Istituto **P**rofessionale per L'**i**ndustria L'**A**rtigianato
"Antonio Guastaferro"

63074 SAN BENEDETTO DEL TRONTO (AP)

Classe : 4A_IPAI -- 5A_IPAI
A.S. : 2020-2021
Docenti : Tufoni Franco -- Enrico Ruggieri
Disciplina : Tecnologie elettriche-elettroniche, dell'automazione e applicazioni

Arduino
Applicazioni

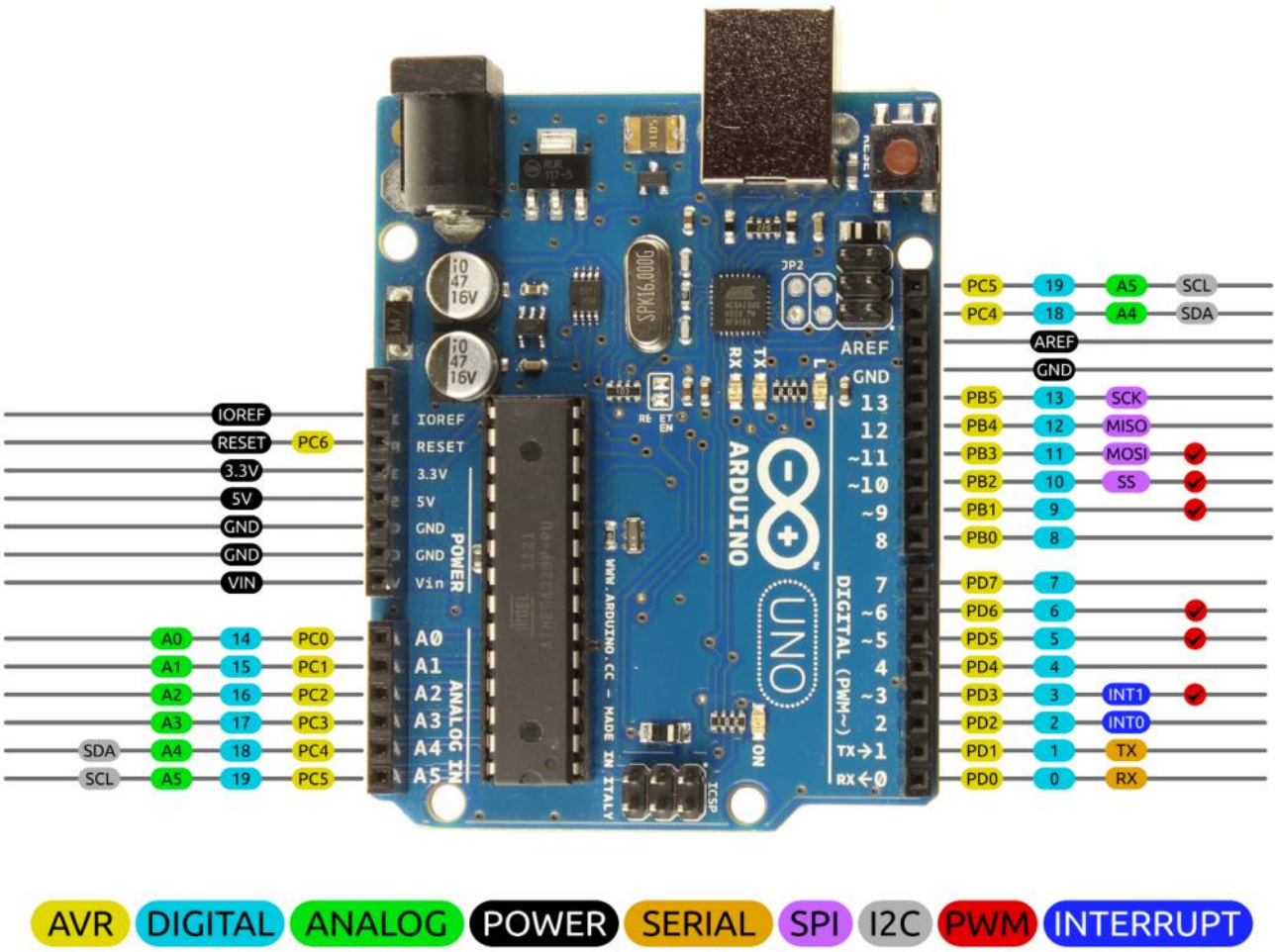
Open Source

Vers. 3.9 –26-10-2020

Indice

1 – Lampeggio Led
2 - Lampeggio Led Rosso-Verde
3 – Lampeggio Led Rosso-Verde ripetuto 15 volte
4 – Controllo Led BiColore-Bipolare
5 – Controllo Led BiColore a catodo comune
6 – Controllo Lampeggio Led con Potenziometro
7 – Controllo LED RGB a catodo comune – Test colori
Base (Red, Green, Blue) e Derivati (Magenta, Cyan, Yellow)
8 – Controllo LED RGB a catodo comune – Test colori 16 M
9– Controllo LED RGB a catodo comune con tre potenziometri
10 – Controllo 8 Led
11 – Accensione Led con un Pulsante: PL1
12 – Accensione e Spegnimento Led con un Pulsante: PL2
13 – Controllo 4 pulsanti e 4 LED con libreria antiribalzo: PL3
14 – Controllo luminosità LED tramite pulsanti: PL4
15 – Controllo Start – Stop (Avvio/Arresto): PL5
16 – Controlli di Flusso – Strutture di Controllo
17 – Controllo LED con sensore di posizione e pulsante
18 – LED Array 1 – Effetto scorrimento con ritardo variabile
19 – LED Array 2 – Effetto scorrimento a blocchi con ritardo variabile
20 – LED BAR – Effetto riempimento
21 – Controllo LED con Fotoresistenza
22 – Controllo quattro LED con Fotoresistenza
23 – Controllo intensità luminosa LED con Fotoresistenza
24 – Rilevazione temperatura con il Sensore TMP 36 e Monitor Seriale
25 – Rilevazione temperatura con il Sensore TMP 36 e Display LCD 16x2
26 – Controllo Servomotore
27 – Controllo Servomotore con due pulsanti
28 – Controllo Servomotore con un potenziometro
29 – Inseguitore solare con Fotoresistenze e Servomotore
30 – Modulo Sensore di fiamma
31 – Controllo Motore a Corrente Continua
32 – Flex Sensor e Servo Motore
33 – Modulo Sensore di Temperatura e Umidità con DHT11
34 – Modulo Sensore magnetico – Effetto Hall
35 – Modulo Sensore di Tilt
36 – Modulo Sensore Umidità del terreno
37 – Calendario con RTC DS 1307 e Display LCD 16x2
38 – Comando 2 Servo con un modulo Laser
39 – Gioco Quiz - 4 Concorrenti
40 – Comando LED tramite Monitor Seriale e Tastiera
41 –Generatore di funzioni con il plotter seriale
42 - Simulazione Legge di Ohm
43 - Simulazione Porte logiche
44 - Simulazione Timer NE555
45 - Es. Manuale Sik Guide Sparkfun (10,16)

Arduino Uno R3 Pinout

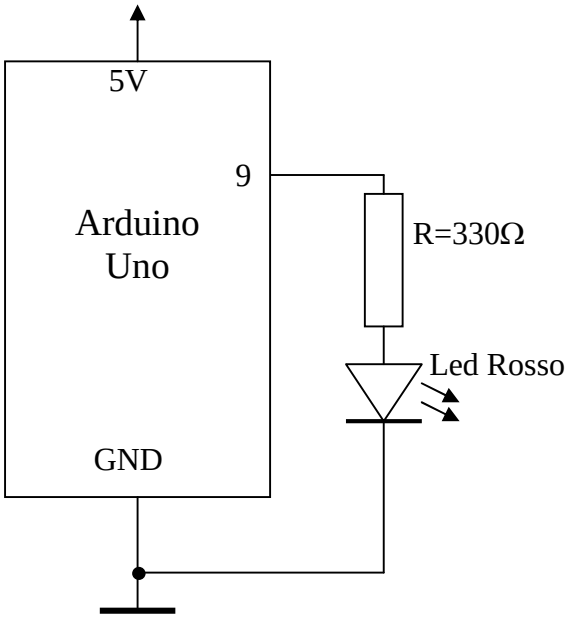


Esercizio 1 – Lampeggio Led

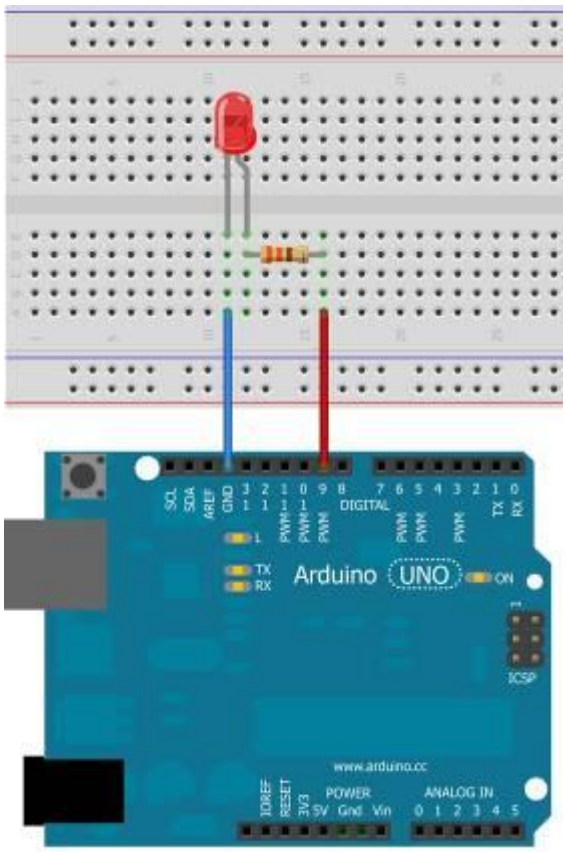
1) Disegnare lo schema elettrico
2) Disegnare lo schema di cablaggio
3) Scrivere il programma

Soluzione

1 – Schema elettrico



2 - Cablaggio



3 - Programma

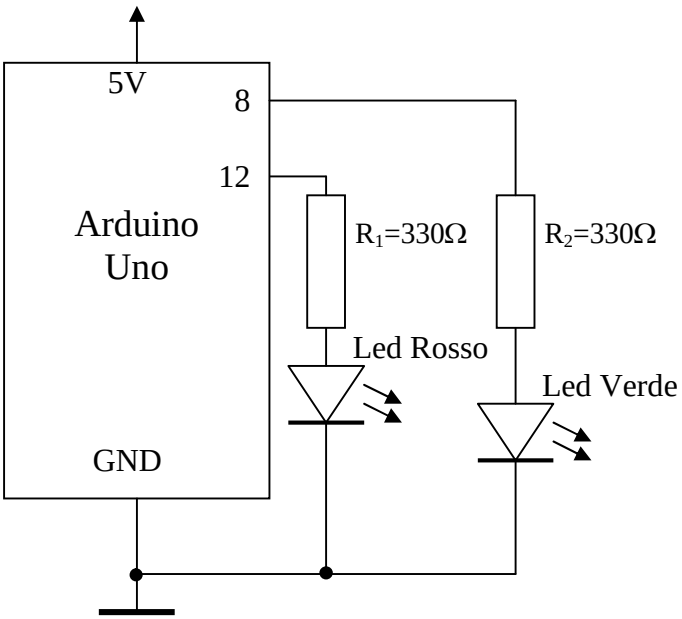
```
// Esercizio 1 - LED Flasher
int ledPin=9;
void setup() {
pinMode(ledPin, OUTPUT);
}
void loop() {
digitalWrite(ledPin, HIGH);
delay(500);
digitalWrite(ledPin, LOW);
delay(500);
}
```

Esercizio 2 – Lampeggio Led Rosso-Verde

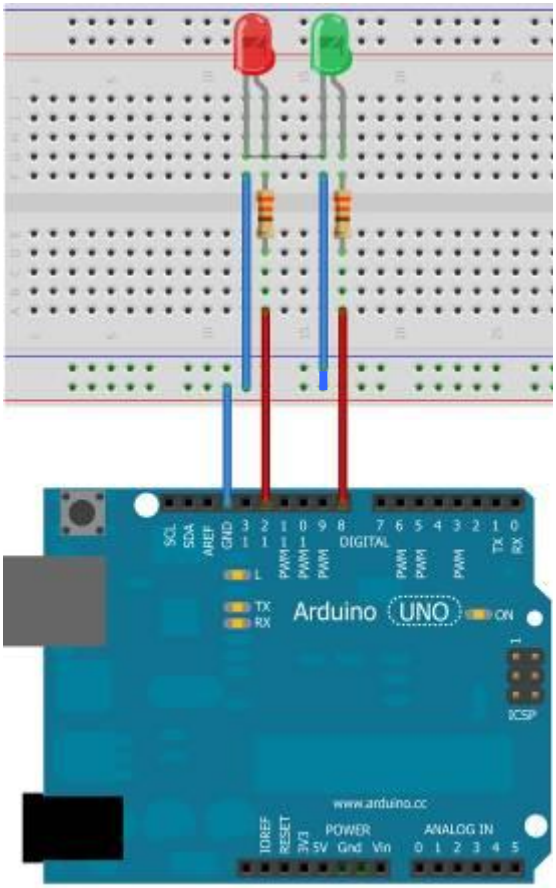
1) Disegnare lo schema elettrico
2) Disegnare lo schema di cablaggio
3) Scrivere il programma

Soluzione

1 – Schema elettrico



2 - Cablaggio



3 - Programma

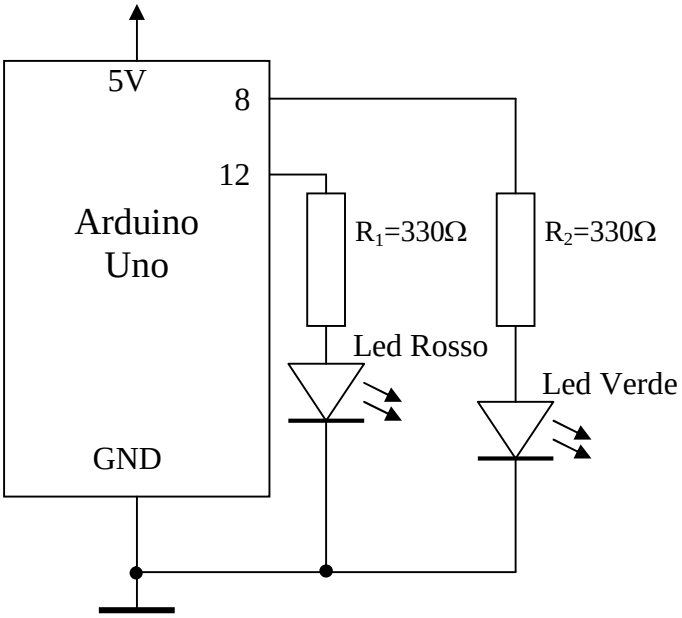
```
// Esercizio 2 - LED Flasher Rosso-Verde
int LedRosso = 12;
int LedVerde=8;

void setup() {
pinMode(LedRosso, OUTPUT);
pinMode(LedVerde, OUTPUT);
}
void loop() {
digitalWrite(LedRosso, HIGH);
digitalWrite(LedVerde, LOW);
delay(500);
digitalWrite(LedRosso, LOW);
digitalWrite(LedVerde, HIGH);
delay(500);
}
```

Esercizio 3 – Lampeggio Led Rosso-Verde ripetuto 15 volte

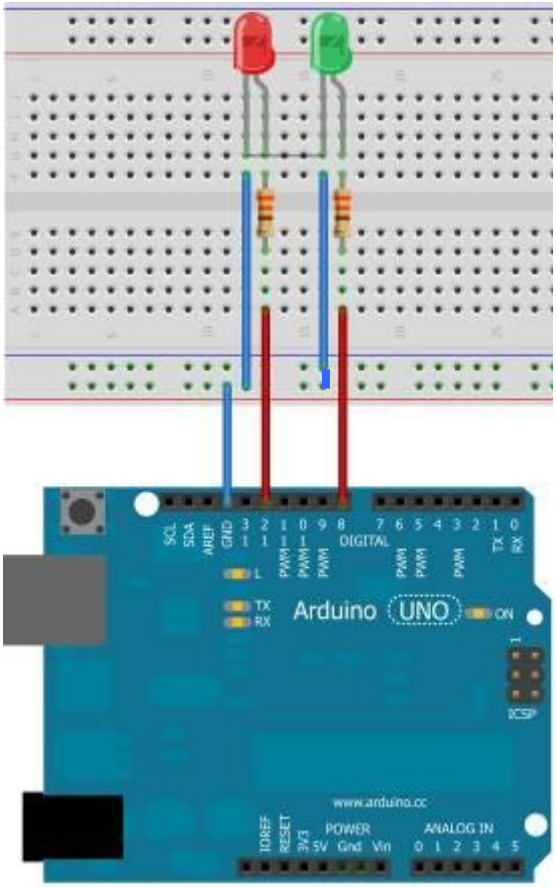
1) Disegnare lo schema elettrico
2) Disegnare lo schema di cablaggio
3) Scrivere il programma

Soluzione

1 – Schema elettrico



2 - Cablaggio



3 - Programma

```
// Esercizio 3 - LED Flasher Rosso-Verde – 15 volte
int LedRosso = 12;
int LedVerde=8;
int conta = 0;                          // variabile contatore
void setup() {
pinMode(LedRosso, OUTPUT);
pinMode(LedVerde, OUTPUT);
}
void loop() {
if(conta<15){
digitalWrite(LedRosso, HIGH);
digitalWrite(LedVerde, LOW);
delay(500);
digitalWrite(LedRosso, LOW);
digitalWrite(LedVerde, HIGH);
delay(500);
conta++;                                // incrementa la variabile conta
}
}
```

Esercizio 4 – Controllo Led BiColore-Bipolare

4) Disegnare lo schema elettrico
5) Disegnare lo schema di cablaggio
6) Scrivere il programma

Soluzione

Il LED bicolore/bipolare è formato da due Led (Es. Rosso/Verde) disposti in antiparallelo sullo stesso involucro (Fig.1).

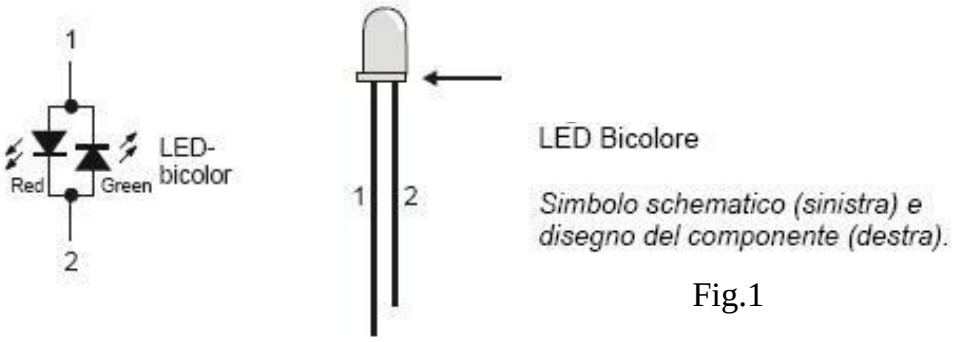


Fig.1

In Fig. 2 sono riportate le modalità di alimentazione

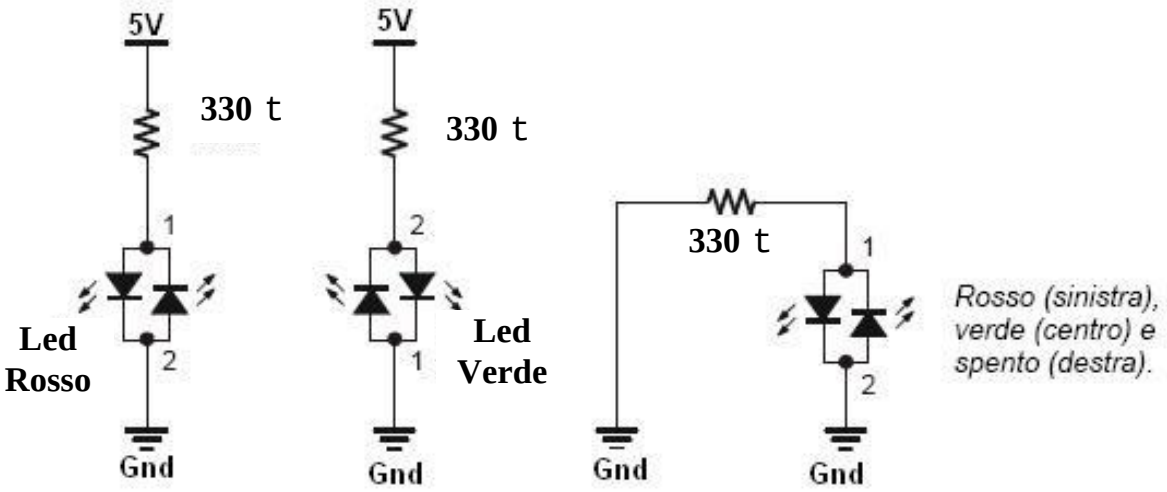


Fig. 2: Modalità di alimentazione

In Fig. 3 è riportato lo schema elettrico del collegamento del Led Bicolore/Bipolare con Arduino. In Fig. 4 il cablaggio con Fritzing.

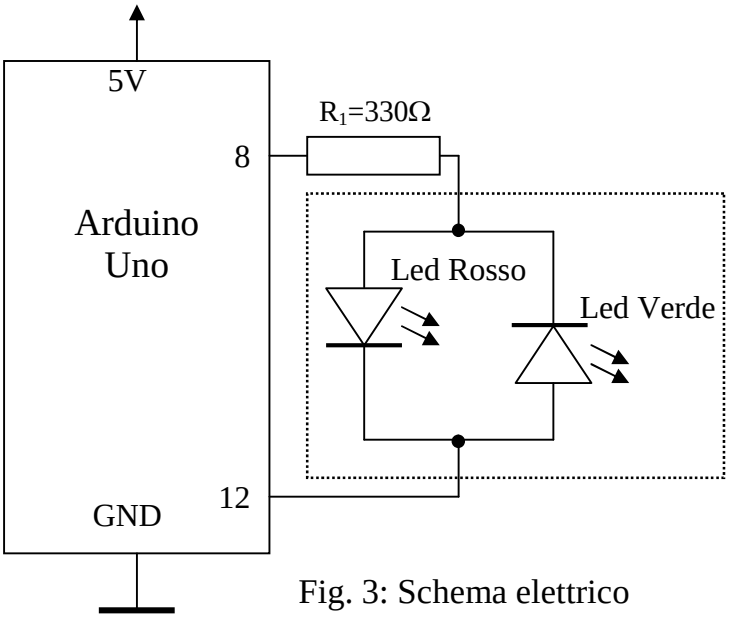


Fig. 3: Schema elettrico

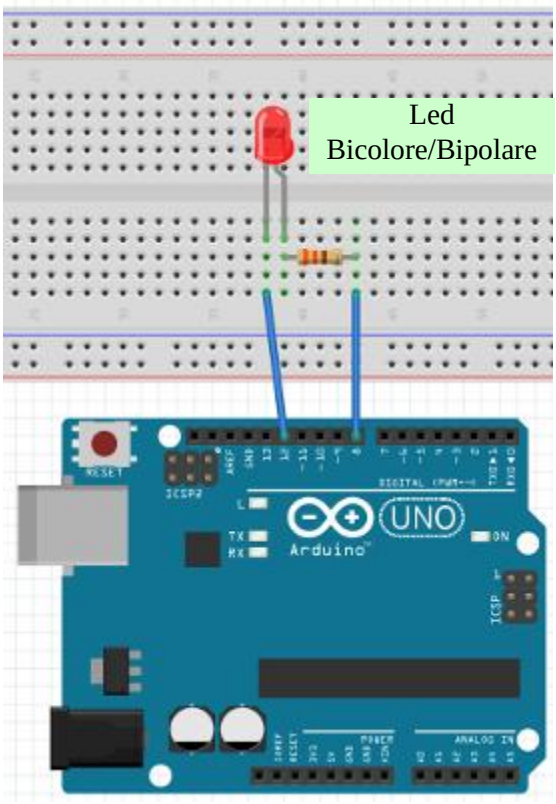


Fig. 4: Cablaggio con Fritzing

Programma

```
// Esercizio 4 - LED Bicolore/Bipolare
int ledK= 12;
int ledA=8;
int ritardo=500;

void setup()

{
   pinMode(ledK,OUTPUT);
   pinMode(ledA,OUTPUT);
}

void loop()
{
  digitalWrite(ledK,HIGH);
  digitalWrite(ledA,LOW);
  delay(ritardo);
  digitalWrite(ledK,LOW);
  digitalWrite(ledA,HIGH);
  delay(ritardo);
}
```

Esercizio 5 – Controllo Led BiColore a catodo comune

1) Disegnare lo schema elettrico
2) Disegnare lo schema di cablaggio
3) Scrivere il programma

Soluzione

Il LED bicolore a catodo comune è formato da due Led (Es. Rosso/Verde) disposti con il catodo in comune sullo stesso involucro (Fig.1).

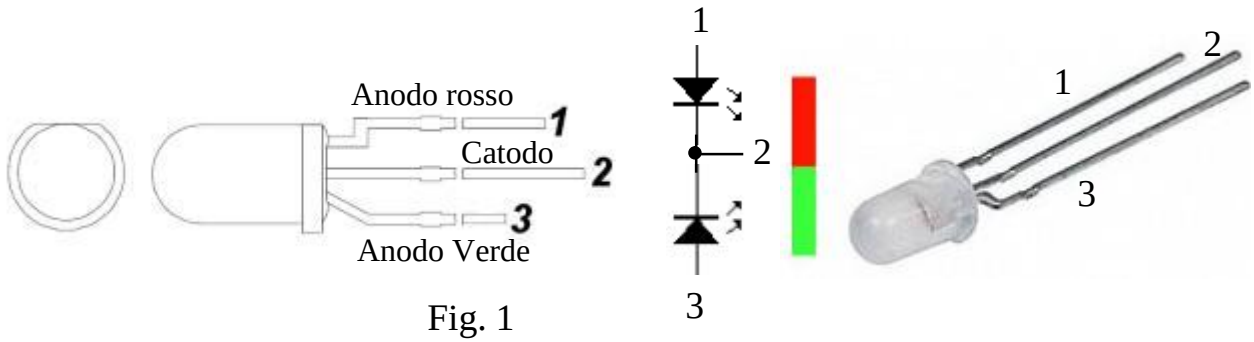


Fig. 1

In Fig. 2 è riportato lo schema elettrico del collegamento del Led Bicolore a catodo comune con Arduino. In Fig. 3 il cablaggio con Fritzing.

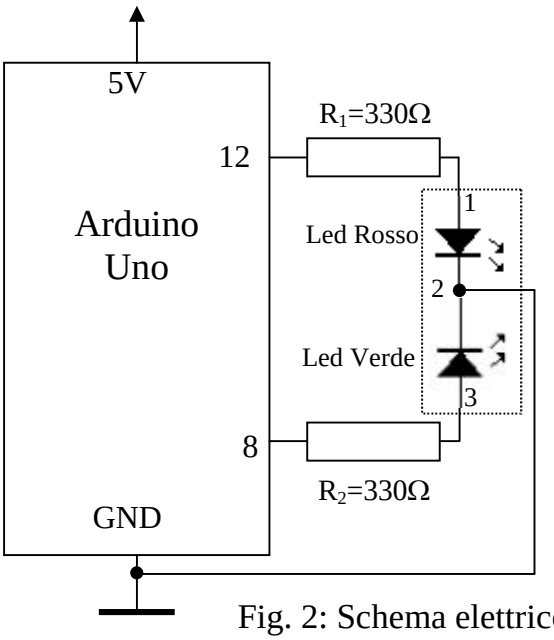


Fig. 2: Schema elettrico

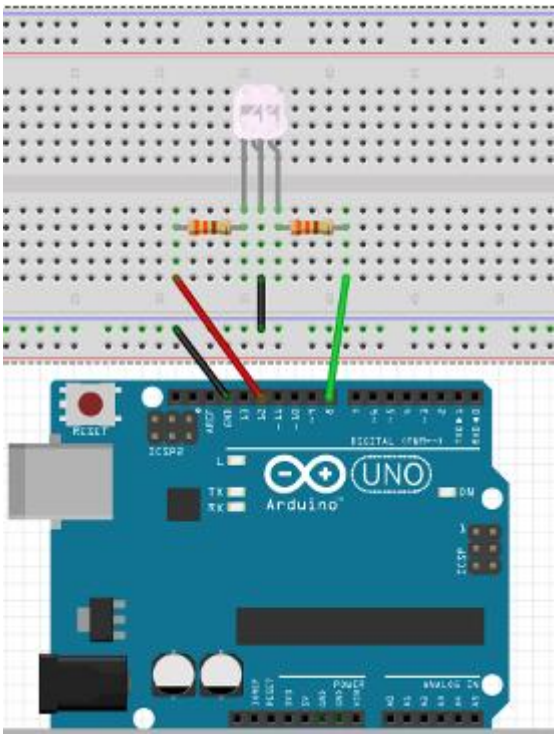


Fig. 3: Cablaggio con Fritzing

Programma -1

```
// Esercizio 5A - LED Bicolore a catodo comune
int ledR= 12;
int ledV=8;
int ritardo=500;

void setup()

{
   pinMode(ledR,OUTPUT);
   pinMode(ledV,OUTPUT);
}

void loop()
{
   digitalWrite(ledR,HIGH);
  digitalWrite(ledV,LOW);
  delay(ritardo);
  digitalWrite(ledR,LOW);
  digitalWrite(ledV,HIGH);
  delay(ritardo);
}
```

Programma -2

```
// Esercizio 5B - LED Bicolore a catodo comune
int ledR= 12;
int ledV=8;
int ritardo=500;

void setup()

{
   pinMode(ledR,OUTPUT);
   pinMode(ledV,OUTPUT);
}

void loop()
{
   digitalWrite(ledR,HIGH);
  digitalWrite(ledV,LOW);
  delay(ritardo);
  digitalWrite(ledR,LOW);
  digitalWrite(ledV,HIGH);
  delay(ritardo);
  digitalWrite(ledR,HIGH);
  digitalWrite(ledV,HIGH);
  delay(ritardo);
  digitalWrite(ledR,LOW);
  digitalWrite(ledV,LOW);
  delay(ritardo);
}
```

Esercizio 6 – Controllo Lampeggio Led con Potenziometro

1) Disegnare lo schema elettrico
2) Disegnare lo schema di cablaggio
3) Scrivere il programma

Soluzione

Il potenziometro con il centrale collegato sull'ingresso Analogico A0 fornisce una tensione variabile tra 0V e 5V.
Il valore analogico tramite il convertitore A/D a 10 bit interno viene convertito in digitale (numero compreso tra 0 e 1023).
Il valore numerico (0 ÷ 1023) viene utilizzato come ritardo nella sequenza di lampeggio (0ms ÷ 1023ms), per aumentare il tempo è sufficiente moltiplicare il numero per una costante (es. 2, 3, 4 ecc).
In Fig. 1 è riportato lo schema elettrico, in fig. 2 lo schema di cablaggio con Fritzing.

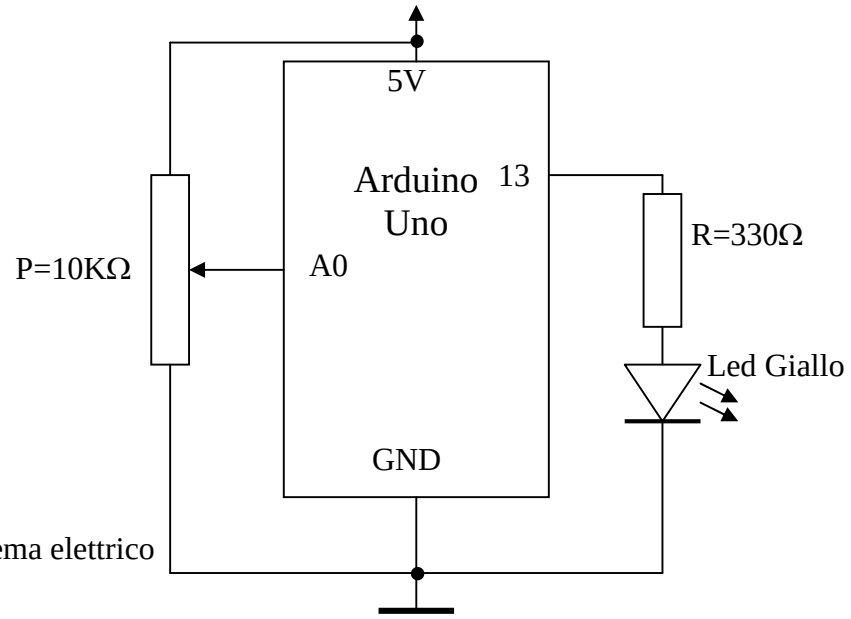


Fig. 1: Schema elettrico

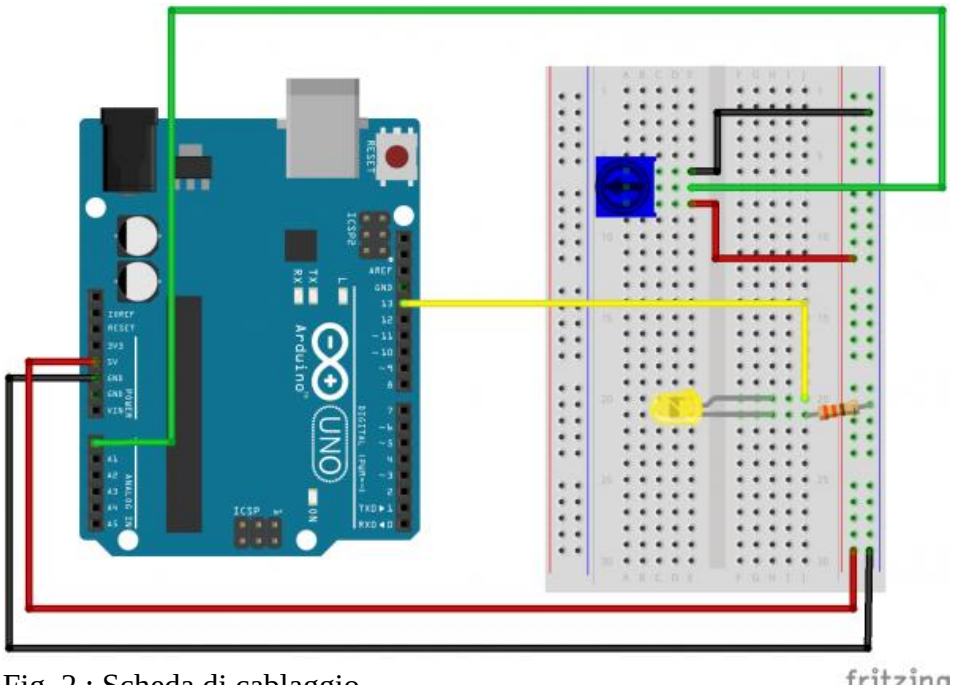


Fig. 2 : Scheda di cablaggio

Programma -1

```
// Esercizio 6A - Controllo Lampeggio Led con Potenziometro (0 ÷ 1023)
int pot = A0;       // seleziona il pin per il potenziometro
int ledPin = 13;    // seleziona il pin per il ED
int valore = 0;     // Variabile per memorizzare il valore digitale (0 ÷1023)

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
 {
  valore = analogRead(pot);          // Legge il valore analogico (0 ÷ 5v)
                                     // Converte in digitale (0 ÷ 1023)
                                     // Memorizza il risultato in valore

  digitalWrite(ledPin, HIGH);
  delay(valore);                     // ritardo, 0 ms ÷ 1023 ms
  digitalWrite(ledPin, LOW);
  delay(valore);
}
```

Programma -2

```
// Esercizio 6B - Controllo Lampeggio Led con Potenziometro (0 ÷ 4092)
int pot = A0;       // seleziona il pin per il potenziometro
int ledPin = 13;    // seleziona il pin per il ED
int valore = 0;     // Variabile per memorizzare il valore digitale (0 ÷1023)

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
 {
  valore = analogRead(pot);          // Legge il valore analogico (0 ÷ 5v)
                                     // Converte in digitale (0 ÷ 1023)
                                     // Memorizza il risultato in valore

  digitalWrite(ledPin, HIGH);
  delay(4*valore);                           // ritardo, 0 ms ÷ 4092 ms
  digitalWrite(ledPin, LOW);
  delay(valore);
}
```

Esercizio 7 – Controllo LED RGB a catodo comune – Test colori Base (Red, Green, Blue) e Derivati (Magenta, Cyan, Yellow)

1) Disegnare lo schema elettrico
2) Disegnare lo schema di cablaggio
3) Scrivere il programma

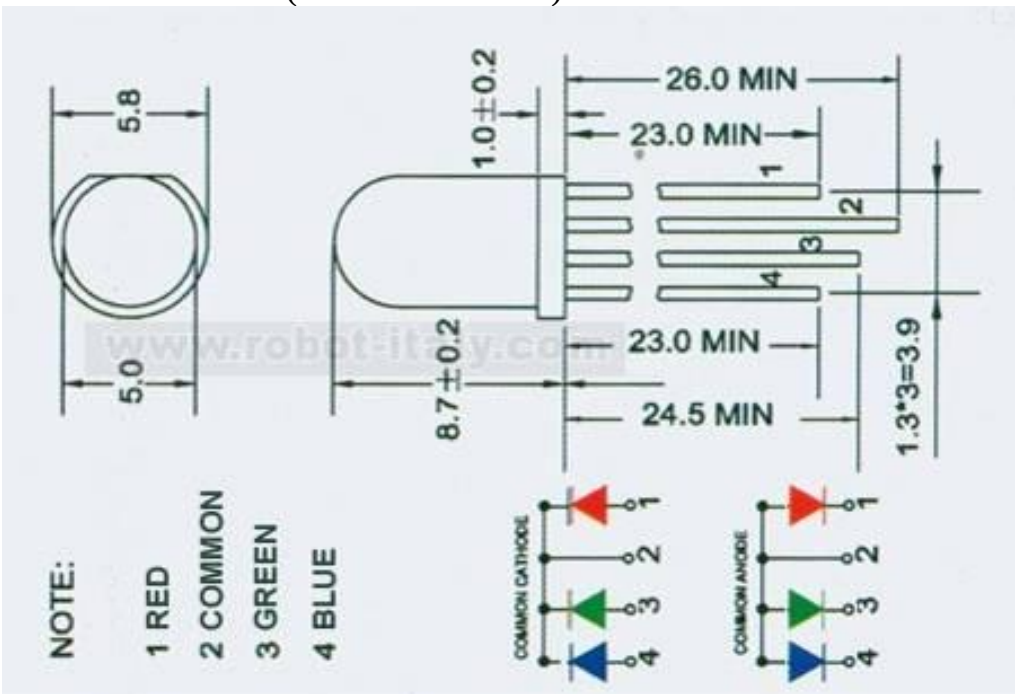
Soluzione

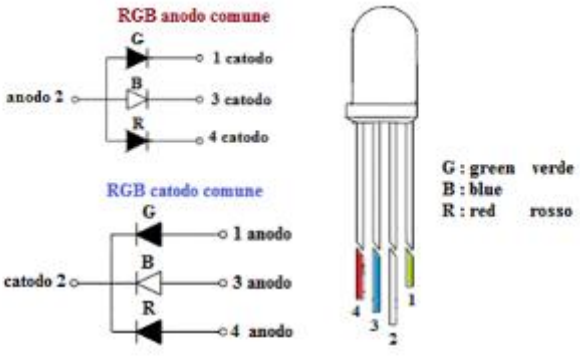
I Led RGB sono Led capaci di produrre 3 differenti lunghezze d'onda:

- Rosso (**R**, red)
- Verde (**G**, green)
- Blu (**B**, blue)

Posseggono 4 terminali e si possono presentare in due tipi di configurazione:

- RGB ad anodo comune (1 anodo e 3 catodi)
- RGB a catodo comune (1 catodo e 3 anodi)





Tensione di polarizzazione diretta:

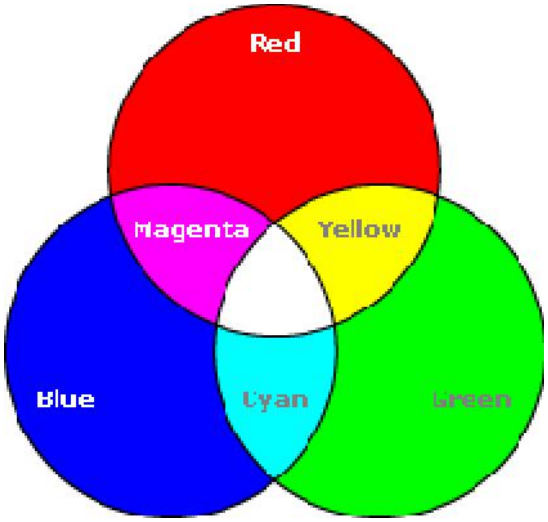
- Rosso: 2V
- Verde: 3,2V
- Blu: 3,2V

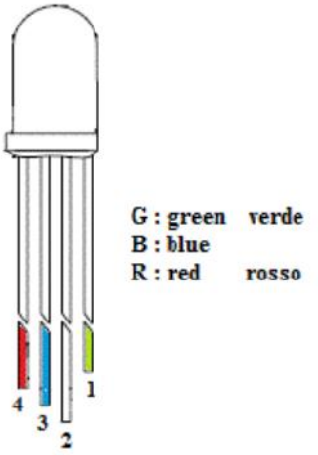
I=20 mA

Luminosità:

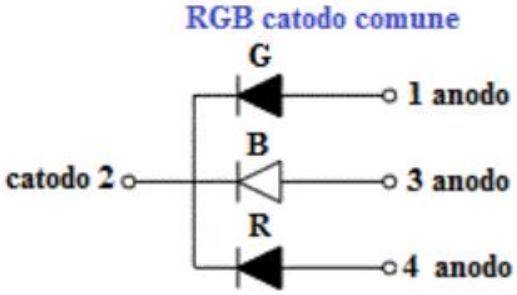
- Rosso: 2800 mcd
- Verde: 6500 mcd
- Blu: 1200 mcd

La mescolanza dei tre colori dà luogo ad una luce di un determinato colore che dipende dall'intensità di ciascuno dei tre colori originari





Negli esempi che seguiranno sono stati utilizzati dei diodi a **catodo comune**.



In serie ad ogni LED sarà inserita una resistenza che consentirà di regolare la corrente circolante nel diodo. La tensione di polarizzazione diretta a parità di corrente (I=20 mA) e diversa per ogni Led (Rosso=2V, Verde=3,2V, Blu=3,2V).
Si sceglie una resistenza di 330Ω per ogni diodo.
In Fig. 1 è riportato lo schema elettrico, in fig. 2 lo schema di cablaggio con Fritzing.

