sketch: LED_Delay.ino)

Supponiamo di voler **variare la luminosità del LED** da minima a massima e viceversa:

```
#define LED_PIN 12
#define PERIOD 30
#define STEP 2
int time_on, inc;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  time_on=0;
  inc=1;
}
```

```
void loop() {
    if(time_on>=0 && time_on<=PERIOD) {
      digitalWrite(LED_PIN, HIGH);
      delay(time_on);
      digitalWrite(LED_PIN, LOW);
      delay(PERIOD-time_on);
    }
    if(inc) {
      if(time_on<PERIOD) { time_on+=STEP; }
      else { inc=0; time_on=PERIOD; }
    }
    else {
      if(time_on>0) { time_on-=STEP; }
      else { inc=1; time_on=0; }
    }
}
```

Cosa succede variando il valore di PERIOD e/o STEP?

Pulse Width Modulation (PWM)

modulazione di larghezza di impulso (sketch: LED_Fade.ino)

Possiamo **variare la luminosità del LED** anche usando la funzione **`analogWrite()`** su un PIN che supporti la **PWM** (PIN il cui numero è preceduto da una **~**, ovvero, i PIN 3, 5, 6, 9, 10 e 11):

```
#define LED_PIN 10
#define PAUSE_TIME 30

void setup() {
  pinMode(LED_PIN, OUTPUT);
}
```

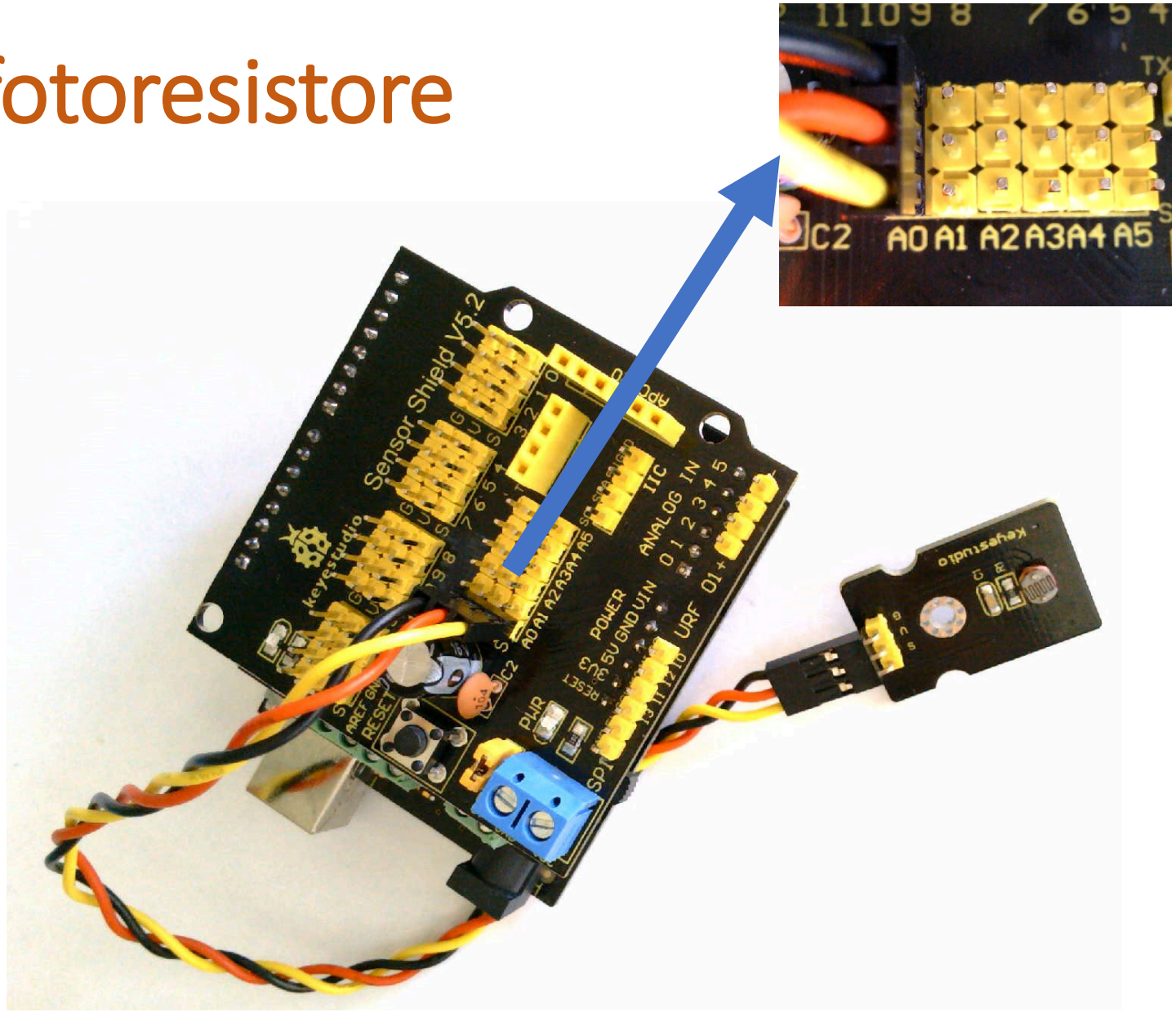
```
void loop() {
  for(int i=0; i<=255; i+=5) {
    analogWrite(LED_PIN, i);
    delay(PAUSE_TIME);
  }
  for(int i=255; i>=0; i-=5) {
    analogWrite(LED_PIN, i);
    delay(PAUSE_TIME);
  }
}
```

Fotoresistore

- Il **fotoresistore** è un componente la cui resistenza è **inversamente proporzionale** alla **quantità di luce** che lo colpisce.

