I.T.I.S. & L.S.
F. GIORDANI
TEC LIM

ITIS-LS "Francesco Giordani" Caserta

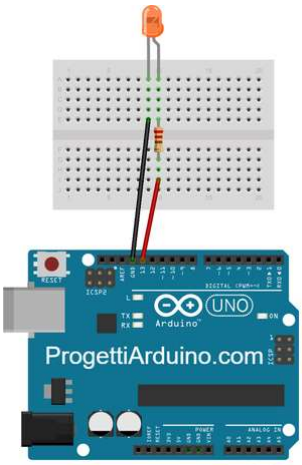
prof. Ennio Ranucci

a.s. 2019-2020

Realizziamo semplici circuiti con Arduino

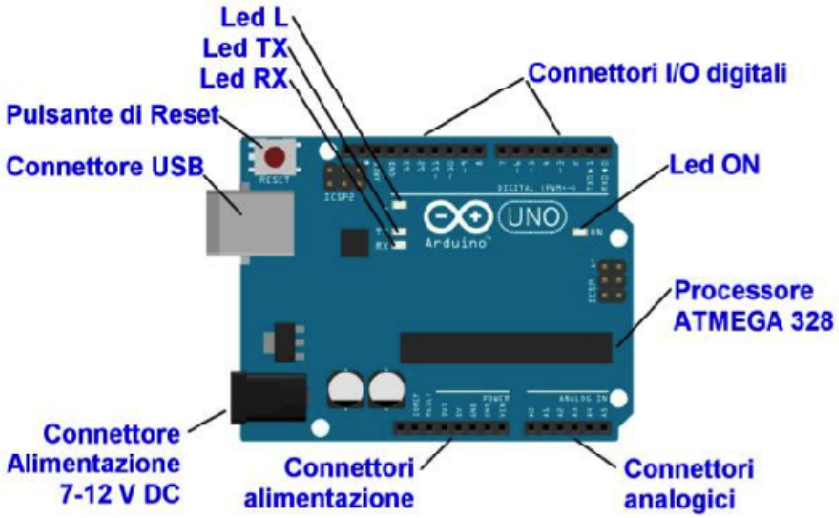
Esercitazioni svolte in ambiente Arduino

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni
Data:
Numero progressivo dell'esercizio: *es0*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Arduino 1.8.10*
Obiettivo didattico: *Conoscere l'ambiente di programmazione;*
Obiettivo del programma:
Programma c++ e circuito per far lampeggiare un LED.




```
// Arduino blink facciamo lampeggiare un led
  #define LED 13          // LED collegato al pin digitale 13
void setup()
{
   pinMode(LED, OUTPUT);     // imposta il pin digitale come output
}
void loop()
{
  digitalWrite(LED, HIGH);  // accende il LED
  delay(1000);              // aspetta un second
  digitalWrite(LED, LOW);   // spegne il LED
  delay(1000);              // aspetta un second
}
```

Arduino Uno è un dispositivo basato su microcontrollore che permette di realizzare diversi tipi di circuiti elettronici. Possiede 15 pin digitali programmabili come ingressi o uscite e 6 ingressi per l'acquisizione ed elaborazione di segnali analogici. Il microcontrollore è l'ATmega328 prodotto da Atmel, ha una velocità di 16MHz, una memoria flash da 32KB, una sram da 2KB e una memoria EEPROM da 1KB. L'alimentazione della scheda avviene tramite porta usb o tramite l'apposito connettore.



1) Arduino-Compatible UNO R3 board x 1

2) UNO Development expansion board x 1

3) Mini bread board x 1

4) Bread board (830 fori)x 1

5) Contenitore di componenti SMD x 1

6) LED (Rossi) x5

7) LED (Gialli) x5

8) LED (Verdi) x5

9) Buzzer x2

10) Bottoni x5

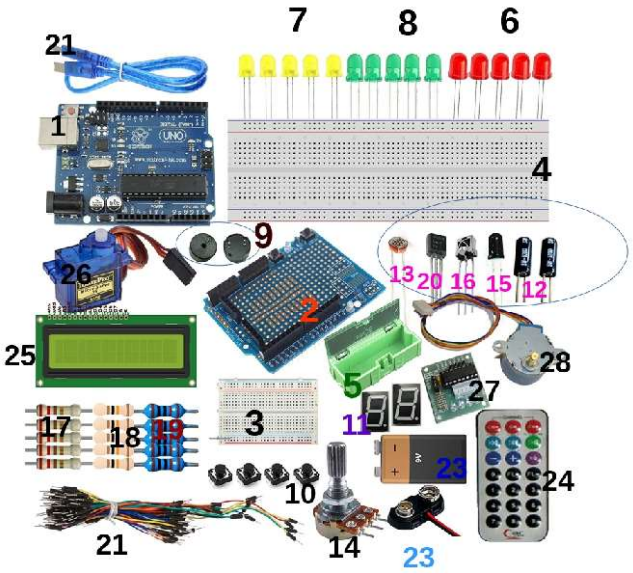
11) LED display a 7 segmenti (1-digit) x 2