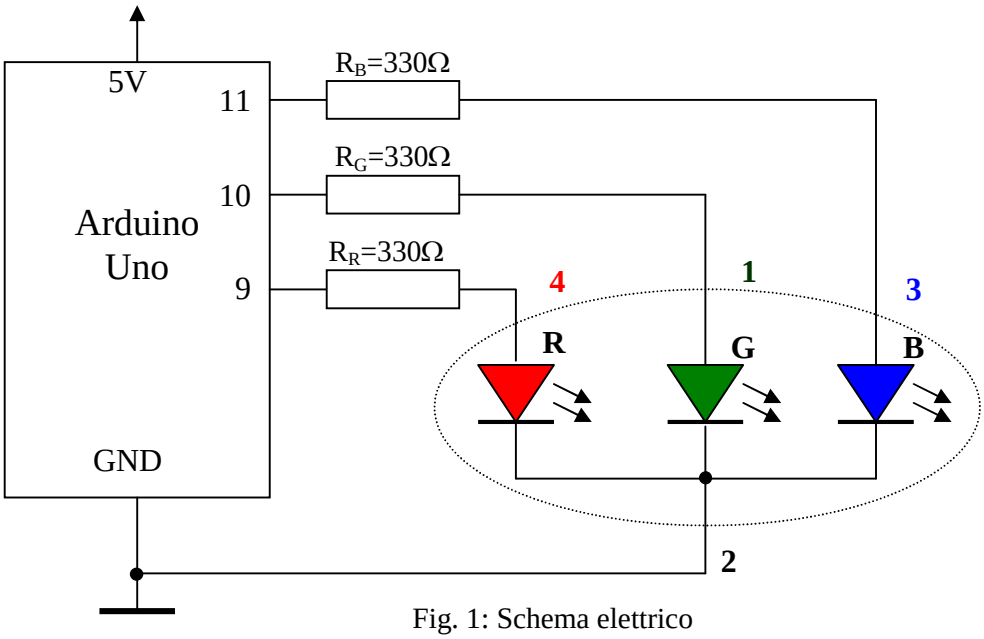


Fig. 1: Schema elettrico

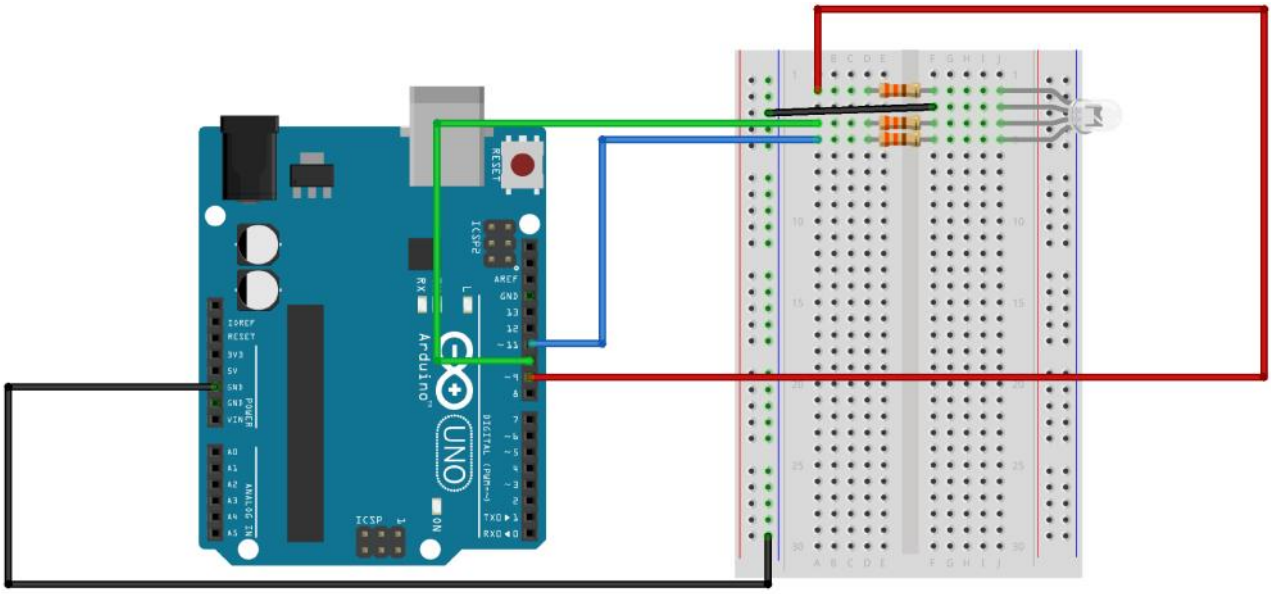
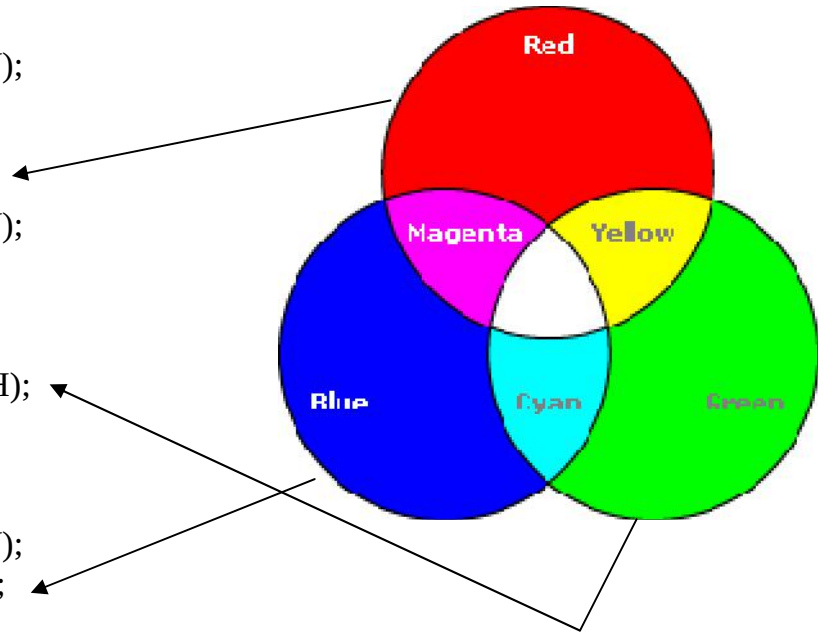


Fig. 2: Schema di cablaggio

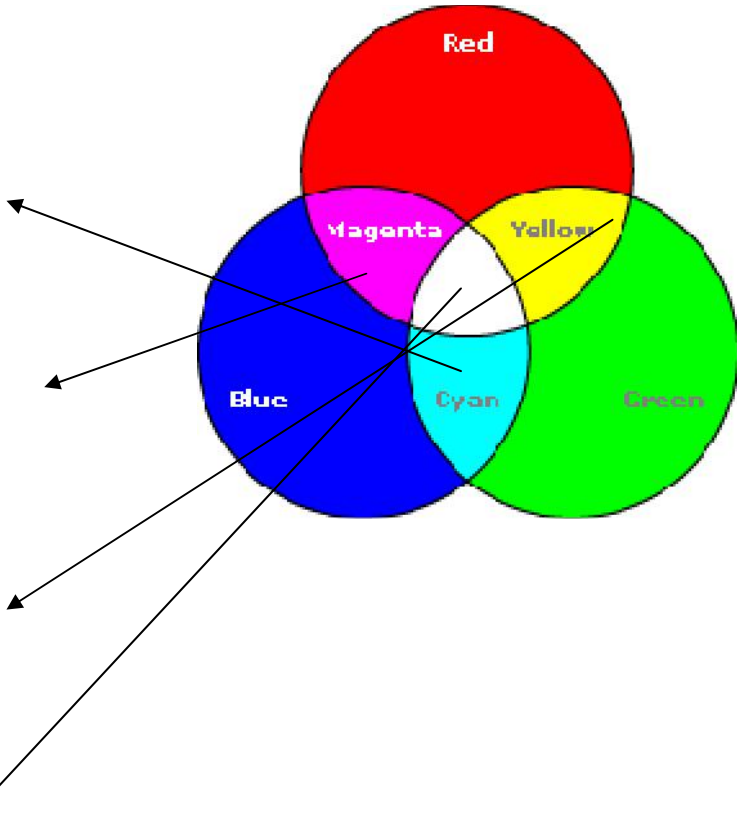
Programma -1 Test colori: Rosso(Red), Verde (Green) e Blu (Blue)

```
// Esercizio 7A – Test colori:Rosso(Red), Verde (Green) e Blu (Blue)
const int RED_PIN = 9;
const int GREEN_PIN = 10;
const int BLUE_PIN = 11;
int ritardo = 1000;  // In milliseconds

void setup()
{
pinMode(RED_PIN, OUTPUT);
pinMode(GREEN_PIN, OUTPUT);
pinMode(BLUE_PIN, OUTPUT);
}

void loop()
{
digitalWrite(RED_PIN, LOW);
digitalWrite(GREEN_PIN, LOW);
digitalWrite(BLUE_PIN, LOW);
delay(ritardo);
digitalWrite(RED_PIN, HIGH);
digitalWrite(GREEN_PIN, LOW);
digitalWrite(BLUE_PIN, LOW);
delay(ritardo);
digitalWrite(RED_PIN, LOW);
digitalWrite(GREEN_PIN, HIGH);
digitalWrite(BLUE_PIN, LOW);
delay(ritardo);
digitalWrite(RED_PIN, LOW);
digitalWrite(GREEN_PIN, LOW);
digitalWrite(BLUE_PIN, HIGH);
delay(ritardo);
}
```



Programma -2 Test colori: Cyan(Blue+Green), Magenta (Red+Blue)
Yellow (Red + Green), White (Red+Green+Blue)

```
// Esercizio 7B – Test Led RGBcolori:
// Cyan(Blue+Green), Magenta (Red+Blue), Yellow (Red +Green), White (Red+Green+Blue)

const int RED_PIN = 9;
const int GREEN_PIN = 10;
const int BLUE_PIN = 11;
int ritardo = 1000;  // In milliseconds

void setup()
{
pinMode(RED_PIN, OUTPUT);
pinMode(GREEN_PIN, OUTPUT);
pinMode(BLUE_PIN, OUTPUT);
}

void loop()
{
digitalWrite(RED_PIN, LOW);
digitalWrite(GREEN_PIN, LOW);
digitalWrite(BLUE_PIN, LOW);
delay(ritardo);

// Cyan (Green e Blue=High, Red=Low):
digitalWrite(RED_PIN, LOW);
digitalWrite(GREEN_PIN, HIGH);
digitalWrite(BLUE_PIN, HIGH);
delay(ritardo);

// Magenta (Red e Blue=High, Green=Low):
digitalWrite(RED_PIN, HIGH);
digitalWrite(GREEN_PIN, LOW);
digitalWrite(BLUE_PIN, HIGH);
delay(ritardo);

// Yellow (Red e Green=High, Blue=Low):
digitalWrite(RED_PIN, HIGH);
digitalWrite(GREEN_PIN, HIGH);
digitalWrite(BLUE_PIN, LOW);
delay(ritardo);

// White (Read, Green e Blue=high):
digitalWrite(RED_PIN, HIGH);
digitalWrite(GREEN_PIN, HIGH);
digitalWrite(BLUE_PIN, HIGH);
delay(1000);
}
```



Esercizio 8 – Controllo LED RGB a catodo comune

a) **Test colori 16 M**
b) **Test Colori Random**

1) Disegnare lo schema elettrico
2) Disegnare lo schema di cablaggio
3) Scrivere il programma

Soluzione

In Fig. 1 è riportato lo schema elettrico, in fig. 2 lo schema di cablaggio con Fritzing.

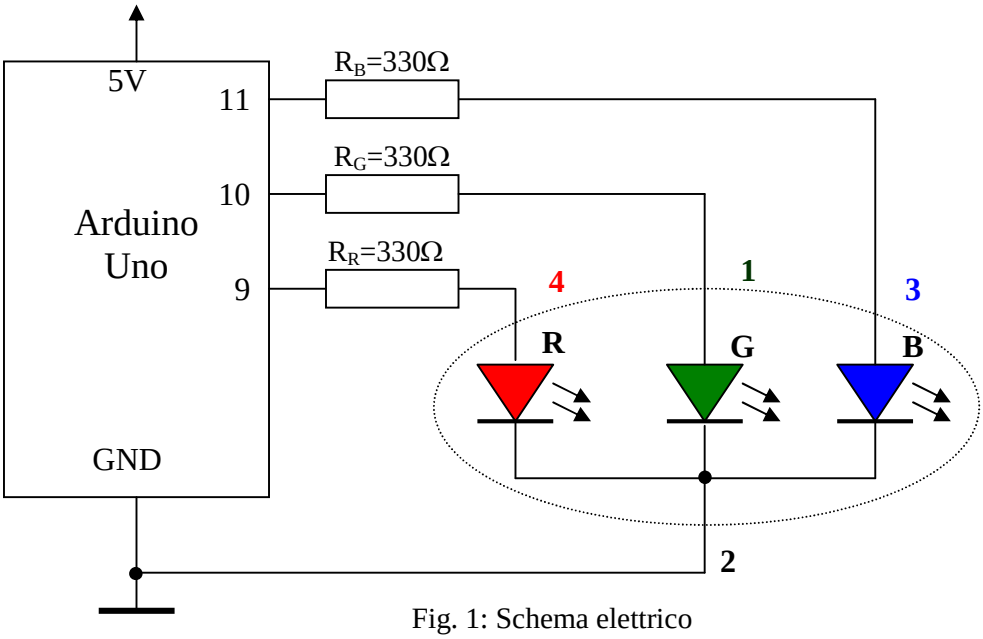


Fig. 1: Schema elettrico

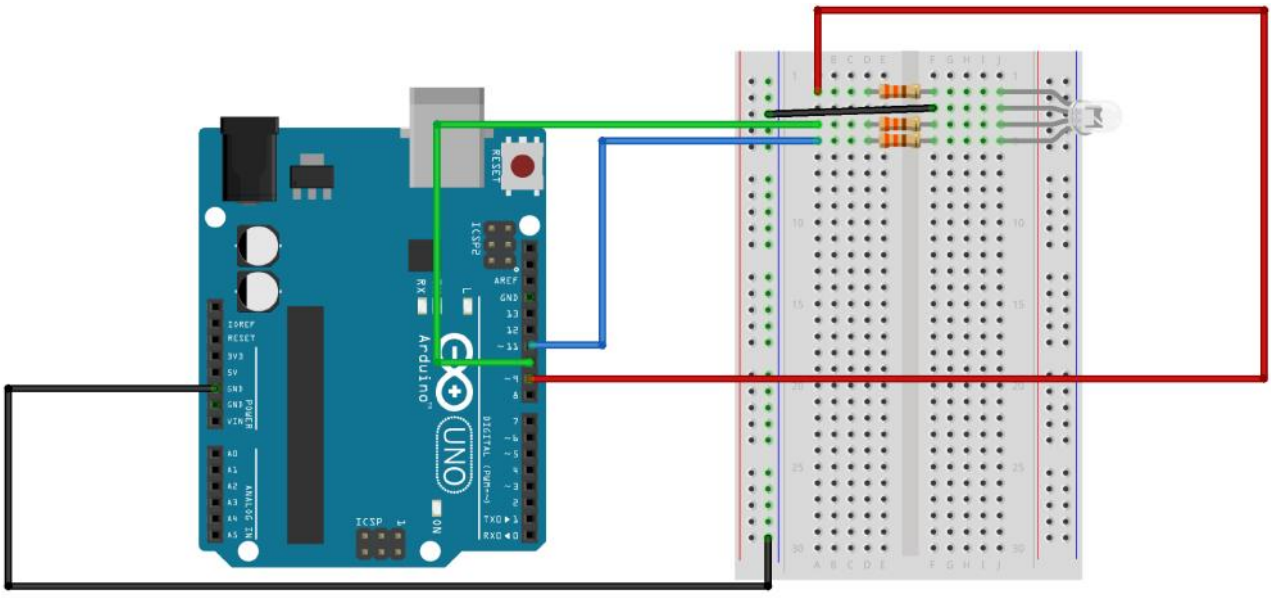


Fig. 2: Schema di cablaggio

Programma -1 Test colori: 16 M

```
// Esercizio 8A – Test colori:16 M
const int RED_PIN = 9;
const int GREEN_PIN=10;
const int BLUE_PIN = 11;
int ritardo = 100;  // In millisecondi
int r=0;
int g=0;
int b=0;
void setup()
{
pinMode(RED_PIN, OUTPUT);
pinMode(GREEN_PIN, OUTPUT);
pinMode(BLUE_PIN, OUTPUT);
}
void loop()
{
for (r=0;r<=255;r++)
{
for (g=0;g<=255;g++)
{
for (b=0;b<=255;b++)
{
analogWrite(RED_PIN, r);
analogWrite(GREEN_PIN, g);
analogWrite(BLUE_PIN, b);
delay(ritardo);
}
}
}
}
```

Programma -2 Test colori random

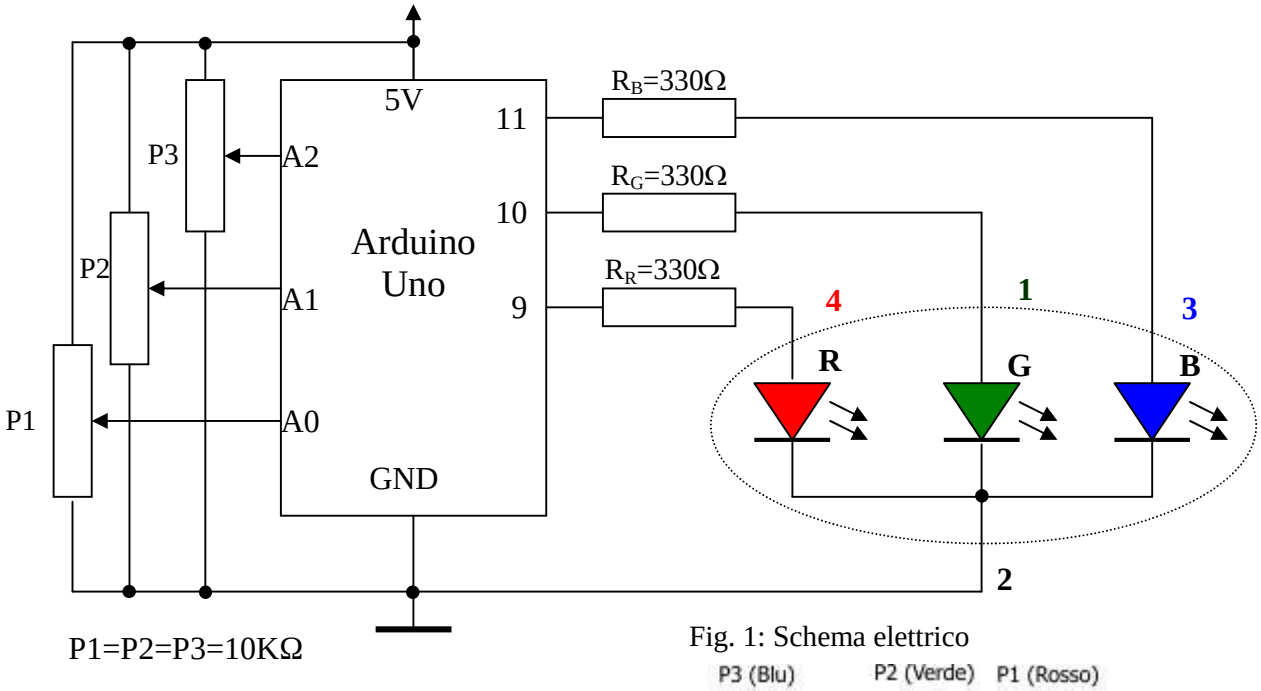
```
// Esercizio 8B – Test colori random
const int RED_PIN = 9;
const int GREEN_PIN=10;
const int BLUE_PIN = 11;
int ritardo = 1000;  // In millisecondi

void setup()
{
pinMode(RED_PIN, OUTPUT);
pinMode(GREEN_PIN, OUTPUT);
pinMode(BLUE_PIN, OUTPUT);
}
void loop()
{
analogWrite(RED_PIN, random(256));
analogWrite(GREEN_PIN, random(256));
analogWrite(BLUE_PIN, random(256));
delay(ritardo);
}
```

Esercizio 9– Controllo LED RGB a catodo comune con tre potenziometri

1) Disegnare lo schema elettrico
2) Disegnare lo schema di cablaggio
3) Scrivere il programma

Soluzione

In Fig. 1 è riportato lo schema elettrico, in fig. 2 lo schema di cablaggio con Fritzing.



P1=P2=P3=10KΩ

Fig. 1: Schema elettrico

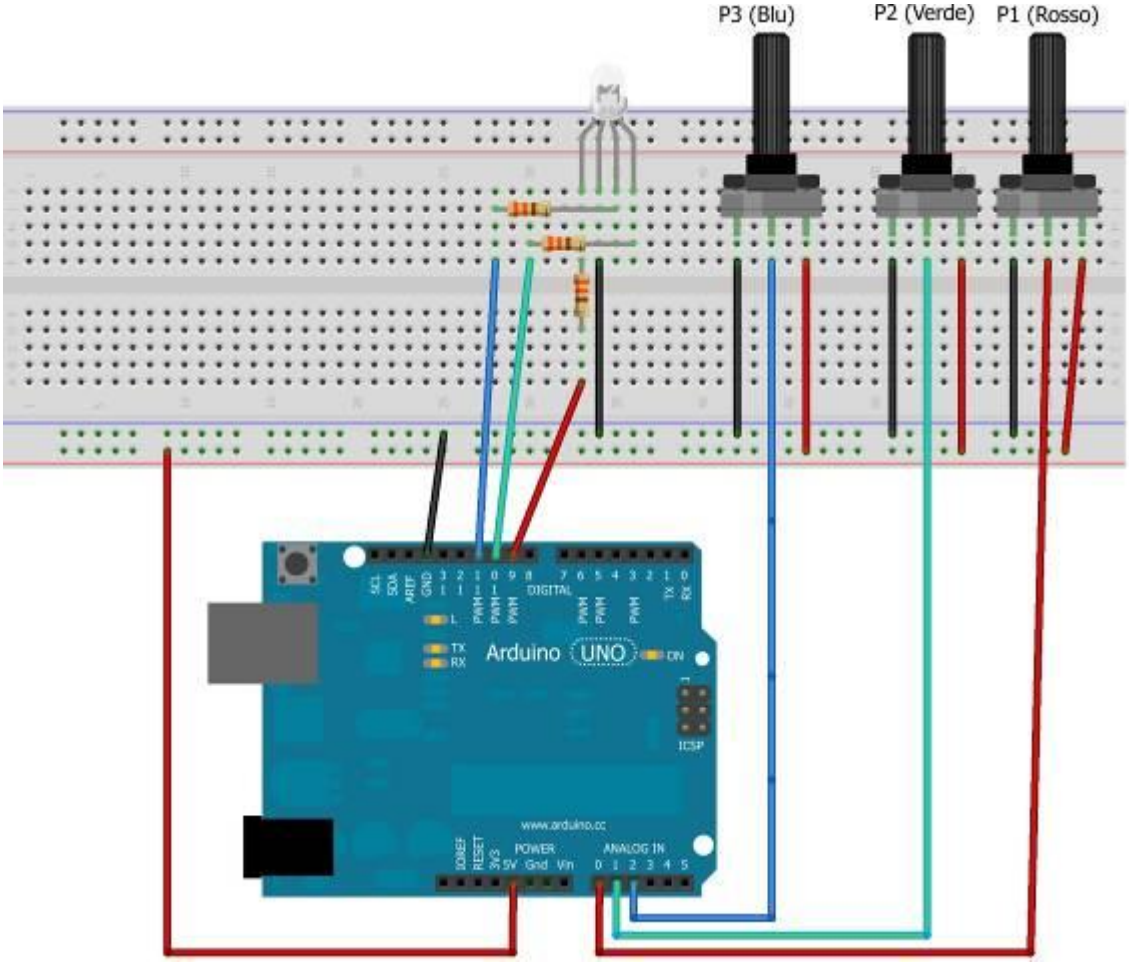


Fig. 2: Cablaggio con Fritzing

Programma -1 Controllo LED RGB a catodo comune con tre potenziometri

```
// Esercizio 9 – Controllo LED RGB a catodo comune con tre potenziometri
const int RED_PIN = 9;
const int GREEN_PIN=10;
const int BLUE_PIN = 11;
int pot1= A0;// seleziona il pin per il potenziometro 1
int pot2= A1;// seleziona il pin per il potenziometro 2
int pot3= A2;// seleziona il pin per il potenziometro 3
// inizializzazione delle variabile per memorizzare i valori letti dai tre potenziometri
int vp1 = 0;
int vp2 = 0;
int vp3 = 0;

void setup()
{
pinMode(RED_PIN, OUTPUT);
pinMode(GREEN_PIN, OUTPUT);
pinMode(BLUE_PIN, OUTPUT);
}

void loop()
{
vp1 = analogRead(pot1);    // lettura posizione potenziometro 1
vp2 = analogRead(pot2);    // lettura posizione potenziometro 2
vp3 = analogRead(pot3);    // lettura posizione potenziometro 3
analogWrite(RED_PIN, vp1/4);
analogWrite(GREEN_PIN, vp2/4);
analogWrite(BLUE_PIN, vp3/4);
}
```

Esercizio 10 – Controllo 8 Led

1) Disegnare lo schema elettrico
2) Disegnare lo schema di cablaggio
3) Scrivere il programma

Soluzione

In Fig. 1 è riportato lo schema elettrico, in fig. 2 lo schema di cablaggio con Fritzing.

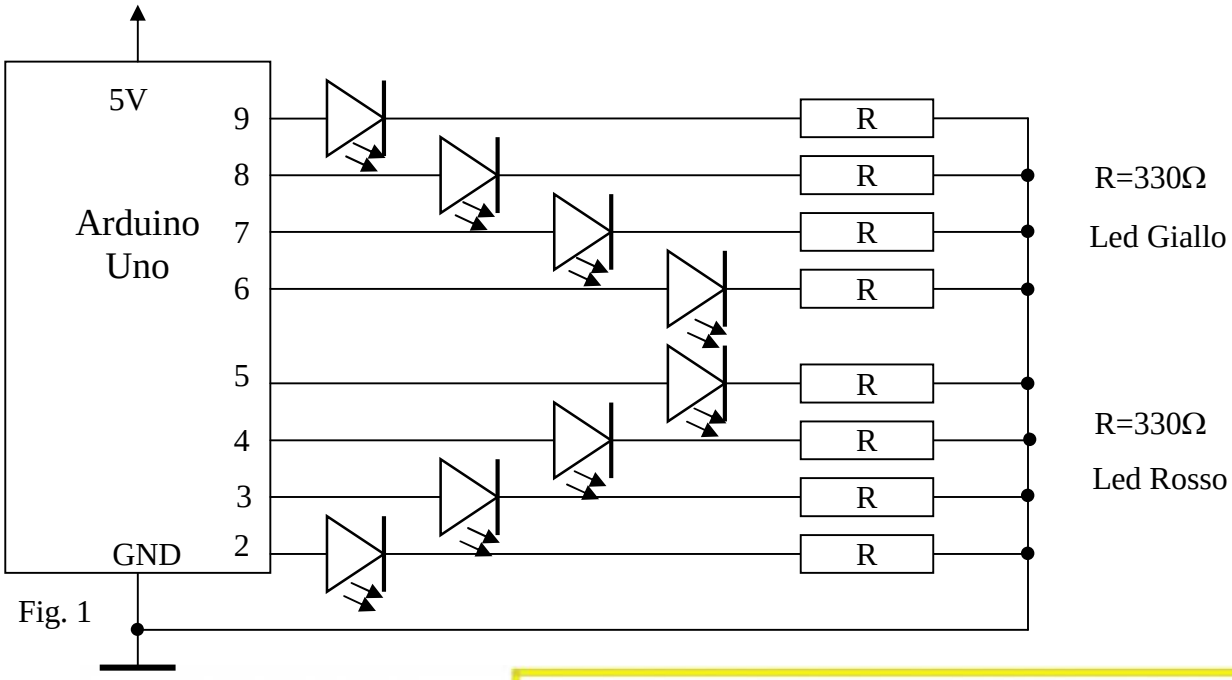


Fig. 1

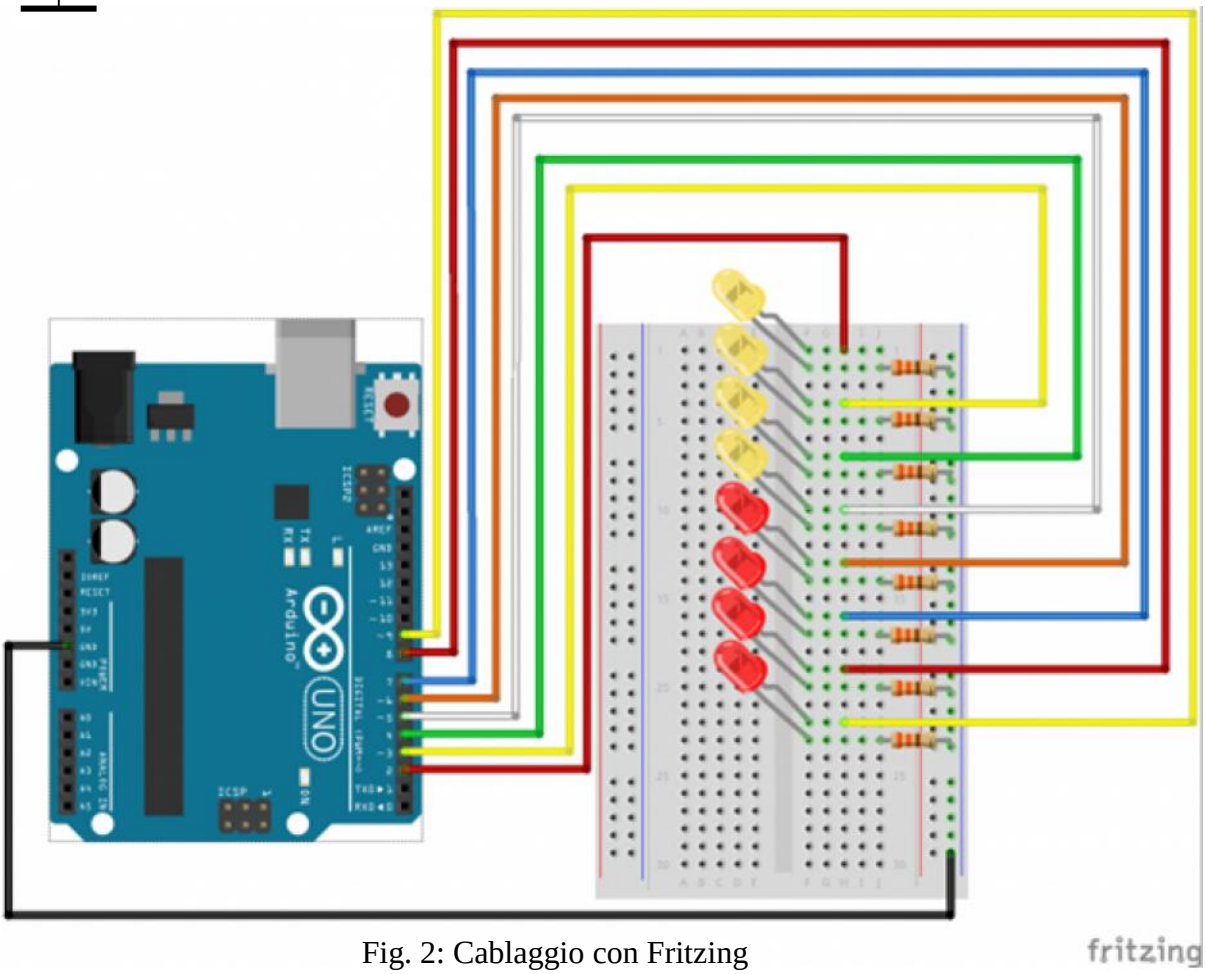


Fig. 2: Cablaggio con Fritzing

Programma -1 Controllo 8 LED

```
// Esercizio 10 – Controllo 8 LED
int ledPins[] = {2,3,4,5,6,7,8,9};

void setup()
{
  int index;
  for(index = 0; index <= 7; index++)
  {
    pinMode(ledPins[index],OUTPUT);
  }
}

void loop()
{
  oneAfterAnotherNoLoop();  // Light up all the LEDs in turn
}
void oneAfterAnotherNoLoop()
{
  int delayTime = 100;
  digitalWrite(ledPins[0], HIGH);  //Turns on LED #0 (pin 2)
  delay(delayTime);
  digitalWrite(ledPins[1], HIGH);  //Turns on LED #1 (pin 3)
  delay(delayTime);
  digitalWrite(ledPins[2], HIGH);  //Turns on LED #2 (pin 4)
  delay(delayTime);
  digitalWrite(ledPins[3], HIGH);  //Turns on LED #3 (pin 5)
  delay(delayTime
  digitalWrite(ledPins[4], HIGH);  //Turns on LED #4 (pin 6)
  delay(delayTime
  digitalWrite(ledPins[5], HIGH);  //Turns on LED #5 (pin 7)
  delay(delayTime);
  digitalWrite(ledPins[6], HIGH);  //Turns on LED #6 (pin 8)
  delay(delayTime
  digitalWrite(ledPins[7], HIGH);  //Turns on LED #7 (pin 9)
  delay(delayTime);
  digitalWrite(ledPins[7], LOW);   //Turn off LED #7 (pin 9)
  delay(delayTime);
  digitalWrite(ledPins[6], LOW);   //Turn off LED #6 (pin 8)
  delay(delayTime);
  digitalWrite(ledPins[5], LOW);   //Turn off LED #5 (pin 7)
  delay(delayTime);              //wait delayTime milliseconds
  digitalWrite(ledPins[4], LOW);   //Turn off LED #4 (pin 6)
  delay(delayTime);              //wait delayTime milliseconds
  digitalWrite(ledPins[3], LOW);   //Turn off LED #3 (pin 5)
  delay(delayTime);              //wait delayTime milliseconds
  digitalWrite(ledPins[2], LOW);   //Turn off LED #2 (pin 4)
```

```
  delay(delayTime);              //wait delayTime milliseconds
  digitalWrite(ledPins[1], LOW);   //Turn off LED #1 (pin 3)
  delay(delayTime);              //wait delayTime milliseconds
  digitalWrite(ledPins[0], LOW);   //Turn off LED #0 (pin 2)
  delay(delayTime);              //wait delayTime milliseconds
}

void oneAfterAnotherLoop()
{
  int index;
  int delayTime = 100; // milliseconds to pause between LEDs
  for(index = 0; index <= 7; index++)
  {
    digitalWrite(ledPins[index], HIGH);
    delay(delayTime);
  }

  for(index = 7; index >= 0; index--)
  {
    digitalWrite(ledPins[index], LOW);
    delay(delayTime);
  }
}

void oneOnAtATime()
{
  int index;
  int delayTime = 100; // milliseconds to pause between LEDs
  for(index = 0; index <= 7; index++)
  {
    digitalWrite(ledPins[index], HIGH);  // turn LED on
    delay(delayTime);                    // pause to slow down
    digitalWrite(ledPins[index], LOW);   // turn LED off
  }
}

void pingPong()
{
  int index;
  int delayTime = 100; // milliseconds to pause between LEDs
  for(index = 0; index <= 7; index++)
  {
    digitalWrite(ledPins[index], HIGH);  // turn LED on
    delay(delayTime);                    // pause to slow down
    digitalWrite(ledPins[index], LOW);   // turn LED off
  }
```

```
  for(index = 7; index >= 0; index--)
  {
    digitalWrite(ledPins[index], HIGH);  // turn LED on
    delay(delayTime);                    // pause to slow down
    digitalWrite(ledPins[index], LOW);   // turn LED off
  }
}
void marquee()
{
  int index;
  int delayTime = 200; // milliseconds to pause between LEDs
  for(index = 0; index <= 3; index++) // Step from 0 to 3
  {
    digitalWrite(ledPins[index], HIGH);    // Turn a LED on
    digitalWrite(ledPins[index+4], HIGH);  // Skip four, and turn that LED on
    delay(delayTime);                      // Pause to slow down the sequence
    digitalWrite(ledPins[index], LOW);     // Turn the LED off
    digitalWrite(ledPins[index+4], LOW);   // Skip four, and turn that LED off
  }
}
void randomLED()
{
  int index;
  int delayTime;
  index = random(8);    // pick a random number between 0 and 7
  delayTime = 100;
  digitalWrite(ledPins[index], HIGH);  // turn LED on
  delay(delayTime);
  digitalWrite(ledPins[index], LOW);   // turn LED off
}
```

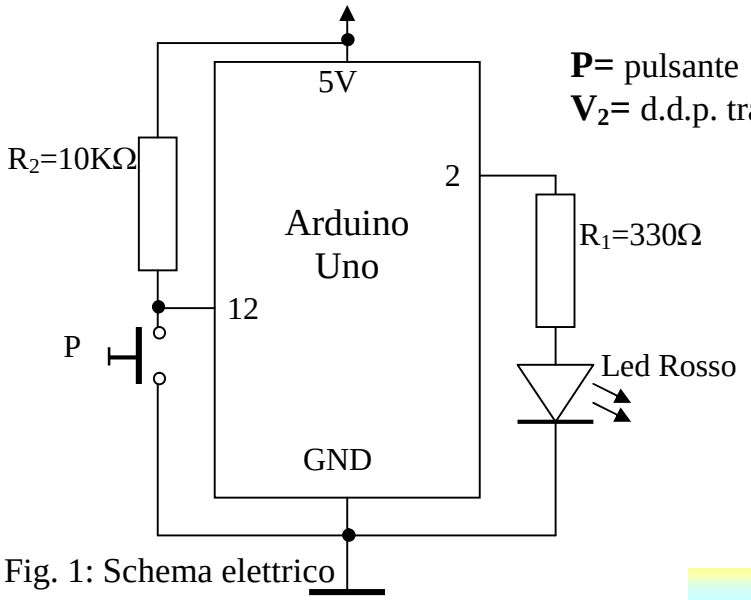
Esercizio 11 – Accensione Led con un Pulsante: PL1

Realizzare un programma che permette di accendere un Led quando premiamo un pulsante e di spegnerlo quando viene rilasciato.

1) Disegnare lo schema elettrico
2) Disegnare lo schema di cablaggio
3) Scrivere il programma

Soluzione

Fig. 1 è riportato lo schema elettrico, in fig. 2 lo schema di cablaggio con Fritzing.



P= pulsante collegato con resistenza di pull-up
V_2= d.d.p. tra il Pin 2 e GND

P=Off → V_2=5V (High)

P=On → V_2=0V (Low)

Fig. 1: Schema elettrico

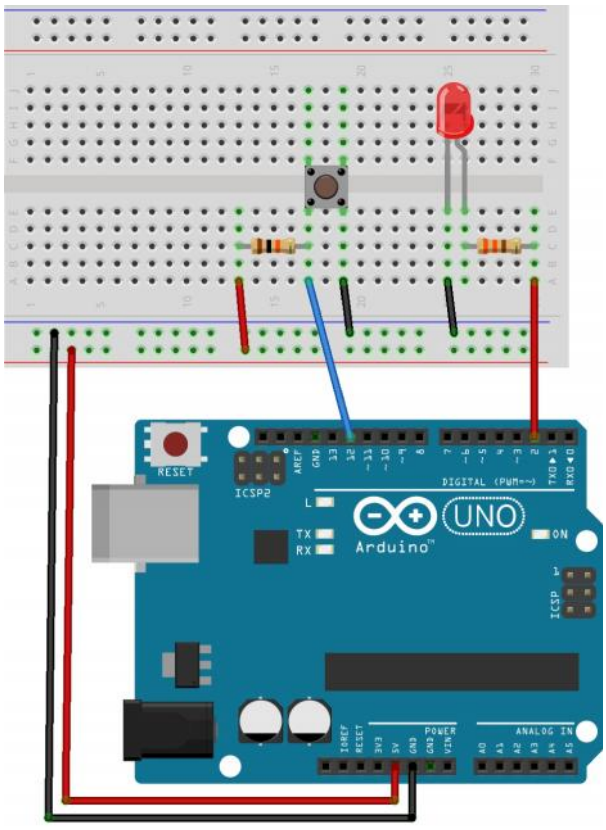


Fig. 2: Cablaggio con Fritzing

Programma

```
// Esercizio 11 – Pulsante-LED-PL1
int led=2;
int pulsante=12;
int v2=0;

void setup()
{
pinMode(led, OUTPUT);
pinMode(pulsante, INPUT);
}

void loop()
{
   v2=digitalRead(pulsante);
   if(v2==LOW)
     {
       digitalWrite(led, HIGH);
     }
  else
    {
      digitalWrite(led, LOW);
    }
}
```

Esercizio 12 – Accensione e Spegnimento Led con un Pulsante: PL2

Realizzare un programma che permette di accendere un Led quando premiamo un pulsante e quando viene nuovamente premuto il pulsante spegne il Led, comportamento analogo a quello che si ha per un impianto di illuminazione.

1) Disegnare lo schema elettrico
2) Disegnare lo schema di cablaggio
3) Scrivere il programma

Soluzione

Fig. 1 è riportato lo schema elettrico, in fig. 2 lo schema di cablaggio con Fritzing.

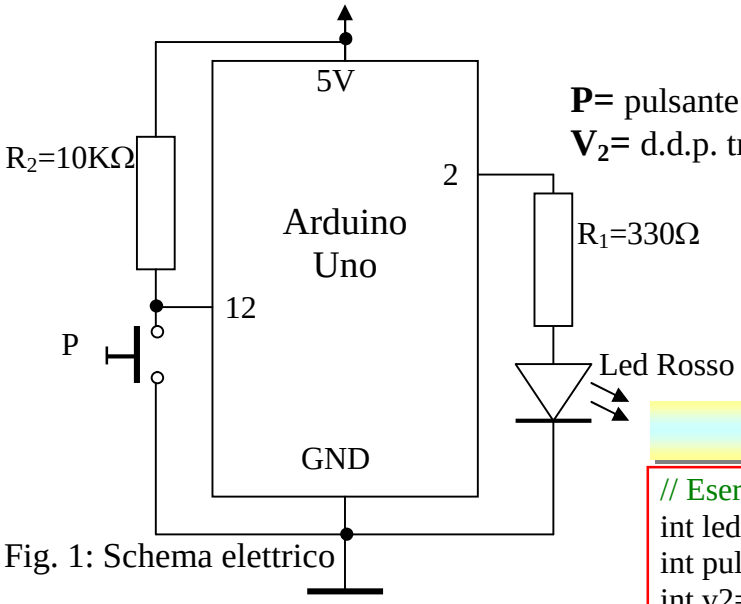


Fig. 1: Schema elettrico

P= pulsante collegato con resistenza di pull-up
V_2= d.d.p. tra il Pin 2 e GND

P=Off → V_2=5V (High)

P=On → V_2=0V (Low)

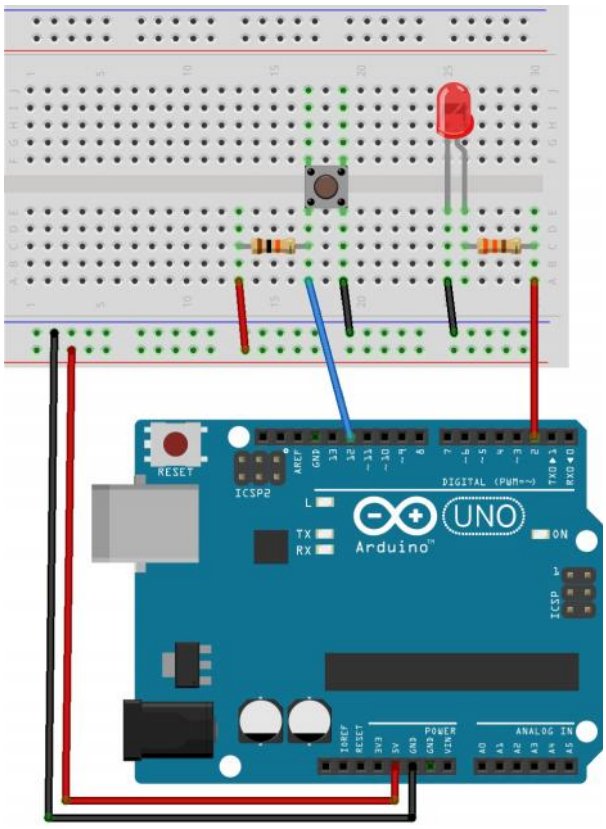


Fig. 2: Cablaggio con Fritzing

Programma

```
// Esercizio 12 – Pulsante-LED-PL2
int led=2;
int pulsante=12;
int v2=0;
int stato=0;

void setup()
{
pinMode(led, OUTPUT);
pinMode(pulsante, INPUT);
}
void loop()
{
   v2=digitalRead(pulsante);
   if(v2==LOW)
      {
        stato=1-stato;
      }

   if(stato==1)
     {
        digitalWrite(led, HIGH);
     }
  else
     {
        digitalWrite(led, LOW);
     }
}
```

Esercizio 13 – Controllo 4 pulsanti e 4 LED con libreria antirimbalzo: PL3

Realizzare un programma in grado di gestire quattro pulsanti e quattro LED con la libreria antirimbalzo. Ogni pulsante deve controllare l'accensione e lo spegnimento del rispettivo LED. (Pulsante1, on/off LED1; Pulsante2, on/off LED2, ….).

1) Disegnare lo schema elettrico
2) Disegnare lo schema di cablaggio
3) Scrivere il programma

Soluzione

Il circuito anti-rimbalzo viene utilizzato in elettronica per produrre un solo impulso stabile in presenza di un ingresso che risulti elettricamente rumoroso.
L'origine dei suddetti disturbi in presenza di pulsanti o interruttori è dovuta al fatto che la lamina di contatto non garantisce, una volta commutata, un impulso immediatamente stabile. Fig. 1 è riportato lo schema elettrico.

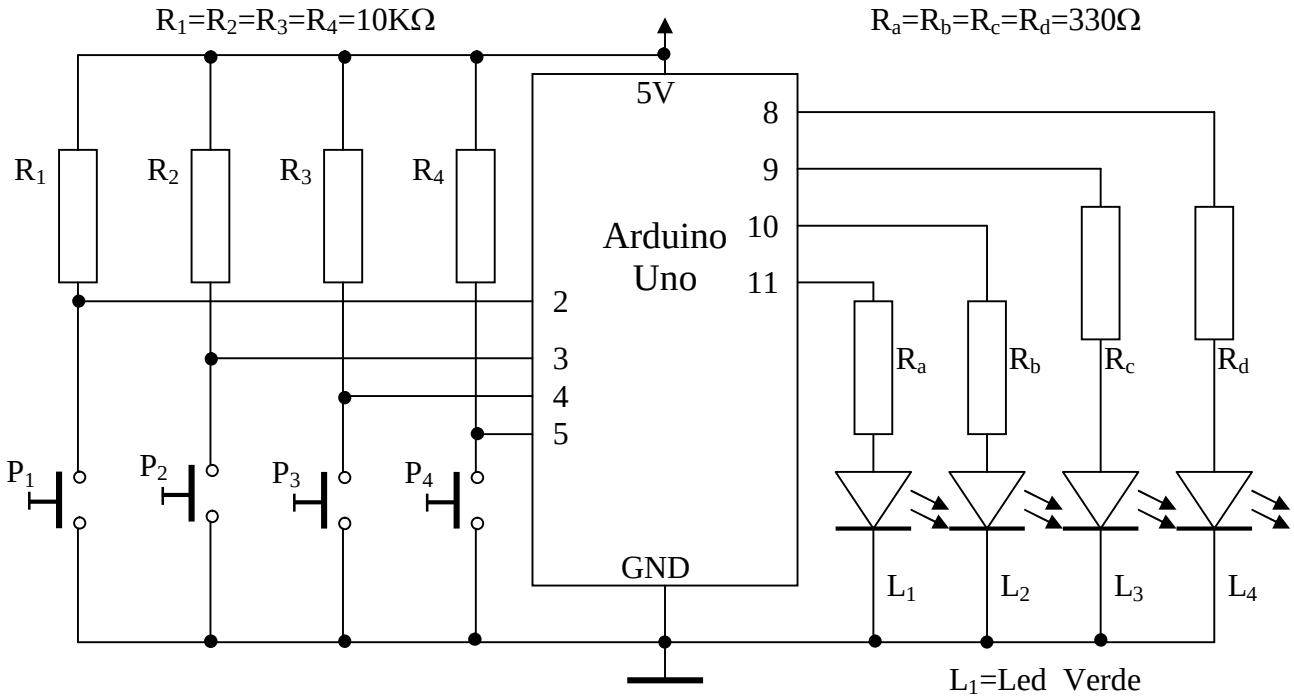


Fig.1

Download e installazione libreria

1) www.ipsia.gov.it/Documenti/Arduino/Librerie/TriggerButton1.2.zip
2) Scompattare il file "**TriggerButton1.2.zip**"
3) Copiare la cartella "**Trigger Button**" nella cartella "**libraries**" di Arduino
4) Inserire il comando **#include <TriggerButton.h>** all'inizio dello Sketch

Programma

```
// Esercizio 13: Controllo 4 pulsanti e 4 LED con libreria antirimbalzo: PL3

#include <TriggerButton.h>
TriggerButton pulsante1, pulsante2, pulsante3, pulsante4;
int ledverde  = 8;
int ledgiallo = 9;
int ledrosso  = 10;
int ledblu    = 11;
int tempo=70; // tempo antirimbalzo 70 ms
void setup()
{
  // Impostazione pulsanti sui pin 2, 3, 4 e 5 (attivi bassi)
  pulsante1.setUp(2, LOW);
  pulsante2.setUp(3, LOW);
  pulsante3.setUp(4, LOW);
  pulsante4.setUp(5, LOW);

  pinMode(ledverde,  OUTPUT);
  pinMode(ledgiallo, OUTPUT);
  pinMode(ledrosso,  OUTPUT);
  pinMode(ledblu,    OUTPUT);
}

void loop()
{
  if (pulsante1.Click(tempo))
  {
   digitalWrite(ledverde, !digitalRead(ledverde));
  }
  if (pulsante2.Click(tempo))
  {
   digitalWrite(ledgiallo, !digitalRead(ledgiallo));
  }
if (pulsante3.Click(tempo))
  {
   digitalWrite(ledrosso, !digitalRead(ledrosso));
  }
    if ( pulsante4.Click(tempo))
  {
   digitalWrite(ledblu, !digitalRead(ledblu));
  }
}
```

Esercizio 14 – Controllo luminosità LED tramite pulsanti: PL4

Realizzare un programma in grado di controllare la luminosità di due LED tramite quattro pulsanti. Con un pulsante si deve incrementare la luminosità con un pulsante si deve diminuire la luminosità.