Connessione del fotoresistore

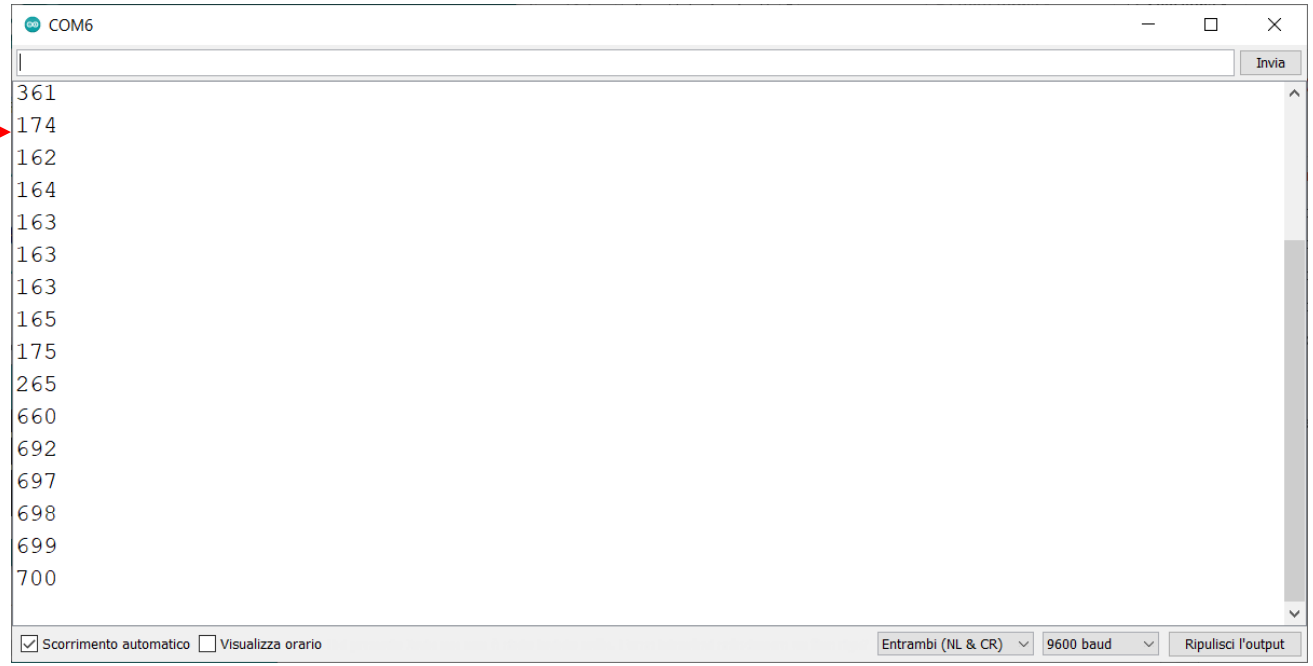
- La connessione è sul PIN analogico A0 (attenzione alla successione dei cavetti: nero -> Ground, rosso -> tensione, giallo -> segnale A0):



Codice di test (Photoresistor.ino)

```
#define sensorPin A0 // imposta il pin del fotoresistore

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(sensorPin);  // legge il valore del sensore
  Serial.println(sensorValue);              // stampa il valore letto sulla seriale
  delay(500);                               // attende mezzo secondo
}
```

Monitor della porta seriale

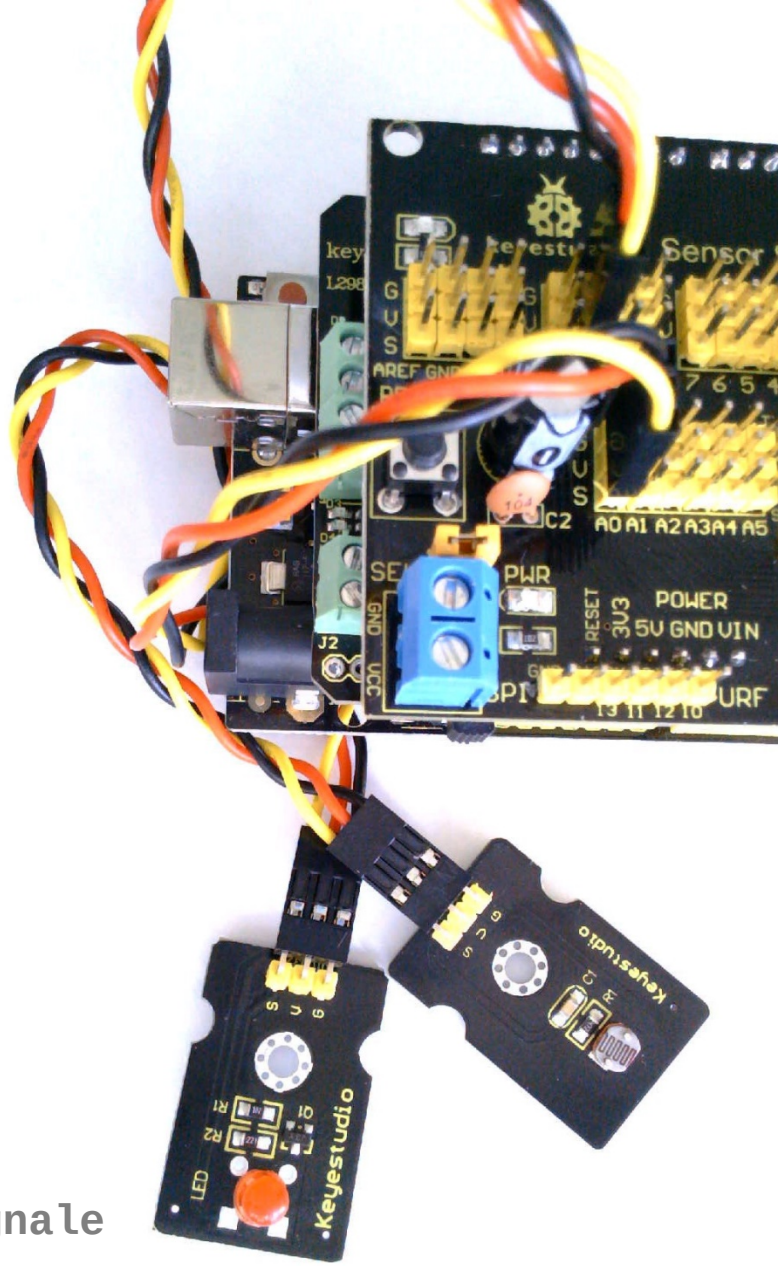


Fotoresistore e LED

- Ricolleghiamo il LED al PIN digitale 10:

```
#define analogInPin A0
#define analogOutPin 10

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(analogInPin);
  // mappa il valore (da 0 a 1023) nell'intervallo 0-255:
  int outputValue = map(sensorValue, 0, 1023, 0, 255);
  analogWrite(analogOutPin, outputValue);
  Serial.println(sensorValue);
  // attende 2 ms per fare in modo che l'ADC stabilizzi il segnale
  delay(2);
}
```