12) Giroscopio x 2

13) Fotoresistenza (light sensor) x 1

14) Potenziometro x5

15) Sensore incendio x 1



16) Sensore a infrarossi x 1

17) 220 Ohm resistore x 5

18) 1k Ohm resistore x 5

19) 10k Ohm resistore x 5

20) LM35 sensore di temperatura x 1

21) Cavo usb x1

22) Connettori (wire jumper) x20

23) Plug batteria 9V + batteria 9V x 1

24) IR Controllo remoto x1

25) 1602 LCD modulo x 1

26) SG90 9G Servo motore x1

27) ULN2003 driver board x 1

28) 5V motore a step x 1

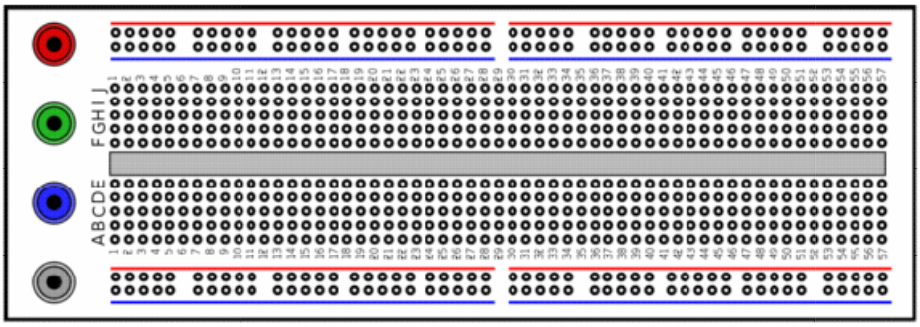
Arduino Uno R3 Board Starter Kit with LCD Servo Motor Sensor

*La **breadboard** permette di realizzare velocemente prototipi di circuiti senza bisogno di fare saldature. E' una basetta con dei fori dove, con una lieve pressione, si inseriscono i terminali dei componenti, collegandoli uno con l'altro. Il circuito così realizzato può essere facilmente modificato e smontato per riutilizzare i componenti.*

I fori della breadboard sono collegati tra loro elettricamente secondo questo schema:

i fori di ogni riga rossa e blu orizzontale sono collegati tra loro

i fori delle righe verticali sono collegati tra loro cinque a cinque



*I diodi **LED** (Light Emetting Diode)sono costantemente impiegati nel fermodellismo in quanto non si riscaldano eccessivamente, hanno un limitato assorbimento di corrente e sono di diversi colori, forme e dimensioni. Il terminale contraddistinto dalla lettera A è l'anodo e per distinguerlo, risulta sempre più lungo del Catodo, contraddistinto dalla lettera K.*
Il catodo può essere identificato anche da una smussatura alla base del corpo cilindrico del led.
Per accendere il led è necessario collegare l'anodo al positivo ed il catodo verso la massa od al negativo di alimentazione. Se accidentalmente si invertisse il collegamento il led rimarrebbe spento.
E' assolutamente necessario interporre in serie all'alimentazione, indifferentemente dal lato del catodo o dell'anodo, una resistenza che limiti il passaggio della corrente, in quanto il valore ottimale per il corretto funzionamento del led è compreso fra 15 e 17 milliampere.
Correnti superiori a 25 milliampere metterebbero presto il led fuori uso, mentre con correnti non inferiori a 3 milliampereè comunque assicurato un corretto funzionamento ma con minore luminosità.
La semplice formula per calcolare il valore della resistenza è la seguente:

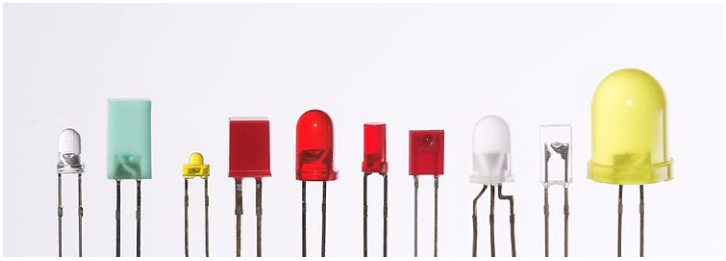
ohm = (Vcc – 1,5) : 0,016

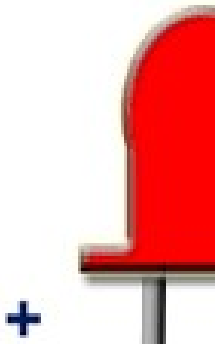
Il risultato (ohm) darà il valore della resistenza da collegare al circuito.