4) Disegnare lo schema elettrico
5) Disegnare lo schema di cablaggio
6) Scrivere il programma

Soluzione

In Fig. 1 è riportato lo schema elettrico del circuito

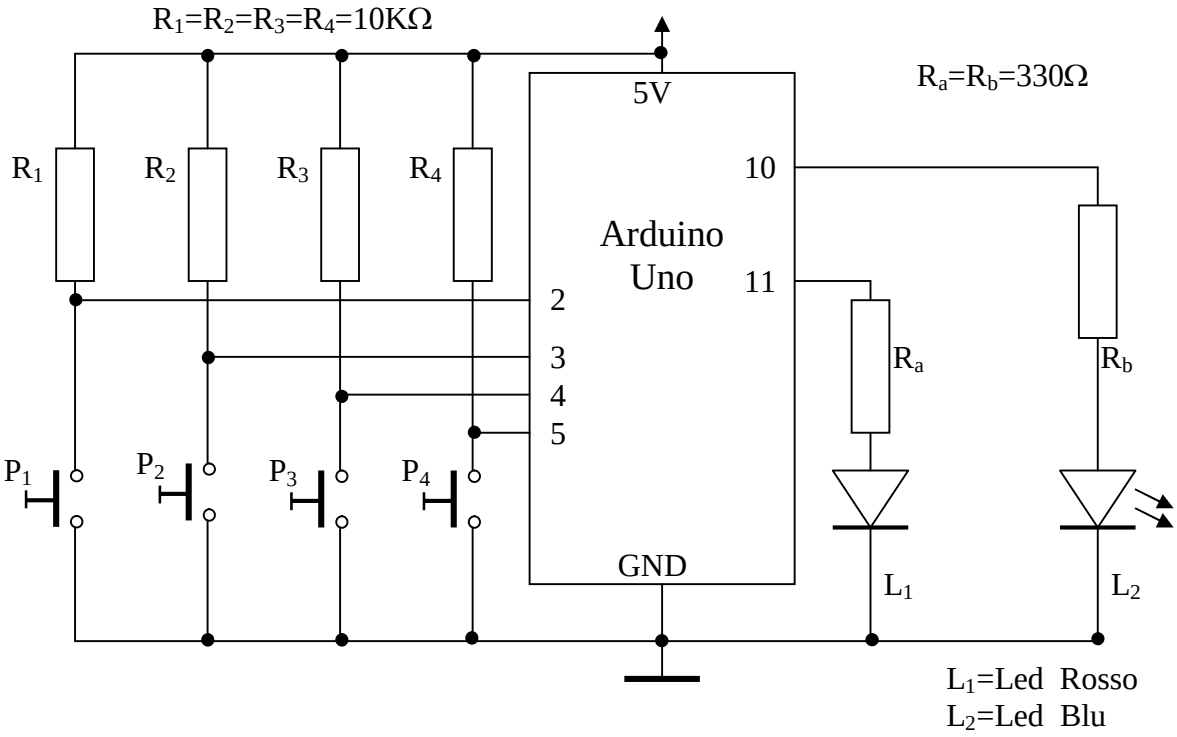


Fig.1

P_1 e P_2 Controllo L_1
P_3 e P_4 Controllo L_2

Programma

```
// Esercizio 14: Controllo luminosità LED tramite pulsanti con libreria antirimbalzo: PL4
#include <TriggerButton.h>
TriggerButton pulsante1, pulsante2, pulsante3, pulsante4;
int ledrosso  = 11;
int ledblu    = 10;
int tempo=70; // tempo antirimbalzo 70 ms
int t=0;
int k=0;
void setup(){
  pulsante1.setUp(2, LOW);
  pulsante2.setUp(3, LOW);
  pulsante3.setUp(4, LOW);
  pulsante4.setUp(5, LOW);
  pinMode(ledrosso,  OUTPUT);
  pinMode(ledblu,    OUTPUT);
}
void loop(){
if (pulsante1.Click(tempo))
{
   analogWrite(ledrosso,t);
   t=t+10;
   if (t>255) { t=0;
      }
}
  if (pulsante2.Click(tempo)){
   analogWrite(ledrosso,t);
   t=t-10;
   if (t<0){t=0;
      }
  }
if (pulsante3.Click(tempo)){
   analogWrite(ledblu,k);
   k=k+10;
   if (k>255){k=0;
      }
   }
  if (pulsante4.Click(tempo)){
   analogWrite(ledblu,k);
   k=k-10;
   if (k<0){k=0;
      }
}
}
```

Esercizio 15 – Controllo Start – Stop (Avvio/Arresto): PL5

Realizzare un programma in grado di accendere/spegnere due LED (Start/Stop) tramite due pulsanti. Un pulsante per accendere /Start) un pulsante per spegnere (Stop).
luminosità.

7) Disegnare lo schema elettrico
8) Disegnare lo schema di cablaggio
9) Scrivere il programma

Soluzione

In Fig. 1 è riportato lo schema elettrico del circuito

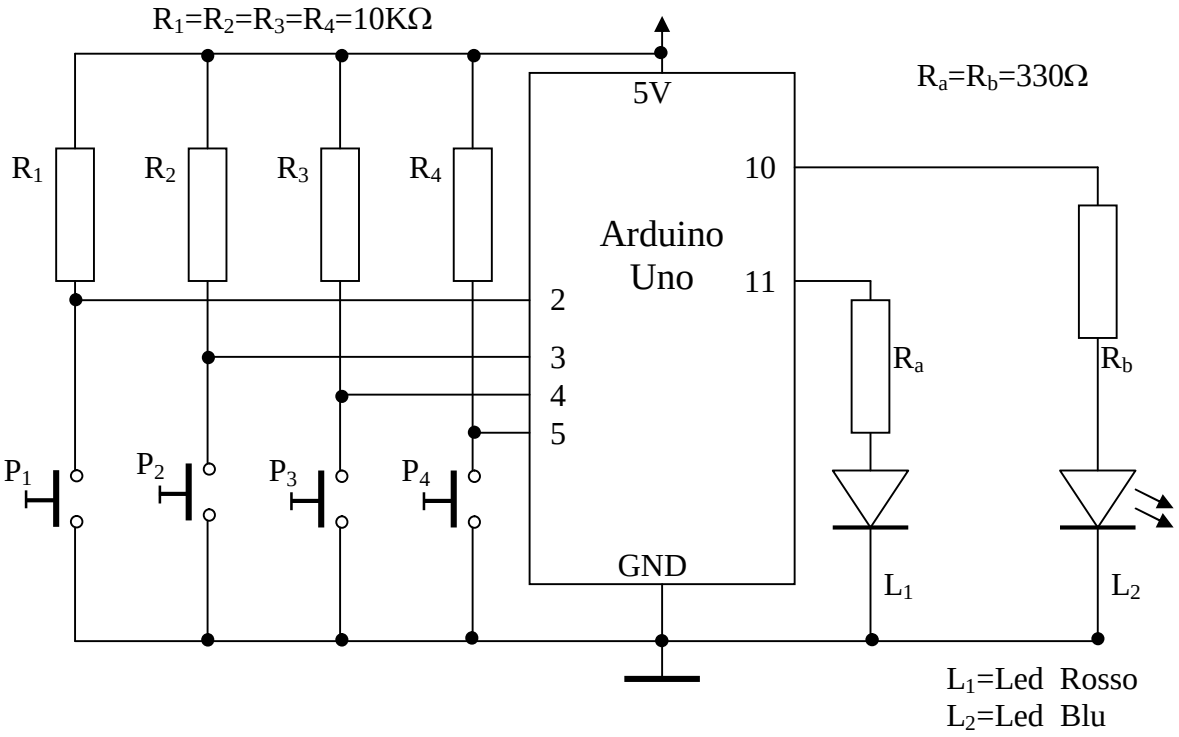


Fig.1

Tabella di funzionamento					
P_1	P_2	Led Rosso (L_1)	P_3	P_4	Led Blu (L_2)
ON		Acceso	ON		Acceso
	ON	Spento		ON	Spento

Programma

```
// Esercizio 15: Controllo Start – Stop (Avvio/Arresto) con libreria antirimbalzo: PL5
#include <TriggerButton.h>
TriggerButton pulsante1, pulsante2, pulsante3, pulsante4;
int ledrosso  = 11;
int ledblu    = 10;
int tempo=70; // tempo antirimbalzo 70 ms
void setup(){
  pulsante1.setUp(2, LOW);
  pulsante2.setUp(3, LOW);
  pulsante3.setUp(4, LOW);
  pulsante4.setUp(5, LOW);
  pinMode(ledrosso,  OUTPUT);
  pinMode(ledblu,    OUTPUT);
}
void loop(){
if (pulsante1.Click(tempo))
{
   digitalWrite(ledrosso,HIGH);
}
  if (pulsante2.Click(tempo))
  {
   digitalWrite(ledrosso,LOW);
  }
if (pulsante3.Click(tempo))
   {
   digitalWrite(ledblu,HIGH);
   }
  if (pulsante4.Click(tempo))
  {
   digitalWrite(ledblu,LOW);
  }
}
```

Esercizio 16 – Controlli di Flusso – Strutture di Controllo

- *Struttura selezione semplice* (if…)
- *Struttura selezione doppia (if…else)*
- *Struttura selezione tripla (if…else if … else)*
- *Struttura iterazione enumerativa (for…)*

Assegnato il circuito di figura 1

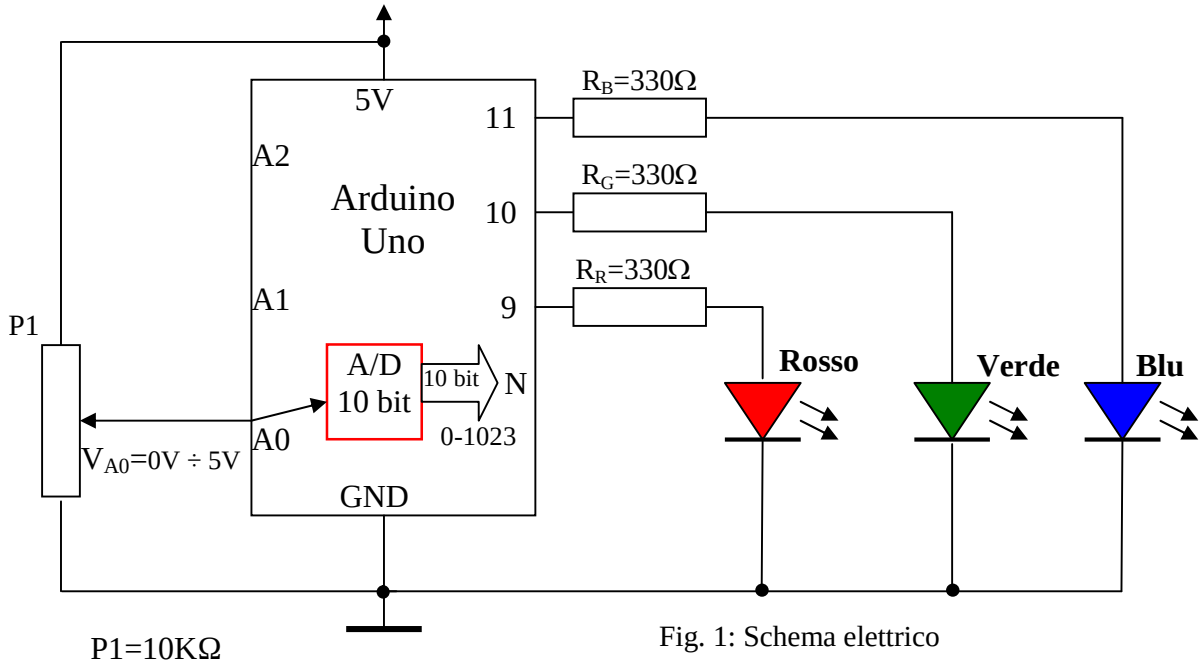


Fig. 1: Schema elettrico

Indicato con N il numero fornito dal convertitore (0 ÷ 1023), disegnare il Flow Chart e scrivere il programma in grado di rispettare le specifiche indicate nell'elenco seguente.

16.1)
- Accendere il LED rosso se N>=500

Struttura selezione semplice (if…)

16.2)
- Accendere il LED rosso se N>=400
- Accendere il LED verde se N< 400

Struttura selezione doppia (if…else)

16.3)
- Accendere il LED rosso se N<=400
- Accendere il LED verde se N>=700
- Accendere il LED blu se 400<N<700

Struttura selezione tripla (if…else if … else)

16.4)
- Accendere il LED verde a luce piena se 500<N<700
- Accendere il LED verde a metà potenza se 70=<N<=1023
- Se N<500 LED Verde spento, LED Rosso acceso e LED Blu acceso

Struttura selezione tripla (if…else if … else)

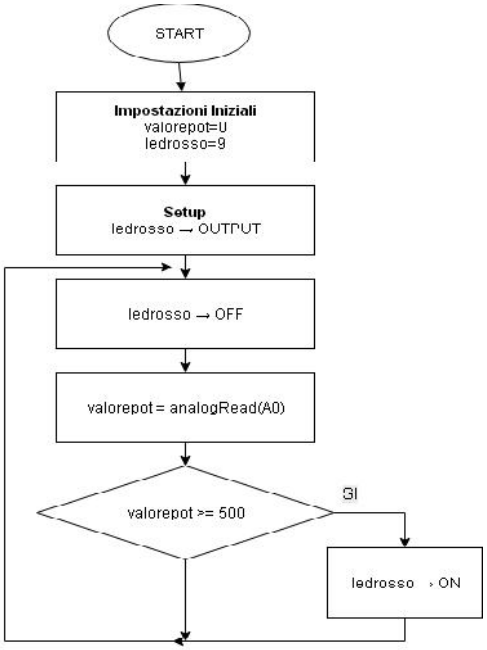
16.5) Accendere con luminosità crescente il LED Rosso e il LED Blu, a luminosità decrescente il LED verde.

Struttura iterazione Enumerativa (for…)

Soluzione

16.1) LED rosso acceso se il convertitore A/D restituisce un numero>=500
Struttura selezione semplice (if) – Disegnare il Flow Chart e scrivere il programma

Flow Chart

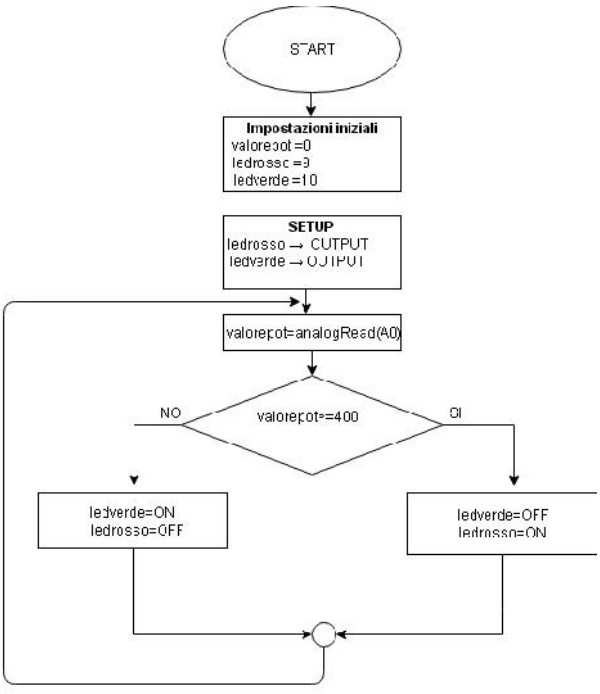


Programma

```
// Esercizio 16.1: Struttura selezione semplice  (if)
int valorepot=0;
int ledrosso=9;
void setup()
{
  pinMode(ledrosso,  OUTPUT);
}
void loop()
{
digitalWrite(ledrosso, LOW);
valorepot=analogRead(A0);
if (valorepot>=500)
 {
   digitalWrite(ledrosso, HIGH);
 }
}
```

16.2) Se il convertitore A/D restituisce un numero>=400: LED rosso ON, LED Verde OFF
Se il convertitore A/D restituisce un numero <400: LED rosso OFF, LED Verde ON
Struttura selezione doppia (if…else) - Flow Chart e programma

Flow Chart

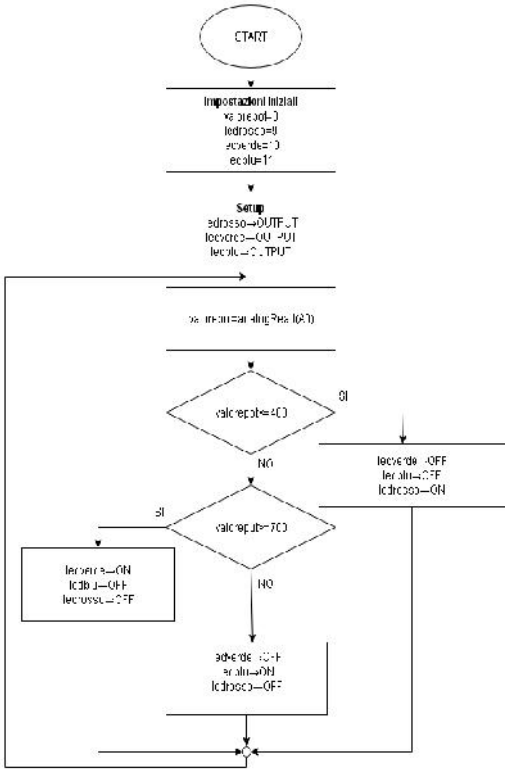


Programma

```
// Esercizio 16.2: Struttura selezione doppia  (if..else)
int valorepot=0;
int ledrosso=9;
int ledverde=10;
void setup()
{
  pinMode(ledrosso,  OUTPUT);
  pinMode(ledverde,  OUTPUT);
}
void loop()
{
valorepot=analogRead(A0);
if (valorepot>=400)
 {
 digitalWrite(ledverde, LOW);
 digitalWrite(ledrosso, HIGH);
 }
else
 {
  digitalWrite(ledverde,HIGH);
  digitalWrite(ledrosso, LOW);
 }
}
```

16.3) **LED rosso ON** se N<=400 (LED verde e LED Blu OFF), **LED verde** ON se N>=700 (LED rosso e LED Blu OFF), **LED blu ON** se 400<N<700 (LED rosso e LED verde OFF). *Struttura selezione tripla (if…else if … else)*

Flow Chart

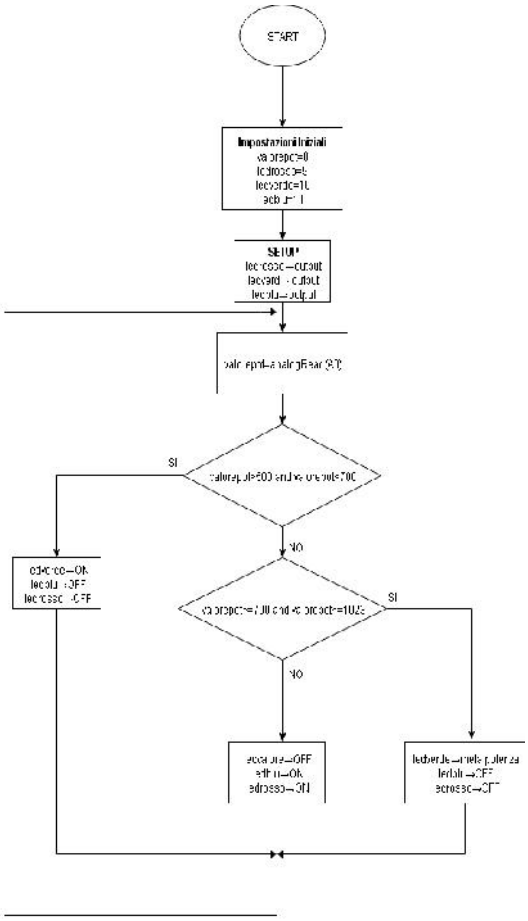


Programma

```
// Esercizio 16.3: Struttura selezione tripla  (if..else if..else)
int valorepot=0;
int ledrosso=9;
int ledverde=10;
int ledblu=11;

void setup()
{
  pinMode(ledrosso,  OUTPUT);
  pinMode(ledverde,  OUTPUT);
  pinMode(ledblu,  OUTPUT);
}
void loop()
{
valorepot=analogRead(A0);
if (valorepot<=400)
  {
  digitalWrite(ledverde, LOW);
  digitalWrite(ledblu, LOW);
  digitalWrite(ledrosso, HIGH);
  }
else if (valorepot>=700)
  {
  digitalWrite(ledverde, HIGH);
  digitalWrite(ledblu, LOW);
  digitalWrite(ledrosso, LOW);
  }
 else
  {
  digitalWrite(ledverde, LOW);
  digitalWrite(ledblu, HIGH);
  digitalWrite(ledrosso, LOW);
  }
}
```

16.4) Se 500<N<700 LED verde a luce piena, LED rosso OFF e LED Blu OFF
Se 700=<N<=1023 LED verde a metà potenza, LED rosso OFF e LED Blu OFF
Se N<500 LED Verde spento, LED Rosso ON e LED Blu ON
Struttura selezione tripla (if…else if … else)

Flow Chart



Programma

```
// Esercizio 16.4: Struttura selezione semplice  (If...else if...else))
int valorepot=0;
int ledrosso=9;
int ledverde=10;
int ledblu=11;

void setup()
{
  pinMode(ledrosso,  OUTPUT);
  pinMode(ledverde,  OUTPUT);
  pinMode(ledblu,  OUTPUT);
}
void loop()
{
valorepot=analogRead(A0);
if (valorepot>500 && valorepot<700)
  {
   digitalWrite(ledverde, HIGH);
   digitalWrite(ledblu, LOW);
   digitalWrite(ledrosso, LOW);
  }
else if (valorepot>=700 && valorepot<=1023)
  {
   analogWrite(ledverde, 90); // Led verde a metà potenza
   digitalWrite(ledblu,LOW);
   digitalWrite(ledrosso,LOW);
  }
 else
  {
   digitalWrite(ledverde, LOW);
   digitalWrite(ledblu, HIGH);
   digitalWrite(ledrosso, HIGH);
  }
}
```

16.5) Accensione con luminosità crescente il LED Rosso e il LED Blu, a luminosità decrescente il LED verde. *Struttura iterazione Enumerativa (for…).*

Flow Chart	Programma

```
// Esercizio 16.5: Struttura Iterazione Enumerativa (for…)
int x;
int ledrosso=9;
int ledverde=10;
int ledblu=11;

void setup()
{
  pinMode(ledrosso,  OUTPUT);
  pinMode(ledverde,  OUTPUT);
  pinMode(ledblu,  OUTPUT);
}
void loop()
{
for (x=0; x<=255; x++)
 {
   analogWrite(ledrosso, x);
   analogWrite(ledverde, 255-x);
   analogWrite(ledblu, x);
   delay (250);
 }
}
```

Esercizio 17 – Controllo LED con sensore di posizione e pulsante

Assegnato il circuito di figura 1

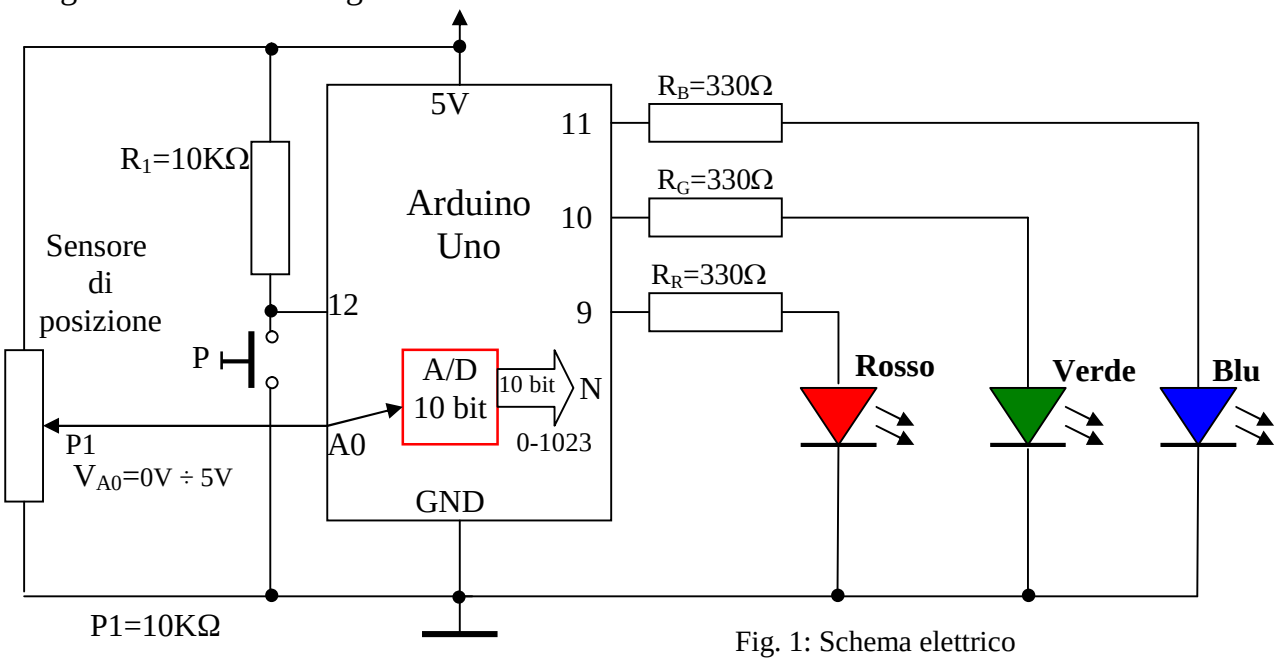


Fig. 1: Schema elettrico

- Se il sensore di posizione restituisce un numero maggiore di 650 e minore di 750 accendere il LED blu e spegnere tutti gli altri.
- Se il sensore di posizione restituisce un numero maggiore di 750 e il pulsante P è premuto accendere solo il LED rosso, altrimenti accendere tutti i LED
- Se il sensore di posizione restituisce un numero minore o uguale a 650 accendere solo il LED Verde.

Disegnare il Flow Chart e scrivere il programma in grado di rispettare le specifiche assegnate.

Soluzione

Programma

```
// Esercizio 17 – Controllo LED con sensore di posizione e pulsante
int ledverde  = 10;
int ledrosso  = 9;
int ledblu    = 11;
int valorepot=0;
int pulsante=12;
void setup()
{
  pinMode(ledverde,  OUTPUT);
  pinMode(ledrosso,  OUTPUT);
  pinMode(ledblu,    OUTPUT);
  pinMode(pulsante,  INPUT);
}

void loop()
{
 valorepot=analogRead(A0);
 if (valorepot>650 && valorepot<750)
  {
   digitalWrite(ledverde, LOW);
   digitalWrite(ledblu, HIGH);
   digitalWrite(ledrosso, LOW);
  }
 else if (valorepot>=750 && digitalRead(pulsante)==LOW)
  {
   digitalWrite(ledverde, LOW);
   digitalWrite(ledblu, LOW);
   digitalWrite(ledrosso, HIGH);
  }
else if (valorepot>=750 && digitalRead(pulsante)==HIGH)
  {
   digitalWrite(ledverde, HIGH);
   digitalWrite(ledblu, HIGH);
   digitalWrite(ledrosso, HIGH);
  }
else
  {
   digitalWrite(ledverde, HIGH);
   digitalWrite(ledblu, LOW);
   digitalWrite(ledrosso, LOW);
  }

}
```

Esercizio 18 – LED Array 1 – Effetto scorrimento con ritardo variabile

Assegnato il circuito di figura

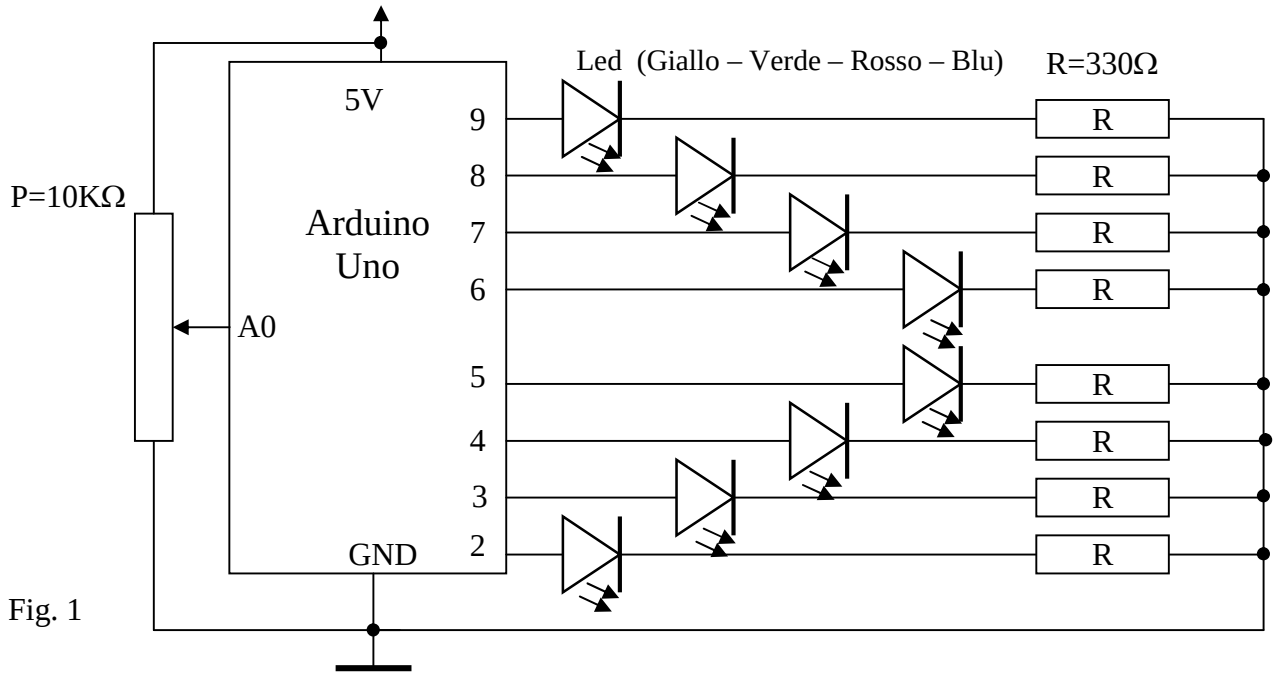


Fig. 1

Scrivere il programma in grado di eseguire una sequenza luminosa (effetto scorrimento con ritardo variabile), tale effetto, deve essere realizzato con un array composto da 8 caselle.
La regolazione del ritardo deve essere effettuata tramite il potenziometro P collegato sull'ingresso analogico A0.

Soluzione

Si definisce un array di nome **pinArray** composto da 8 caselle, su ogni singola casella memorizziamo il numero del pin collegato LED.

PinArray [8]

0	1	2	3	4	5	6	7
2	3	4	5	6	7	8	9

Stato LED Pin

2	3	4	5	6	7	8	9
H							
	H						
		H					
			H				
				H			
					H		
						H	
							H
							H
						H	
					H		
				H			
			H				
		H					
	H						
H							

H = LED acceso
Casella vuota = LED Spento

Programma

```
// Esercizio 18 – LED Array – Effetto scorrimento con ritardo variabile

int pinArray[] = {2, 3, 4, 5, 6, 7, 8, 9};      // Definizione Array pin
int count = 0;                                  // Contatore, indice dell'array
int timer;                                      // Ritardo, temporizzatore

void setup()
{
  for (count=0;count<8;count++)
   {
   pinMode(pinArray[count], OUTPUT);    // Configura i pin 2 ÷ 9 come OUTPUT
   }
}
 void loop()
{
 timer=analogRead(A0);                          // Lettura valore analogico – Conv A/D (0-1023)
  for (count=0;count<8;count++)
   {
      digitalWrite(pinArray[count], HIGH);
      delay(timer);
      digitalWrite(pinArray[count], LOW);
      delay(timer);
    }
  for (count=7;count>=0;count--)
   {
      digitalWrite(pinArray[count], HIGH);
      delay(timer);
      digitalWrite(pinArray[count], LOW);
      delay(timer);
    }
}
```

Esercizio 19 – LED Array 2 – Effetto scorrimento a blocchi con ritardo variabile

Assegnato il circuito di figura

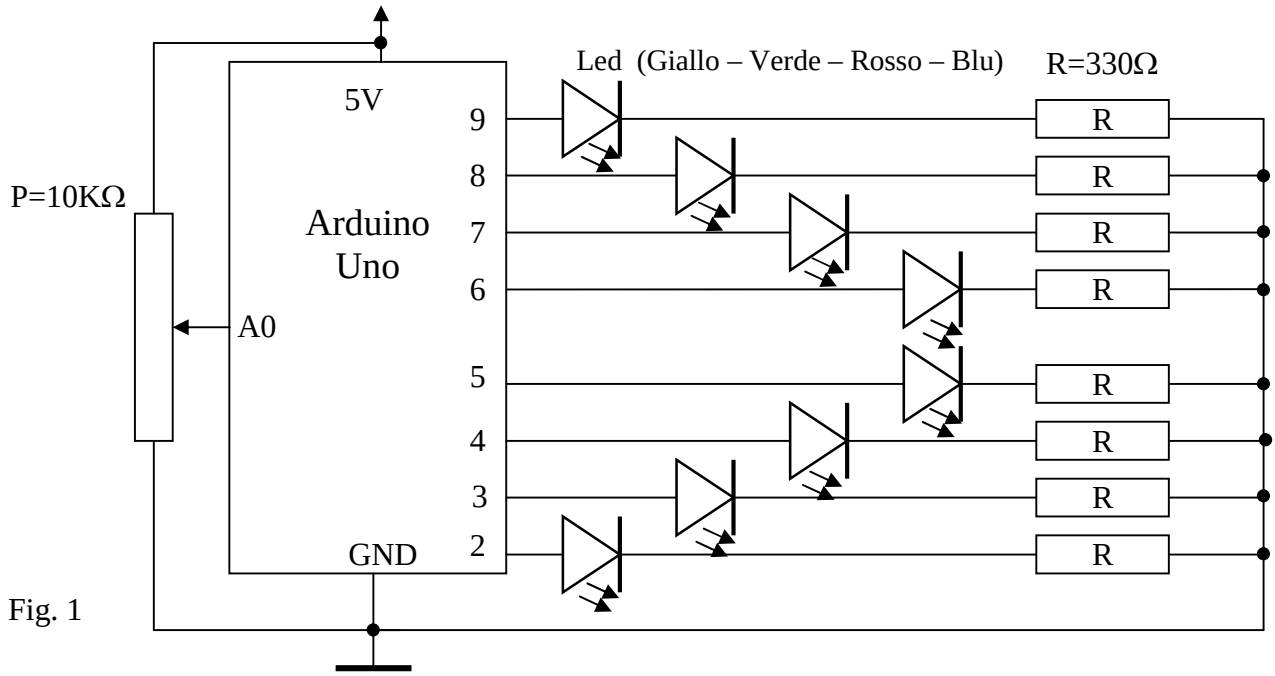


Fig. 1

Scrivere il programma in grado di eseguire una sequenza luminosa (effetto scorrimento a blocchi con ritardo variabile), tale effetto, deve essere realizzato con un array composto da 8 caselle.

La regolazione del ritardo deve essere effettuata tramite il potenziometro P collegato sull'ingresso analogico A0.

Soluzione

Si definisce un array di nome **pinArray** composto da 8 caselle, su ogni singola casella memorizziamo il numero del pin collegato LED.

PinArray [8]

0	1	2	3	4	5	6	7
2	3	4	5	6	7	8	9

Stato LED Pin

2	3	4	5	6	7	8	9
H				H			
	H				H		
		H				H	
			H				H
			H				H
		H				H	
	H				H		
H				H			

H = LED acceso
Casella vuota = LED Spento

Programma

```
// Esercizio 19 – LED Array 2  – Effetto scorrimento a blocchi con ritardo variabile

int pinArray[] = {2, 3, 4, 5, 6, 7, 8, 9};      // Definizione Array pin
int count = 0;                                  // Contatore, indice dell'array
int timer;                                      // Ritardo, temporizzatore

void setup()
{
  for (count=0;count<8;count++)
   {
   pinMode(pinArray[count], OUTPUT);    // Configura i pin 2 ÷ 9 come OUTPUT
   }
}
 void loop()
{
 timer=analogRead(A0);
  for (count=0;count<4;count++)
    {
       digitalWrite(pinArray[count], HIGH);
       digitalWrite(pinArray[count+4], HIGH);
       delay(timer);
       digitalWrite(pinArray[count], LOW);
       digitalWrite(pinArray[count+4], LOW);
       delay(timer);
     }
  for (count=7;count>3;count--)
    {
       digitalWrite(pinArray[count], HIGH);
       digitalWrite(pinArray[count-4], HIGH);
       delay(timer);
       digitalWrite(pinArray[count], LOW);
       digitalWrite(pinArray[count-4], LOW);
       delay(timer);
     }
}
```

Esercizio 20 – LED BAR – Effetto riempimento

Assegnato il circuito di figura

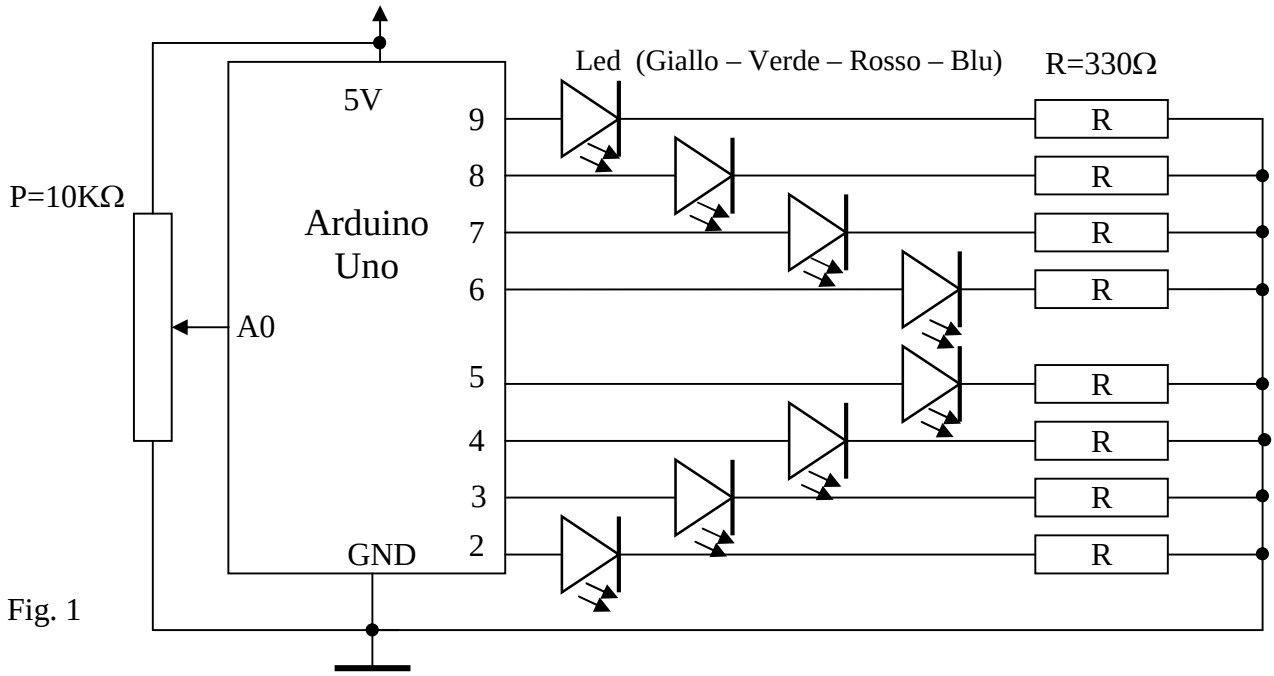


Fig. 1

Scrivere il programma in grado di accendere i LED in base alla posizione del potenziometro.
Utilizzare l'istruzione MAP per mappare il risultato della conversione (0 ÷ 1023) nell'intervallo (1 ÷ 8).

Sintassi di **map**:

***variabile*=map(*valore, fromLow, fromHigh, toLow, toHigh*)**

L'istruzione mappa un valore dell'intervallo compreso fra *fromLow* e *fromHigh* sull'intervallo compreso fra *toLow* e *toHigh* (esegue una proporzione).
Molto utile per elaborare valori che provengono da sensori analogici (es. Temperatura, Posizione, Luminosità, ecc).

Esempio:

```
val=map(analogRead(A0), 0, 1023, 1, 8);
 // mappa il valore letto sulla porta A0 su un valore compreso tra 1 e 8
```

Soluzione

Si definisce un array di nome **pinArray** composto da 8 caselle, su ogni singola casella memorizziamo il numero del pin collegato LED.

PinArray [8]

0	1	2	3	4	5	6	7
2	3	4	5	6	7	8	9

La soluzione si base sulla struttura di controllo **Switch …. Case**

Programma

```
// Esercizio 20 – LED BAR – Effetto riempimento
int pinArray[] = {2, 3, 4, 5, 6, 7, 8, 9};        // Definizione Array pin
int count = 0;                                    // Contatore, indice dell'array
int val;                                          // Risultato mappatura
void setup()
{
  for (count=0;count<8;count++)
   {
   pinMode(pinArray[count], OUTPUT);     // Configura i pin 2 ÷ 9 come OUTPUT
   }
}
 void loop()
{
 val=map (analogRead(A0),0,1023,1,8); // Mappa il valore nell'intervallo 1 ÷ 8
 for (count=0;count<8;count++)
   {
   digitalWrite(pinArray[count], LOW);      // Spegne tutti i Led
   }
switch (val)
  {
case 1:
          digitalWrite(pinArray[0],HIGH);
          break;
case 2:
          digitalWrite(pinArray[0],HIGH);
          digitalWrite(pinArray[1],HIGH);
          break;
case 3:
          digitalWrite(pinArray[0],HIGH);
          digitalWrite(pinArray[1],HIGH);
          digitalWrite(pinArray[2],HIGH);
          break;
case 4:
          digitalWrite(pinArray[0],HIGH);
          digitalWrite(pinArray[1],HIGH);
          digitalWrite(pinArray[2],HIGH);
          digitalWrite(pinArray[3],HIGH);
          break;
case 5:
          digitalWrite(pinArray[0],HIGH);
          digitalWrite(pinArray[1],HIGH);
          digitalWrite(pinArray[2],HIGH);
          digitalWrite(pinArray[3],HIGH);
          digitalWrite(pinArray[4],HIGH);
          break;
case 6:
          digitalWrite(pinArray[0],HIGH);
          digitalWrite(pinArray[1],HIGH);
          digitalWrite(pinArray[2],HIGH);
          digitalWrite(pinArray[3],HIGH);
          digitalWrite(pinArray[4],HIGH);
          digitalWrite(pinArray[5],HIGH);
          break;
case 7:
          digitalWrite(pinArray[0],HIGH);
          digitalWrite(pinArray[1],HIGH);
          digitalWrite(pinArray[2],HIGH);
          digitalWrite(pinArray[3],HIGH);
          digitalWrite(pinArray[4],HIGH);
          digitalWrite(pinArray[5],HIGH);
          digitalWrite(pinArray[6],HIGH);
          break;
case 8:
          digitalWrite(pinArray[0],HIGH);
          digitalWrite(pinArray[1],HIGH);
          digitalWrite(pinArray[2],HIGH);
          digitalWrite(pinArray[3],HIGH);
          digitalWrite(pinArray[4],HIGH);
          digitalWrite(pinArray[5],HIGH);
          digitalWrite(pinArray[6],HIGH);
          digitalWrite(pinArray[7],HIGH);
          break;
  }
}
```

Esercizio 21 – Controllo LED con Fotoresistenza

Dimensionare e programmare un circuito in grado di accendere un LED se l'intensità di luce scende sotto un certo valore.

1. Disegnare lo schema elettrico
2. Disegnare lo schema di cablaggio
3. Scrivere il programma

Soluzione

La **fotoresistenza** (LDR) (Light Dependent Resistor) è un componente elettronico la cui resistenza è inversamente proporzionale alla quantità di luce che lo colpisce. Si comporta come un tradizionale resistore, ma il suo valore in ohm diminuisce mano a mano che aumenta l'intensità della luce che la colpisce. Ciò comporta che la corrente elettrica che transita attraverso tale componente è proporzionale all'intensità di una sorgente luminosa. Il valore varia da pochi Ohm (forte intensità di luce) a migliaia di Ohm (Buio).

Le fotoresistenze LDR (Light Dependent Resistor), ma in generale i sensori di luce, possono essere utilizzati con la scheda Arduino per:

- realizzare un circuito crepuscolare, che accende le luci di casa quando c'è buio
- misurare la luce in ufficio o in casa per regolare al meglio la luce delle lampadine alogene o a led (dimmer).
- dirigere il robot con piattaforma Arduino verso fonti di luce
- regolare la direzione dei pannelli fotovoltaici automaticamente per costruire un inseguitore solare con Arduino
- regolatore di volume audio

Fondamentalmente essa è composta da materiale semiconduttore. L'energia radiante fornita ad un semiconduttore provoca la produzione di coppie elettrone-lacuna in eccesso rispetto a quelle generate termicamente che causa una diminuzione della resistenza elettrica del materiale (effetto fotoconduttivo). Quando la radiazione incidente viene interrotta i portatori di carica in eccesso si ricombinano riportando la conducibilità del semiconduttore al suo valore iniziale in condizioni di oscurità.