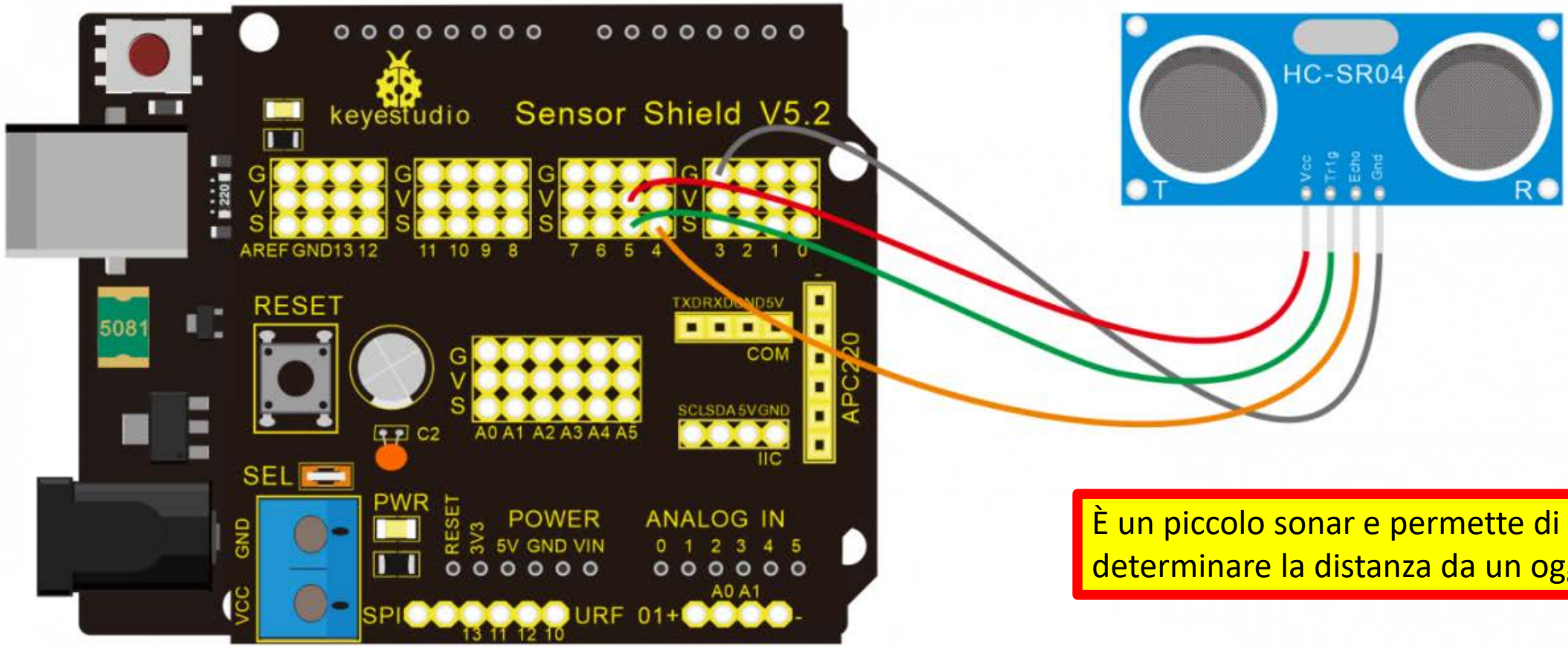
Coprendo il fotoresistore, il LED emana meno luce



Sensore a ultrasuoni

Wiring guide:

Ultrasonic sensor		keyestudio V5 sensor shield
VCC	→	5v(V)
Trig	→	5(S)
Echo	→	4(S)
Gnd	→	Gnd(G)

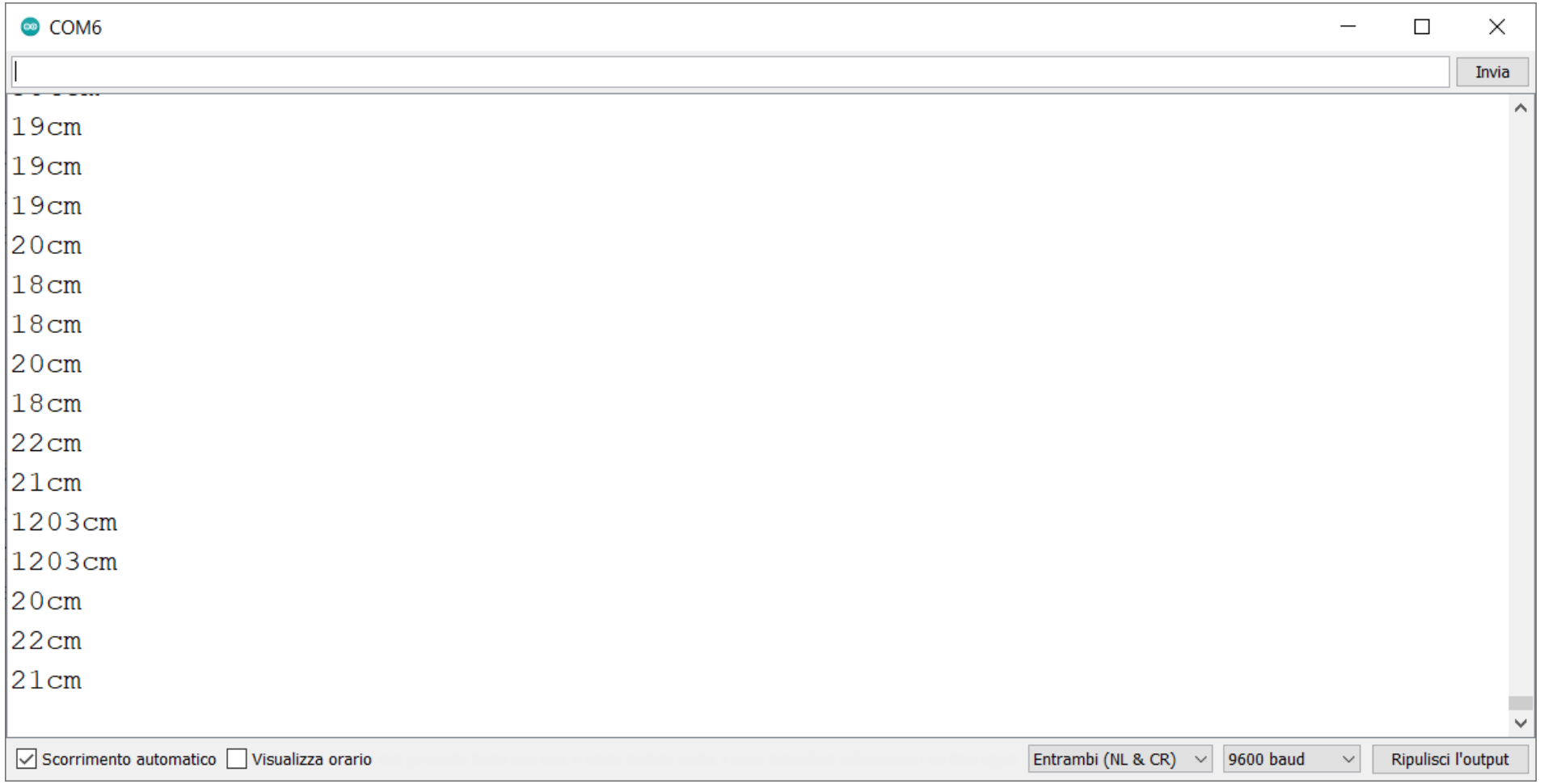


È un piccolo sonar e permette di determinare la distanza da un oggetto

Codice (Ultrasonic.ino)

```
#define trigPin 5    // Trigger
#define echoPin 4    // Echo

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
void loop() {
  // Il segnale di trigger deve restare HIGH per almeno 10 ms.
  // Quindi si emette prima un breve segnale LOW, per assicurare il rilevamento del trigger HIGH:
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // duration è la durata in ms dall'emissione del ping alla ricezione dell'echo per riflessione da un oggetto.
  long duration = pulseIn(echoPin, HIGH);
  // Conversione del tempo in distanza
  long cm = (duration/2) / 29.1;     // divisione per 29.1 o moltiplicazione per 0.0343
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(250);
}
```

Monitor della seriale



Variante con LED (Ultrasonic_LED.ino)

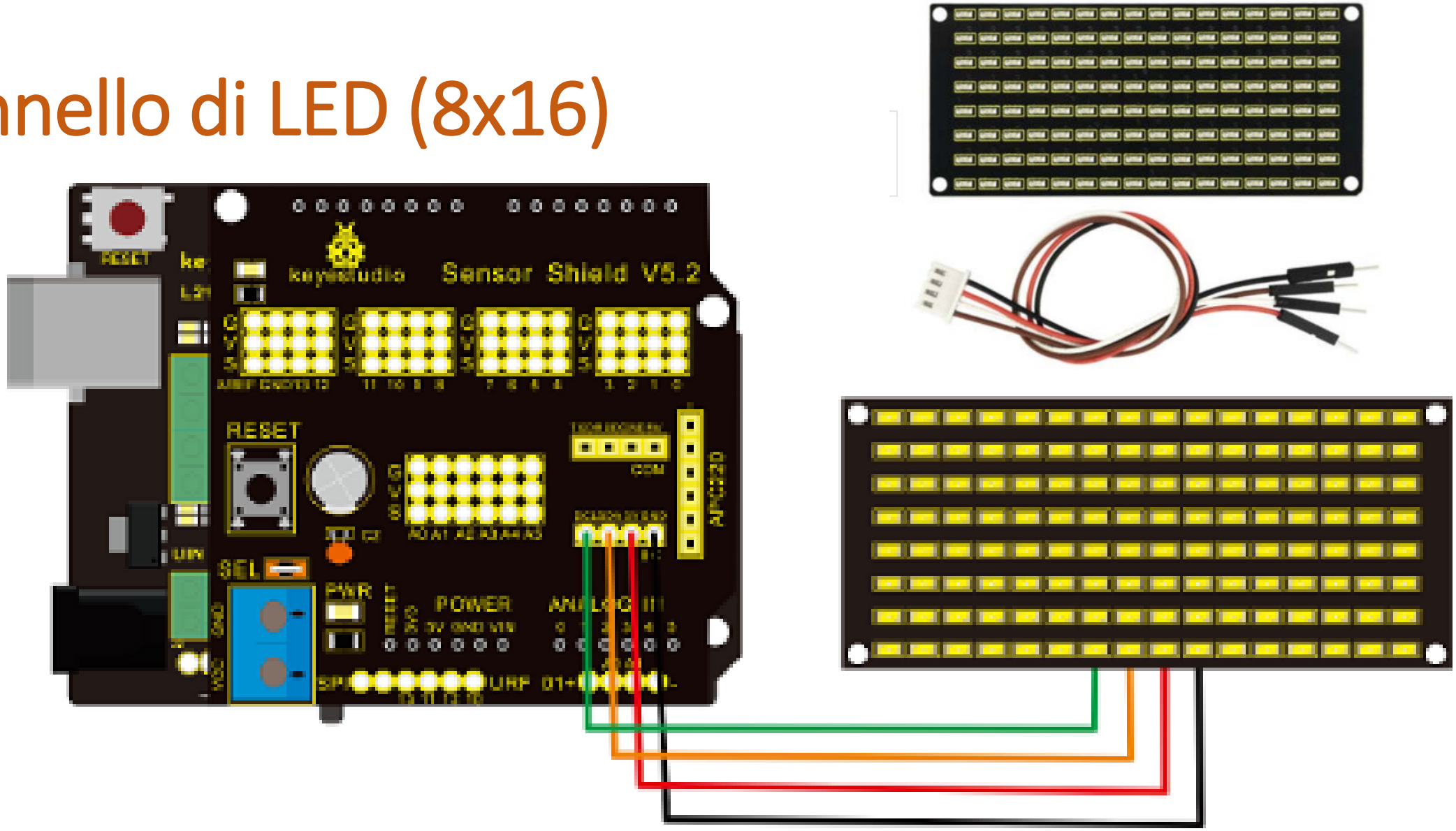
- Con l'aggiunta di un LED si può aggiungere un allarme visivo.
- Basta collegare il LED al solito pin 10 ed aggiungere al loop le righe seguenti, prima della sua fine:

```
if (cm>=2 && cm<=10)
  digitalWrite(alarm, HIGH);
delay(1000);
digitalWrite(alarm, LOW);
delay(1000);
```

- Dove alarm è definito tramite la direttiva seguente:

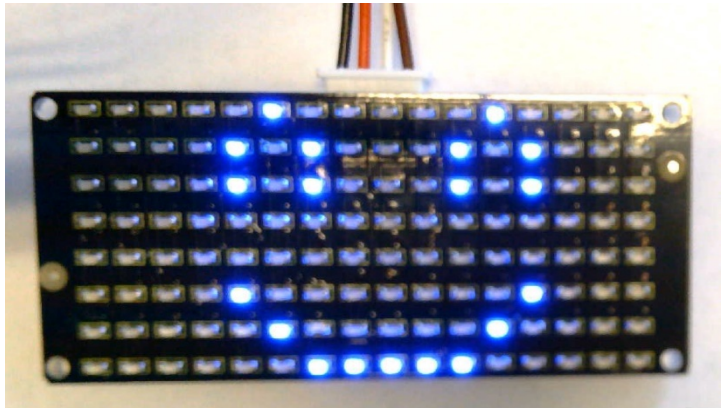
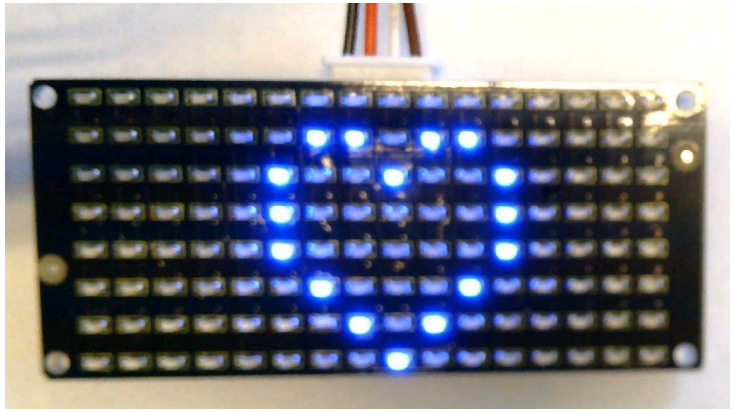
```
#define alarm 10
```