In Fig. 1 è riportato lo schema elettrico, in fig. 2 lo schema di cablaggio con Fritzing.

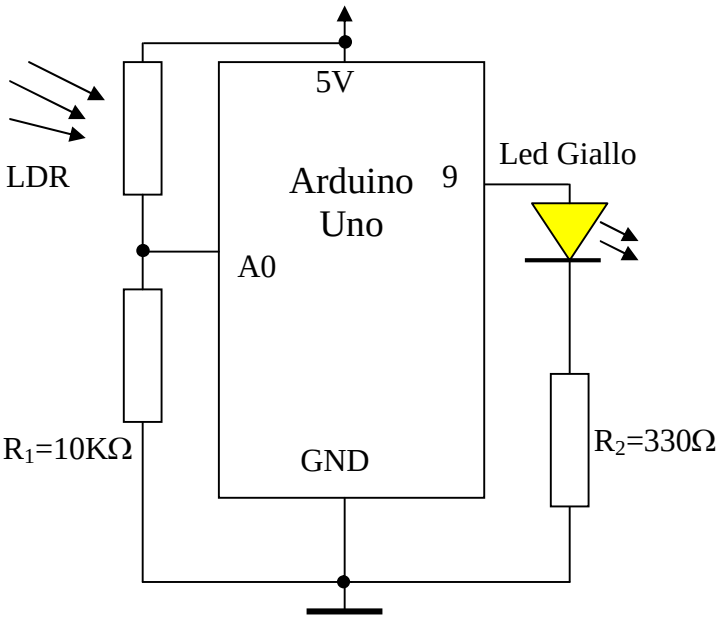


Fig. 1: Schema elettrico

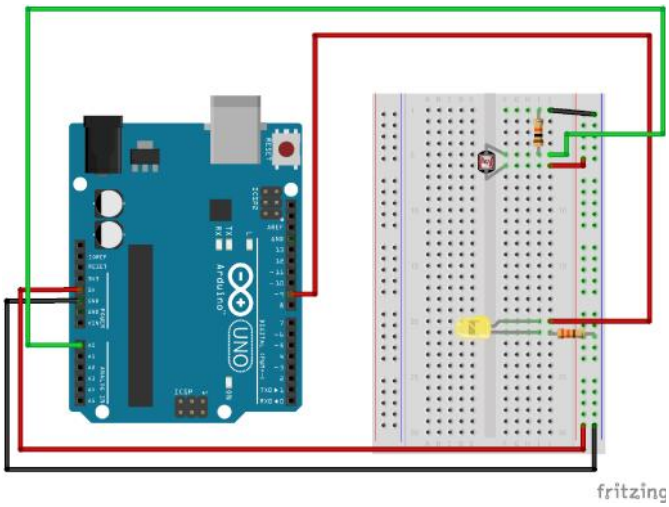


Fig. 2: Cablaggio con Fritzing

Programma

```
// Esercizio 21 - Controllo LED con Fotoresistenza
int valoreldr; // variabile in cui viene inserito il valore analogico (da 0 a 1023)
               // della tensione rilevata sul partitore R1-LDR.
byte led=9;
void setup()
{
pinMode(led, OUTPUT); // definisce la porta 9 come porta di OUTPUT
}
void loop()
{
valoreldr = analogRead(0); // legge il valore della tensione fornito dal partitore R1-LDR
if(valoreldr<=512)
  {
    digitalWrite(led, HIGH); // accende il LED con luce ambiente bassa
  }
else
  {
    digitalWrite(led, LOW); // spegne il LED con luce ambiente alta
  }
}
```

Esercizio 22 – Controllo quattro LED con Fotoresistenza

Dimensionare e programmare un circuito in grado di accendere quattro LED (Giallo,Verde, Rosso, Blu) in funzione dell'intensità di luce.

1. Disegnare lo schema elettrico
2. Disegnare lo schema di cablaggio
3. Scrivere il programma

Soluzione

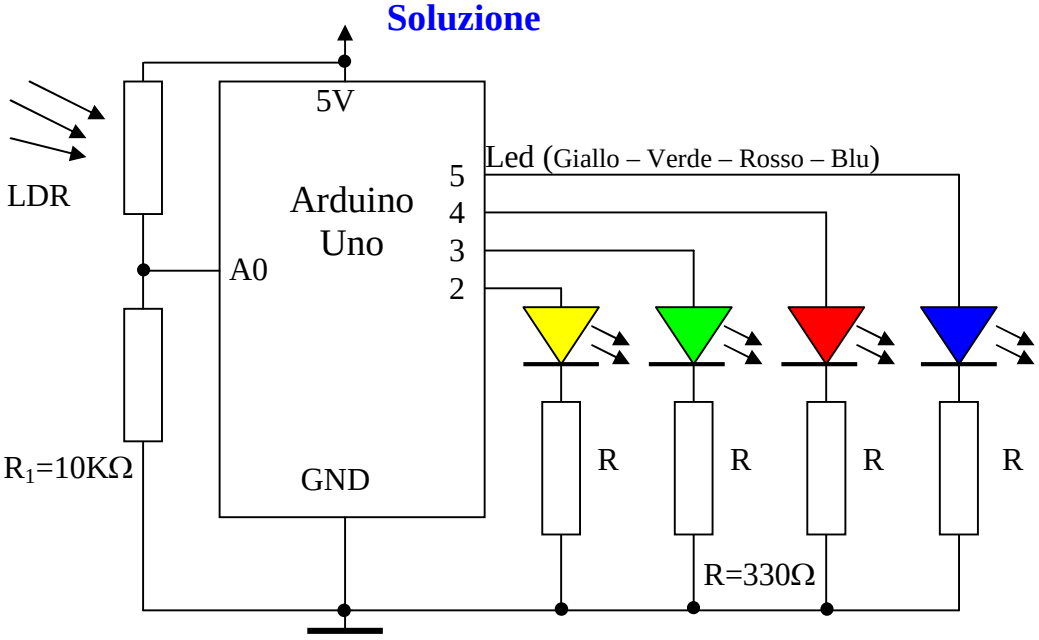


Fig. 1: Schema elettrico

Programma

```
// Esercizio 22 - Controllo quattro LED con Fotoresistenza
int valoreldr; // variabile in cui viene inserito il valore analogico (da 0 a 1023)
               // della tensione rilevata sul  partitore R1-LDR
byte ledg=2;
byte ledv=3;
byte ledr=4;
byte ledb=5;
byte i;
void setup()
{
 pinMode(ledg, OUTPUT);
 pinMode(ledv, OUTPUT);
 pinMode(ledr, OUTPUT);
 pinMode(ledb, OUTPUT);
}
void loop()
{
  valoreldr = analogRead(0); // legge il valore della tensione fornito dal partitore R1-LDR
 for(i=2;i<=5;i++)              // ciclo di for utilizzato per spegnere tutti i led
digitalWrite (i, LOW);
        if (valoreldr < 127) { digitalWrite (ledg, HIGH); } // accende i LED
        if (valoreldr < 255) { digitalWrite (ledv, HIGH); } // in funzione
        if (valoreldr < 511) { digitalWrite (ledr, HIGH); } // della tensione
        if (valoreldr < 767) { digitalWrite (ledb, HIGH); } // rilevata
 }
```

Esercizio 23 – Controllo intensità luminosa LED con Fotoresistenza

Dimensionare e programmare un circuito in grado di controllare la luminosità di un LED in funzione dell'intensità di luce (minima intensità=Massima Luminosità; . Massima intensità=Minima Luminosità).

1. Disegnare lo schema elettrico
2. Disegnare lo schema di cablaggio
3. Scrivere il programma

Soluzione

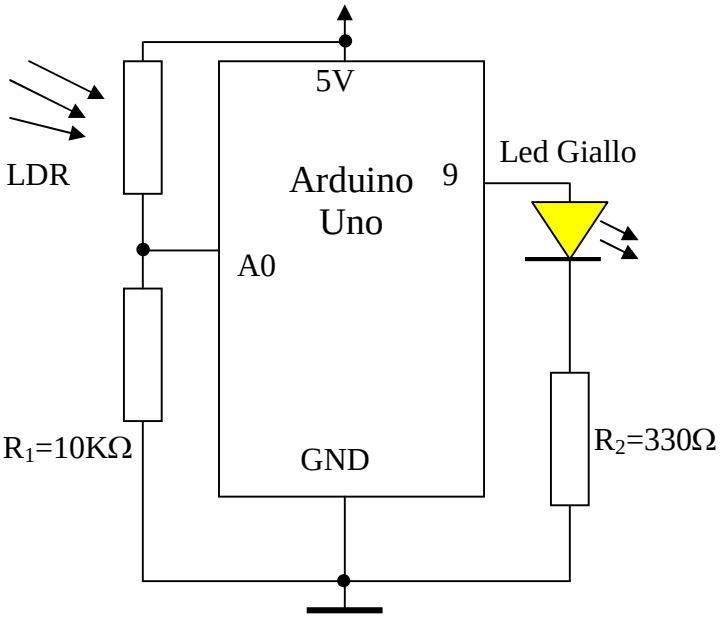


Fig. 1: Schema elettrico

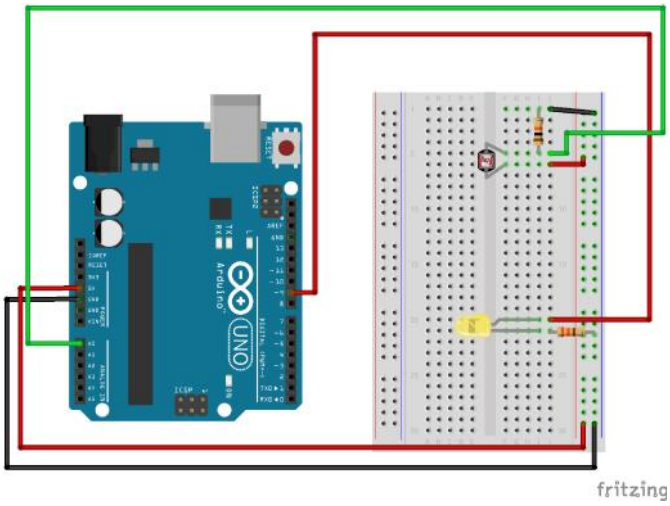


Fig. 2: Cablaggio con Fritzing

Programma

```
// Esercizio 23 - Controllo intensità luminosa LED con Fotoresistenza
int valoreldr;       // variabile in cui viene inserito il valore analogico (da 0 a 1023)
                     // della tensione rilevata sul partitore R1-LDR
byte ledg=9;

void setup()
{
 pinMode(ledg, OUTPUT);
}
void loop()
{
  valoreldr = analogRead(0);    // legge il valore della tensione fornito dal partitore R1-LDR
  valoreldr=map(valoreldr,0,1023,0,255); // mappa il valoreldr nell'intervallo 0 - 255
  analogWrite (ledg, 255-valoreldr); // Controlla la luminosità del LED
}
```

Esercizio 24 – Rilevazione temperatura con il Sensore TMP 36 e Monitor Seriale

Dimensionare e programmare un circuito in grado di rilevare la temperatura ambiente con il sensore TMP36 e visualizzare il valore tramite il Monitor Seriale dell'IDE

1. Disegnare lo schema elettrico
2. Disegnare lo schema di cablaggio
3. Scrivere il programma

Soluzione

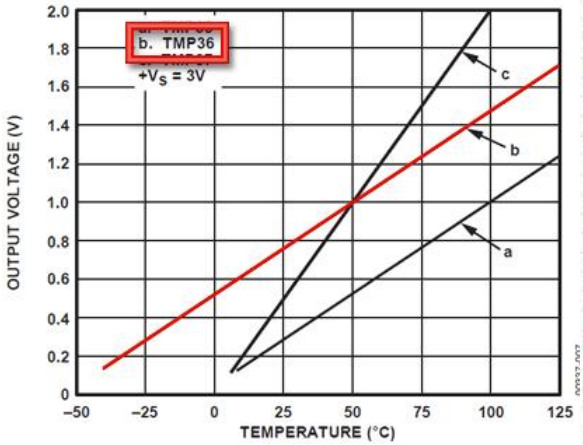
Il TMP36 è un sensore di temperatura analogico, fornisce un'uscita lineare pari a 10mv/°C con un offset di 500mV nel range -40°C +125°C, alla temperatura di **0°C** il sensore eroga una tensione di **500mV**.
Il legame Temperatura misurata (°C) e tensione di uscita è dato dalla seguente caratteristica di trasferimento:

V_{out}=K•T+0,5 dove K=10mv/°C, T=temperatura espressa in °C e 0,5 rappresenta l'Offset.

Trasformazione in:

T(°F)=1,8*T+32 T(K)=273,15+T

In figura 1 Grafico (*Vout in funzione della Temperatura*) e Caratteristiche tecniche.



Altezza	5.2mm
Profondità	4.19mm
Lunghezza	5.2mm
Package fornitore	TO-92
Funzione	Sensore di temperatura
Numero pin	3
Precisione	± 2 ° C
Linearità	± 0,5 ° C
Sensibilità	10 mV / °C fattore di scala
Campo temperatura minima	-40 ° C a +125 ° C, funzionamento a +150 ° C
Tensione tipica di funzionamento	2,7 V a 5,5 V
Tipo uscita	Analogica
Corrente di riposo	Meno di 50 µA
	Basso auto-riscaldamento

Fig. 1

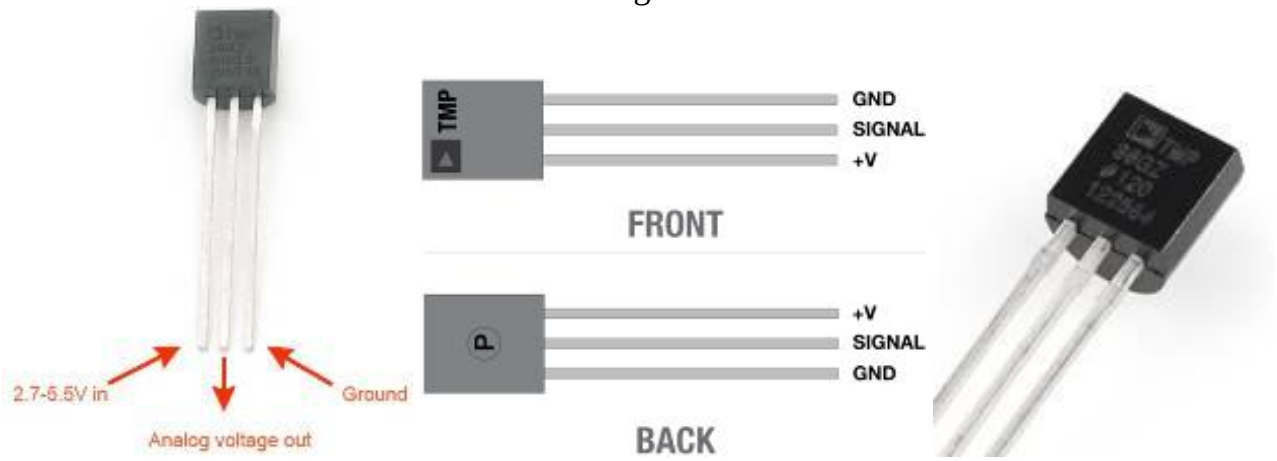


Fig. 2 (Piedinatura e foto)

In Fig. 3 è riportato lo schema elettrico, in fig. 4 lo schema di cablaggio con Fritzing.

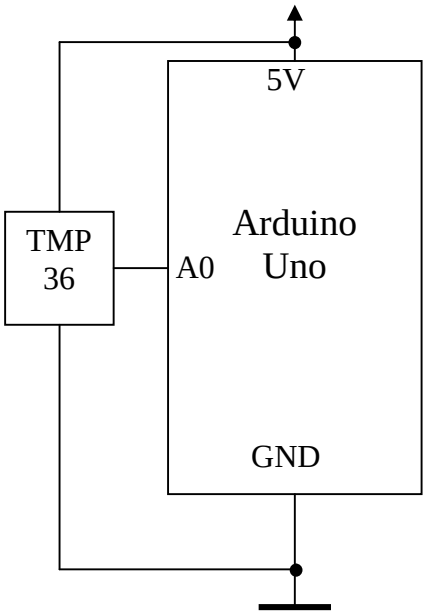


Fig. 3 – Schema elettrico

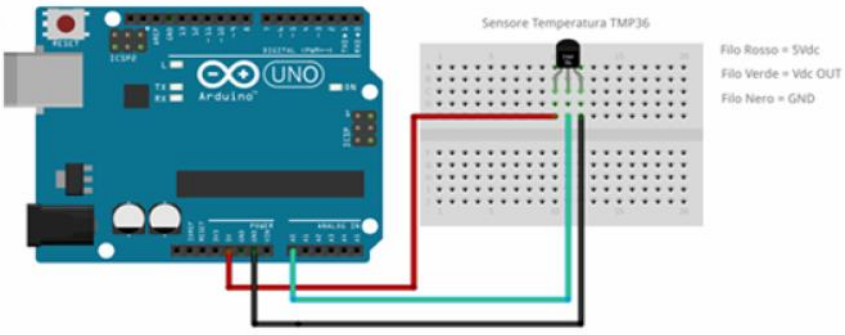


Fig. 4: Cablaggio con Fritzing

Il segnale analogico fornito dal sensore (V_{out}) viene inviato al convertitore A/D interno (Pin A0) che lo trasforma in un numero nel range (0÷1023).
Tramite software si trasforma il numero nel valore di Tensione e nel valore di Temperatura .

Formule:

V_{out}=K•T+0,5 [V] dove K=10mv/°C, T=temperatura espressa in °C e 0,5 rappresenta l'Offset.

$$N = \frac{Vout}{\frac{5}{1024}} \text{ [N. out A/D]} \qquad Vout = \frac{N \bullet 5}{1024} \text{ [V]} \qquad T = \frac{Vout - 0{,}5}{K} \text{ [°C]}$$

Programma

```
// Esercizio 24 - Rilevazione temperatura con il Sensore TMP 36 e Monitor Seriale
void setup()
{
Serial.begin(9600); // Inizializzazione monitor seriale
}
void loop()
{
   int valoreADC = analogRead(0);                  // Conv A/D di Vout del sensore TMP36
   float voltage = (valoreADC/1024.0)* 5.0;        // trasformazione valore ADC in Tensione
   float temp=(voltage-0.5)/0.01;                  // trasformazione tensione in Temperatura
// Invio dati al PC tramite seriale
   Serial.print("Vout=");
   Serial.print(voltage,4);
   Serial.print("V  -  Valore ADC=");
   Serial.print(valoreADC);
   Serial.print("  -  Temp=");
   Serial.print(temp,2);
   Serial.println("C");
   delay (1000);
}
```

Esempio visualizzazione sul Monitor

Esercizio 25 – Rilevazione temperatura con il Sensore TMP 36 e Display LCD 16x2

Dimensionare e programmare un circuito in grado di rilevare la temperatura ambiente con il sensore TMP36 e visualizzare il valore tramite il Display LCD 16x2.

1. Disegnare lo schema elettrico
2. Disegnare lo schema di cablaggio
3. Scrivere il programma

Soluzione

In Fig. 1 è riportato lo schema elettrico.

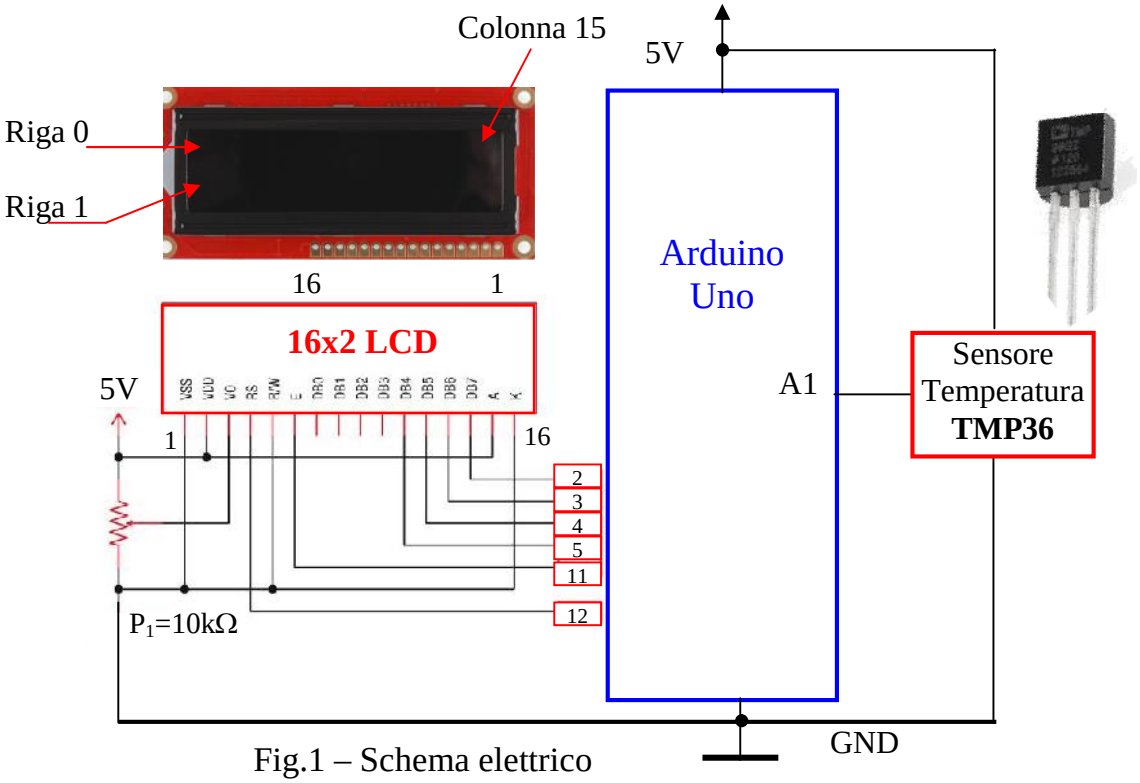


Fig.1 – Schema elettrico

Programma

```
/* Esercizio 25 - Rilevazione temperatura con il Sensore TMP 36 e Display LCD 16x2
 Descrizione pin:
 Pin +5V          -> Alimentazione
 Pin GND          -> Alimentazione
 Pin Digital 2    -> Bus DB7 dati LCD
 Pin Digital 3    -> Bus DB6 dati LCD
 Pin Digital 4    -> Bus DB5 dati LCD
 Pin Digital 5    -> Bus DB4 dati LCD
 Pin Digital 11    -> terminale E display LCD
 Pin Digital 12    -> terminale RS display LCD
 Pin Analogico A1 -> lettura sensore TMP36
*/
#include <LiquidCrystal.h>  // Libreria gestione Display LCD
// Selezione dei pin utilizzati dal display LCD
// Sintassi LiquidCrystal(rs, enable, d4, d5, d6, d7)
LiquidCrystal lcd(12,11,5,4,3,2);
void setup()
{
lcd.begin (16,2); // Inizializzazione libreria per display LCD 16x2
}
void loop()
{
int valoreADC = analogRead(1);
float voltage = (valoreADC/1024.0)* 5.0;
float temp=(voltage-0.5)/0.01;
// Invio dati al Display LCD
lcd.clear(); // Pulisce lo schermo LCD
lcd.setCursor(0, 0);
lcd.print("Vout=");
lcd.setCursor(5, 0);
lcd.print(voltage,3);
lcd.setCursor(10, 0);
lcd.print("V");
lcd.setCursor(0, 1);
lcd.print("Temp=");
lcd.print(temp,2);
lcd.setCursor(10, 1);
lcd.print("C");
delay (2000); //Pausa di 2 secondi
}
```

Esercizio 26 – Controllo Servomotore

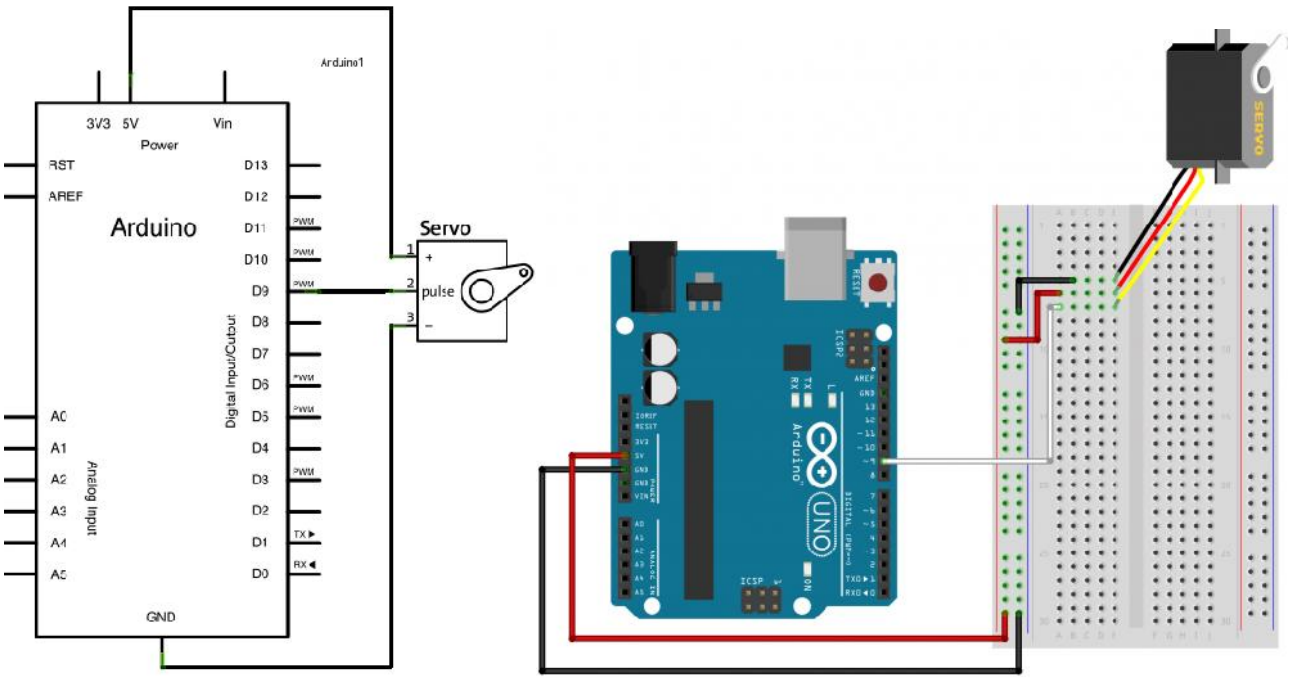
Dimensionare e programmare un circuito in grado di far ruotare il perno di un servo motore:

a) da 0° a 180° con incremento di 1°;
b) da 180° a 0° con decremento di 1°;
c) da 0° a 180° con incremento di 30°;

1. Disegnare lo schema elettrico
2. Disegnare lo schema di cablaggio
3. Scrivere il programma

Soluzione

In Fig. 1 è riportato lo schema elettrico, in fig. 2 lo schema di cablaggio con Fritzing.



Programma

```
/* Esercizio 26 - Controllo Servomotore
#include <Servo.h> // richiama la libreria di gestione dei servomotori
Servo myservo;      // crea il servo oggetto “myservo” da utilizzare nel programma
                    // in un programma possono essere creati, al massimo otto servo oggetti
int pos = 0;        // variabile per la memorizzazione della posizione (in gradi angolari) del perno
void setup()
{
 myservo.attach(9); // assegna il servo oggetto “myservo” alla porta 9
}
void loop()
{
 myservo.write(0); // posiziona il perno nella posizione iniziale 0
 delay (1000);      // attende 1 secondo
 for(pos = 0; pos <= 180; pos += 1) // sposta di un grado per volta il perno, da 0 a 180
  {
 myservo.write(pos); // indirizza il perno alla posizione memorizzata in 'pos'
 delay(15);          // attende 15ms per consentire al servomotore di raggiungere la posizione
```

```
 }
 for(pos = 180; pos>=1; pos-=1) // diminuisce di un grado per volta, da 180 a 0
{
 myservo.write(pos); // indirizza il perno alla posizione memorizzata in 'pos'
 delay(15);              // attende 15ms per consentire al servomotore di raggiungere la posizione
 }
 delay (1000);
 myservo.write(0); // indirizza il perno alla posizione 0
 delay(1000); // attende 1 secondo prima di iniziare la nuova sequenza
 for (pos =0; pos <= 180; pos = pos + 30) // seleziona le posizioni 30, 60, 90, 120, 150 e 180
 {
 myservo.write(pos); // indirizza il perno alla posizione memorizzata in 'pos'
 delay(1000); // attende 1 secondo prima di andare alla successiva posizione
 }
 delay (3000); // attende 3 secondi prima di ricominciare il ciclo
}
```

Esercizio 27 – Controllo Servomotore con due pulsanti

Dimensionare e programmare un circuito in grado di far ruotare il perno di un servo motore con due pulsanti (Up, Down) e visualizzare l'angolo relativo alla posizione del perno sul monitor seriale.

1. Disegnare lo schema elettrico
2. Disegnare lo schema di cablaggio
3. Scrivere il programma in grado di far ruotare il perno di ± 5° ad ogni click dei pulsanti
4. Scrivere il programma in grado di far ruotare il perno di ± 1° con pulsanti premuti

Soluzione

In Fig. 1 è riportato lo schema elettrico, in fig. 2 lo schema di cablaggio con Fritzing.

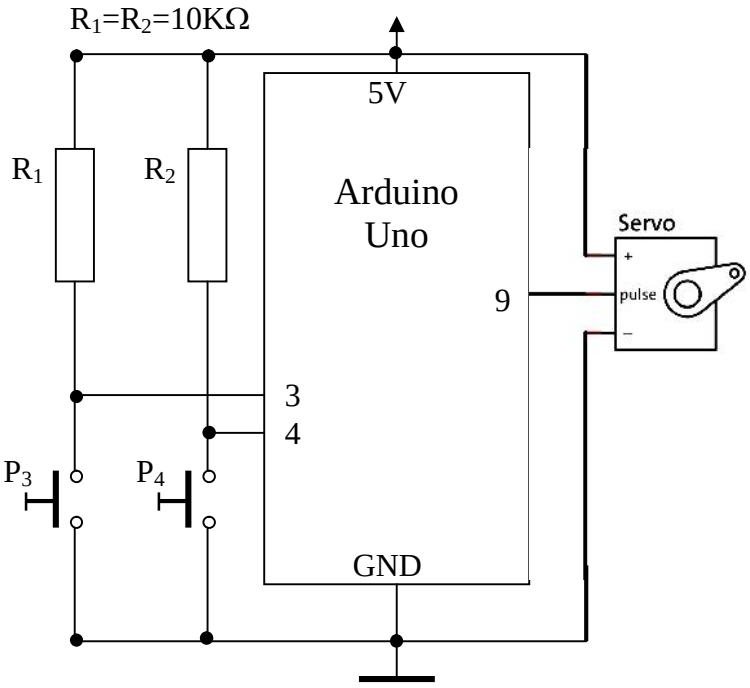


Fig.1

Programma 27a: ± 5° ad ogni click dei pulsanti

```
// Esercizio 27a: Controllo Servomotore con due pulsanti e libreria antirimbalzo
//                ± 5° ad ogni click dei pulsanti
#include <TriggerButton.h> // richiama la libreria antirimbalzo
TriggerButton pulsante1, pulsante2;
int tempo=70; // tempo antirimbalzo 70 ms

#include <Servo.h> // richiama la libreria di gestione dei servomotori
Servo myservo; // crea il servo oggetto “myservo”
int pos = 0; // variabile per la memorizzazione della posizione (in gradi angolari) del perno
void setup()
{
 Serial.begin(9600);
   pulsante1.setUp(3, LOW); //imposta il pulsante 1 sul pin 3 attivo basso
  pulsante2.setUp(4, LOW); //imposta il pulsante 2 sul pin 4 attivo basso
  myservo.attach(9); // assegna il servo oggetto “myservo” alla porta 9
  myservo.write(0); // sposta il perno nella posizione iniziale
}

void loop()
{
  if (pulsante1.Click(tempo))
  {
    pos=pos+5;
    if ( pos > 180) { pos=180; }
    Serial.println(pos);  // Visualizza sul monitor seriale la posizione del perno
    myservo.write(pos);
    delay(15);
  }
  if (pulsante2.Click(tempo))
  {
   pos=pos-5;
   if ( pos < 0) { pos=0; }
   Serial.println(pos); // Visualizza sul monitor seriale la posizione del perno
   myservo.write(pos);
   delay(15);
  }
}
```

Programma 27b: ± 1° con pulsanti premuti

```
// Esercizio 27b: Controllo Servomotore con due pulsanti:  ± 1° con pulsanti premuti
#include <Servo.h> // richiama la libreria di gestione dei servomotori
Servo myservo; // crea il servo oggetto “myservo”
int pos = 0; // variabile per la memorizzazione della posizione (in gradi angolari) del perno
byte pulsante1=3;
byte pulsante2=4;
void setup()
{
 Serial.begin(9600);
  myservo.attach(9); // assegna il servo oggetto “myservo” alla porta 9
  myservo.write(0); // sposta il perno nella posizione iniziale
}
void loop()
{
  if (digitalRead(pulsante1)==LOW)
  {
    pos=pos+1;
    if ( pos > 180) { pos=180; }
    Serial.println(pos);  // Visualizza sul monitor seriale la posizione del perno
    myservo.write(pos);
    delay(15);
  }
  if (digitalRead(pulsante2)==LOW)
  {
   pos=pos-1;
   if ( pos < 0) { pos=0; }
   Serial.println(pos); // Visualizza sul monitor seriale la posizione del perno
   myservo.write(pos);
   delay(15);
  }
}
```

Esercizio 28 – Controllo Servomotore con un potenziometro

Soluzione

In Fig. 1 è riportato lo schema elettrico, in fig. 2 lo schema di cablaggio con Fritzing.

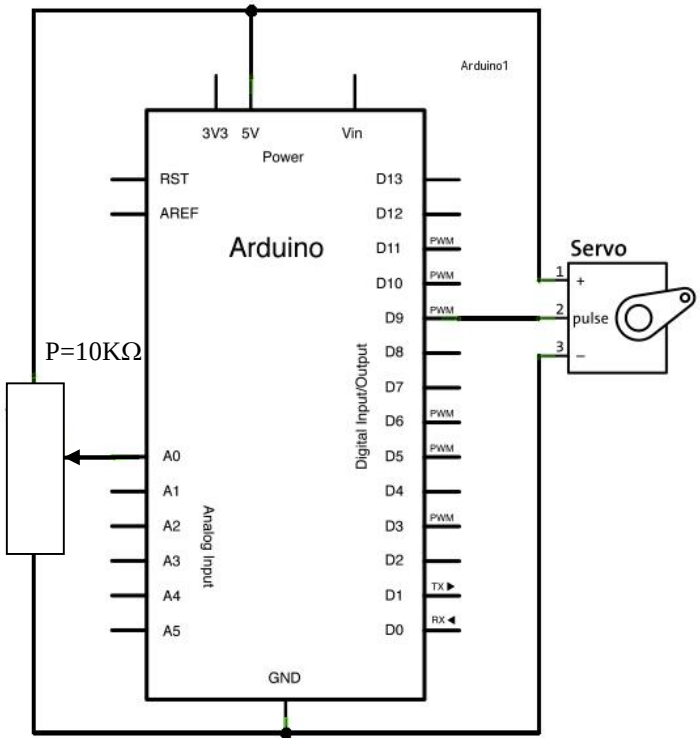
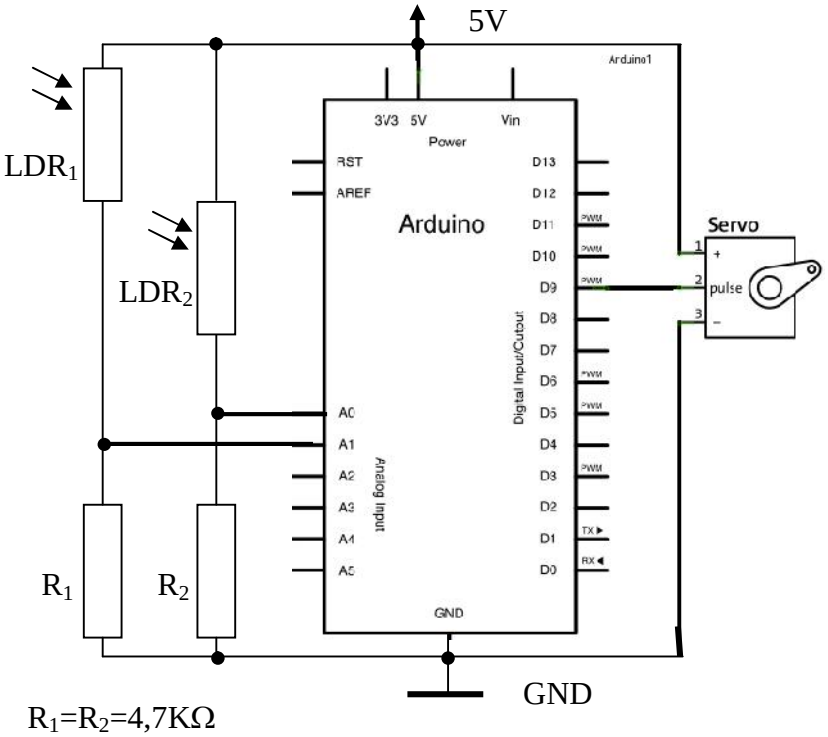


Fig.1 – Schema elettrico

Programma

```
// Esercizio 28: Controllo Servomotore con un potenziometro
#include <Servo.h> // richiama la libreria di gestione dei servomotori
Servo myservo; // crea il servo oggetto “myservo”
int pos = 0; // variabile per la memorizzazione della posizione (in gradi angolari) del perno
void setup()
{
 Serial.begin(9600);
  myservo.attach(9); // assegna il servo oggetto “myservo” alla porta 9
  myservo.write(0); // sposta il perno nella posizione iniziale
}
void loop()
{
  pos=map(analogRead(0), 0, 1023, 0, 180);      // legge il valore sulla porta A0
                                                // e mappa il valore nell’intervallo 0° ± 180°
  myservo.write(pos); // sposta il perno nella posizione memorizzata in “pos”
  delay(15);
}
```

Esercizio 29 – Inseguitore solare con Fotoresistenze e Servomotore



$R_1=R_2=4,7K\Omega$

Fig. 1: Schema elettrico

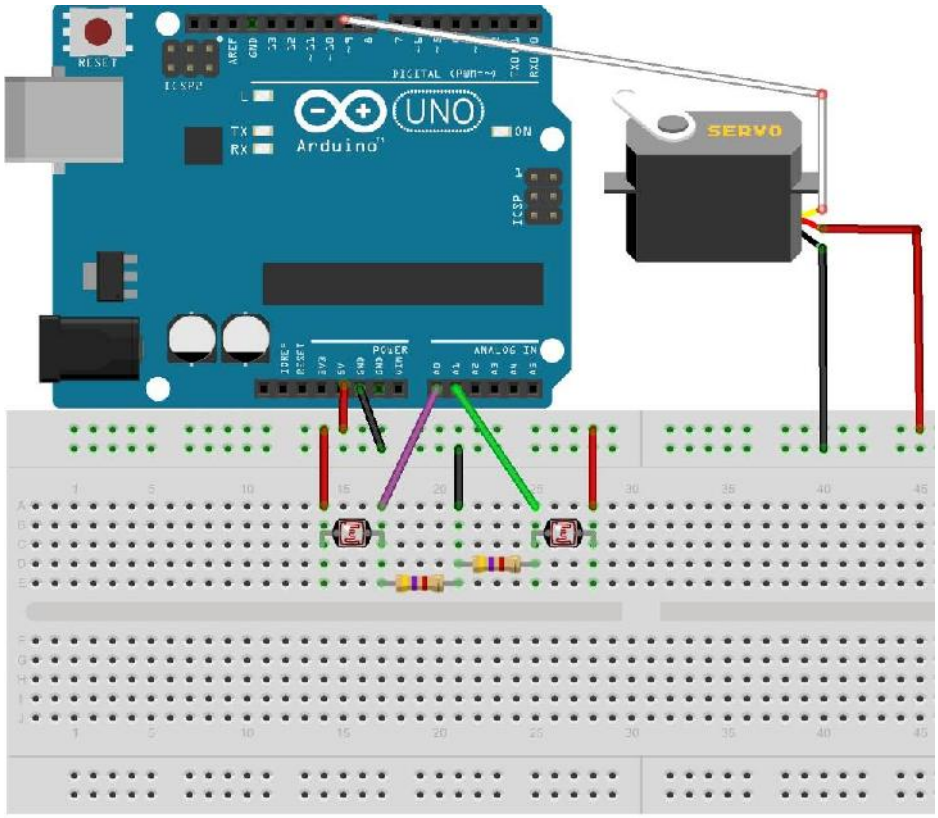


Fig. 2: Cablaggio con Fritzing

Programma

```
// Esercizio 29: Inseguitore solare con Fotoresistenze e Servomotore
#include <Servo.h> // richiama la libreria di gestione dei servomotori
Servo myservo; // crea il servo oggetto “myservo”

 int Valore;
int Centro=90; // Centro del servo

void setup() {
  Serial.begin(9600);
  myservo.attach(9); // Collega il servo al PIN 9
}

void loop() {
  int sensorValue = analogRead(A0);  // Legge il valore del sensore LDR1 di sinistra
  int sensorValue2 = analogRead(A1); // Legge il valore del sensore LDR2 di destra

  Valore=(sensorValue-sensorValue2)/10;
  /*
  Calcola la differenza tra i due sensori e la divide per 10,
  in questo modo una differenza di 10 equivale a 1, una differenza
  di 53 equivale a 5 e cosi via
  */

 if (Valore==0) myservo.detach(); else myservo.attach(9);
  /*
  Se la differenza è uguale a 0 scollega il servo altrimenti lo
  collega (serve per risparmiare energia)
  */
  if (Valore>30) Valore=30;
  if (Valore<-30) Valore=-30;
  // Limita la differenza massima tra -30 e + 30

  Serial.println(Valore); // Visualizza il valore sulla seriale
  myservo.write(Centro+Valore);
  /*
Sposta il perno nella posizione Centro+Valore -  Max=90+30=120; Min=90-30=60
Se “Valore” è positivo ruota il perno a DX,
Se “Valore” è negativo ruota il perni a SX.
La velocità è direttamente proporzionale al valore assoluto di Valore
  */
  delay(15);
 }
```

Esercizio 30 – Modulo Sensore di fiamma

Dimensionare e programmare un circuito basato sul sensore di fiamma YG1006 in grado, tramite un display LCD 16x2 e due Led (Rosso, Verde), di attivare un allarme in presenza di una fiamma.

1. Disegnare lo schema elettrico
2. Scrivere il programma in grado di visualizzare su un display LCD 16x2 lo stato della fiamma, attivare un Led Rosso in presenza di fiamma e un Led verde in assenza di fiamma.

Soluzione

Il sensore di fiamma YG1006, assemblato sulla scheda della Waveshare (Fig.1) è un fototransistor che si presenta come un diodo LED di colore nero. E' sensibile ai raggi elettromagnetici luminosi con lunghezza d'onda tra 760 e 1100 nm (Fig. 2) e con un picco di sensibilità intorno ai 950 nm.

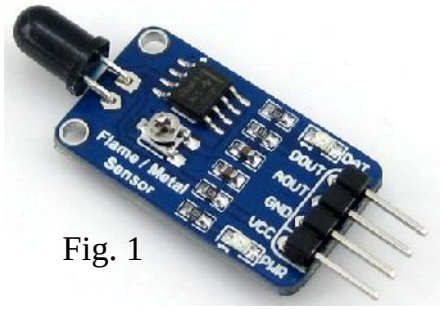


Fig. 1

Il sensore è sensibile allo spettro elettromagnetico tipicamente emesso dalle fiamme in una combustione, quindi, può essere utilizzato per rilevare incendi in sistemi d'allarme. Viene usato in molti progetti con Arduino e in robot didattici antincendio.

L'angolo di sensibilità della sonda è di 60 gradi(Fig.3), la temperatura operativa è compresa tra -25 gradi e 85 gradi Celsius (centigradi).

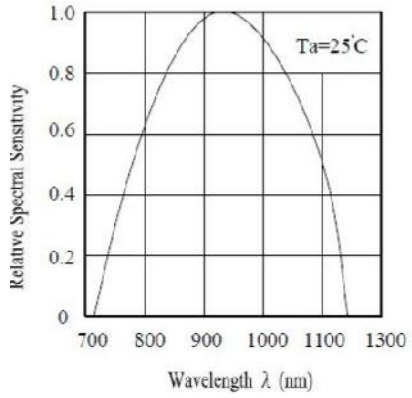


Fig. 2

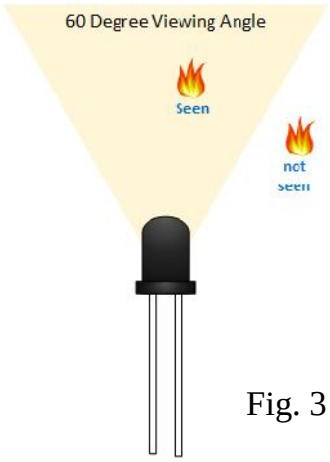


Fig. 3

In Fig.4 è riportato lo schema elettrico della scheda Sensore di fiamma.
La parte principale è costituita da un comparatore semplice (LM393).

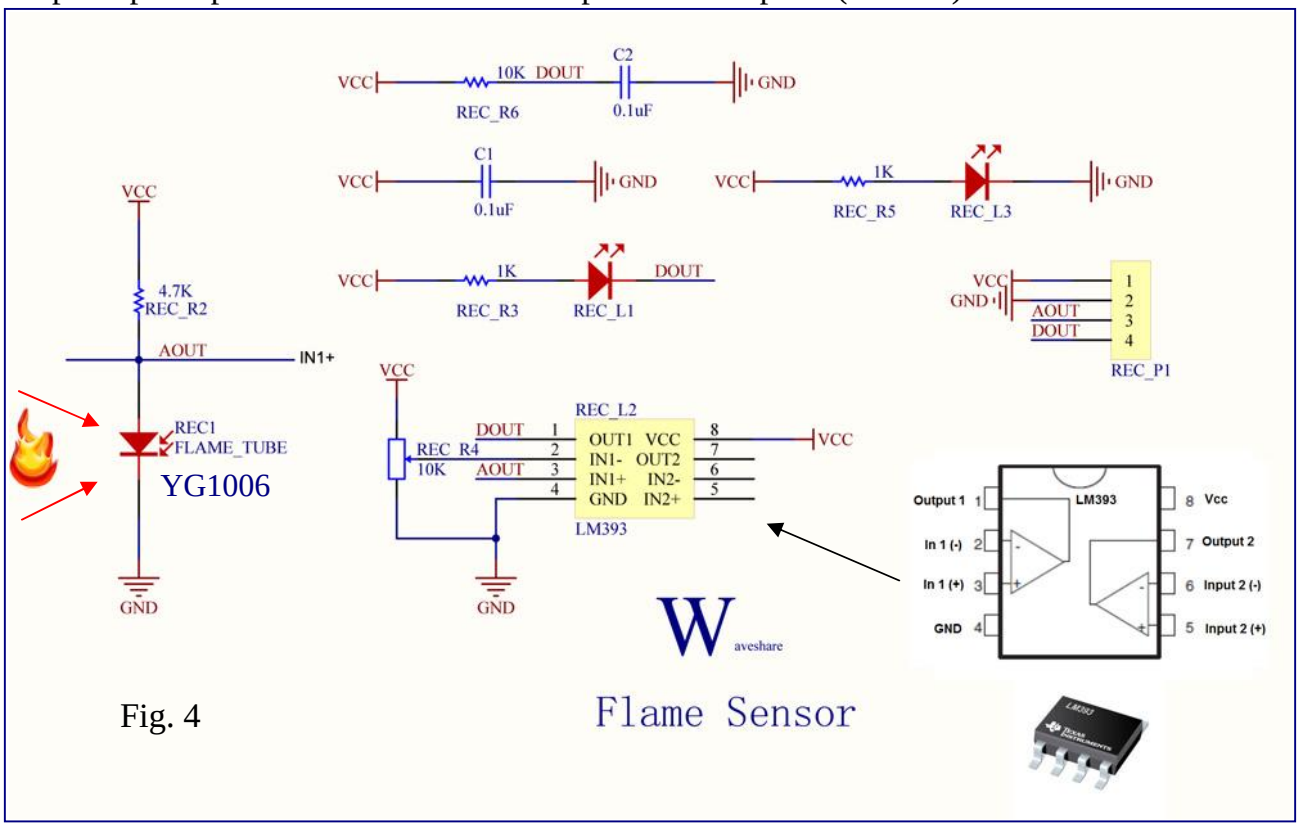


Fig. 4

Il trimmer collegato a IN- permette di impostare una tensione di riferimento mentre l'ingresso IN+ è collegato al diodo YG1006 (REC1 FLAME_TUBE).
L'uscita del comparatore (Dout) dipende dal confronto tra IN- e IN+.

Dout=LOW se IN->IN+ **Dout=HIGH se IN-<IN+**

In presenza di fiamma il diodo YG1006 si porta in saturazione (massima conduzione), il valore Aout tende ad un valore molto basso, prevale l'ingresso IN-, l'uscita del comparatore si porta a livello basso e si accende il diodo REC L1. In assenza di fiamma il diodo YG1006 risulta interdetto (minima conduzione), il valore Aout tende a Vcc, prevale l'ingresso IN+, l'uscita del comparatore si porta a livello alto e il diodo REC L1 risulta spento.

Il circuito dispone quattro terminali (Vcc, GND, DOUT, AOUT) e due diodi Led:

1)**Dout**: uscita digitale (*Livello Basso presenza di fiamma, livello Alto assenza di fiamma*).
2)**Aout**: uscita analogica (*tensione (0V÷5V) inversamente proporzionale all'intensità della fiamma: il valore diminuisce all'aumentare dell'intensità della fiamma*).
3)**Led Rec_L**.3: Led Power
4)**Led Rec_L1**: Led acceso in presenza di fiamma, Led spento in assenza di fiamma, la soglia di commutazione dipenda dal trimmer REC R4.

In Fig. 5 è riportato lo schema elettrico del sistema.

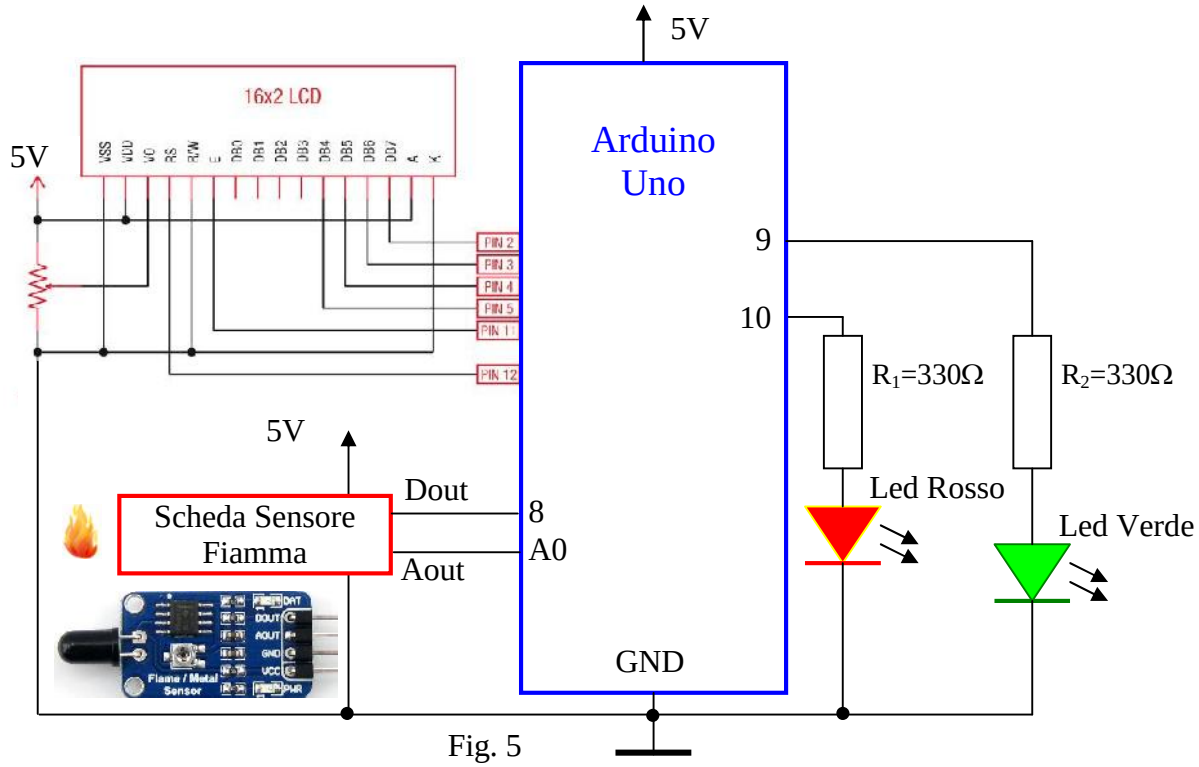


Fig. 5

Programma

```
// Esercizio 30: Sensore di fiamma
int flame_din=8;
int flame_ain=A0;
int ad_value;
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
void setup()
{
  pinMode(flame_din, INPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(flame_ain,INPUT);
  lcd.begin(16, 2);
  lcd.clear();
  lcd.print("Sensore di Fiamma");
}
void loop()
{
  ad_value=analogRead(flame_ain);
  lcd.setCursor(0,1);
  lcd.print("     ");
  lcd.setCursor(0,1);
  lcd.print(ad_value);
```

```
if (digitalRead(flame_din)==LOW)
    {
lcd.setCursor(5,1);
lcd.print("All. Fiamma");
    digitalWrite(9, LOW);
    digitalWrite(10, HIGH);
      }
  else
  {
   lcd.setCursor(5,1);
  lcd.print("  No Fiamma");
  digitalWrite(10, LOW);
  digitalWrite(9, HIGH);
  }
  delay(500);
  }
```

Esercizio 31 – Controllo motore a corrente continua (ON-OFF) e due velocità

Dimensionare e programmare un circuito per il controllo di un motore a corrente continua. Scrivere due programmi in grado di:

1) Tramite due pulsanti (**pr**, **pv**) attivare il controllo ON-OFF
2) Tramite due pulsanti (**pr**, **pv**) attivare il motore a due velocità (ridotta e massima)
   a. **Pr** avvia il motore a velocità ridotta,
   b. **Pv** avvia il motore alla massima velocità.
   c. La pressione simultanea di **Pr** e **Pv** ferma il motore

Lo stato di funzionamento deve essere visualizzato sul display LCD 16x2

Soluzione

In Fig. 1 è riportato lo schema elettrico del sistema.
Per i pulsanti (pin 8 , 9) si sfrutta la configurazione INPUT_PULLUP. Sul pin 10 il circuito driver per il motore mentre il display LCD 16x2 viene pilotato tramite i pin 2,3,4,5,11,12

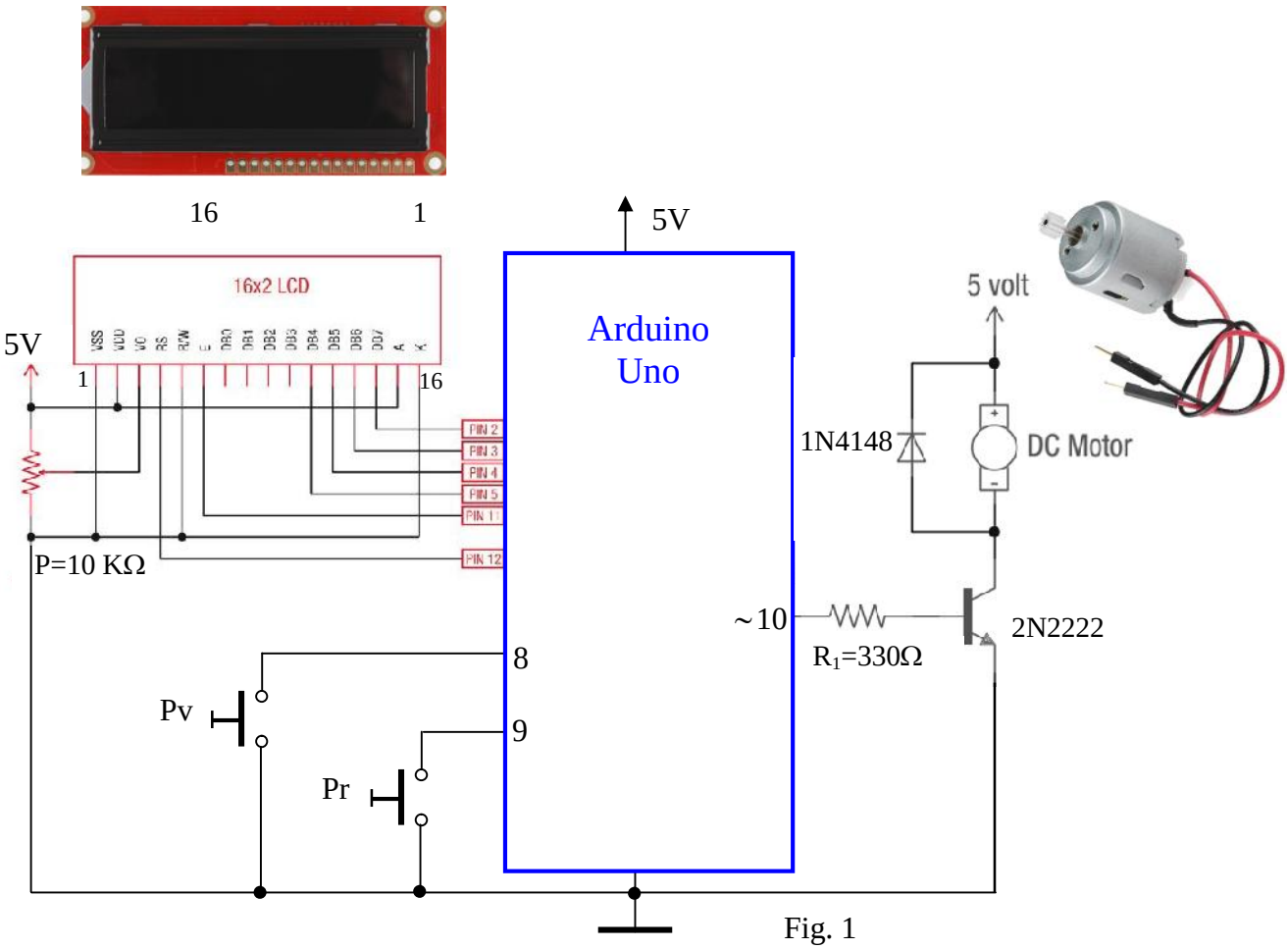


Fig. 1

Di seguito sono riportati i due programmi di gestione il primo relativo al punto 1 ed il secondo relativo al punto 2.

Programma 1

```
// Esercizio 31a: Controllo Motore in Corrente Continua - ON-OFF
int pr=9;   // Pin pulsante Avvio motore
int pv=8;  // Pin pulsante Stop motore
int m=10; // Pin Motore
int v=0;    // Var. velocità motore v=0 (Motore OFF); V=1 (Motore ON)

#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
void setup()
{
  pinMode(pr, INPUT_PULLUP);
  pinMode(pv, INPUT_PULLUP);
  pinMode(m, OUTPUT);
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Motore CC On-Off");
  lcd.setCursor(0,1);
  lcd.print("M=Off");
 }
void loop()
{
  if (digitalRead(pr)==LOW)
  {
    digitalWrite(m,HIGH);
    lcd.setCursor(0,1);
    lcd.print("M=On ");
    }
  if (digitalRead(pv)==LOW)
  {
    digitalWrite(m,LOW);
    lcd.setCursor(0,1);
    lcd.print("M=Off");
   }
    delay(600);
}
```

Programma 2

```
// Esercizio 31b: Controllo Motore in Corrente Continua – due velocità
int pr=9;    // Pin pulsante Velocità RIDOTTA
int pv=8;   // Pin pulsante Velocità MASSIMA
int m=10; // Pin Motore

#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
void setup()
{
  pinMode(pr,INPUT_PULLUP);
  pinMode(pv,INPUT_PULLUP);
  pinMode(m, OUTPUT);
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Motore CC 2 Vel.");
  lcd.setCursor(0,1);
  lcd.print("M=Off");
 }
void loop()
{
  if (digitalRead(pr)==LOW)
  {
    analogWrite(m,90);
    lcd.setCursor(0,1);
    lcd.print("Vel. =Ridotta");
    }

  if (digitalRead(pv)==LOW)
  {
    analogWrite(m,255);
    lcd.setCursor(0,1);
    lcd.print("Vel. =Massima");
   }
 if (digitalRead(pv)==LOW && digitalRead(pr)==LOW)
{
    digitalWrite(m,LOW);
    lcd.setCursor(0,1);
    lcd.print("M=Off        ");
   }
delay(700);
}
```

Esercizio 32 – Flex Sensor e Servo Motore

Dimensionare e programmare un circuito in grado di far ruotare il perno di un servo motore da 0°a 180°tramite la flessione di un Flex sensor.

1. Disegnare lo schema elettrico
2. Scrivere il programma in grado di far ruotare il perno da 0°a 180° tramite la flessione del Flex Sensor e visualizzare la posizione sul diplay LCD 16x2

Soluzione

In Fig. 1 è riportato lo schema elettrico del sistema.

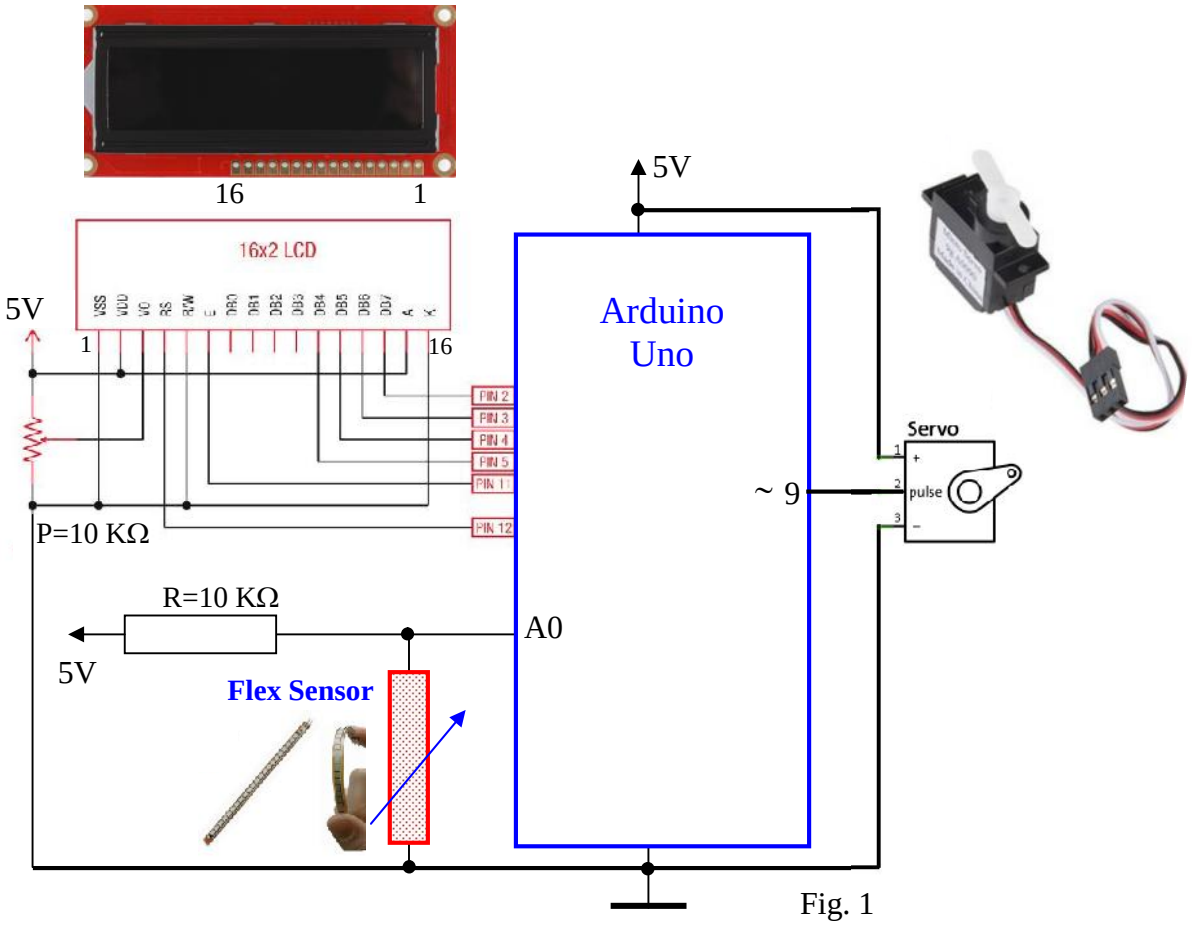


Fig. 1

Programma

```
// Esercizio 32: Flex Sensor e Servo Motore
int pservo=9;         // Pin Servo
int flexposition;    // Valore Flex Sensor
int servoposition;  // Posizione Servo

#include <Servo.h>
Servo servo1;

#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
void setup()
{
  servo1.attach(pservo);

  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("FlexSensor-Servo");

 }
void loop()
{
  flexposition = analogRead(A0);
  servoposition = map(flexposition, 700, 900, 0, 180);
  servoposition = constrain(servoposition, 0, 180);
  servo1.write(servoposition);
    lcd.setCursor(0,1);
    lcd.print("Flex:");
    lcd.setCursor(5,1);
    lcd.print("    ");
    lcd.setCursor(5,1);
    lcd.print(flexposition);
lcd.setCursor(9,1);
    lcd.print(">");
    lcd.setCursor(11,1);
    lcd.print("S:");
    lcd.setCursor(13,1);
    lcd.print("   ");
    lcd.setCursor(13,1);
    lcd.print(servoposition);
delay(55);
}
```

Esercizio 33 – Modulo Sensore Temperatura e Umidità con DHT11

Dimensionare e programmare un circuito basato sul modulo Sensore Temperatura Umidità DHT11 in grado di visualizzare su un display LCD 16x2 la temperatura (°C), l'umidità (%) e la temperatura di rugiada (°C).

Soluzione

In Fig. 1 è riportato lo schema elettrico del sistema.

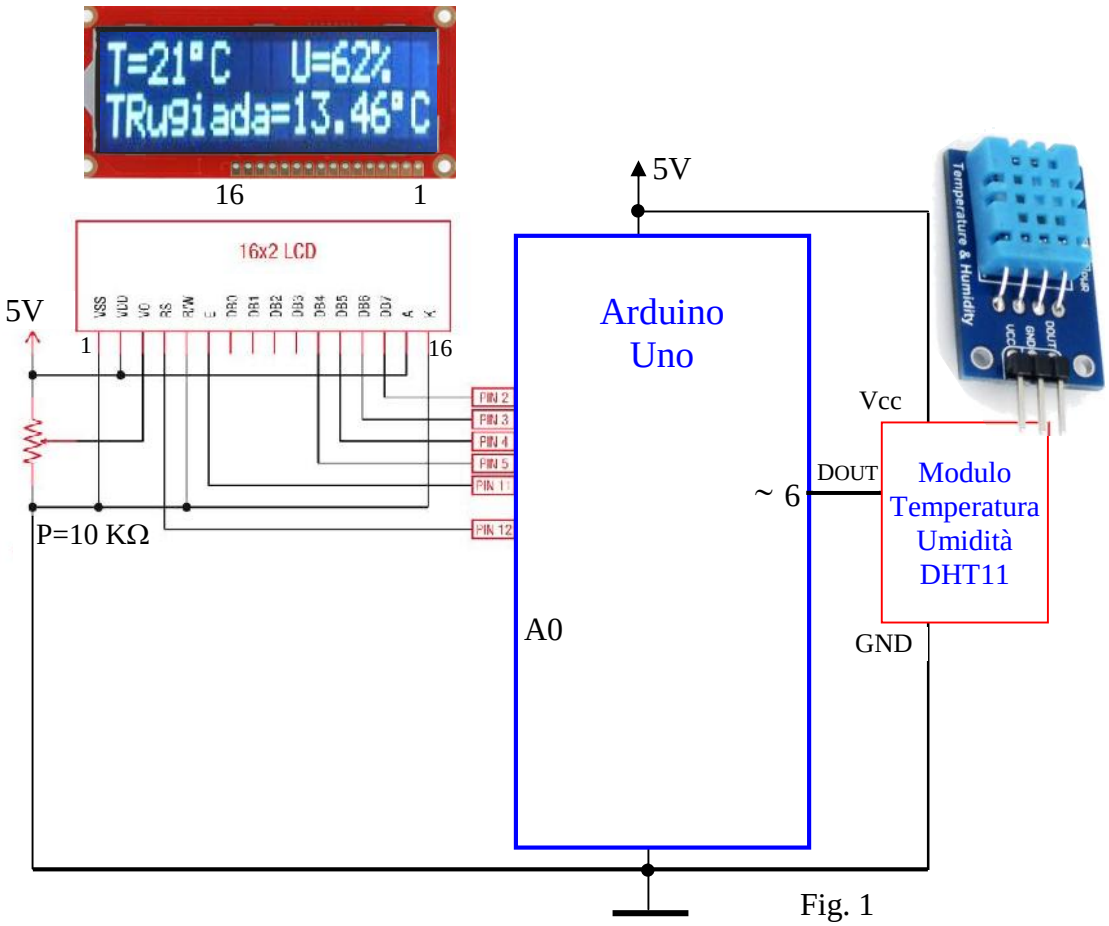


Fig. 1

Modulo Temperatura Umidità DHT11

Il modulo utilizza il sensore DHT11, sensore di temperatura e umidità con uscita dei dati in formato digitale. Esso rileva i valori di umidità e temperatura e, attraverso un

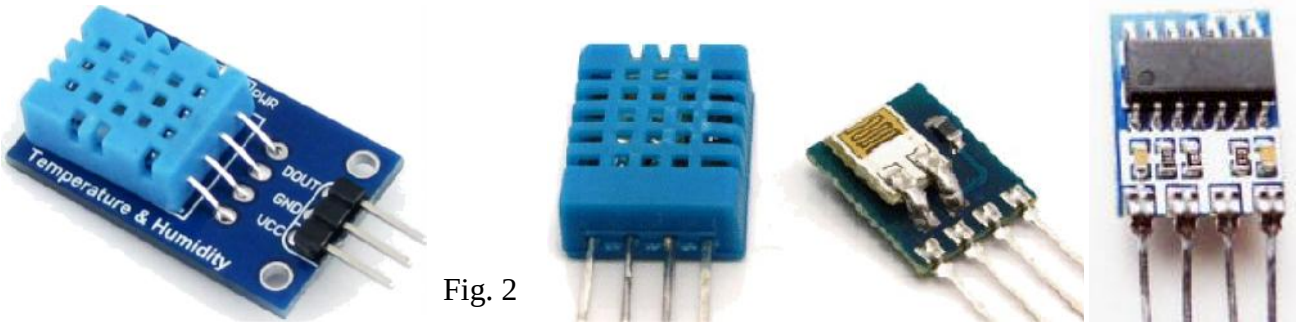


Fig. 2

microcontrollore ad 8 bit in esso racchiuso, li trasforma in segnali digitali.

Si tratta di quindi di un componente evoluto, composto da un sensore di umidità di tipo resistivo, un sensore di temperatura di tipo NTC (Negative Temperature Coefficient – un sensore analogico che diminuisce l'impedenza di una resistenza all'aumentare della temperatura) e da un microcontrollore.

Esistono diversi moduli della serie DHT, il modulo 11, trattato in questo esercizio, è il più semplice ed il più economico. Il DHT11 fornisce solo .00 per la parte decimale.
Questo sensore, su sollecitazione di Arduino, trasmette sul pin DOUT (Uscita digitale) un treno di quaranta bit:

- 8 bit per indicare la parte intera del valore di umidità
- 8 bit per indicare la parte decimale dell'umidità (.00 nel modello DHT11)
- 8 bit per indicare la parte intera della temperatura
- 8 bit per indicare la parte decimale della temperatura (.00 nel modello DHT11)
- 8 bit per indicare il numero di controllo (per validare il valore dei precedenti 32 bit)

Modello	DHT11
Alimentazione	3-5.5V DC
Segnale di uscita	digitale del segnale tramite single-bus
Elemento sensibile	Resistenza in Polimero
Campo di misura umidità	20-90% di umidità relativa, temperatura di 0-50 gradi Celsius
Precisione	umidità + -4% RH (Max + -5% di umidità relativa), temperatura +-2.0Celsius
Risoluzione o la sensibilità	umidità 1% di umidità relativa, temperatura 0.1Celsius
Ripetibilità umidità	+ -1% di umidità relativa temperatura +-1Celsius
Umidità isteresi	+ -1% RH
Stabilità a lungo termine	+ -0.5% UR / anno
Tempo di rilevazione	medio: 2s
Dimensioni	12 * 15,5 * 5,5 millimetri

Caratteristiche tecniche

In fig. 3 è riportato lo schema elettrico del modulo.

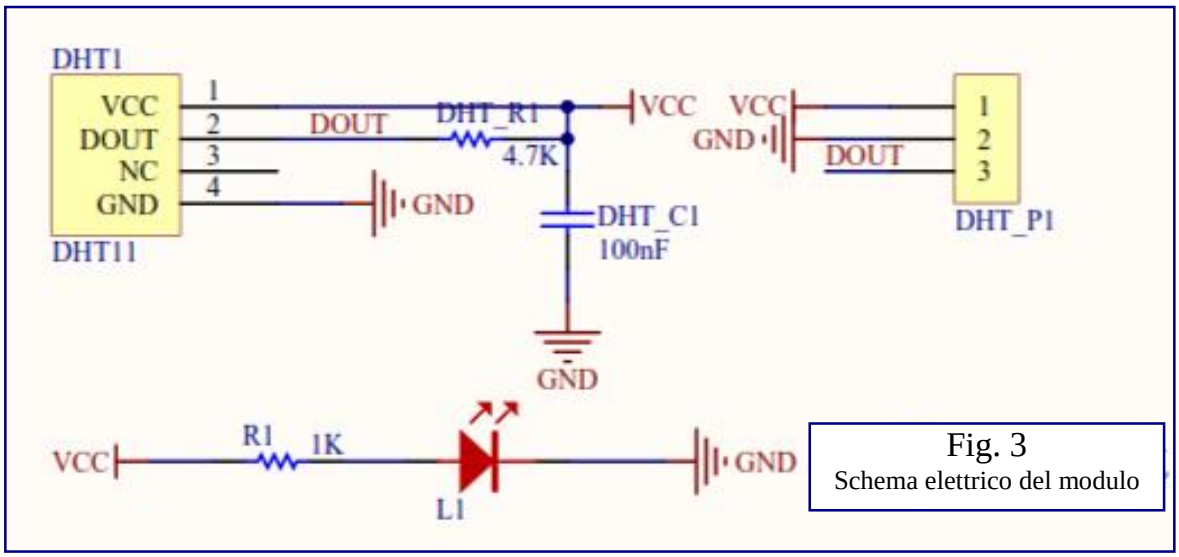


Fig. 3
Schema elettrico del modulo

La gestione del segnale è a carico di una libreria (**dht11.h**) che deve essere scaricata ed installata prima della compilazione del programma.

Download e installazione libreria

1. Link libreria: www.ipsia.gov.it/Documenti/Arduino/Librerie/dht11.zip
2. Installare la libreria : Menu Sketch → Inclusione librerie → Aggiungi una libreria da file .ZIP

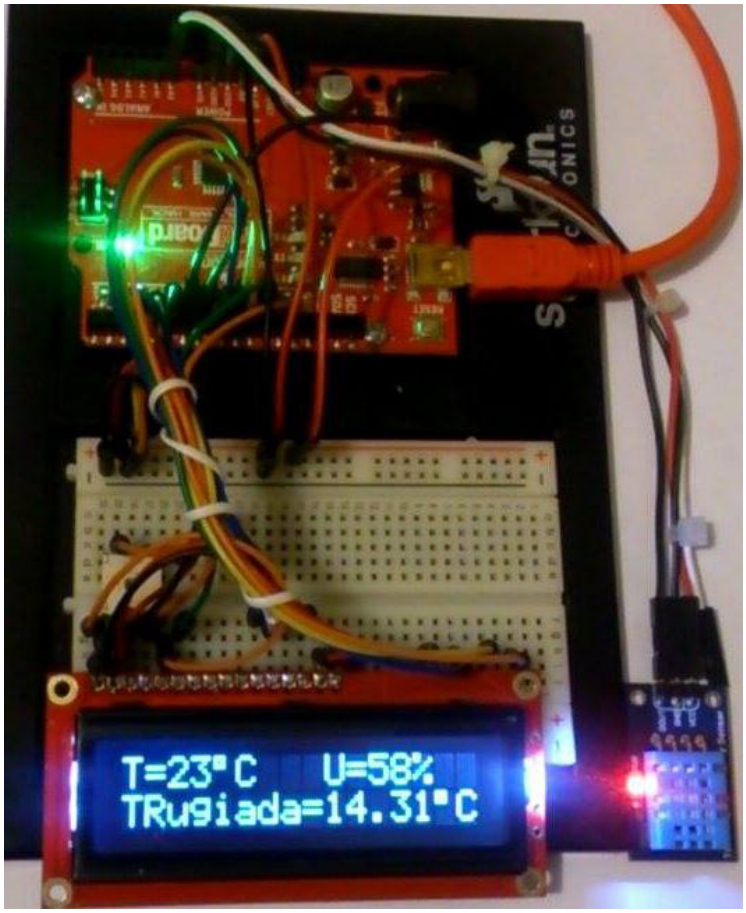
Temperatura di rugiada

Con punto di rugiada o temperatura di rugiada ("**dew point**") si intende la temperatura alla quale, a pressione costante, l'aria (o, più precisamente, la miscela aria-vapore) diventa satura di vapore acqueo. In meteorologia in particolare, indica a che temperatura deve essere portata l'aria per farla condensare in rugiada, senza alcun cambiamento di pressione. Se il punto di rugiada cade sotto 0 °C, esso viene chiamato anche punto di brina.
In estate essa ci fornisce un'idea immediata della sensazione di calore sul nostro organismo: temperature di rugiada superiori ai 17°C sono sintomo di una debole afa, quando invece superano i 21°C, l'afa comincia a diventare fastidiosa.
Inoltre il dew point da una rappresentazione anche di quello che è il "carburante" disponibile per lo sviluppo di temporali, più il dew point è elevato più gli eventuali fenomeni temporaleschi potranno essere intensi.
Nel programma il calcolo viene effettuato dalla funzione **dewPoint**, all'interno l'algoritmo che determina il valore della temperatura di rugiada in base al valore della temperatura e dell'umidità.

Programma

```
/* Esercizio 33: Modulo Sensore Temperatura e Umidità con DHT11
 Il programma rileva visualizza sul display i seguenti dati:
 Temperatura in °C - Umidità in % - Temperatura di rugiada in °C
 La temperatura di rugiada viene calcolata con la funzione dewPoint
*/
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);

#include <dht11.h>
dht11 DHT11;
#define DHT11PIN 6

double dewPoint(double celsius, double humidity)  // Calcolo temperature di rugiada
{
      double A0= 373.15/(273.15 + celsius);
      double SUM = -7.90298 * (A0-1);
      SUM += 5.02808 * log10(A0);
      SUM += -1.3816e-7 * (pow(10, (11.344*(1-1/A0)))-1) ;
      SUM += 8.1328e-3 * (pow(10,(-3.49149*(A0-1)))-1) ;
      SUM += log10(1013.246);
      double VP = pow(10, SUM-3) * humidity;
      double T = log(VP/0.61078);   // temp var
      return (241.88 * T) / (17.558-T);
}

void setup()
{
  lcd.begin(16, 2);
  lcd.clear();
  }
```

```
void loop()
{
 int chk = DHT11.read(DHT11PIN);
 int h = DHT11.humidity;
 int t = DHT11.temperature;

   lcd.setCursor(0,0);
   lcd.print("T=");
   lcd.print(t,DEC);
   lcd.print(char(223));
   lcd.print("C");
   lcd.setCursor(9, 0);
   lcd.print("U=");
   lcd.print(h,DEC);
   lcd.print("%");
   lcd.setCursor(0,1);
   lcd.print("TRugiada=");
   lcd.print(dewPoint(DHT11.temperature, DHT11.humidity),2);
   lcd.print(char(223));
   lcd.print("C");

delay(2000);  //ritardo 2 sec
}
```



Cablaggio e collaudo

Esercizio 34 – Modulo Sensore magnetico – Effetto Hall

Dimensionare e programmare un circuito basato sul modulo Sensore Magnetico ad Effetto Hall in grado di visualizzare su un display LCD 16x2 il numero decimale in uscita al convertitore A/D, la tensione di uscita del modulo e l'induzione magnetica B in milliTesla [mT].

Soluzione

In Fig. 1 è riportato lo schema elettrico del sistema

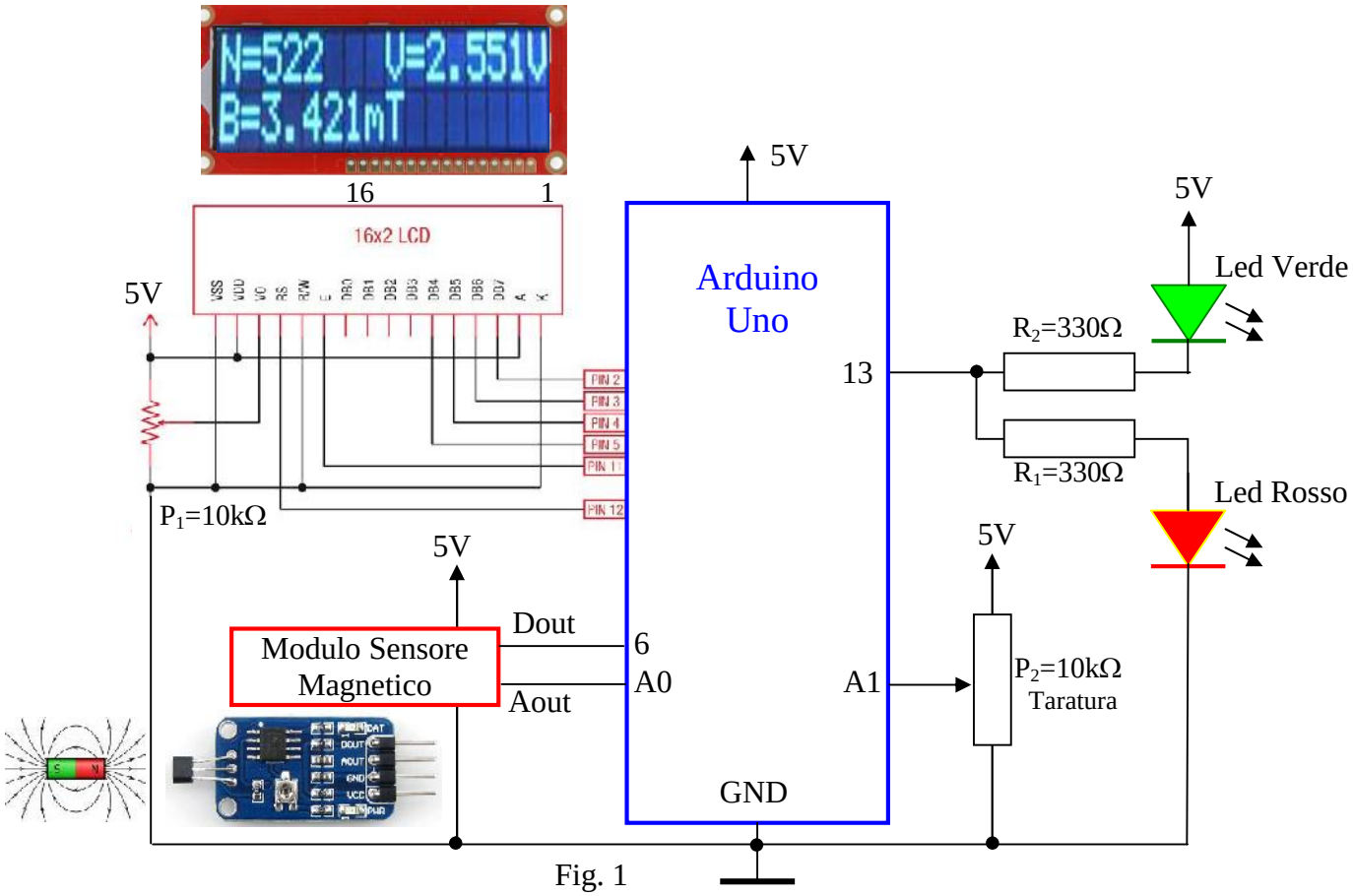
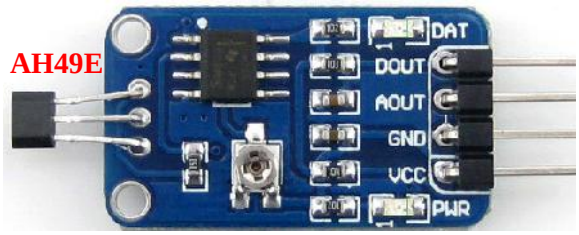


Fig. 1

Modulo magnetico ad effetto Hall

Il modulo utilizza il sensore AH49E in grado di rilevare la presenza di un campo magnetico (induzione magnetica B) generato da un magnete permanente (calamita) o elettromagnete.



Il sensore AH49E sfrutta l'effetto Hall e fornisce in uscita una tensione direttamente proporzionale all'induzione magnetica B a cui è sottoposto.

In fig. 2 è riportata la caratteristica di trasferimento (V=f(B)).

Transfer Characteristics (V_{CC}=5V)

When there is no ouside magnetic field (B=0GS), the quiescent output voltage is one-half the supply voltage in general. If a south magnetic pole approches to the front face (the side with marking ID) of the Halleffect sensor, the circuit will drive the output voltage higher. Contrary, a north magnetic pole will drive the output voltage lower. The variations of voltage level up or down are symmetrical.

Greatest magnetic sensitivity is obtained with a supply voltage of 6V, but at the cost of increased supply current and a slight loss of output symmetry. So, it is not recommended to work in such condition unless the output voltage magnitude is a main issue. The output signal can be capacitively coupled to an amplifier for boosting further if the changing frequency of the magnetic field is high.

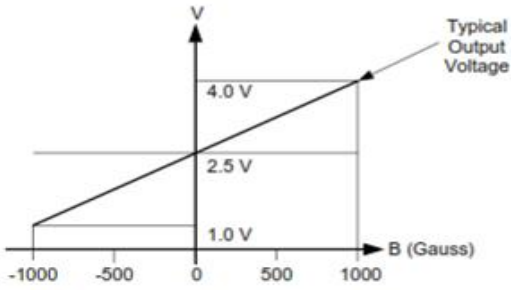


Fig. 2 The Transfer Characteristics of AH49E

$$V = \frac{3}{2000} \cdot B + 2{,}5 \qquad B = \frac{(V - 2{,}5) \cdot 2000}{3}$$

In fig. 3 sono riportate le caratteristiche generali del sensore AH49.

LINEAR HALL-EFFECT IC **AH49E**

General Description

The AH49E is a small, versatile linear Hall-effect device that is operated by the magnetic field from a permanent magnet or an electromagnet. The output voltage is set by the supply voltage and varies in proportion to the strength of the magnetic field.

The integrated circuitry features low noise output, which makes it unnecessary to use external filtering. It also includes precision resistors to provide increased temperature stability and accuracy. The operating temperature range of these linear Hall sensors is -40°C to 85°C, appropriate for commercial, consumer and industrial applications.

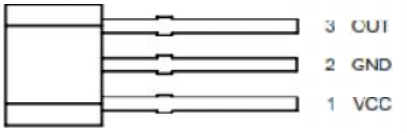
The AH49E is available in standard TO-92S package.

Features

- Miniature Construction
- Power Consumption of 3.5mA at V_{CC}=5V for Energy Efficiency
- Single Current Sourcing Output
- Linear Output for Circuit Design Flexibility
- Low Noise Output Virtually Eliminates the Need for Filtering
- A Stable and Accurate Output
- Temperature Range of -40°C to 85°C
- Responds to Either Positive or Negative Gauss

Applications

- Current Sensing
- Motor Control
- Position Sensing
- Magnetic Code Reading
- Ferrous Metal Detector
- Vibration Sensing
- Liquid Level Sensing
- Weight Sensing



Pin Configuration of AH49E (Bottom View)

Electrical Characteristics

(V_{CC}=5V, T_A=25°C, unless otherwise specified.)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Supply Current	I_{CC}			3.5	4.5	mA
Quiescent Output Voltage	V_{NULL}	@ B=0GS	2.25	2.5	2.75	V
Output Voltage Sensitivity		B=0GS to ±1000GS	1.1	1.6	2.1	mV/GS
Output Voltage Span	V_{OS}		1.0 to (V_{CC}-1.0)	0.8 to (V_{CC}-0.8)		V
Output Resistor	R_O			60	120	Ω
Magnetic Field Range	B		±650	±1000		GS
Linearity of Span				0.7		%
Output Noise		BW=10Hz to 10kHz		90		μV

Fig. 3 – Estratto dal datasheet

Nota: nel SI l'unita di misura dell'induzione magnetica B è il in Tesla [T] (1T=10-^{4} GS)

In Fig. 4 è riportato lo schema elettrico del modulo magnetico ad effetto Hall basato sul sensore AH49E.

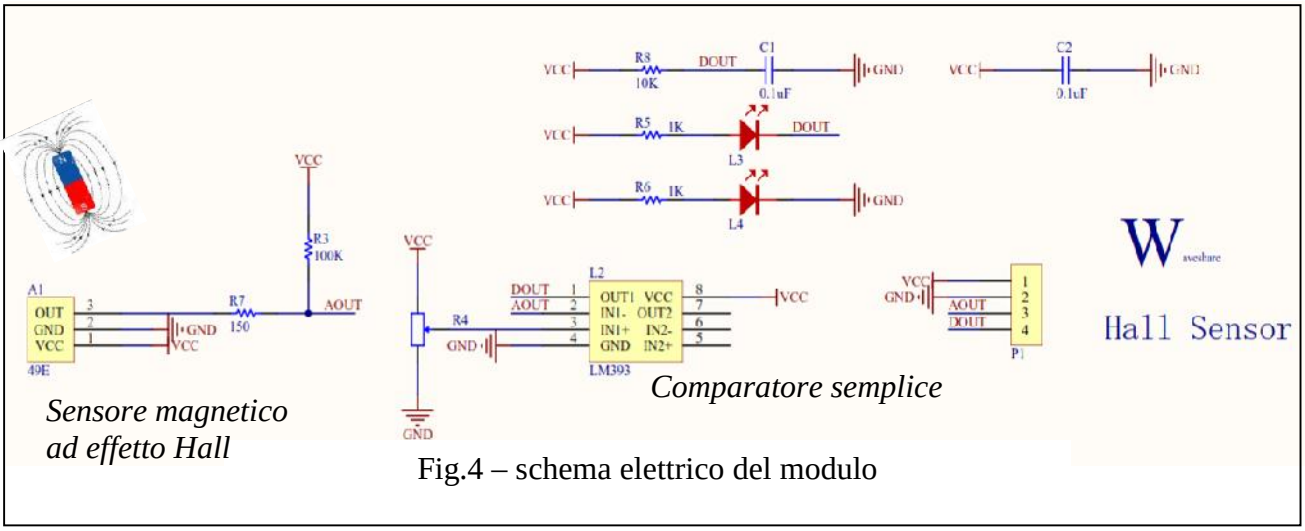


Fig.4 – schema elettrico del modulo

Il modulo fornisce due uscite AOUT (analogica) e DOUT (digitale, uscita del comparatore LM393). Il trimmer collegato a IN+ permette di impostare una tensione di riferimento mentre l'ingresso IN- è collegato ad AOUT (uscita analogica fornita da sensore tramite il partitore R3, R7)

L'uscita del comparatore (Dout) dipende dal confronto tra IN- e IN+.