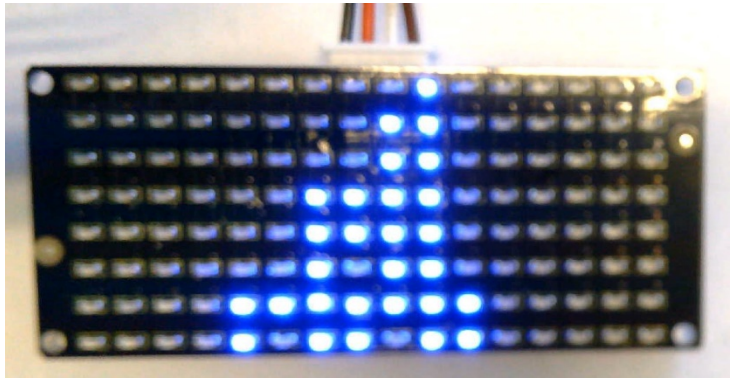
Pannello di LED (8x16)



Sketch di esempio (LED_Panel.ino)

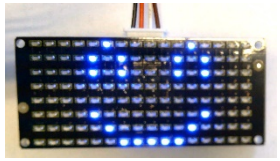
- Disegna in successione 4 immagini sul pannello:



Codifica delle immagini (I)

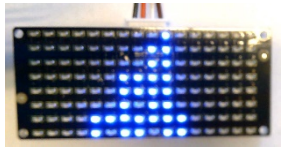
- Le immagini sono codificate nella matrice **`unsigned char table[4][16]`**
- Ogni immagine è rappresentata da 16 numeri esadecimali (ogni numero rappresenta la configurazione acceso/spento degli 8 LED di una colonna del pannello):



0x00,0x00,0x00,0x00,0x26,0x41,0x86,0x80,0x80,0x80,0x86,0x41,0x26,0x00,0x00,0x00

0x00,0x00,0x00,0x00,0x00,0x1C,0x22,0x42,0x84,0x42,0x22,0x1C,0x00,0x00,0x00,0x00

0x00,0x00,0x00,0x00,0x20,0x44,0x42,0x84,0x80,0x84,0x42,0x44,0x20,0x00,0x00,0x00



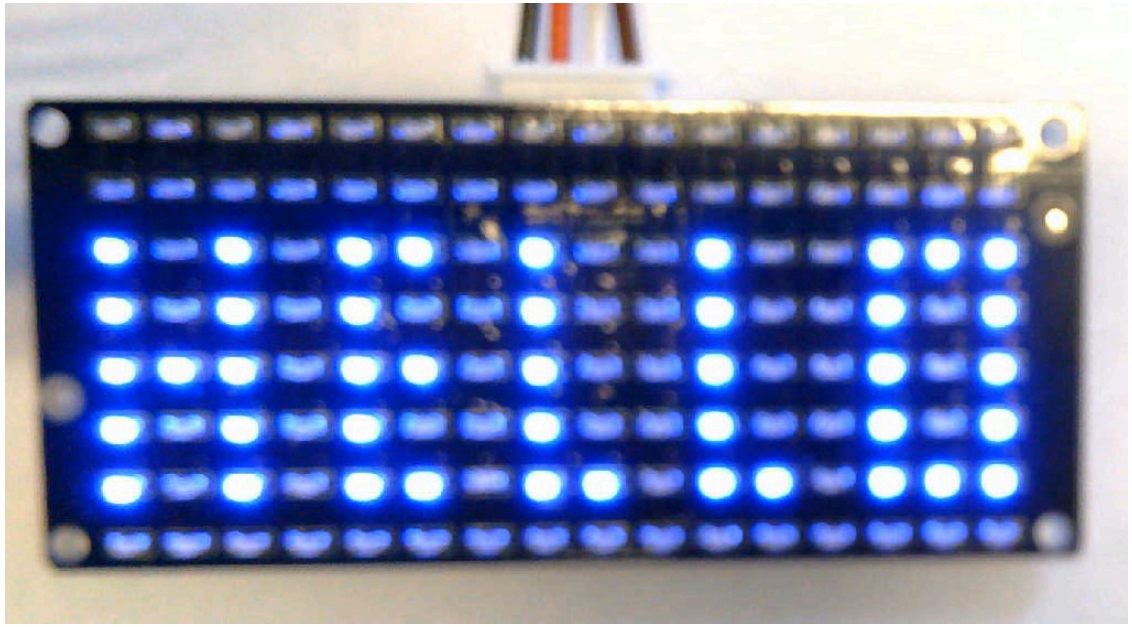
0x00,0x00,0x00,0x00,0xC0,0x40,0xF8,0xD8,0x7E,0xFF,0xC0,0x00,0x00,0x00,0x00,0x00

Codifica delle immagini (II)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0
0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x00	0x00	0x00	0x00	0x00	0x1C	0x22	0x42	0x84	0x42	0x22	0x1C	0x00	0x00	0x00	0x00

Esercizio

- Aggiungere una nuova linea alla variabile table in modo che codifichi la stringa «HELLO».