Dout=LOW se IN->IN+ **Dout=HIGH se IN-<IN+**

In presenza di campo magnetico prevale l'ingresso IN-, l'uscita del comparatore si porta a livello basso e si accende il diodo L3 (altrimenti i diodo risulta spento).

Il circuito dispone quattro terminali (Vcc, GND, DOUT, AOUT) e due diodi Led:

1) **Dout**: uscita digitale (*Livello Basso presenza di campo magnetico*).
2) **Aout**: uscita analogica (*tensione (1V÷4 V) pe B=±1000 G).*
3) **Led L4**: Led Power
4) **Led L3**: Led acceso in presenza di campo magnetico.

L'Effetto HALL

L'effetto Hall (Fig.5 e Fig.6) è la formazione di una differenza di potenziale, detto potenziale di Hall (VH), sulle facce opposte di un conduttore elettrico dovuta a un campo magnetico (induzione magnetica B) perpendicolare alla corrente elettrica (I) che scorre in esso.

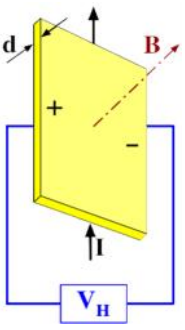


Fig. 5

L'effetto prende il nome dal fisico Edwin Hall che per primo lo scoprì nel 1879.

Quando sui terminali di un conduttore si applica una tensione continua V (Fig. 6), questa promuove un flusso di elettroni uniforme dal punto A verso il punto B, senza che, tra due punti estremi di una sezione trasversale del conduttore (C - D) sussista alcuna differenza di potenziale.

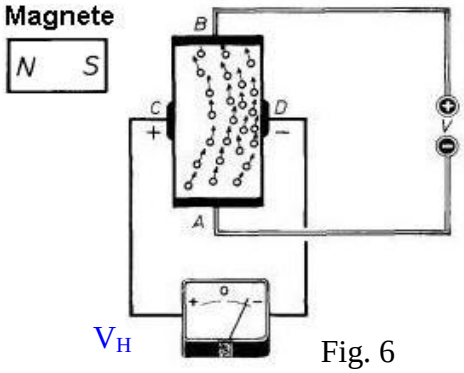


Fig. 6

L'indice del voltmetro, infatti, rimane fermo sullo zero centrale.

Se, invece, come indicato nello schema (Fig. 5), si avvicina un magnete (Polo SUD) al conduttore, il flusso di elettroni subisce una deviazione dal percorso rettilineo, con un certo ammassamento verso il punto D ed un diradamento nella zona prossima al punto C.

Il risultato elettrico più appariscente è quello della presenza di una tensione, fra i punti C - D, segnalata dall'indice del voltmetro. Invertendo le polarità del magnete, anche il corrispondente concentramento di elettroni e la deviazione dell'indice dello strumento si invertono. Con il risultato che il punto C è questa volta più negativo del punto D e l'indice del voltmetro flette verso i valori positivi.

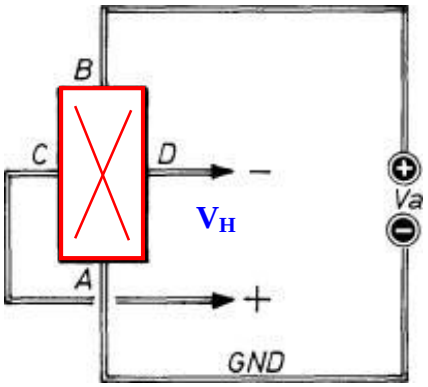


Fig. 7 - Schema simbolico di un sensore ad effetto Hall

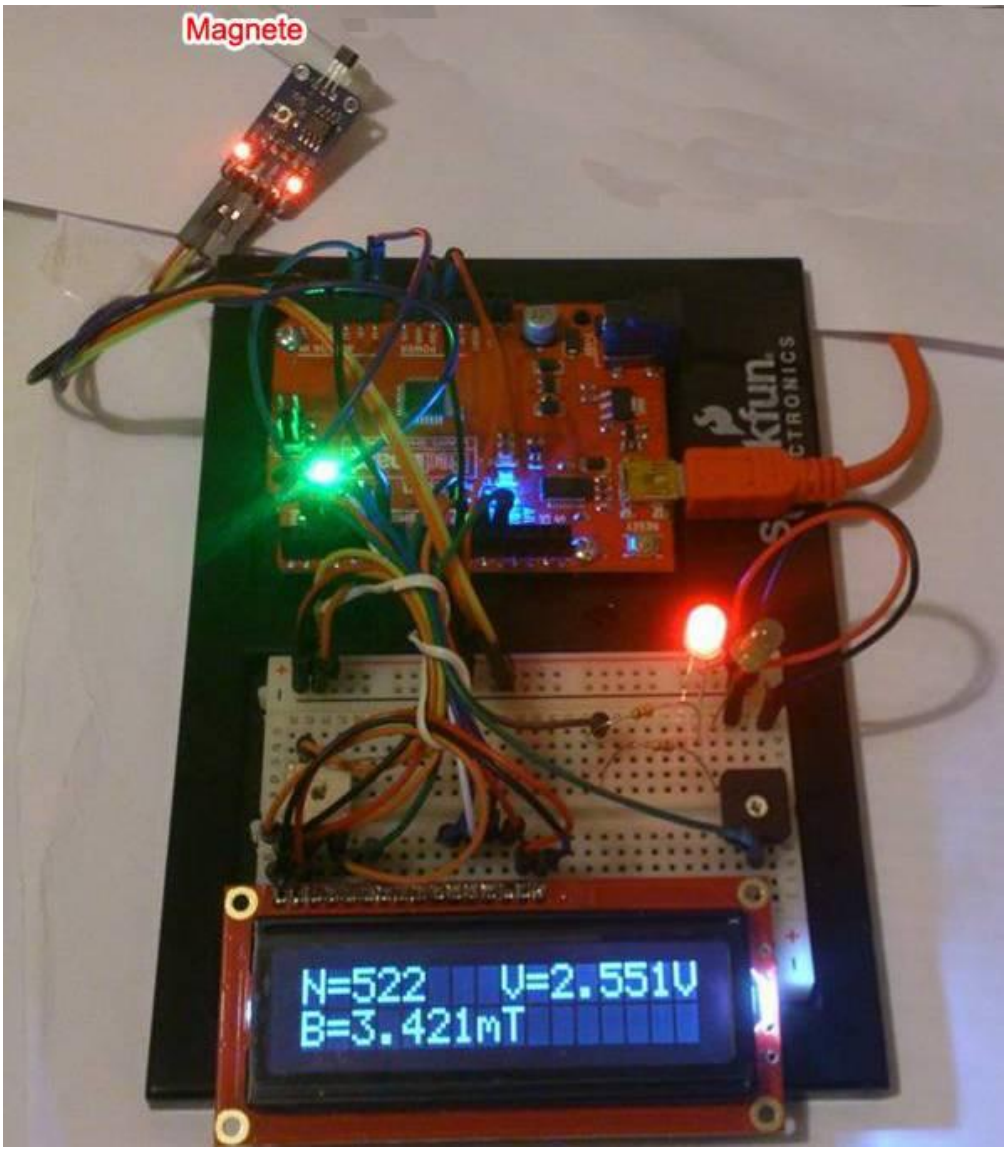
Più simbolicamente il concetto di tensione di Hall (V_H) si esprime attraverso lo schema riportato in Fig.7, nel quale con Va si indica la tensione di alimentazione del circuito, mentre con GND è segnalata la linea di terra (ground = terra).

In generale, dunque, la tensione di Hall si manifesta quando un campo magnetico (induzione magnetica B) coinvolge, trasversalmente, un conduttore percorso da corrente ed è rilevabile fra le estremità delle due sezioni perpendicolari.

Programma

```
/* Esercizio 34: Modulo Sensore Magnetico - Effetto Hall
 --- Rileva l'induzione magnetica B e visualizza:
---  N (uscita del conv. A/D),  la tensione d' uscita del modulo
 --- L'induzione magnetica B [mT]
*/
#include <LiquidCrystal.h>  // Libreria gestione display
LiquidCrystal lcd(12,11,5,4,3,2);
int pindout=6;       // Uscita digitale del modulo
int pinaout=A0;      // Uscita analogica del modulo
int pintaratura=A1;  // Pin taratura
int ledvero=13;      // Pin Led Rosso e Verde

void setup()
{
  lcd.begin(16, 2);
  lcd.clear();
  pinMode(pindout,INPUT);
  pinMode(ledvero,OUTPUT);
}

void loop()
{
  if(digitalRead(pindout)==LOW)
  {
    digitalWrite(ledvero,HIGH);
  }
  else
  {
    digitalWrite(ledvero,LOW);
  }
 int a=analogRead(pinaout);
```

```
 int at=map(analogRead(pintaratura), 0,1023, -50, 50); // valore taratura
 a=a+at;
 float v=a*5.0/1023;
 float b=(v-2.5)*2000/3.0;  // B in Gauss
 double bt=b/10.0; // B in MilliTesla
   lcd.setCursor(0,0);
   lcd.print("N=");
   lcd.print(a,DEC);
   lcd.print("   V=");
   lcd.print(v,3);
   lcd.print("V");
   lcd.setCursor(0, 1);
   lcd.print("B=           ");
   lcd.setCursor(2, 1);
   lcd.print(bt,3);
   lcd.print("mT");
delay(500);
}
```



Cablaggio e collaudo

Esercizio 35 – Modulo Sensore di Tilt

Dimensionare e programmare un circuito basato sul modulo Sensore di Tilt in grado di rilevare con un avviso acustico e luminoso se un oggetto si è spostato (Vibrazione oppure Rotazione) rispetto ad una posizione originale di riferimento.

Soluzione

In Fig. 1 è riportato lo schema elettrico del sistema.

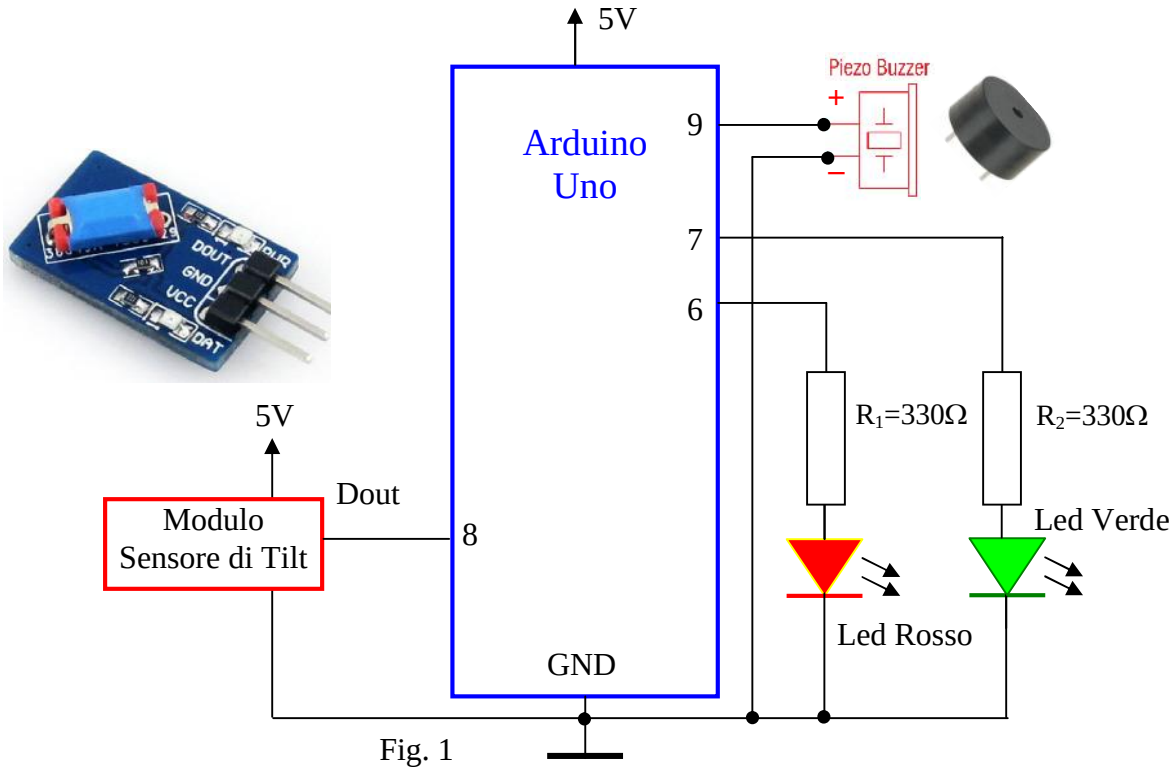


Fig. 1

Modulo sensore di Tilt

Il modulo (Fig. 2) contiene un piccolo tubo all'interno del quale scorre una pallina di metallo. Se si inclina il tubo da una parte, la pallina raggiunge un estremo e chiude un contatto. Si tratta di un interruttore sensibile all'inclinazione che apre oppure chiude un contatto in base alla posizione della pallina.

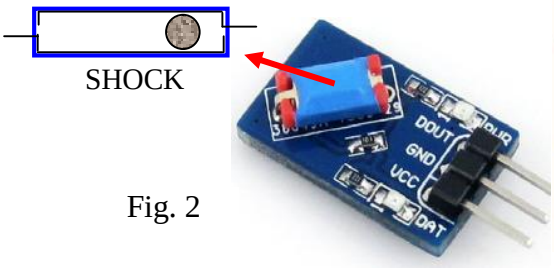


Fig. 2

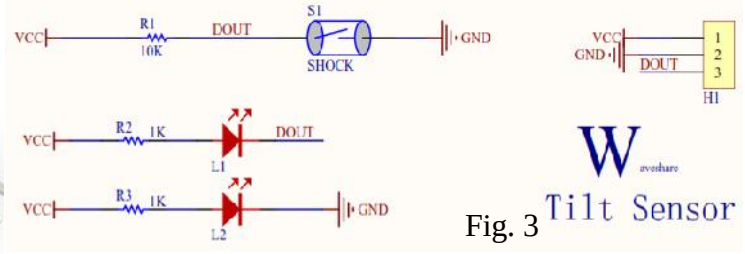


Fig. 3

In Fig. 3 è riportato lo schema elettrico del modulo sensore di Tilt

Il circuito dispone tre terminali (Vcc, GND, DOUT) e due diodi Led:

1) **Dout**: uscita digitale (*SHOCK=Chiuso → Dout=Livello Basso).*
2) **Led L1**: Led Power
3) **Led L2**: Led acceso in presenza di movimento (*SHOCK=Chiuso*).

Viene utilizzato oltre che per rilevare del movimento anche per rilevare vibrazione, in allarmi antifurto.

Programma

```
/* Esercizio 35: Modulo Sensore di Tilt
 --- Rileva con un avviso acustico e luminoso se un oggetto
 --- si è spostato (Vibrazione oppure Rotazione)
 --- rispetto ad una posizione originale di riferimento
*/

int tilt_pin=8;
int ledverde=7;
int ledrosso=6;
int buzzer=9;
int frequenza=800;

void setup()
{
  pinMode(tilt_pin,INPUT);
  pinMode(ledverde, OUTPUT);
  pinMode(ledrosso, OUTPUT);
  pinMode(buzzer, OUTPUT);
}
void loop()
{
  if(digitalRead(tilt_pin)==LOW)
  {
    digitalWrite(ledverde,LOW);
    digitalWrite(ledrosso,HIGH);
    tone(buzzer,frequenza,200);
    delay (100);
    digitalWrite(ledrosso,LOW);
    delay (100);
    frequenza=frequenza+100;
    if (frequenza>5000)    {frequenza=800;}
  }
  else
  {
    frequenza=800;
    digitalWrite(ledverde,HIGH);
    digitalWrite(ledrosso,LOW);
    noTone(buzzer);
  }

}
```

Esercizio 36 – Modulo Sensore Umidità del terreno

Dimensionare e programmare un circuito basato sul modulo Sensore umidità del terreno in grado di:

1. Rilevare e visualizzare su un display Lcd 16x2 l'umidità rilevata dal sensore;
2. se l'umidità è inferiore al 20%:
   a. visualizzare "Asciutto", accendere un Led Rosso e attivare un segnale acustico
   b. attivare una pompa per l'irrigazione goccia a goccia;
3. se l'umidità è compresa tra il 20% e il 60%
   a. visualizzare "Umido" e far lampeggiare il Led verde (situazione ottimale)
4. se l'umidità è superiore al 60%
   a. visualizzare "Bagnato" e far lampeggiare i led rosso e verde

Nei punti 2 e 3 la pompa deve essere disattivata, inoltre deve essere previsto un pulsante per il test del sistema.

Soluzione

In Fig. 1 è riportato lo schema elettrico del sistema.

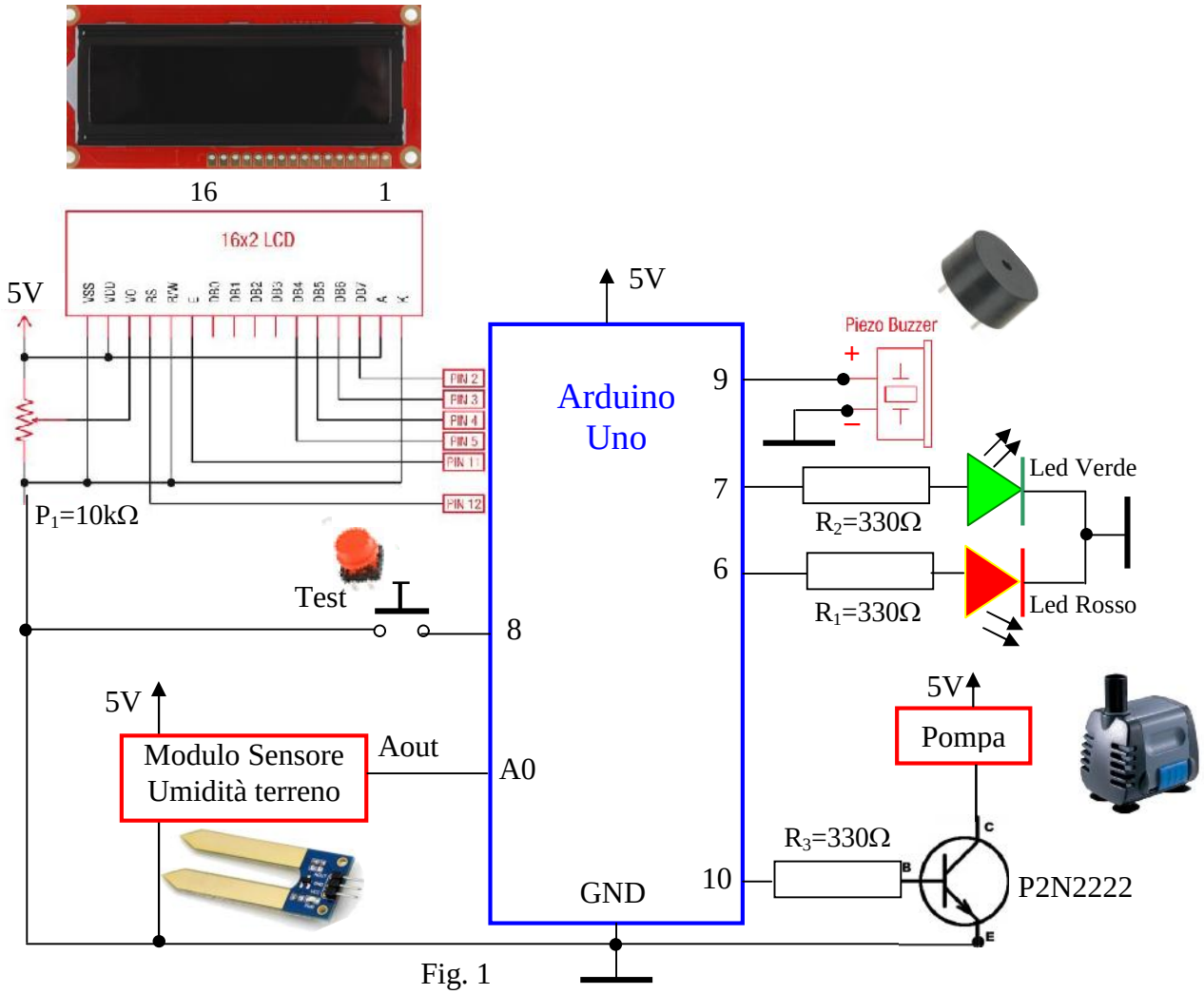


Fig. 1

Modulo sensore umidità terreno

In Fig. 2 è riportato lo schema elettrico del modulo sensore di umidità

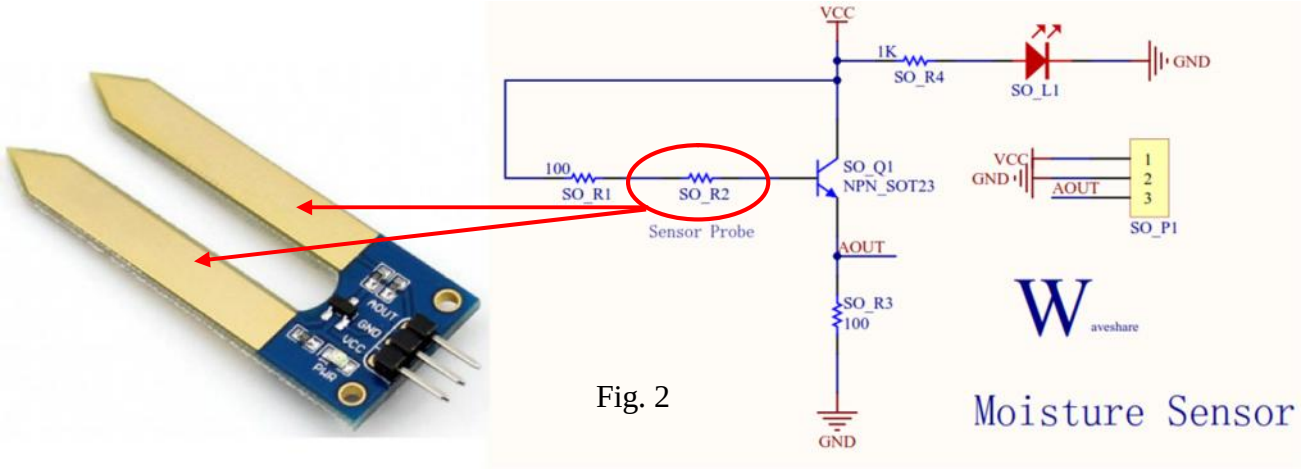


Fig. 2

Lo schema di Fig. 2 mostra un tipico amplificatore di corrente a transistor.
La resistenza di base del transistor è costituita da due resistenze in serie: una fissa da 100 Ω (SO_R1) e da una variabile il cui valore dipende dal grado di umidità del suolo (SO_R2).
La tensione di uscita (AOUT), funzione della corrente di emettitore, è prelevata ai capi della resistenza di emettitore SO_R3.
Se il suolo presenta un grado di umidità basso (terreno secco) il valore di SO_R2 è elevato, la corrente di base del transistor è bassa.
Di conseguenza la corrente di emettitore è bassa e AOUT assume valori bassi.
Man mano che l'umidità aumenta, la resistenza SO_R2 diminuisce, la corrente di base del transistor aumenta facendo aumentare la corrente di emettitore con relativo aumento di AOUT. Questa tensione viene applicata all'ingresso del convertitore A/D di Arduino tramite il pin A0.
La tensione AOUT è funzione dell'umidità relativa a cui è sottoposto in sensore immerso nel terreno.
Il modulo dispone tre terminali (Vcc, GND, AOUT) e un diodo Led:

1. **Aout**: uscita analogica.
2. **Led SO_L1**: Led Power
3. **Vcc e GND (** Vcc=5V, GND=Massa).

Pulsante test

Il pulsante (Fig. 3) è collegato tra il pin 8 e GND, il pin deve essere abilitato tramite software come input con resistenza di PullUp.

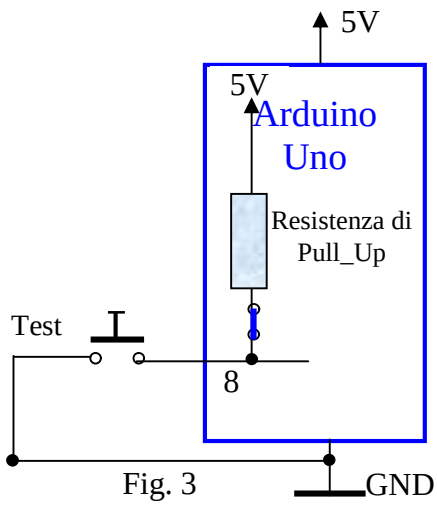


Fig. 3

1) **Test=OFF** (pulsante aperto) il pin 8 si trova a livello alto (5V), il software legge il valore dell'umidità fornita da sensore (ingresso A0).
2) **Test=ON** (pulsante chiuso) il pin 8 si trova a livello basso (GND), il software esclude A0 e tramite un contatore viene simulata l'umidità da 0% a 85% in questo modo è possibile testare il funzionamento delle tre fasi.

Gestione pompa

La pompa alimentata a 5V viene attività tramite il transistor P2N2222 (Fig 4).

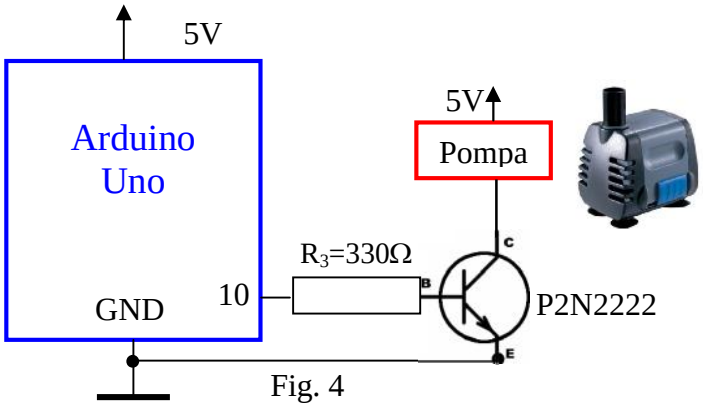
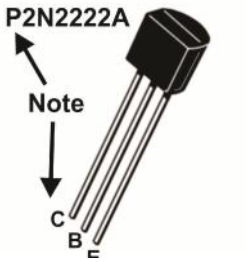


Fig. 4

Si utilizza il circuito di figura perché la corrente di uscita sul Pin 10 non è sufficiente a pilotare la pompa.
Il transistor (NPN) viene polarizzato per il funzionamento come interruttore.



1) **Pin10=HIGH** (*transistor in saturazione*) massima corrente di base, massima corrente di collettore, **pompa on** (attivata).
2) **Pin10=LOW** (*transistor interdetto*) minima corrente di base, minima corrente di collettore, **pompa off** (spenta).

Programma

```
/* Esercizio 36: Modulo Sensore Umidità Terreno
1.  Rileva e visualizza su un display Lcd 16x2 l'umidità rilevata dal sensore;
2.  se l'umidità è inferiore al 20%:visualizza "Asciutto"
      a) accende un Led Rosso e attiva un segnale acustico
      b) attiva una pompa per l'irrigazione goccia a goccia;
3.  se l'umidità è compresa tra il 20% e il 60%
      a) visualizza "Umido"  e  lampeggia  il Led Verde (situazione ottimale)
4.  se l'umidità è superiore al 60%
      a) visualizza "Bagnato" e lampeggia i Led Rosso e Verde
*/
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
int Moisture_ain=A0;
int ledverde=7;
int ledrosso=6;
int ad_value=0;
int buzzer=9;
int pompa=10;
int pulsantetest=8;
int frequenza=700;
void setup()
{
  pinMode(Moisture_ain, INPUT);
  pinMode(ledverde, OUTPUT);
  pinMode(ledrosso, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(pompa, OUTPUT);
  pinMode(pulsantetest, INPUT_PULLUP);
  lcd.begin(16,2);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("U=    %");
  lcd.setCursor(0,1);
  lcd.print("Pompa:");
}
void loop()
{
byte B=digitalRead(pulsantetest);
if (B==0)
{
ad_value=ad_value+2;
if(ad_value>850) {ad_value=0;}
}
else
{
ad_value=analogRead(Moisture_ain);
}
```

```
int u=map(ad_value,0,1023,0,1000);
float um=u/10.0;
 if(um<20.0)          // Gestione "Asciutto"
 {
   lcd.setCursor(2,0);
   lcd.print("    ");
   lcd.setCursor(2,0);
   lcd.print(um,1);
   lcd.setCursor(8,0);
   lcd.print("Asciutto");
   digitalWrite(ledverde,LOW);
   digitalWrite(ledrosso,HIGH);
   digitalWrite(pompa,HIGH);     // attiva la pompa per l'irrigazione
   lcd.setCursor(6,1);
   lcd.print("ON ");
   tone(buzzer,frequenza,200);
   delay (50);
   frequenza=frequenza+100;
   if (frequenza>3000)    {frequenza=700;}
 }
 else if (um>20.0 && um<60.0)  // Gestione "Umido"
 {
   digitalWrite(pompa,LOW);      // Spegne la pompa per l'irrigazione
   lcd.setCursor(6,1);
   lcd.print("OFF");
   lcd.setCursor(2,0);
   lcd.print("    ");
   lcd.setCursor(2,0);
   lcd.print(um,1);
   lcd.setCursor(8,0);
   lcd.print("Umido    ");
   digitalWrite(ledverde,HIGH);
   digitalWrite(ledrosso,LOW);
   delay (100);
   digitalWrite(ledverde,LOW);
   delay (100);
  }
 else                                  // Gestione "Bagnato"
 {
   digitalWrite(pompa,LOW);      // Spegne la pompa per l'irrigazione
   lcd.setCursor(6,1);
   lcd.print("OFF");
   lcd.setCursor(2,0);
   lcd.print("    ");
   lcd.setCursor(2,0);
   lcd.print(um,1);
   lcd.setCursor(8,0);
   lcd.setCursor(8,0);
```

```
    lcd.print("Bagnato    ");
    digitalWrite(ledverde,LOW);
    digitalWrite(ledrosso,HIGH);
    delay (100);
    digitalWrite(ledverde,HIGH);
    digitalWrite(ledrosso,LOW);
    delay (100);
   }
 }
```

Esercizio 37 - Calendario con RTC DS 1307 e Display LCD 16x2

Dimensionare e programmare un circuito basato sul modulo RTC DS 1307 in grado di visualizzare su un display LCD 16x2:

1. Prima riga – Data nel formato gg mese aaaa e giorno della settimana (Es.: 26 Nov 2016 Sab)
2. Seconda riga – Ora nel formato hh:mm:ss (Es.:17:56:45)

Inoltre inserire un Led verde con lampeggio alla frequenza di 1 Hz

Soluzione

In Fig. 1 è riportato lo schema elettrico del sistema.

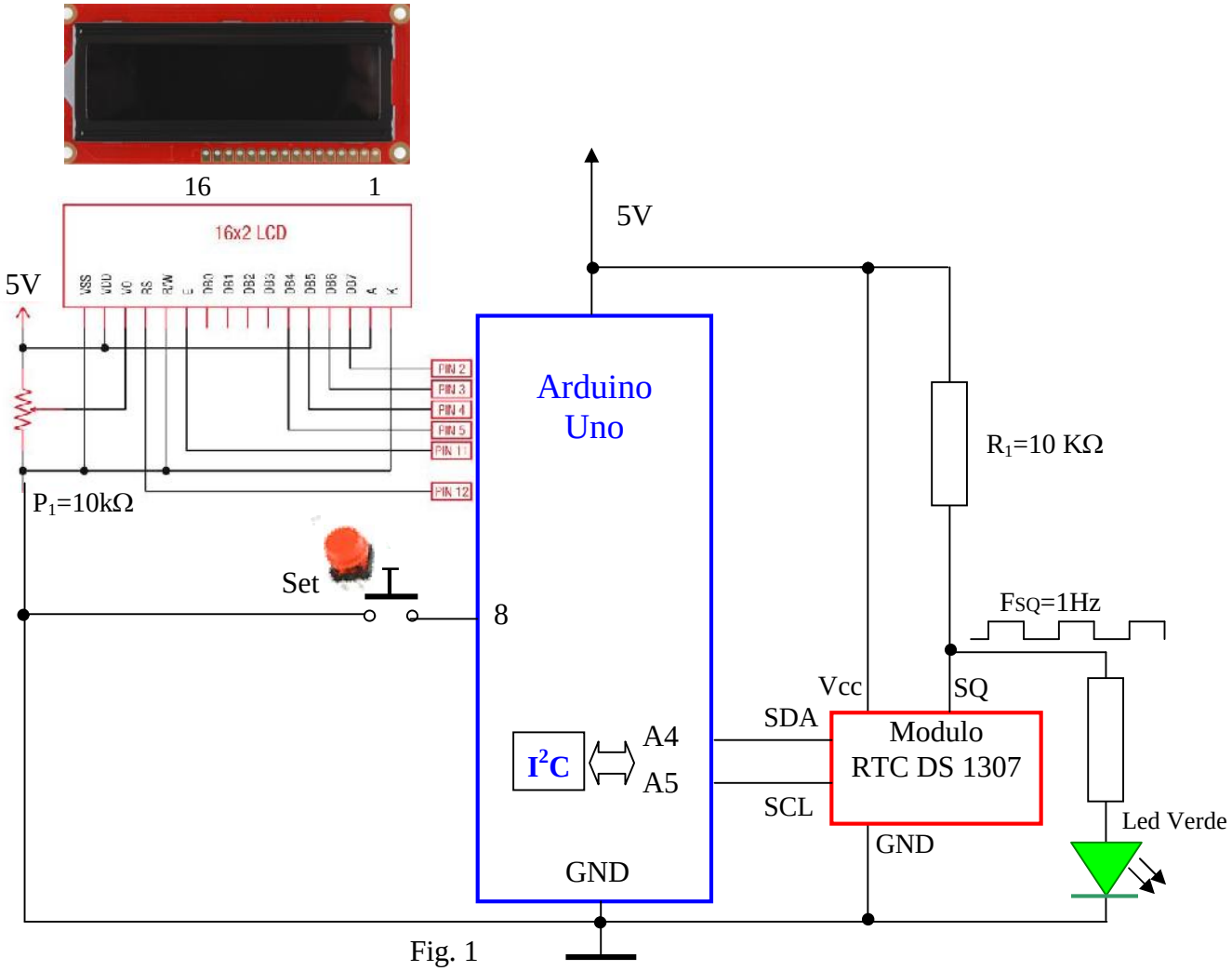


Fig. 1

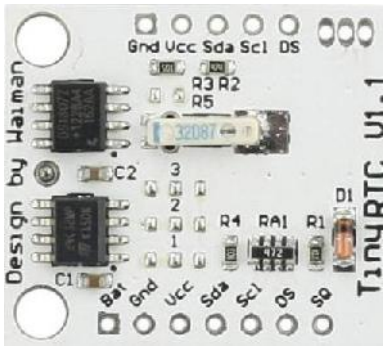
Modulo RTC

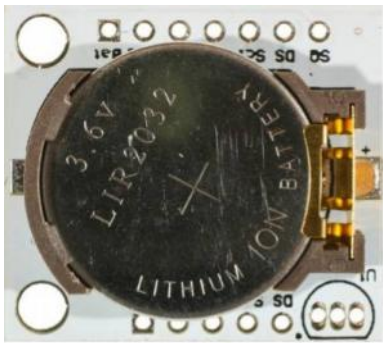
Generalità

Un RTC (Real-Time Clock, orologio in tempo reale, è un dispositivo con funzione di orologio, solitamente costituito da un processore a circuito integrato specializzato per questa funzione, il quale conteggia il tempo reale (anno, mese, giorno, ore, minuti e secondi) anche quando l'utilizzatore viene spento. Viene usato in molti tipi di computer ed è presente in tutti i PC. L'RTC è presente anche in molti sistemi embedded nonché viene utilizzato anche in circuiti elettronici dove è necessario avere un preciso riferimento temporale.
Alcuni modelli di RTC integrati sono il DS1307 di Maxim, il PCF8563 di NXP Semiconductors (ex Philips) e le diverse soluzioni di Integrated Device Technology (IDT).
Per poter mantenere il conteggio del tempo anche a circuito non alimentato, i real-time clock hanno un oscillatore al quarzo a loro dedicato e sono alimentati da una speciale batteria autonoma rispetto all'alimentazione principale; in alcuni tipi anche il quarzo è installato all'interno del package. Al contrario, i clock che non sono real-time non funzionano quando il dispositivo è spento.
Gli RTC furono introdotti nei computer agli inizi degli anni ottanta: uno dei primi ad integrare un orologio in tempo reale fu l'Apple III. Successivamente anche IBM utilizzò un RTC nel suo PC AT del 1984, che integrava un RTC MC146818. Successivamente anche Dallas Semiconductor realizzò degli RTC. Gli orologi in tempo reale erano facilmente individuabili sulle schede madri dei vecchi PC grazie alla batteria tampone che avevano vicino e a dei disegni che indicavano la funzione del chip. Nei computer più recenti gli RTC sono integrati direttamente nel chipset del sistema. Gli RTC non devono essere confusi con il real-time o il clock della CPU.

RTC DS 1307 con interfaccia I^2C

Il modulo si basa sull'integrato **DS1307 I2C** RTC (vedi datasheet) è dotato di EEPROM (**AT24C32** - Erasable and Programmable Read Only Memory) con 32kbyte memoria. Il modello in esame integra il circuito per l'istallazione del sensore di temperatura DS18B20 (Fig. 2).
È presente una batteria tampone ricaricabile LIR2032 al litio che consente di alimentare il dispositivo e garantisce un funzionamento a lungo termine anche quando è disconnesso da Arduino. Il circuito risulta essere così totalmente indipendente: incrementa l'ora senza aver bisogno di un microcontrollore e con la batteria completamente carica è in grado di funzionare per 1 anno intero quando il sensore di temperatura risulta spento oppure non presente.
In Fig 3 è riportato lo schema elettrico del modulo.
L'orario e il calendario sono salvati nei registri in formato BCD. In questo formato ogni cifra di un numero è rappresentata con un codice binario a quattro bit, il cui valore è compreso tra 0000 (0) e 1001 (9). Ad esempio il numero 127 in formato BCD viene registrato in questo modo: 0001 0010 0111.
Sebbene il BCD comporti un notevole spreco di bit (circa 1/6 di memoria inutilizzata in packed BCD), in alcuni casi è preferibile perché ha una diretta corrispondenza con il codice ASCII.





Alloggiamento per **DS18B20**

Fig. 2

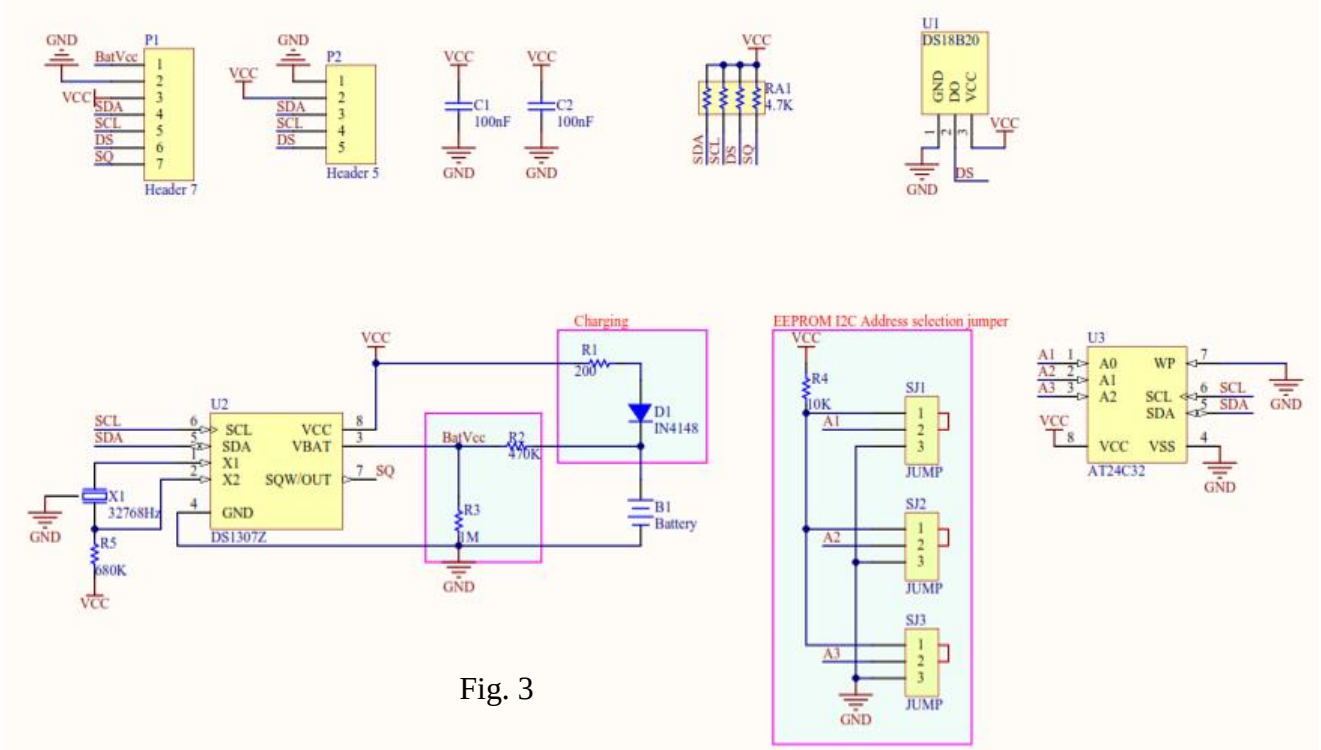


Fig. 3

È sufficiente infatti sostituire i primi quattro bit inutilizzati con 0011 per ottenere il corrispondente ASCII (0011 + BCD => 0011 + 0000 => '0').
Il codice BCD è molto usato in elettronica, specialmente in circuiti digitali privi di microprocessore, perché facilita la visualizzazione di lunghe cifre su display a sette segmenti dove ad ogni display fisico corrisponde esattamente una cifra. Esistono appositi circuiti integrati che effettuano la conversione da BCD nella corrispondente sequenza di accensione dei segmenti. Anche l'esecuzione di semplici calcoli aritmetici è più semplice da effettuarsi su cifre BCD per circuiti logici combinatori.
In Fig. 4 la tabella degli indirizzi dei registri interni dell'RTC.

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	secondi (decine)			secondi (unità)				SECONDI	00–59
01h	0	minuti (decine)			minuti (unità)				MINUTI	00–59
02h	0	12 1 / 24 0	ore (decine) / PM/AM	ore (decine)	ore (unità)				ORE	1–12 +AM/PM 00–23
03h	0	0	0	0	0	giorno settimana			GG SETTIMANA	01–07
04h	0	0	giorno mese (decine)		giorno mese (unità)				GG MESE	01–31
05h	0	0	0	mese (decine)	mese (unità)				MESE	01–12
06h	Anno (decine)				Anno (unità)				ANNO	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	08h–FFh

Fig. 4

I registri dell'RTC vanno dall'indirizzo 00h a 07h.
I registri utilizzabili come RAM vanno da 08h a 3Fh.
- Il Bit 7 del **registro dei secondi** (00h) è il bit di halt (**CH**) dell'orologio. Quando questo bit è posto a 1 l'oscillatore è spento mentre se posto a 0 l'oscillatore viene riattivato.

- I valori che corrispondono al giorno della settimana vengono definiti dall'utente (solitamente 1 = domenica, 2 Lunedì etc.). Allo scadere della mezzanotte questi vengono incrementati di uno (se vale 7 viene riportato a 1).
- Il Bit 6 del **registro delle ore** (02h) indica se l'orario è definito con modalità 12-ore (quando vale 1) oppure 24-ore (se vale 0). Nel caso si utilizzi la modalità a 12 ore il bit 5 (bit AM/PM) viene posto a 1 se è PM, 0 se è AM. Nel caso sia abilitata la modalità 24-ore il 5 bit è usato per la codifica delle decine delle ore. Se si cambia da 12 a 24 ore l'orario deve essere resettato.
- il **registro di controllo** (08h) è utilizzato per pilotare l'uscita SQ (che è un'onda quadra). Infatti il nostro modulo RTC ha la possibilità di generare un onda quadra di 1Hz, 4096KHz, 8192KHz, 32768KHz. Tramite software viene programmato il registro per generare un'onda quadra ad 1 Hz. Il segnale viene utilizzato per il lampeggio di un LED verde (Fig. 1)
Sul Datasheet sono riportati tutti i dettagli per la configurazione dei registri.

Il modulo utilizza i pin analogici Arduino **A5** (SCL) e **A4** (SDA) per la comunicazione seriale "I^2C" (impostazione predefinita per la libreria <wire.h>). (http://arduino.cc/en/Reference/Wire)

Il DS1307 supporta il protocollo I^2C. Un dispositivo che invia i dati sul bus è definito trasmettitore mentre quella che li legge ricevitore. Il dispositivo che controlla gli altri, detti slave, è detto master e deve generare il segnale di clock, controllare l'accesso al bus e generare le condizioni di START e STOP.

Il DS1307 opera come slave sul bus I^2C ed è allocato all'indirizzo 0x68 (Nel linguaccia C si usa il suffisso 0x per indicare un numero esadecimale).

Protocollo I^2C

…….

Programma

```
/* Esercizio 37 - Calendario con RTC DS 1307 e Display LCD 16x2
1.  Prima riga – Data nel formato gg mese aaaa e giorno della settimana (Es.: 26 Nov 2016  Sab)
2.  Seconda riga – Ora nel formato hh:mm:ss (Es.:17:56:45)
*/
#include <Wire.h>
#include <RTClib.h>
#include <LiquidCrystal.h>
RTC_DS1307 RTC;
#define DS1307_I2C_ADDRESS 0x68
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
char ora[9];
char data[12];
char giorno[7] [4] = {"Dom","Lun", "Mar", "Mer", "Gio", "Ven", "Sab"};
char mese [12] [4] = {"Dic", "Gen","Feb", "Mar", "Apr", "Mag", "Giu", "Lug", "Ago", "Set", "Ott", "Nov"};

int pulsanteset = 6;

void setup ()
{
  pinMode (pulsanteset, INPUT_PULLUP);
  Wire.begin();
  RTC.begin();
  lcd.begin(16, 2);
 Wire.beginTransmission(DS1307_I2C_ADDRESS);
 Wire.write(7);
 Wire.write(B10010000); // Imposta FSQ=1Hz
 Wire.endTransmission();
   }
void loop ()
{
  boolean set = digitalRead (pulsanteset);
  if (set == LOW)
  {
    RTC.adjust(DateTime(__DATE__, __TIME__)); // Imposta Data e Ora del Computer
  }
  DateTime now = RTC.now();  // Legge i dati dal modulo RTC DS1307
  int g=now.dayOfTheWeek();  // Determina il giorno della settimana (0=Domenica, 1=Lunedì, ...)
  int m=now.month();         // Determia ilnumero del mese (0=Dicembre, 1=Gennaio, ...)
  sprintf(data, "%02d %s %d", now.day(), mese [m], now.year());  // Composizione stringa data
 sprintf(ora,  "%02d:%02d:%02d", now.hour(), now.minute(), now.second()); //  Stringa ora
  lcd.setCursor(0,0);
  lcd.print(data);
  lcd.setCursor(13,0);
  lcd.print(giorno[g]);
  lcd.setCursor(8,1);
  lcd.print(ora);

  delay(1000);
}
```

Esercizio 38 - Comando 2 Servo con un modulo Laser

Dimensionare e programmare un circuito che tramite un modulo Laser (RX-TX) sia in grado di:

1. Riconoscere entro un tempo di 50s la posizione di cinque lettere (I, P, S, I, A) poste a una certa distanza fra di loro in uno spazio 3D.
2. Visualizzare sul display la direzione del raggio laser (Angolo orizzontale angolo verticale rispetto ad una posizione di riferimento (0,0)).
3. Visualizzare sul display ogni lettera rilevata e comporre la scritta IPSIA.
4. Riavviarsi dopo un determinato tempo.

Se entro il tempo prestabilito il sistema riesce a rilevare tutte e cinque le lettere, sul display sarà visualizzata la scritta “Bravo”. In caso contrario verrà visualizzata la scritta “Game Over”. In ogni caso, al termine del tempo impostato o una volta rilevate tutte e cinque le lettere il sistema si dovrà riavviare.
Per conoscere la posizione delle lettere il sistema deve rilevare e visualizzare l'angolo orizzontale e l'angolo verticale rispetto ad un punto di riferimento.

Soluzione

Si propone una soluzione con due servo motori sovrapposti comandati tramite due potenziometri. Un servo per lo spostamento orizzontale da 0°-180° ed un servo per lo spostamento verticale 0° - 180°. Sul perno del servo orizzontale viene innestato il servo per lo spostamento verticale, il modulo laser viene montato sul perno del servo verticale.
In questo modo è possibile indirizzare il raggio laser con un raggio di azione molto ampio, sufficiente per individuare un oggetto nello spazio. In Fig. 1 è riportato lo schema elettrico del sistema, nella descrizione del software il suo funzionamento.

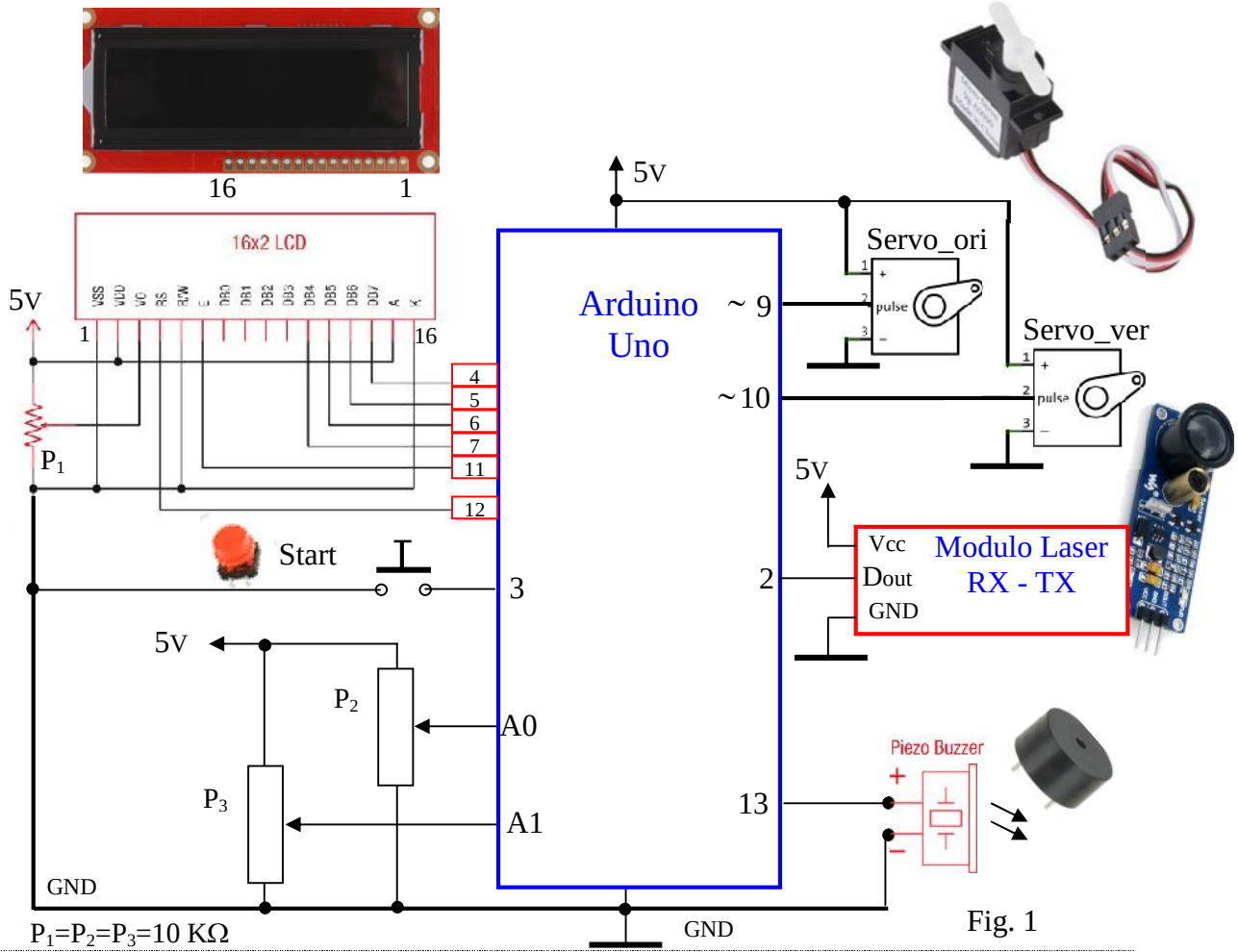


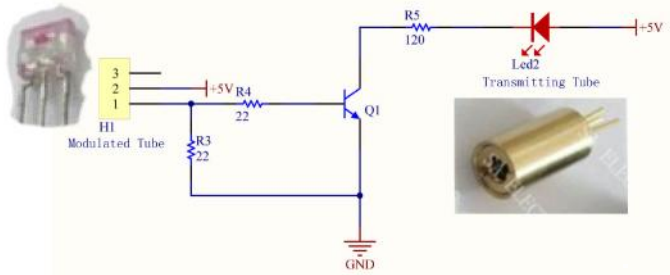
Fig. 1

Modulo Laser (TX - RX)

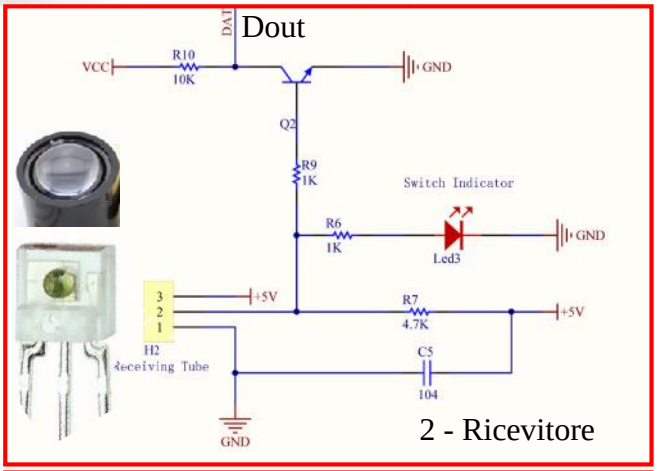
In fig. 2 è riportato lo schema elettrico del modulo

Boost converter chip	PT1301
Operating voltage	2.5V-5.0V
Dimensions	53.0mm*18.0mm
Fixing hole size	2.0mm
Effective distance	4m-5m
Distanza rilevamento	0.8m(typ), 1.5m(max)

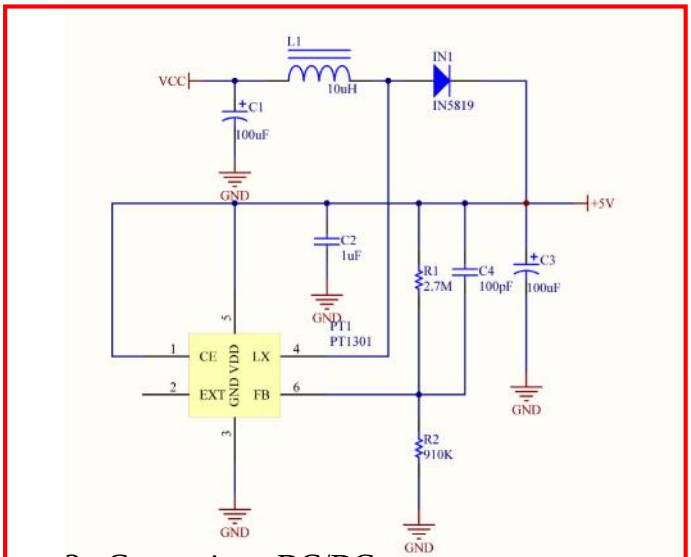
Ricevitore **Modulatore**



1 - Trasmettitore

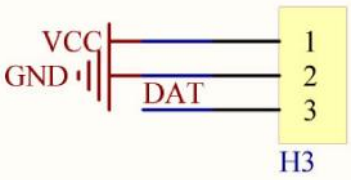


2 - Ricevitore



3 - Convertitore DC/DC

4 - Led Power



5 - Connettore

Fig. 2

Lo schema è suddiviso in 5 parti.

1) Trasmettitore

Nel trasmettitore è presente un tubo oscillatore che può generare un segnale a 180 KHz. Dopo essere stato amplificato da un transistor (Q1) il segnale è applicato al diodo laser per il sistema di eccitazione. La lunghezza d'onda ottica emessa è 650nm e può raggiungere una distanza di circa 5 m.

2) Ricevitore

Il ricevitore è composto da un tubo ricevitore, compatibile con il tubo trasmettitore, che riceve la luce riflessa. E 'in grado di identificare la luce visibile modulata (650--950nm).
Il ricevitore può solo ricevere la luce riflessa alla stessa frequenza con cui viene modulata dal trasmettitore Il trasmettitore emette il raggio laser, se sulla traiettoria incontra un oggetto questo viene riflesso e captato dal chip ricevitore collegato sul connettore H2 (Fig. 2).
La sensibilità è tale da ricevere un segnale riflesso ad una distanza tipica di 80 cm (max 1,5 m). Il terminale 2 del connettore H2 si porta a livello alto, si accende il Led *Switch Indicator* (oggetto intercettato), il transistor Q2 si porta in saturazione, il livello Dout si porta a livello basso. Questo livello viene letto tramite software sul pin 2 di Arduino (fig.1).

3) Convertitore DC/DC

Convertitore basato sull'integrato **PT1301**.

PowTech

PT1301

High Efficiency
Low Voltage Step-up DC/DC Converter

GENERATION DESCRIPTION

The PT1301 is a compact, high efficiency, and low voltage step-up DC/DC converter with an Adaptive Current Mode PWM control loop. It comprises of an error amplifier, a ramp generator, a PWM comparator, a switch pass element and the driver. It provides stable and high efficient operation over a wide range of load currents without external compensation. The below 1V start-up input voltage makes PT1301 suitable for single battery cell applications. The built-in power transistor is able to provide up to 300mA output current while working under Li-Battery Supply. Besides, it provides extra pin to drive external power devices (NMOS or NPN) in case higher output current is needed. The output voltage is set with two external resistors. The 500KHz high switching rate reduces the size of external components. Besides, the 14μA low quiescent current together with high efficiency maintains long battery lifetime.

FEATURES

- Low Quiescent (Switch-off) Supply Current: 14μA
- Low Start-up Input Voltage: typical 0.8V
- High Supply Capability: Deliver 3.3V 100mA with 1Alkaline Cell; 5V 300mA with 1 Li-Cell
- Zero Shutdown Mode Supply Current
- High efficiency: 90%
- Fixed switching frequency: 500KHz
- Options for internal or external power switches
- Package type: SOT-23-6, SOT-89-5,MSOP8

APPLICATIONS

- MP3, PDA, Electronic
- Dictionary, DSC, LCD, RF-Tag,
- Portable Devices, Wireless Devices, etc.

4) Led Power
Led acceso in presenza di alimentazione (5V).

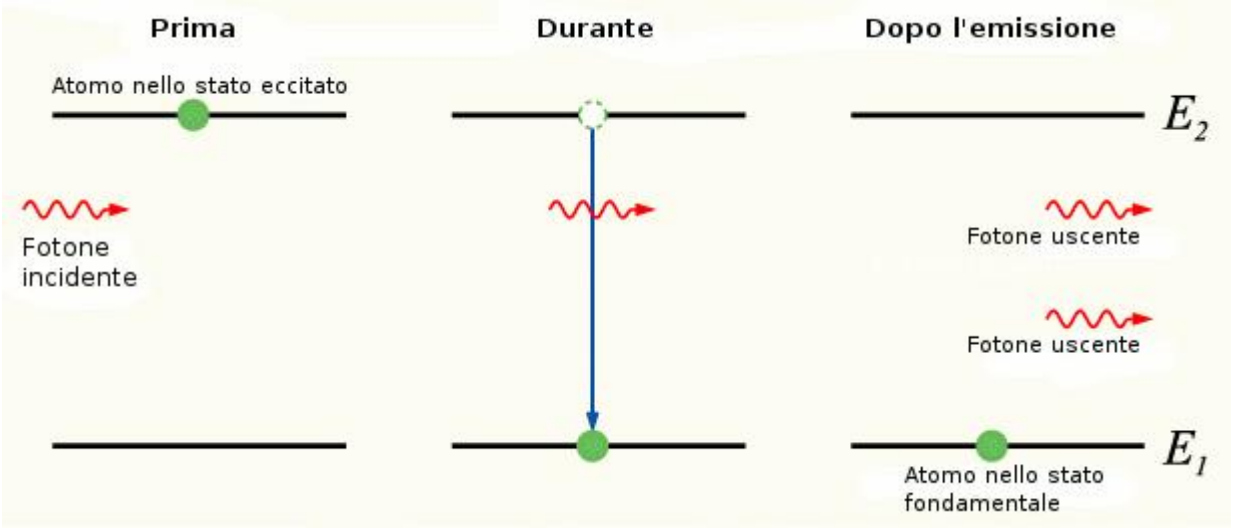
5) Connettore H3

Pin	Simbolo	Descrizione
1	DOUT	Uscita digitale (Pin 2 Arduino)
2	GND	Massa
3	Vcc	Alimentazione (2,5 V – 5.0 V)

Approfondimento: Principi di funzionamento del L.A.S.E.R.

La parola laser rappresenta una sigla, per la precisione questa sigla sta a significare "**L**ight **A**mplification by **S**timulated **E**mission of **R**adiation", quindi sostanzialmente amplificazione luminosa provocata dall'emissione stimolata di radiazioni elettromagnetiche.

Emissione stimolata e spontanea
Cominciamo dunque spiegando brevemente la differenza tra emissione di radiazioni spontanea e stimolata:
1) Nel caso di emissione spontanea il sistema in esame (un atomo ad esempio) può rimanere in uno stato energeticamente eccitato per un tempo più o meno lungo, a seconda del delta (differenza) di energia in gioco. Quando il sistema si diseccita e fa ritorno allo stato energetico fondamentale ecco che l'energia persa viene rilasciata sotto forma di radiazione elettromagnetica (luce, ma non sempre!). Tale radiazione in uscita potrà direzionarsi in qualsiasi direzione.
2) Quando si parla invece di emissione stimolata ecco che il ritorno del sistema allo stato energetico iniziale viene provocato (stimolato) da un fotone incidente. I fotoni in uscita però in questo caso sono nella stessa direzione dei fotoni che hanno provocato il diseccitamento. Inoltre, fondamentale, i fotoni in uscita sono tutti in FASE tra loro. Questo significa che tutte le creste delle loro onde sono allineate, così come le "conche" (e questo porta all'amplificazione del fascio luminoso).

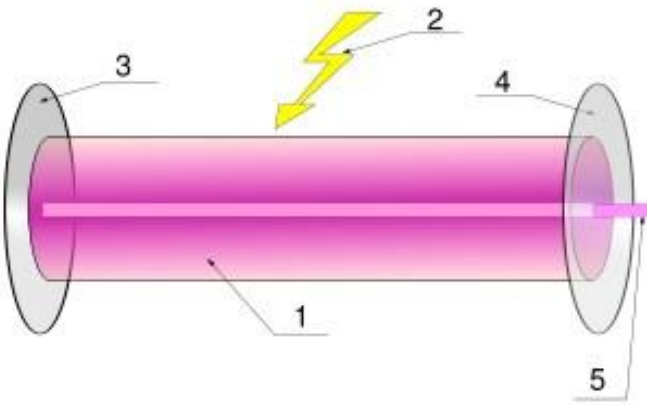


Dunque ricapitolando, i fotoni emessi per emissione stimolata presentano tre importanti caratteristiche:
- il fotone emesso è in fase con il fotone incidente.
- il fotone emesso ha la stessa lambda (lunghezza d'onda) del fotone incidente.
- il fotone emesso viaggia nella stessa direzione del fotone incidente.

Quindi prendendo in esame i componenti (es. atomi) di un materiale adatto, all'equilibrio termico ci sono moltissimi atomi nello stato fondamentale e pochi in quello eccitato.
Gli atomi vengono poi "pompati" nello stato eccitato, fornendo energia in vari metodi: in questo modo si crea quella che viene definita come "inversione di popolazione".
In seguito un fotone emesso, anche spontaneamente, nel sistema provoca il diseccitamento degli altri atomi eccitati, con conseguente emissione di altri fotoni (sempre tutti in fase e nella stessa direzione). Questo è fondamentalmente il principio di base.
Dal punto di vista tecnico invece, un laser comune è costituito da un mezzo attivo e da una cavità. Andiamo a vedere di cosa si tratta.

Mezzo attivo e cavità

Il mezzo attivo è il materiale che emette i fotoni in fase, la Cavità invece è una vera e propria cavità, composta, nella sua forma più semplice, da due specchi (di cui uno semiriflettente, per permettere poi l'uscita del raggio) che confinano il mezzo attivo.
La funzione dei due specchi è quella di provocare, mediante riflessioni, numerosissimi passaggi dei fotoni emessi attraverso il mezzo attivo, per provocare un aumento dell'intensità del fascio luminoso. Inoltre la lunghezza della cavità permette anche di selezionare la lunghezza d'onda dei fotoni emessi.
In più i due specchi permettono di far si che solo i fotoni che si muovono orizzontalmente rispetto alla cavità possano subire riflessioni, e quindi amplificazione. Tutti gli altri si annullano, con l'ottenimento di un raggio altamente focalizzato
Nell'immagine possiamo vedere:



1) Mezzo ottico
2) Energia fornita per il "pompaggio"
3) Specchio
4) Specchio semiriflettente
5) Fascio laser in uscita

Pulsante Start

Il pulsante (Fig. 3) è collegato tra il pin 3 e GND, il pin deve essere abilitato tramite software come input con resistenza di PullUp.

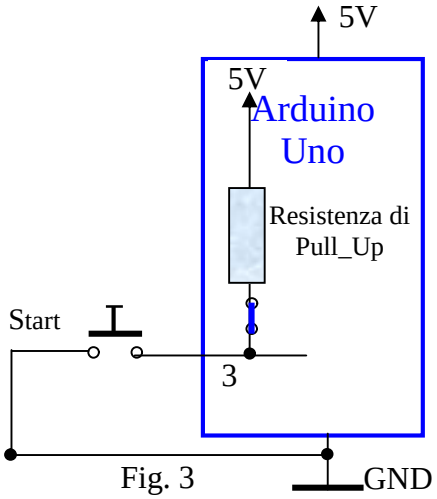


Fig. 3

1) **Start=OFF** (pulsante aperto) il pin 3 si trova a livello alto (5V), il software legge il valore alto e tramite un ciclo do... while ferma l'esecuzione in attesa della pressione.
2) **Start=ON** (pulsante chiuso) il pin 3 si trova a livello basso (GND), il software esce da ciclo do... while e prosegue per il normale funzionamento.

```
do{
}while (digitalRead(pinStart)==HIGH); // Attesa pressione
                                      pulsante Start
```

Gruppo Potenziometri -Servo

In fig. 4 è riportato il gruppo Potenziometri-Servo

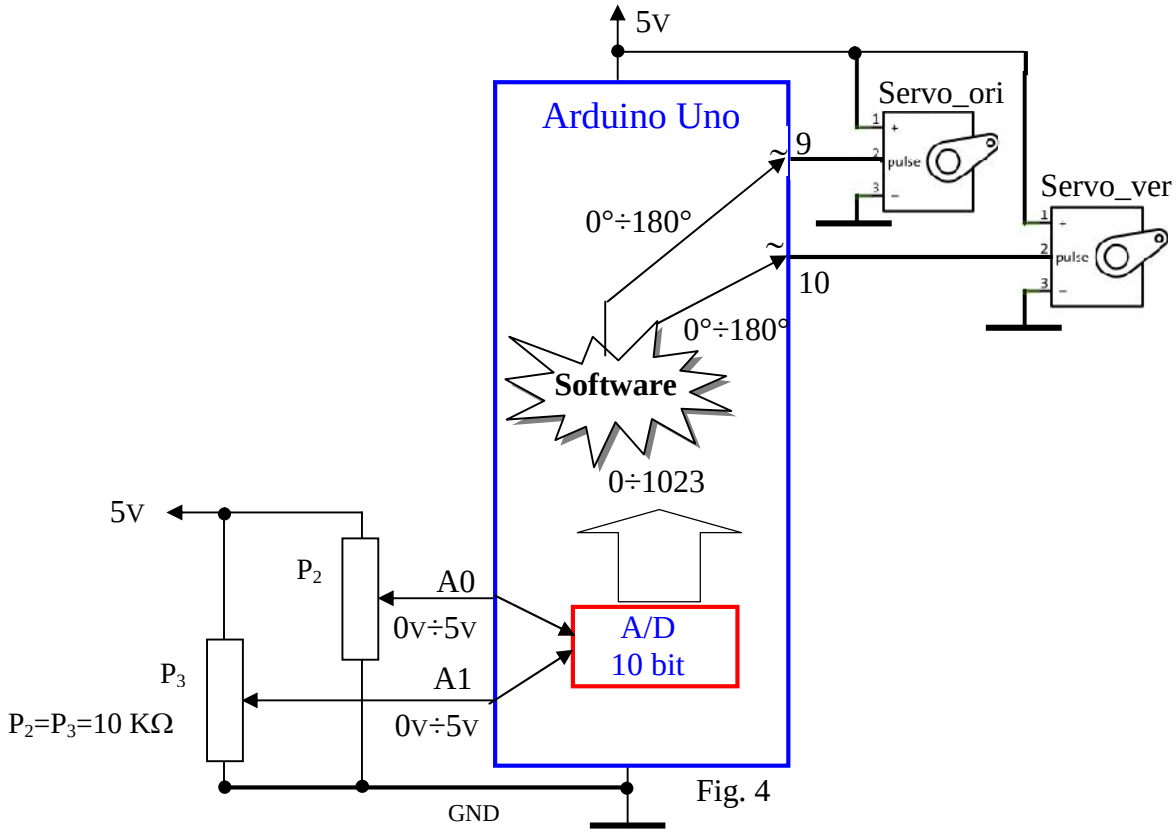


Fig. 4

La rotazione dei potenziometri P2 e P3 fornisce sui Pin A0 e A1 una tensione variabile 0V÷ 5V, tale valore viene convertito in digitale dal convertitore A/D interno nel range 0÷1023 e mappato in angolo 0°÷180° per il comando dei due Servo (P2 --> Servo orizzontale, P2 --> Servo Verticale).

```
pos_ori=map(analogRead (A0),0,1023,0,180);
pos_ver=map(analogRead (A1),0,1023,0,180);
```

Lettura e conversione

Nota:
Convertitore interno a 10 bit, range di input 0V÷ 5V, range di output 0÷1023, risoluzione analogica 4,88 mV.

Programma

Di seguito sono riportati i punti fondamentali del programma.

1) Inizializzazione delle variabili e richiamo delle librerie (Display e Servo). Memorizzazione della posizione (Angolo Verticale e Angolo Orizzontale) delle 5 lettere (I P S I A)
2) Fase di Setup: Setup del Display, Servo e modalità di funzionamento dei Pin.
3) Con l'istruzione Do....While si attende la pressione del pulsante Start .
4) Con la pressione del pulsante Start inizia la fase di riconoscimento (max 70 s).
5) La rotazione dei potenziometri P2 e P3 fornisce sui Pin A0 e A1 una tensione variabile 0V÷ 5V, tale valore viene convertito in digitale nel range 0÷1023 e mappato in angolo 0÷180 per il comando dei due Servo (P1 --> Servo orizzontale, P2 --> Servo Verticale)
6) Il movimento dei Servo sposta il raggio laser e visualizza sul display la relativa posizione (angolo Orizzontale e Angolo Verticale).
7) Se il raggio intercetta in un punto, marcato con un cuoricino sulla lettera, questo viene riflesso e captato dal ricevitore.
8) Il software rileva la riflessione, verifica la posizione e visualizza sul display la lettera intercettata (Le lettere devono essere intercettate nella giusta sequenza (I - P - S - I - A).
9) Trascorsi 50s viene effettuato in automatico il Reset.
10) Il display visualizza tutte le fasi del funzionamento.

Codice

```
/* Esercizio 38 - Comando 2 Servo con un modulo Laser
    1. Riconosce entro un tempo di 70s la posizione di cinque lettere (I, P, S, I, A) poste a una certa distanza fra di loro in uno spazio 3D.
    2. Visualizza sul display la direzione del raggio laser (Angolo orizzontale angolo verticale rispetto ad una posizione di riferimento (0,0)).
    3. Visualizza sul display ogni lettera rilevata e compone la scritta IPSIA.
    4. Reset dopo 50 s.
 */

#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,7,6,5,4);
#include <Servo.h> // richiama la libreria di gestione dei servomotori
Servo myservo_ori;     // crea il servo oggetto “myservo_ori”
Servo myservo_ver;     // crea il servo oggetto “myservo_ver”
int pos_ori = 0;       // variabile per la memorizzazione della posizione orizzontale (in gradi angolari) del perno
int pos_ver = 0;       // variabile per la memorizzazione della posizione verticale (in gradi angolari) del perno
int laser_RX=2;        // pin Laser RX
int pinStart=3;        // pin Pulsante di Start
int pinOggetto=13;     // pin Led Oggetto
int x;                 // variabile ciclo for
byte I=0,P=0,S=0,i=0,A=0;              // variabili conteggio lettere

void(* Riavvia)(void) = 0; //Dichiarazione di funzione che punta all’indirizzo zero (Reset Arduino)
```

```
int I_pos_v=121;          //angolo verticale lettera  I
int I_pos_o=121;          //angolo orizzontale lettera  I
int P_pos_v=143;          //angolo verticale lettera  P
int P_pos_o=112;          //angolo orizzontale lettera  P
int S_pos_v=93;          //angolo verticale lettera  S
int S_pos_o=97;          //angolo orizzontale lettera  S
int i_pos_v=141;          //angolo verticale lettera  I
int i_pos_o=90;          //angolo orizzontale lettera  I
int A_pos_v=117;          //angolo verticale lettera  A
int A_pos_o=71;          //angolo orizzontale lettera  A
int dx=2;               // tolleranza 2°
int frequenza=700;
unsigned long time1;
unsigned long time2 = 50000; // Tempo assegnato per la rilevazione delle 5 lettere
long time3;
void setup()
{

 lcd.begin(16, 2);
 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("Premi Start");
 lcd.setCursor(13,0);
 lcd.print(time2/1000);
 myservo_ori.attach(9);   // assegna il servo oggetto “myservo_ori” alla porta 9
 myservo_ver.attach(10);  // assegna il servo oggetto “myservo_ver” alla porta 10
 myservo_ori.write(0);
  myservo_ver.write(0);
 pinMode(laser_RX,INPUT);
 pinMode(pinStart,INPUT_PULLUP);
 pinMode(pinOggetto,OUTPUT);
do{
}while (digitalRead(pinStart)==HIGH); // Attesa pressione pulsante Start
time1 = millis();
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Po=");
lcd.setCursor(0,1);
lcd.print("Pv=");
}
void loop()
{
  pos_ori=map(analogRead (A0),0,1023,0,180);
  pos_ver=map(analogRead (A1),0,1023,0,180);
  lcd.setCursor(3,0);
  lcd.print("   ");
  lcd.setCursor(3,0);
  lcd.print(pos_ori);
```

```
  lcd.setCursor(3,1);
  lcd.print("   ");
  lcd.setCursor(3,1);
  lcd.print(pos_ver);
  myservo_ori.write(pos_ori);
  myservo_ver.write(pos_ver);
 delay(45);
  digitalWrite(pinOggetto,LOW);
if (digitalRead(laser_RX)==LOW && (pos_ver<=I_pos_v+dx && pos_ver>=I_pos_v-dx)
&& (pos_ori<=I_pos_o+dx && pos_ori>=I_pos_o-dx))
{

tone (pinOggetto,frequenza,200);
lcd.setCursor(7,2);
lcd.print("I");
I=1;
}
if (digitalRead(laser_RX)==LOW && I==1 && (pos_ver<=P_pos_v+dx &&
pos_ver>=P_pos_v-dx) && (pos_ori<=P_pos_o+dx && pos_ori>=P_pos_o-dx))
{
tone (pinOggetto,frequenza+100,200);
lcd.setCursor(8,2);
lcd.print("P");
P=1;
}
if (digitalRead(laser_RX)==LOW && P==1 && (pos_ver<=S_pos_v+dx &&
pos_ver>=S_pos_v-dx) && (pos_ori<=S_pos_o+dx && pos_ori>=S_pos_o-dx))
{
tone (pinOggetto,frequenza+150,200);
lcd.setCursor(9,2);
lcd.print("S");
S=1;
}
if (digitalRead(laser_RX)==LOW && S==1 && (pos_ver<=i_pos_v+dx &&
pos_ver>=i_pos_v-dx) && (pos_ori<=i_pos_o+dx && pos_ori>=i_pos_o-dx))
{
tone (pinOggetto,frequenza+200,200);
lcd.setCursor(10,2);
lcd.print("I");
i=1;
}
if (digitalRead(laser_RX)==LOW && i==1 && (pos_ver<=A_pos_v+dx &&
pos_ver>=A_pos_v-dx) && (pos_ori<=A_pos_o+dx && pos_ori>=A_pos_o-dx))
{
tone (pinOggetto,frequenza+250,200);
lcd.setCursor(11,2);
lcd.print("A");
A=1;
```

```
}
time3 = time1+time2- millis();
lcd.setCursor(13,0);
lcd.print("   ");
lcd.setCursor(13,0);
lcd.print(time3/1000);
if (time3<=0)
  {
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print(" Game Over ");
   for (x=0; x<=15;x++)
    {
     lcd.setCursor(x,1);
     lcd.print(char(255));
     delay (400);
    }
   Riavvia(); // Reset del software
  }
  if (I+P+S+i+A==5)
  {
   lcd.setCursor(7,0);
   lcd.print("Bravo");


if(frequenza<3500)
{
  frequenza=frequenza+50;
  tone (pinOggetto,frequenza,200);
 delay (50);
}

   }
}
```

Esercizio 39 - Gioco Quiz - 4 Concorrenti

Dimensionare e programmare un circuito in grado di gestire un gioco quiz per quattro concorrenti (4 pulsanti, 4 led e 1 buzzer).

Soluzione

Il circuito riportato in Fig.1 controlla quattro concorrenti. La pressione di uno dei pulsanti, rilevata dal software, attiva la specifica routine che illumina il corrispondente indicatore (Led - Buzzer).

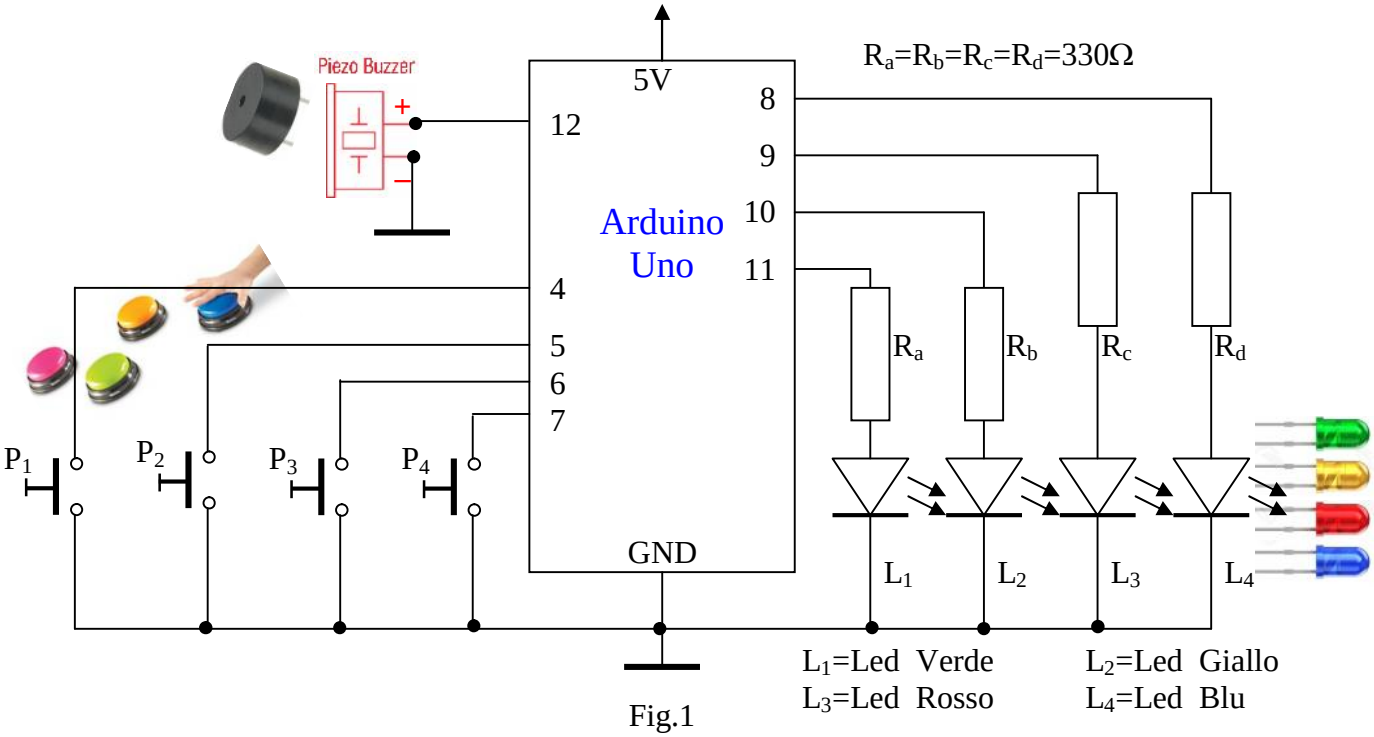


Fig.1

Programma/sketch

La funzione setup() configura i pine 4, 5, 6 e 7 per i pulsanti dei concorrenti (Input con Resistore di PullUP), i pin 8, 9, 10 e 11 quali uscite digitali, per il pilotaggio dei diodi LED e il pin 12 per il pilotaggio del Buzzer.
Nella funzione loop(), in un ciclo "while" vengono controllati continuamente, e molto velocemente, i quattro pulsanti. Non appena uno dei quattro concorrenti preme il suo pulsante, la variabile "ChiHaPremuto" riceve il numero identificativo del giocatore e il ciclo "while" termina. Quindi viene chiamata una funzione, alla quale è passato, come parametro, il numero della porta outtpu da gestire, si accende il relativo LED ed il Buzzer emette un suono.

```
/* Esercizio 39  Gioco Quiz - 4 concorrenti
-- 4 Pulsanti, 4 Led,  1 Buzzer
------
*/
int ChiMaPremuto;

byte buzzer=12;
void setup() {
  pinMode (4, INPUT_PULLUP);          // Pulsante concorrente N.1
  pinMode (5, INPUT_PULLUP);          // Pulsante concorrente N.2
  pinMode (6, INPUT_PULLUP);          // Pulsante concorrente N.3
  pinMode (7, INPUT_PULLUP);          // Pulsante concorrente  N.4

  pinMode (8, OUTPUT);                // LED concorrente N.1
  pinMode (9, OUTPUT);                // LED concorrente N.2
  pinMode (10, OUTPUT);               // LED concorrente N.3
 pinMode (11, OUTPUT);                // LED concorrente N.4
  pinMode (11, OUTPUT);

  digitalWrite (8, LOW);
  digitalWrite (9, LOW);
  digitalWrite (10, LOW);
  digitalWrite (11, LOW);
}
void loop()
{
// Controllo pulsanti
  while (1)
  {
    if (digitalRead(4) == LOW)
    {
      ChiMaPremuto = 1;      // Pressione Pulsante concorrente N.1
      break;
    }
    if (digitalRead(5) == LOW)
    {
      ChiMaPremuto = 2;      // Pressione Pulsante concorrente N.2
      break;
    }
    if (digitalRead(6) == LOW)
    {
      ChiMaPremuto = 3;      // Pressione Pulsante concorrente N.3
      break;
    }
    if (digitalRead(7) == LOW)
    {
      ChiMaPremuto = 4;      // Pressione Pulsante concorrente N.4
      break;
    }
```

```
// Accende il LED del concorrente
   if (ChiMaPremuto == 1)
     accensione(8);
    if (ChiMaPremuto == 2)
     accensione(9);
    if (ChiMaPremuto == 3)
     accensione(10);
    if (ChiMaPremuto == 4)
     accensione(11);
  }
}
// Funzione Lampeggio LED e attivazione suono
void accensione(int porta)
{
tone(buzzer,255,100);
  for (int k = 1; k <= 5; k++)
  {
    digitalWrite(porta, HIGH);
    delay(250);
    digitalWrite(porta, LOW);
    delay(250);
  }
  digitalWrite(porta, HIGH);
  while (1); // Blocca il programma - premere Reset sulla Board di Arduino per un altro ciclo
}
```

Esercizio 40 - Comando LED tramite Monitor Seriale e Tastiera

Dimensionare e programmare un circuito in grado di Accendere/Spegnere 4 LED tramite Monitor Seriale e tastiera.

Soluzione

In Fig. 1 è riportato lo schema elettrico del sistema.

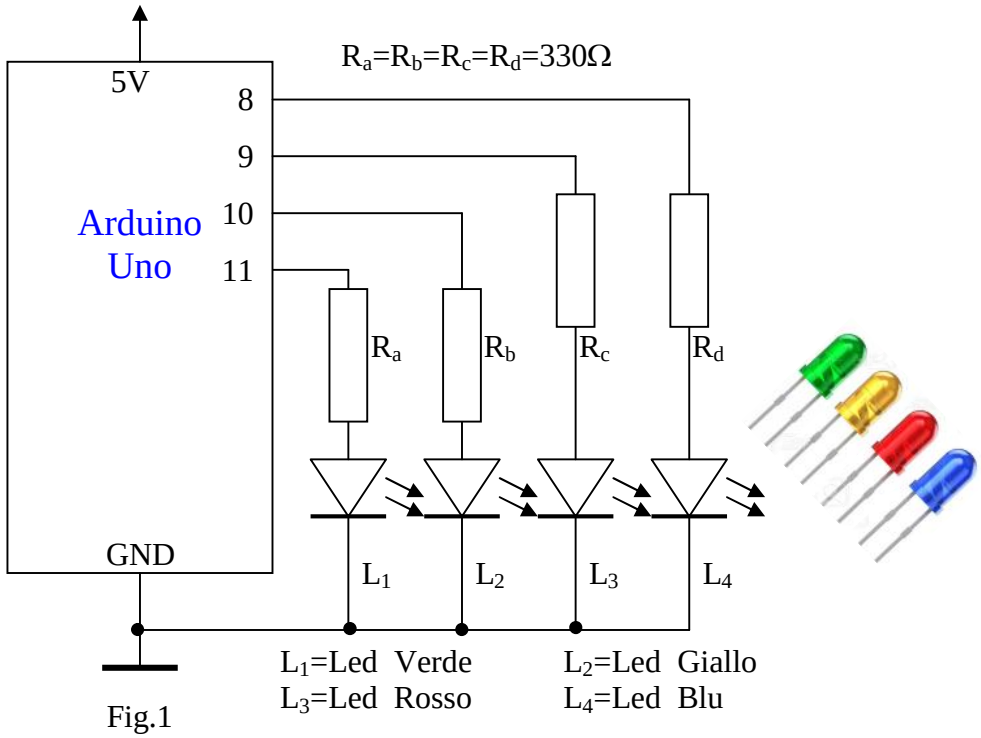


Fig.1

Programma/sketch

```
/* Esercizio 40  Comando 4 LED tramite Monitor Seriale Tastiera
------
*/
byte val = 0;
byte led1,led2,led3,led4 = LOW;

void setup() {
pinMode (8, OUTPUT);pinMode (9, OUTPUT);
pinMode (10, OUTPUT);pinMode (11, OUTPUT);
Serial.begin (9600); //Velocità monitor seriale
Serial.println ("Inserisci un valore tra 1 e 4");
}
void loop()
{
if (Serial.available() > 0)
{
val = Serial.parseInt (); //Legge il codice decimale del valore inserito tramite tastiera
Serial.println (val);
switch (val){
  case 1:
    led1 = !led1; //inverte lo stato di led1
    digitalWrite (8, led1);
    break;
```

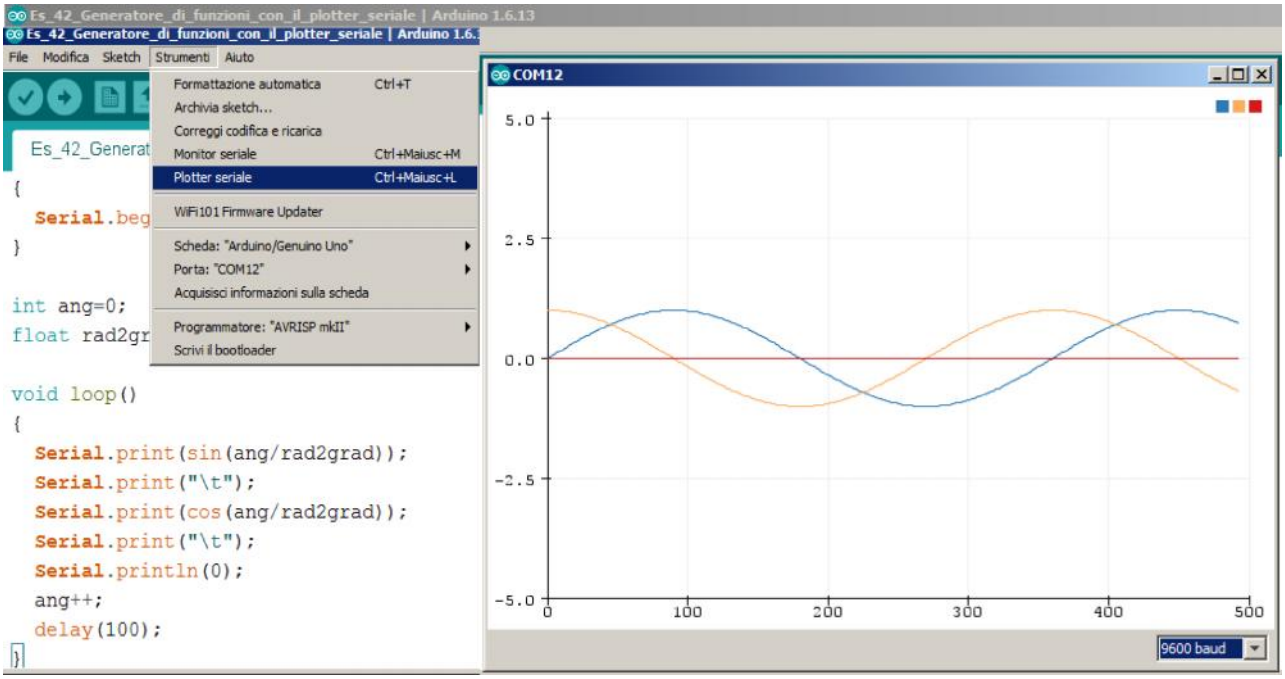
```
  case 2:
    led2 = !led2; //inverte lo stato di led2
    digitalWrite (9, led2);
    break;

  case 3:
    led3 = !led3; //inverte lo stato di led3
    digitalWrite (10, led3);
    break;

  case 4:
    led4 = !led4; //inverte lo stato di led4
    digitalWrite (11, led4);
    break;
 }
 }
}
```

Esercizio 41 - Generatore di funzioni con il plotter seriale

Il plotter seriale è una funzione aggiunta nell'IDE Arduino dalla versione 1.6.7, essa permette di rappresentare graficamente nel tempo valori numerici (temperatura, umidità, pressione, ecc). In questo esercizio analizzeremo un genaratore di funzioni (Seno, Coseno, Dente di sega, Onda Quadra e Triangolare)

Plotter Seriale

Lo strumento Plotter seriale si attiva dal menu strumenti:



si apre sullo schermo del computer una finestra con rappresentato sul lato sinistro un asse cartesiano riportanti i valori dal basso verso l'alto min, 0, max:

- **max**: indica il valore massimo atteso dal grafico, si adatta ai valori che gli arrivano crescendo o decrescendo in funzione dei valori ricevuti e si presente all'apertura impostato su 5.0 ossia il voltaggio massimo sui pin della scheda;
- **0**: indica il valore centrale del grafico durante tutto il periodo di utilizzo;
- **min**: si comporta come il **max** ma per i valori min;

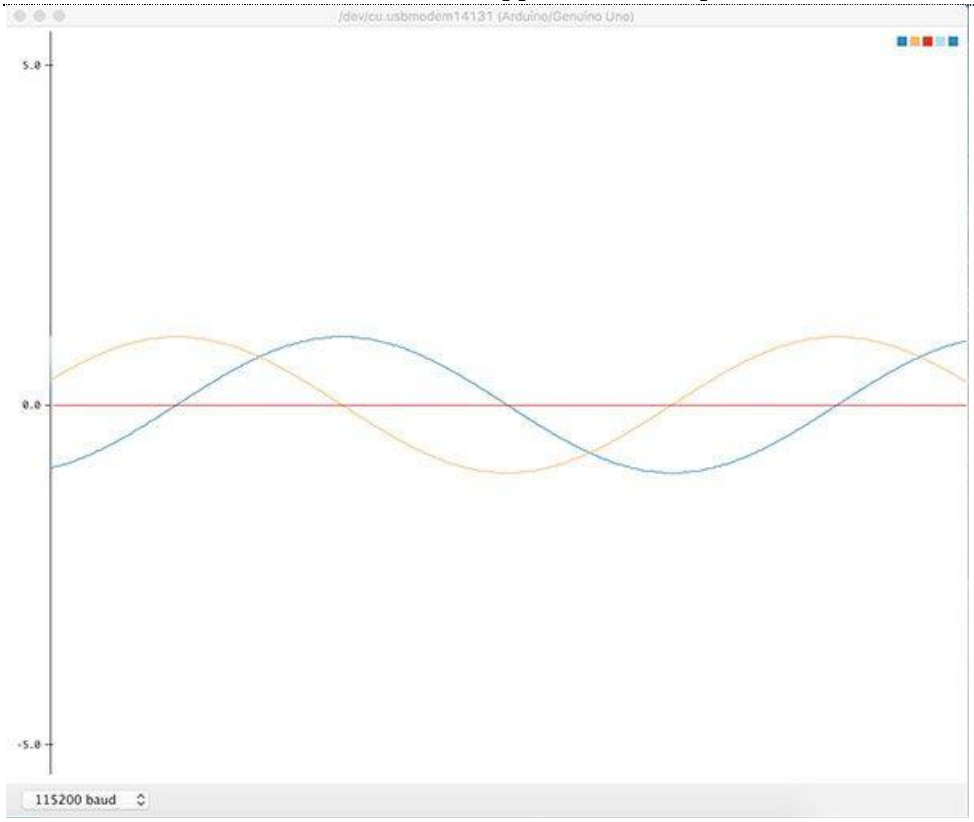
Come funziona il plotter seriale

La funzione di plotter seriale esegue il render grafico di valori numerici inviati mediante i comandi Serial.print e **Serial.println**.