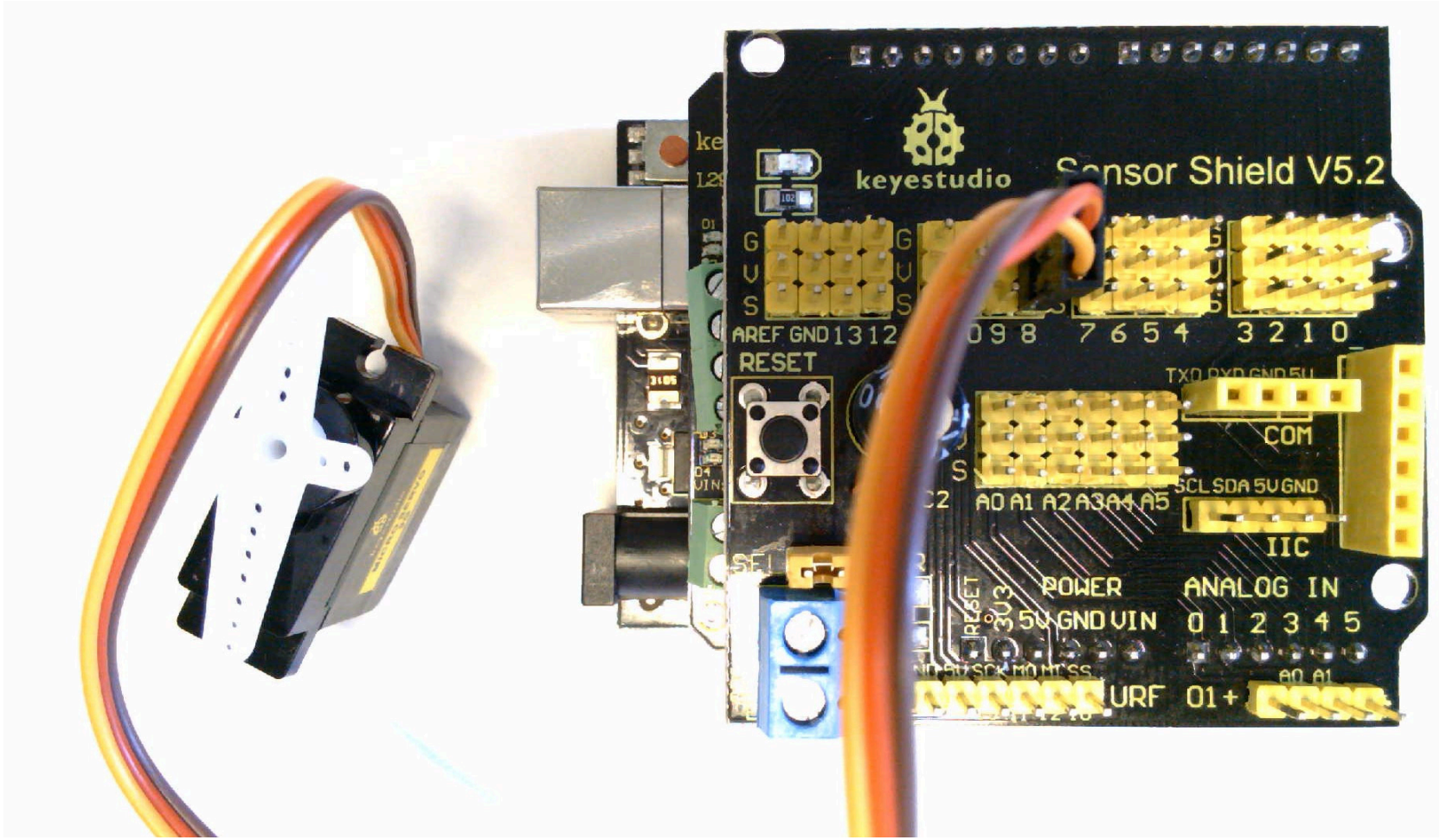
Suggerimento: utilizzare la calcolatrice di Windows per trasformare i pattern di 0 e 1 (spento/acceso) in numeri esadecimali

Servomotore

- Si tratta di un tipo di **motore** che **può ruotare di 180 gradi**: gli impulsi ricevuti dicono al motore quale posizione assumere.

Servomotore: connessione al Sensor Shield

Connessione del cavo al PIN 8:

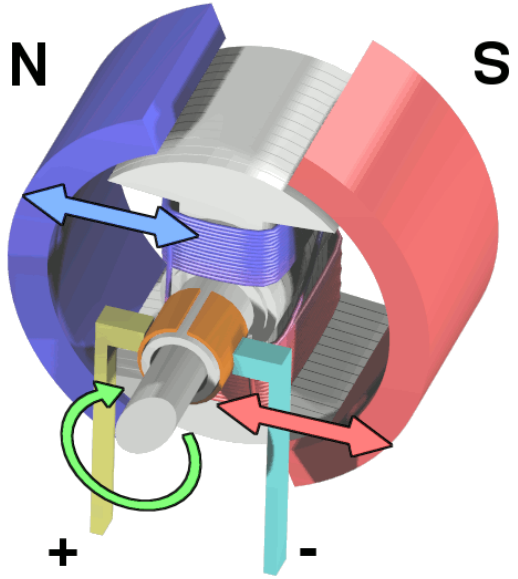


Servomotore: codice (sketch Servo.ino)

```
#include <Servo.h>
#define MOTOR_PIN 8                    // PIN (digitale) del Servomotore
#define SERIAL_SPEED 9600

Servo motor;

void setup() {
  motor.attach(MOTOR_PIN);
  Serial.begin(SERIAL_SPEED);
  delay(1000);
}
void loop() {
  int val=-1;

  if (Serial.available()) {
    val = Serial.parseInt();
    Serial.readStringUntil('\n');
  }
  if (val >= 0 && val <= 180) {
      Serial.println(val);
      motor.write(val);
  }
}
```