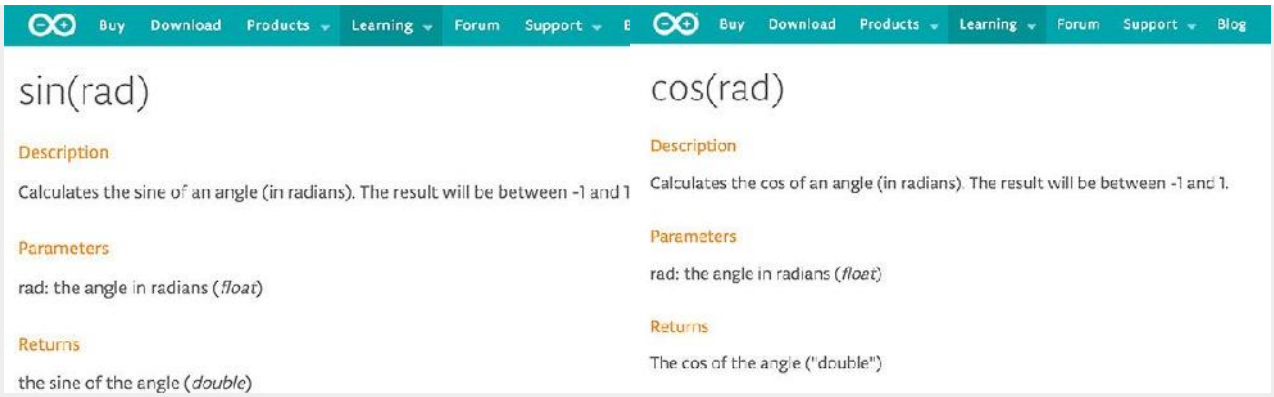
Ciascuno dei valori viene visualizzato con una linea di colore differente e deve essere intervallato da uno spazio o da un "tab" perché il plotter seriale lo riconosca come valore univoco da rappresentare lungo l'asse delle ascisse che rappresenta il trascorrere del tempo.

Esempio 1

Come primo esempio rappresentiamo il grafico delle funzioni **seno** e **coseno** (I valori di queste funzioni oscillano tra -1 ed 1, linea blu funzione seno, linea arancio funzione coseno, linea rossa asse del tempo)



Funzione **sin** (seno): Funzione **cos** (coseno):



Fonte: Reference arduino.cc

Conversione gradi - radianti: **rad = grad / (180 / Pi)**
in cui:

- rad: radianti
- grad: gradi
- Pi: Pi Greco (π)

Programma/sketch

```
/* Esercizio 41  Generatore di funzioni con il plotter seriale
-- Seno, Coseno, Dente di sega, Onda Quadra, Triangolare ----
*/
void setup()
{
Serial.begin(9600);
}
void loop()
{
float angolo = 0;
```

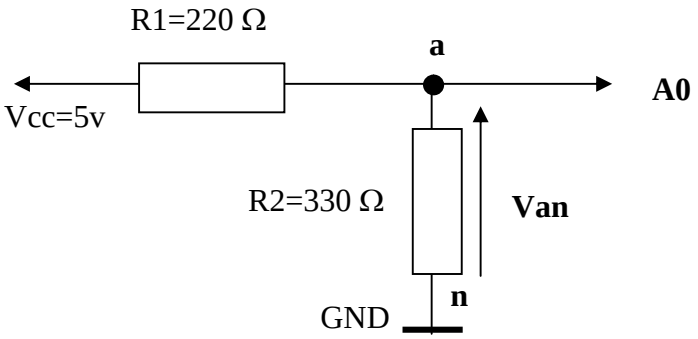
```
for (angolo = 0; angolo <= 90; angolo = angolo + 0.1)//Sfasamento seno e coseno
{
float seno = sin(angolo);// Variabile angolo seno
float coseno = cos(angolo);// Variabile angolo coseno
Serial.print(seno);
Serial.print(" ");
Serial.println(coseno);
delay(1);
}
for (float i = 0; i <= 90; i = i + 1)//Generatore funzione dente di sega
{
Serial.println(i);
delay(1);
}
for (int a = 0; a <= 100; a++)//Generatore funzione onda quadra
{
int b = 0;
Serial.println(b);
delay(1);
}
for (int a = 0; a <= 100; a++)
{
int b = 100;
Serial.println(b);
delay(1);
}
for (float i = 0; i <= 100; i = i + 1)//Generatore funzione onda triangolare
{
Serial.println(i);
delay(1);
}
for (float i = 100; i >= 0; i = i - 1)
{
Serial.println(i);
delay(1);
}
}
```

Sketch di prova:

```
void setup() {
  Serial.begin(115200);
}
int ang=0;
float rad2grad=57.295779513;
void loop() {
  Serial.print(sin(ang/rad2grad));
  Serial.print("\t");
  Serial.print(cos(ang/rad2grad));
    Serial.print("\t");
  Serial.println(0);
  ang++;
  delay(100);
}
```

Esercizio 42 - Simulazione Legge di Ohm

a) Arduino Ohm 1

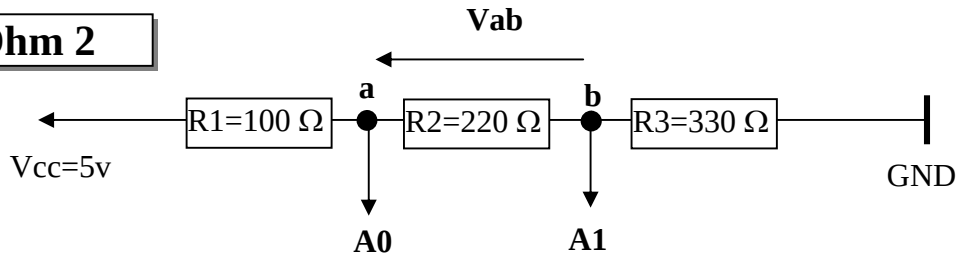


Programma

```
/* ------------------------------------------------------------------------
- Arduino Ohm 1:Partitore con 2 Resistori (R1, R2)
-- Calcolo: RTot, Corrente, Caduta di tensione ai capi di R2
-- Potenza Erogata da Vcc, potenza dissipata sui singoli Resistori
---- IPSIA - Antonio Guastaferro - San Benedetto del Tronto
---- Vers 1.0 - A.S.2017-2018 - CL:4A_IPAI
------------------------------------------------------------------------
*/

float Van,I,Ima,Pr1,Pr2,PE,Rt;
float R1=220,R2=330;
float Vcc=5.0;

void setup()
{
  Serial.begin(9600); // Inizializzazione monitor seriale
}

void loop()
{
 Rt=R1+R2;
 Van=5.0*analogRead(A0)/1024.0;

 I=Van/R2;Ima=1000.0*I;
 Pr1=1000.0*R1*I*I;Pr2=1000.0*R2*I*I;PE=Vcc*Ima;
// Invio dati al PC tramite seriale
Serial.print("Rt=");Serial.print(Rt,0);Serial.print(" Ohm");
Serial.print("  Van=");Serial.print(Van,3);Serial.print("V");
Serial.print("  I=");Serial.print(I,5);Serial.print("A");
Serial.print("  Ima=");Serial.print(Ima,2);Serial.print("mA");
Serial.print("  Pr1=");Serial.print(Pr1,2);Serial.print("mW");
Serial.print("  Pr2=");Serial.print(Pr2,2);Serial.print("mW");
Serial.print("  PE =");Serial.print(PE,2);Serial.println("mW");
delay (5000);
}
```

b) Arduino Ohm 2

Circuito



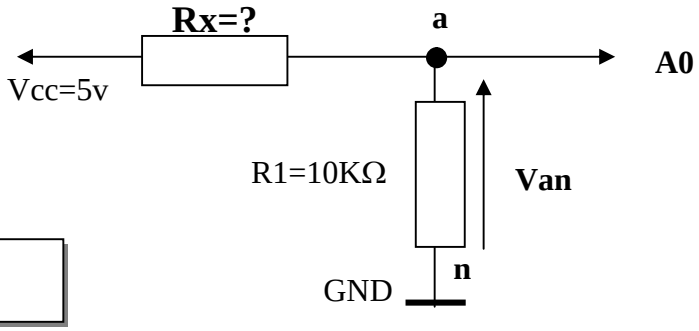
Programma

```
/* ------------------------------------------------------------------------
- Arduino Ohm 2:Partitore con 3 Resistori (R1, R2, R3)
-- Calcolo: RTot, Corrente, Caduta di tensione ai capi di R2
-- Potenza Erogata da Vcc, potenza dissipata sui singoli Resistori
---- IPSIA - Antonio Guastaferro - San Benedetto del Tronto
---- Vers 1.0 - A.S.2017-2018 - CL:4A_IPAI
------------------------------------------------------------------------------
*/
float Va,Vb,Vab,I,Ima,Pr1,Pr2,Pr3,PE,Rt;
float R1=100,R2=220,R3=330;
float Vcc=5.0;
void setup()
{
  Serial.begin(9600); // Inizializzazione monitor seriale
}
void loop()
{
 Rt=R1+R2+R3;
 Va=5.0*analogRead(A0)/1024.0;
 Vb=5.0*analogRead(A1)/1024.0;
 Vab=(Va-Vb); I=Vab/R2;Ima=1000.0*I;
 Pr1=1000.0*R1*I*I;Pr2=1000.0*R2*I*I;Pr3=1000.0*R3*I*I;PE=Vcc*Ima;
// Invio dati al PC tramite seriale
Serial.print("Rt=");Serial.print(Rt,0);Serial.print(" Ohm ");
Serial.print("  Vab=");Serial.print(Vab,3);Serial.print("V ");
Serial.print("  I=");Serial.print(I,5);Serial.print("A");
Serial.print("  Ima=");Serial.print(Ima,2);Serial.print("mA ");
Serial.print("  Pr1=");Serial.print(Pr1,2);Serial.print("mW ");
Serial.print("  Pr2=");Serial.print(Pr2,2);Serial.print("mW ");
Serial.print("  Pr3=");Serial.print(Pr3,2);Serial.print("mW ");
Serial.print("  PE =");Serial.print(PE,2);Serial.println("mW ");
delay (5000);
}
```

Risultato

```
Rt=650 Ohm  Vab=1.689V  I=0.00768A  Ima=7.68mA  Pr1=5.90mW  Pr2=12.97mW  Pr3=19.46mW  PE =38.40mW
Rt=650 Ohm  Vab=1.689V  I=0.00768A  Ima=7.68mA  Pr1=5.90mW  Pr2=12.97mW  Pr3=19.46mW  PE =38.40mW
Rt=650 Ohm  Vab=1.689V  I=0.00768A  Ima=7.68mA  Pr1=5.90mW  Pr2=12.97mW  Pr3=19.46mW  PE =38.40mW
Rt=650 Ohm  Vab=1.694V  I=0.00770A  Ima=7.70mA  Pr1=5.93mW  Pr2=13.05mW  Pr3=19.57mW  PE =38.51mW
Rt=650 Ohm  Vab=1.689V  I=0.00768A  Ima=7.68mA  Pr1=5.90mW  Pr2=12.97mW  Pr3=19.46mW  PE =38.40mW
```

c) Arduino Ohm 3 - Ohmmetro oppure Ohmetro

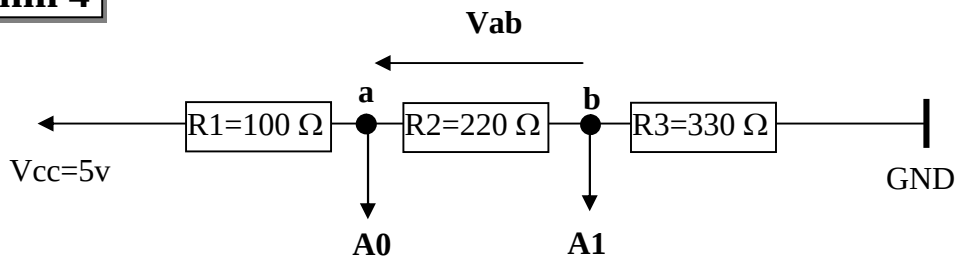


Range ottimale: 50Ω ÷ 120KΩ

```
/* --------------------------------
- Arduino Ohm 3:Ohmetro
-- Lettura valore Resistore Rx
--
---- IPSIA - Antonio Guastaferro - San Benedetto del Tronto
---- Vers 1.0 - A.S.2017-2018 - CL:4A_IPAI
-------------------------------------------------------------------------------------
*/
float Rx,Van,I;
float R1=10000;
float Vcc=5.0;

void setup()
{
  Serial.begin(9600); // Inizializzazione monitor seriale
}

void loop()
{
 Van=5.0*analogRead(A0)/1024.0;
 I=Van/R1;
 Rx=(Vcc-Van)/I;
// Invio dati al PC tramite seriale
Serial.print("Rx=");Serial.print(Rx,0);Serial.println(" Ohm");
delay (5000);
}
```

d) Arduino Ohm 4

Circuito

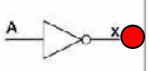
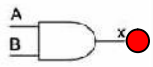
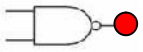
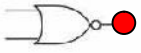
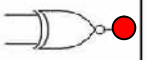


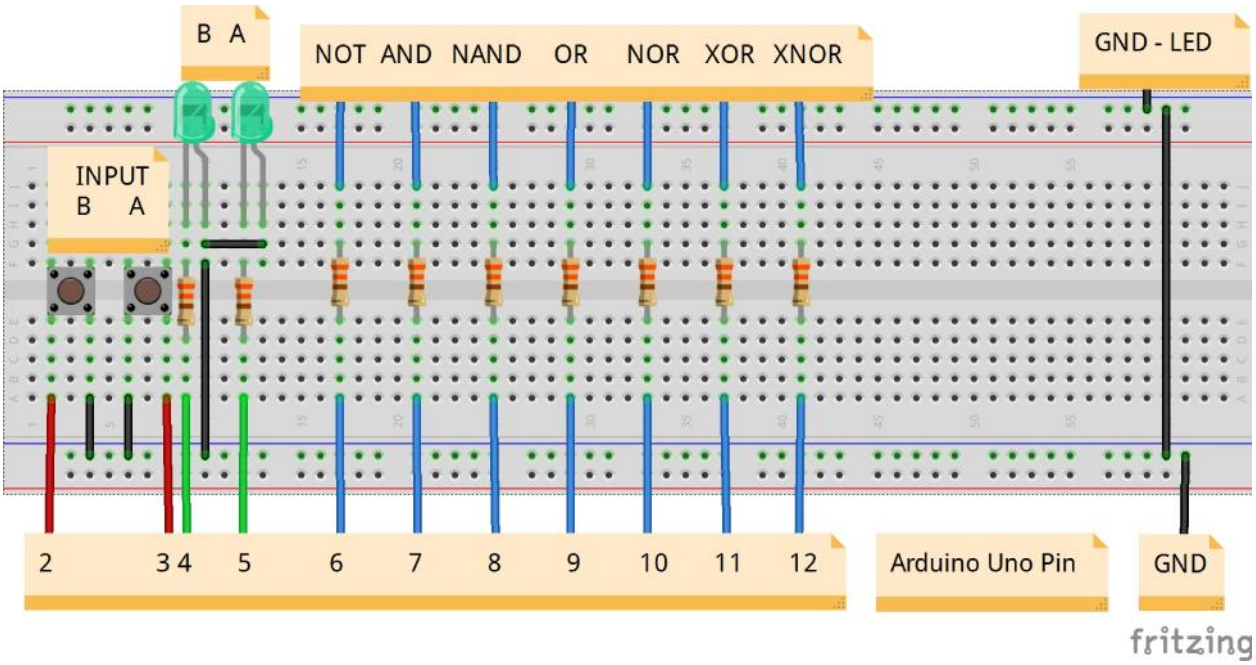
Programma

```
/* ------------------------------------------------------------------------
- Arduino Ohm 4: Struttura a ponte con 4 resistori
-- Determinazione Vab (Diagonale del Ponte)
---- IPSIA - Antonio Guastaferro - San Benedetto del Tronto
---- Vers 1.0 - A.S.2017-2018 - CL:4A_IPAI
------------------------------------------------------------------------
*/
float Va,Vb,Vab;
void setup()
{
  Serial.begin(9600); // Inizializzazione monitor seriale
}
void loop()
{

 Va=5.0*analogRead(A0)/1024.0;
 Vb=5.0*analogRead(A1)/1024.0;
 Vab=(Va-Vb);

// Invio dati al PC tramite seriale

Serial.print("  Vab=");Serial.print(Vab,3);Serial.print("V ");
delay (5000);
}
```

Esercizio 43 - Simulazione porte logiche

Logic Gates

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol							
Truth Table	A X 0 1 1 0	B A X 0 0 0 0 1 0 1 0 0 1 1 1	B A X 0 0 1 0 1 1 1 0 1 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 1	B A X 0 0 1 0 1 0 1 0 0 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 0	B A X 0 0 1 0 1 0 1 0 0 1 1 1

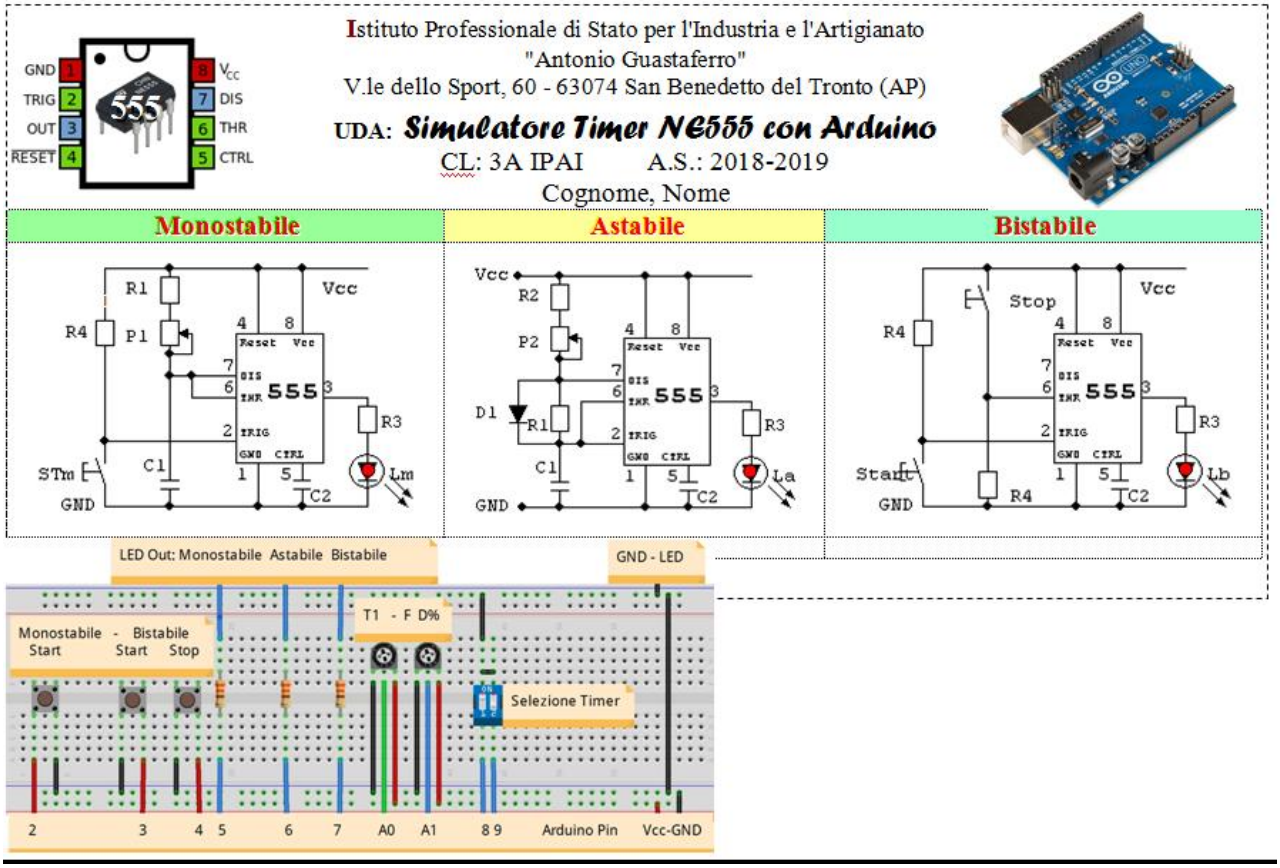



```
/* ------------------------------------------------------------
- Arduino Digitale 1:Simulazione porte logiche a due ingressi A,B
-- AND, OR, NOT, NAND, EX-OR, X-NOR
---- IPSIA - Antonio Guastaferro - San Benedetto del Tronto (AP)
---- Di Cosmo Daniel Vita Francesco Guido Santoni Fabio;
---- Vers 1.0 - A.S.2017-2018 - CL:2A_MAN
------------------------------------------------------------------
*/
void setup()
{
  pinMode(2,INPUT_PULLUP); pinMode(3,INPUT_PULLUP); // Pulsanti A,B (Input)
  pinMode(4,OUTPUT); pinMode(5,OUTPUT); // Led Input B e Led Input A
  pinMode(6,OUTPUT); pinMode(7,OUTPUT); // Led Out NOT e Led Out AND
  pinMode(8,OUTPUT); pinMode(9,OUTPUT); // Led Out NAND e Led Out OR
  pinMode(10,OUTPUT);pinMode(11,OUTPUT);// Led Out NOR e Led Out EX-OR
  pinMode(12,OUTPUT);                   // Led Out EX-NOR
}
```

```
void loop()
{
 digitalWrite(4,digitalRead(2));
 digitalWrite(5,digitalRead(3));

// ----------------------
// Porta NOT
 if(digitalRead(2)==false)
 {
 digitalWrite(6,HIGH);
 }
 else
 {
  digitalWrite(6,LOW);
 }
 // ----------------------
 // Porta AND
 if (digitalRead(2)&&digitalRead(3))
 {
  digitalWrite(7,HIGH);
 }
 else
 {
  digitalWrite(7,LOW);
 }
 // ------------------------
  // Porta NAND
 if (!(digitalRead(2)&&digitalRead(3)))
 {
  digitalWrite(8,HIGH);
 }
 else
 {
  digitalWrite(8,LOW);
 }
 // ------------------------
 // Porta OR
 if (digitalRead(2)||digitalRead(3))
 {
  digitalWrite(9,HIGH);
 }
 else
 {
  digitalWrite(9,LOW);
 }
// ------------------------
 // Porta NOR
 if (!(digitalRead(2)||digitalRead(3)))
 {
  digitalWrite(10,HIGH);
 }
 else
```

```
 {
  digitalWrite(10,LOW);
 }
 // ----------------------
// Porta EX-OR
if ((digitalRead(2)&&!digitalRead(3))||(!digitalRead(2)&&digitalRead(3)))
 {
  digitalWrite(11,HIGH);
 }
 else
 {
  digitalWrite(11,LOW);
 }
// ----------------------
// Porta EX-NOR
if (!((digitalRead(2)&&!digitalRead(3))||(!digitalRead(2)&&digitalRead(3))))
 {
  digitalWrite(12,HIGH);
 }
 else
 {
  digitalWrite(12,LOW);
 }

}
```

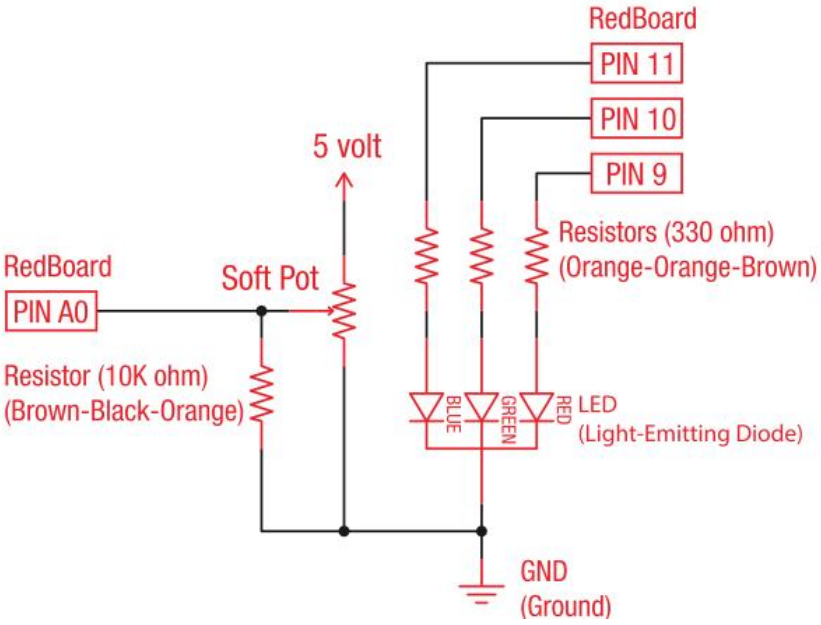
Esercizio 44 - Simulazione Timer NE555



Esercizio 45 - Es. Manuale Sik Guide Sparkfun

Circuit #10: Reading a Soft Potentiometer

In this circuit, we are going to use yet another kind of variable resistor – this time, a soft potentiometer (or soft pot). This is a thin and flexible strip that can detect where pressure is being applied. By pressing down on various parts of the strip, you can vary the resistance from 100 to 10k ohms. You can use this ability to track movement on the soft pot, or simply as a button. In this circuit, we'll get the soft pot up and running to control an RGB LED.

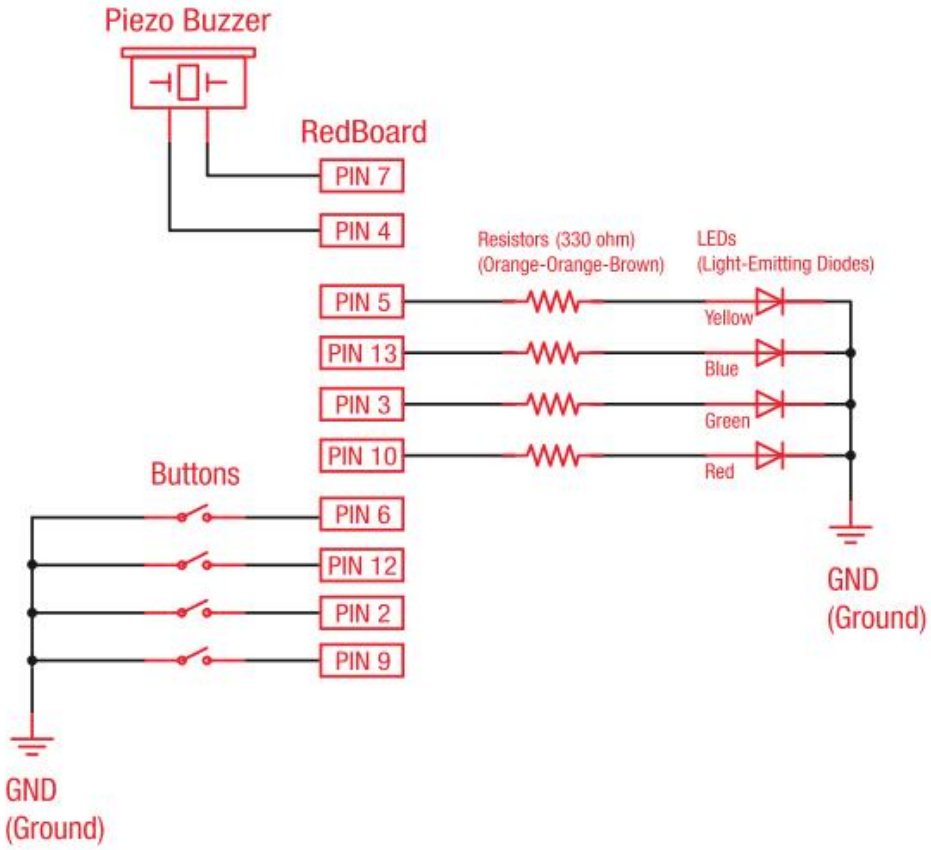


Programma

```
// Example sketch 10 - SOFT POTENTIOMETER
const int RED_LED_PIN = 9;    // Red LED Pin
const int GREEN_LED_PIN = 10; // Green LED Pin
const int BLUE_LED_PIN = 11;  // Blue LED Pin
const int SENSOR_PIN = 0;     // Analog input pin
int redValue, greenValue, blueValue;
void setup()
{
}
void loop()
{
  int sensorValue;
  sensorValue = analogRead(SENSOR_PIN);
  setRGB(sensorValue);
}
void setRGB(int RGBposition)
{
  int mapRGB1, mapRGB2, constrained1, constrained2;
  mapRGB1 = map(RGBposition, 0, 341, 255, 0);
  constrained1 = constrain(mapRGB1, 0, 255);
  mapRGB2 = map(RGBposition, 682, 1023, 0, 255);
  constrained2 = constrain(mapRGB2, 0, 255);
  redValue = constrained1 + constrained2;
  greenValue = constrain(map(RGBposition, 0, 341, 0, 255), 0, 255)
          - constrain(map(RGBposition, 341, 682, 0,255), 0, 255);
  blueValue = constrain(map(RGBposition, 341, 682, 0, 255), 0, 255)
         - constrain(map(RGBposition, 682, 1023, 0, 255), 0, 255);
  analogWrite(RED_LED_PIN, redValue);
  analogWrite(GREEN_LED_PIN, greenValue);
  analogWrite(BLUE_LED_PIN, blueValue);
}
```

Esercizio 16 - Simon Say

Now that we've learned all the basics behind the components in the SIK, let's put them together and have some fun. This circuit will show you how to create your own Simon Says game. Using some LEDs, some buttons, a buzzer and some resistors, you can create this and other exciting games with your SIK.



Programma

```
/*
Example sketch 16
 Spark Fun Electronics
 Oct. 7, 2014
 Simon Says is a memory game. Start the game by pressing one of the four buttons. When a button lights up,
 press the button, repeating the sequence. The sequence will get longer and longer. The game is won after
 13 rounds.
 Generates random sequence, plays music, and displays button lights.
 Simon tones from Wikipedia
 - A (red, upper left) - 440Hz - 2.272ms - 1.136ms pulse
 - a (green, upper right, an octave higher than A) - 880Hz - 1.136ms,
 0.568ms pulse
 - D (blue, lower left, a perfect fourth higher than the upper left)
```

```
 587.33Hz - 1.702ms - 0.851ms pulse
 - G (yellow, lower right, a perfect fourth higher than the lower left) -
 784Hz - 1.276ms - 0.638ms pulse
 Simon Says game originally written in C for the PIC16F88.
 Ported for the ATmega168, then ATmega328, then Arduino 1.0.
 Fixes and cleanup by Joshua Neal <joshua[at]trochotron.com>
 This sketch was written by SparkFun Electronics,
 with lots of help from the Arduino community.
 This code is completely free for any use.
 Visit http://www.arduino.cc to learn about the Arduino.
 */
/*************************************************
* Public Constants
*************************************************/
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
```

```
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
#define CHOICE_OFF      0 //Used to control LEDs
```

```
#define CHOICE_NONE     0 //Used to check buttons
#define CHOICE_RED  (1 << 0)
#define CHOICE_GREEN    (1 << 1)
#define CHOICE_BLUE (1 << 2)
#define CHOICE_YELLOW   (1 << 3)
#define LED_RED     10
#define LED_GREEN   3
#define LED_BLUE    13
#define LED_YELLOW  5
// Button pin definitions
#define BUTTON_RED    9
#define BUTTON_GREEN  2
#define BUTTON_BLUE   12
#define BUTTON_YELLOW 6
// Buzzer pin definitions
#define BUZZER1  4
#define BUZZER2  7
// Define game parameters
#define ROUNDS_TO_WIN      13 //Number of rounds to succesfully remember before you win. 13 is do-able.
#define ENTRY_TIME_LIMIT   3000 //Amount of time to press a button before game times out. 3000ms = 3 sec
#define MODE_MEMORY  0
#define MODE_BATTLE  1
#define MODE_BEEGEES 2
// Game state variables
byte gameMode = MODE_MEMORY; //By default, let's play the memory game
byte gameBoard[32]; //Contains the combination of buttons as we advance
byte gameRound = 0; //Counts the number of succesful rounds the player has made it through
void setup()
{
  pinMode(BUTTON_RED, INPUT_PULLUP);
  pinMode(BUTTON_GREEN, INPUT_PULLUP);
  pinMode(BUTTON_BLUE, INPUT_PULLUP);
  pinMode(BUTTON_YELLOW, INPUT_PULLUP);
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_BLUE, OUTPUT);
  pinMode(LED_YELLOW, OUTPUT);
  pinMode(BUZZER1, OUTPUT);
  pinMode(BUZZER2, OUTPUT);
  //Mode checking
  gameMode = MODE_MEMORY; // By default, we're going to play the memory game
  if (checkButton() == CHOICE_YELLOW) play_beegees();
  if (checkButton() == CHOICE_GREEN)
  {
    gameMode = MODE_BATTLE; //Put game into battle mode
    setLEDs(CHOICE_GREEN);
    toner(CHOICE_GREEN, 150);
    setLEDs(CHOICE_RED | CHOICE_BLUE | CHOICE_YELLOW);
    while(checkButton() != CHOICE_NONE) ; // Wait for user to stop pressing button
    //Now do nothing. Battle mode will be serviced in the main routine
  }
  play_winner(); // After setup is complete, say hello to the world
```

```
}
void loop()
{
  attractMode(); // Blink lights while waiting for user to press a button
  setLEDs(CHOICE_RED | CHOICE_GREEN | CHOICE_BLUE | CHOICE_YELLOW);
  delay(1000);
  setLEDs(CHOICE_OFF); // Turn off LEDs
  delay(250);
  if (gameMode == MODE_MEMORY)
  {
    if (play_memory() == true)
      play_winner(); // Player won, play winner tones
    else
      play_loser(); // Player lost, play loser tones
  }

  if (gameMode == MODE_BATTLE)
  {
    play_battle(); // Play game until someone loses

    play_loser(); // Player lost, play loser tones
  }
}
boolean play_memory(void)
{
  randomSeed(millis()); // Seed the random generator with random amount of millis()
  gameRound = 0; // Reset the game to the beginning
  while (gameRound < ROUNDS_TO_WIN)
  {
    add_to_moves(); // Add a button to the current moves, then play them back
    playMoves(); // Play back the current game board
    for (byte currentMove = 0 ; currentMove < gameRound ; currentMove++)
    {
      byte choice = wait_for_button(); // See what button the user presses
      if (choice == 0) return false; // If wait timed out, player loses
      if (choice != gameBoard[currentMove]) return false; // If the choice is incorect, player loses
    }
    delay(1000); // Player was correct, delay before playing moves
  }
  return true; // Player made it through all the rounds to win!
}
boolean play_battle(void)
{
  gameRound = 0; // Reset the game frame back to one frame
  while (1) // Loop until someone fails
  {
    byte newButton = wait_for_button(); // Wait for user to input next move
    gameBoard[gameRound++] = newButton; // Add this new button to the game array
    for (byte currentMove = 0 ; currentMove < gameRound ; currentMove++)
    {
      byte choice = wait_for_button();
      if (choice == 0) return false; // If wait timed out, player loses.
```

```
      if (choice != gameBoard[currentMove]) return false; // If the choice is incorect, player loses.
    }
    delay(100); // Give the user an extra 100ms to hand the game to the other player
  }
  return true; // We should never get here
}
void playMoves(void)
{
  for (byte currentMove = 0 ; currentMove < gameRound ; currentMove++)
  {
    toner(gameBoard[currentMove], 150);
    delay(150); // 150 works well. 75 gets fast.
  }
}
void add_to_moves(void)
{
  byte newButton = random(0, 4); //min (included), max (exluded)
  if(newButton == 0) newButton = CHOICE_RED;
  else if(newButton == 1) newButton = CHOICE_GREEN;
  else if(newButton == 2) newButton = CHOICE_BLUE;
  else if(newButton == 3) newButton = CHOICE_YELLOW;
  gameBoard[gameRound++] = newButton; // Add this new button to the game array
}
void setLEDs(byte leds)
{
  if ((leds & CHOICE_RED) != 0)
    digitalWrite(LED_RED, HIGH);
  else
    digitalWrite(LED_RED, LOW);
  if ((leds & CHOICE_GREEN) != 0)
    digitalWrite(LED_GREEN, HIGH);
  else
    digitalWrite(LED_GREEN, LOW);
  if ((leds & CHOICE_BLUE) != 0)
    digitalWrite(LED_BLUE, HIGH);
  else
    digitalWrite(LED_BLUE, LOW);
  if ((leds & CHOICE_YELLOW) != 0)
    digitalWrite(LED_YELLOW, HIGH);
  else
    digitalWrite(LED_YELLOW, LOW);
}
byte wait_for_button(void)
{
  long startTime = millis(); // Remember the time we started the this loop
  while ( (millis() - startTime) < ENTRY_TIME_LIMIT) // Loop until too much time has passed
  {
    byte button = checkButton();
    if (button != CHOICE_NONE)
    {
      toner(button, 150); // Play the button the user just pressed
      while(checkButton() != CHOICE_NONE) ;  // Now let's wait for user to release button
```

```
      delay(10); // This helps with debouncing and accidental double taps
      return button;
    }
  }
  return CHOICE_NONE; // If we get here, we've timed out!
}
// Returns a '1' bit in the position corresponding to CHOICE_RED, CHOICE_GREEN, etc.
byte checkButton(void)
{
  if (digitalRead(BUTTON_RED) == 0) return(CHOICE_RED);
  else if (digitalRead(BUTTON_GREEN) == 0) return(CHOICE_GREEN);
  else if (digitalRead(BUTTON_BLUE) == 0) return(CHOICE_BLUE);
  else if (digitalRead(BUTTON_YELLOW) == 0) return(CHOICE_YELLOW);
  return(CHOICE_NONE); // If no button is pressed, return none
}

// Light an LED and play tone
// Red, upper left:     440Hz - 2.272ms - 1.136ms pulse
// Green, upper right:  880Hz - 1.136ms - 0.568ms pulse
// Blue, lower left:    587.33Hz - 1.702ms - 0.851ms pulse
// Yellow, lower right: 784Hz - 1.276ms - 0.638ms pulse
void toner(byte which, int buzz_length_ms)
{
  setLEDs(which); //Turn on a given LED

  //Play the sound associated with the given LED
  switch(which)
  {
  case CHOICE_RED:
    buzz_sound(buzz_length_ms, 1136);
    break;
  case CHOICE_GREEN:
    buzz_sound(buzz_length_ms, 568);
    break;
  case CHOICE_BLUE:
    buzz_sound(buzz_length_ms, 851);
    break;
  case CHOICE_YELLOW:
    buzz_sound(buzz_length_ms, 638);
    break;
  }

  setLEDs(CHOICE_OFF); // Turn off all LEDs
}

// Toggle buzzer every buzz_delay_us, for a duration of buzz_length_ms.
void buzz_sound(int buzz_length_ms, int buzz_delay_us)
{
  // Convert total play time from milliseconds to microseconds
  long buzz_length_us = buzz_length_ms * (long)1000;

  // Loop until the remaining play time is less than a single buzz_delay_us
```

```
  while (buzz_length_us > (buzz_delay_us * 2))
  {
    buzz_length_us -= buzz_delay_us * 2; //Decrease the remaining play time

    // Toggle the buzzer at various speeds
    digitalWrite(BUZZER1, LOW);
    digitalWrite(BUZZER2, HIGH);
    delayMicroseconds(buzz_delay_us);

    digitalWrite(BUZZER1, HIGH);
    digitalWrite(BUZZER2, LOW);
    delayMicroseconds(buzz_delay_us);
  }
}

// Play the winner sound and lights
void play_winner(void)
{
  setLEDs(CHOICE_GREEN | CHOICE_BLUE);
  winner_sound();
  setLEDs(CHOICE_RED | CHOICE_YELLOW);
  winner_sound();
  setLEDs(CHOICE_GREEN | CHOICE_BLUE);
  winner_sound();
  setLEDs(CHOICE_RED | CHOICE_YELLOW);
  winner_sound();
}

// Play the winner sound
// This is just a unique (annoying) sound we came up with, there is no magic to it
void winner_sound(void)
{
  // Toggle the buzzer at various speeds
  for (byte x = 250 ; x > 70 ; x--)
  {
    for (byte y = 0 ; y < 3 ; y++)
    {
      digitalWrite(BUZZER2, HIGH);
      digitalWrite(BUZZER1, LOW);
      delayMicroseconds(x);

      digitalWrite(BUZZER2, LOW);
      digitalWrite(BUZZER1, HIGH);
      delayMicroseconds(x);
    }
  }
}

// Play the loser sound/lights
void play_loser(void)
{
  setLEDs(CHOICE_RED | CHOICE_GREEN);
```

```
  buzz_sound(255, 1500);

  setLEDs(CHOICE_BLUE | CHOICE_YELLOW);
  buzz_sound(255, 1500);

  setLEDs(CHOICE_RED | CHOICE_GREEN);
  buzz_sound(255, 1500);

  setLEDs(CHOICE_BLUE | CHOICE_YELLOW);
  buzz_sound(255, 1500);
}

// Show an "attract mode" display while waiting for user to press button.
void attractMode(void)
{
  while(1)
  {
    setLEDs(CHOICE_RED);
    delay(100);
    if (checkButton() != CHOICE_NONE) return;

    setLEDs(CHOICE_BLUE);
    delay(100);
    if (checkButton() != CHOICE_NONE) return;

    setLEDs(CHOICE_GREEN);
    delay(100);
    if (checkButton() != CHOICE_NONE) return;

    setLEDs(CHOICE_YELLOW);
    delay(100);
    if (checkButton() != CHOICE_NONE) return;
  }
}
int melody[] = {
  NOTE_G4, NOTE_A4, 0, NOTE_C5, 0, 0, NOTE_G4, 0, 0, 0,
  NOTE_E4, 0, NOTE_D4, NOTE_E4, NOTE_G4, 0,
  NOTE_D4, NOTE_E4, 0, NOTE_G4, 0, 0,
  NOTE_D4, 0, NOTE_E4, 0, NOTE_G4, 0, NOTE_A4, 0, NOTE_C5, 0};
int noteDuration = 115;
int LEDnumber = 0; // Keeps track of which LED we are on during the beegees loop
void play_beegees()
{
  setLEDs(CHOICE_YELLOW);
  toner(CHOICE_YELLOW, 150);
  setLEDs(CHOICE_RED | CHOICE_GREEN | CHOICE_BLUE);
  while(checkButton() != CHOICE_NONE) ; // Wait for user to stop pressing button
  setLEDs(CHOICE_NONE); // Turn off LEDs
  delay(1000); // Wait a second before playing song
  digitalWrite(BUZZER1, LOW); .
  while(checkButton() == CHOICE_NONE) //Play song until you press a button
  {
```

```
    for (int thisNote = 0; thisNote < 32; thisNote++) {
      changeLED();
      tone(BUZZER2, melody[thisNote],noteDuration);
      int pauseBetweenNotes = noteDuration * 1.30;
      delay(pauseBetweenNotes);
      noTone(BUZZER2);
    }
  }
}

void changeLED(void)
{
  setLEDs(1 << LEDnumber); // Change the LED

  LEDnumber++; // Goto the next LED
  if(LEDnumber > 3) LEDnumber = 0; // Wrap the counter if needed
}
```