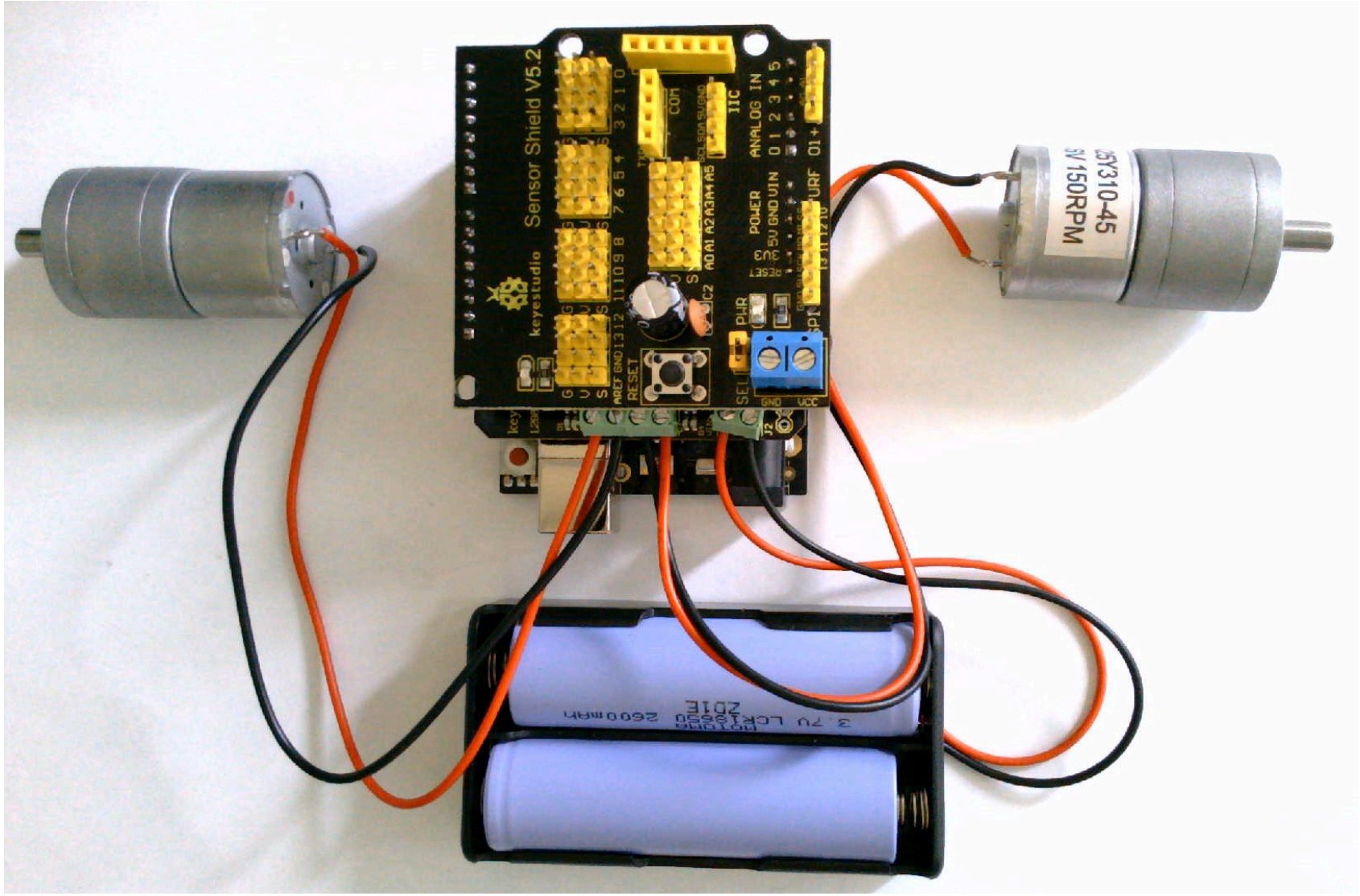
Motore a corrente continua (DC Motor)

- Un motore a corrente continua converte **energia elettrica** in **energia meccanica**.



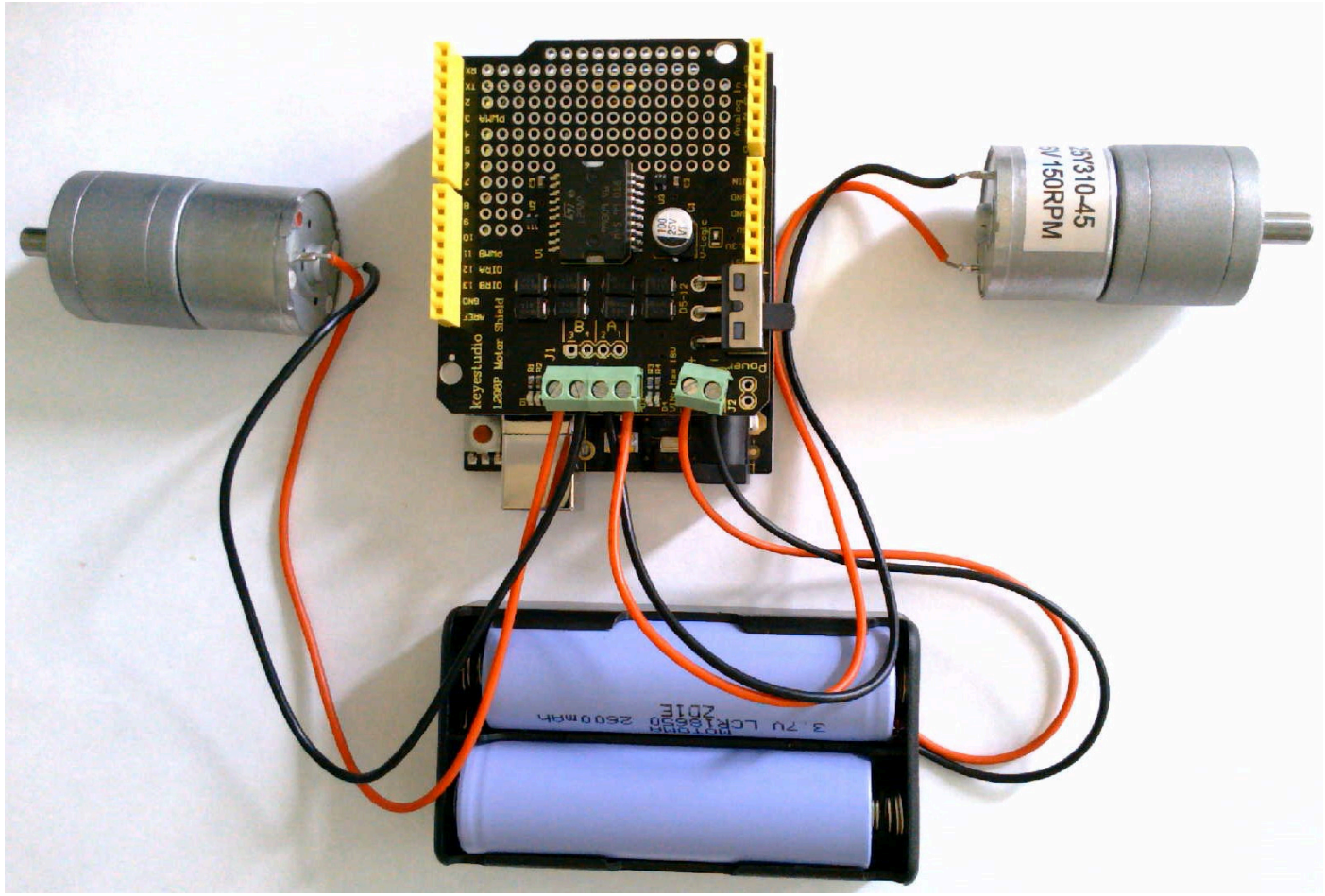
Quando la corrente scorre negli avvolgimenti, si genera un campo magnetico intorno al rotore. La parte sinistra del rotore è respinta dal magnete di sinistra ed attirata da quello di destra. Analogamente fa la parte in basso a destra. La coppia genera la rotazione. (Fonte: Wikipedia)

Connessioni dei motori e della batteria

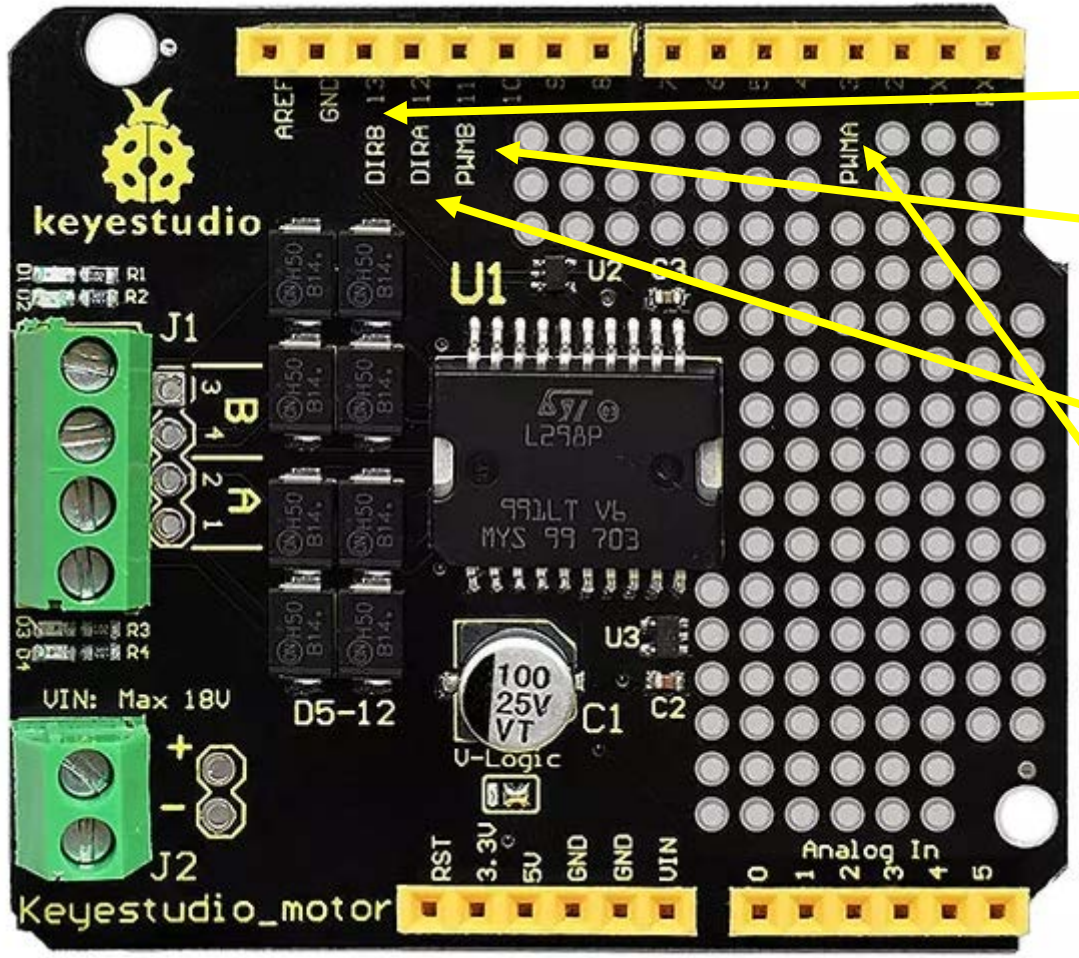


Attenzione alle connessioni (fili rossi e neri)

Sbagliare la connessione dei motori li porterà a girare in senso opposto a quello previsto.
Attenzione alla polarità delle batterie quando vengono inserite nel portabatterie.
Attenzione a collegare correttamente i fili del portabatterie allo shield L298P.
Usare il cacciavite a taglio per serrare i contatti.



Schema delle connessioni e dei PIN



Motore sinistro (B)

+

-

Motore destro (A)

-

+

Portabatteria

+

-

Direzione motore sinistro (PIN D13)

Velocità motore sinistro (PIN D11)

Direzione motore destro (PIN D12)

Velocità motore destro (PIN D3)

Codice di test – Motor_Test.ino (I)

```
#define ML_Ctrl 13 // associa il pin di controllo di direzione del motore di sinistra
#define ML_PWM 11  // associa il pin di velocità del motore di sinistra
#define MR_Ctrl 12 // associa il pin di controllo di direzione del motore di destra
#define MR_PWM 3   // associa il pin di velocità del motore di destra

void setup() {
  pinMode(ML_Ctrl, OUTPUT); // imposta il pin della dir. del motore sinistro come output
  pinMode(ML_PWM, OUTPUT);  // imposta il pin della vel. del motore sinistro come output
  pinMode(MR_Ctrl, OUTPUT); // imposta il pin della dir. del motore destro come output
  pinMode(MR_PWM, OUTPUT);  // imposta il pin della vel. del motore destro come output
}

void loop() {
  // va avanti
  digitalWrite(ML_Ctrl,LOW); // direzione di rotazione del motore sinistro: avanti (LOW)
  analogWrite(ML_PWM,200);   // velocità del motore sinistro: 200
  digitalWrite(MR_Ctrl,LOW); // direzione di rotazione del motore destro: avanti (LOW)
  analogWrite(MR_PWM,200);   // velocità del motore destro: 200
  delay(2000);               // attende 2s
  ...
```

Codice di test – Motor_Test.ino (II)

```
// va indietro
digitalWrite(ML_Ctrl,HIGH); // direzione di rotazione del motore sinistro: indietro (HIGH)
analogWrite(ML_PWM,200);    // velocità del motore sinistro: 200
digitalWrite(MR_Ctrl,HIGH); // direzione di rotazione del motore destro: indietro (HIGH)
analogWrite(MR_PWM,200);    // velocità del motore destro: 200
delay(2000);                // attende 2s
// gira a sinistra
digitalWrite(ML_Ctrl,HIGH); // direzione di rotazione del motore sinistro: indietro (HIGH)
analogWrite(ML_PWM,200);    // velocità del motore sinistro: 200
digitalWrite(MR_Ctrl,LOW);  // direzione di rotazione del motore destro: avanti (LOW)
analogWrite(MR_PWM,200);    // velocità del motore destro: 200
delay(2000);                // attende 2s
// gira a destra
digitalWrite(ML_Ctrl,LOW);  // direzione di rotazione del motore sinistro: avanti (LOW)
analogWrite(ML_PWM,200);    // velocità del motore sinistro: 200
digitalWrite(MR_Ctrl,HIGH); // direzione di rotazione del motore destro: indietro (HIGH)
analogWrite(MR_PWM,200);    // velocità del motore destro: 200
delay(2000);                // attende 2s
// stop (azzera la velocità dei motori)
analogWrite(ML_PWM,0);      // velocità del motore sinistro: 0
analogWrite(MR_PWM,0);      // velocità del motore destro: 0
delay(2000);                // attende 2s
}
```