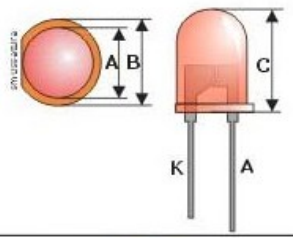
– Vcc è la tensione continua di alimentazione indicata in Volt

– 1,5 è il valore caratteristico della caduta interna del led

– 0,016 rappresenta il valore medio di corrente in amper.

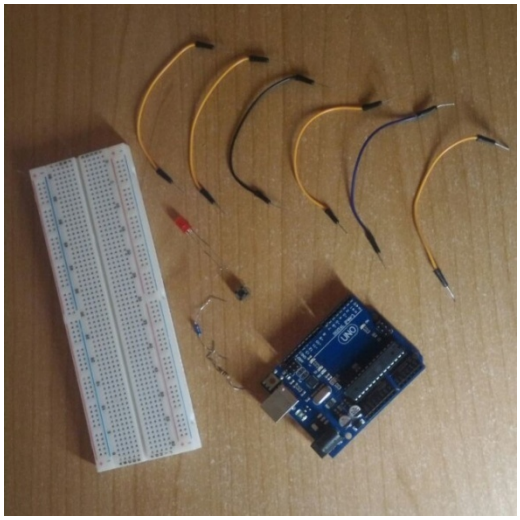


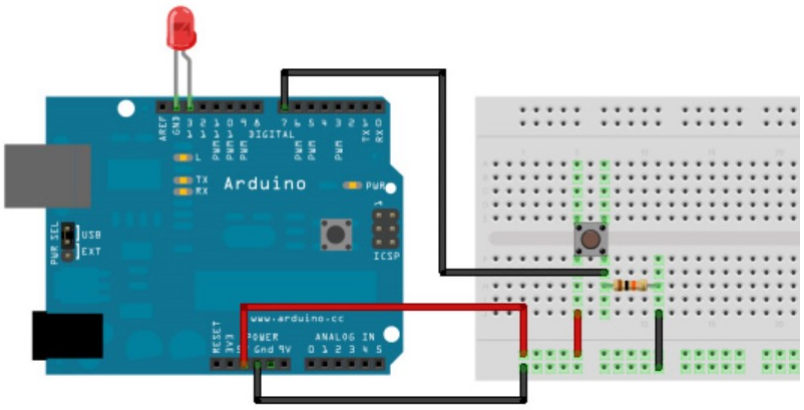


	Simbolo	Misure in mm.	Colore	Note
	 K = Catodo A = Anodo	A = 5 - B = 5,8 C = 9 A = 3 - B = 4 C = 5,5	Rosso Giallo Verde	Monocolore. La smussatura e la dimensione (più corto) indica il Catodo

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni*– articolazione Informatica*
Data:
Numero progressivo dell'esercizio: *es1*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Arduino 1.8.10*
Obiettivo didattico: *Utilizzare il costrutto "if then else"*
Obiettivo del programma:
far illuminare un led mediante un pulsante

Elenco attrezzature e materiali:
scheda arduino-led-pulsante-resistore 220 ohms ,
resistore 10 k-cavi di collegamento (x6)
breadboard-cavo usb per collegare arduino con il pc



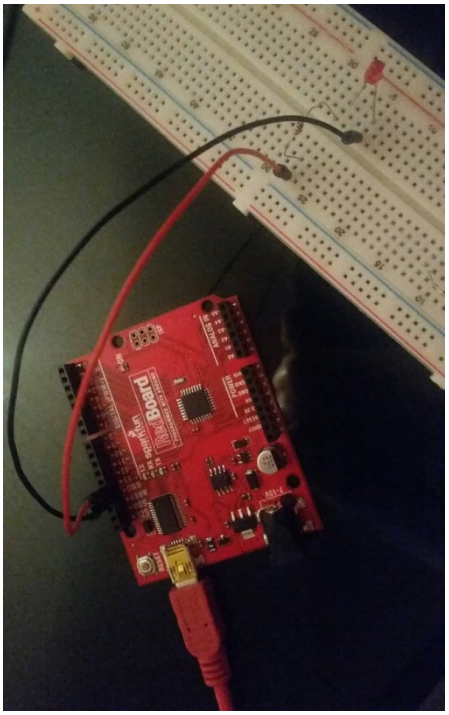


//pulsanteconled

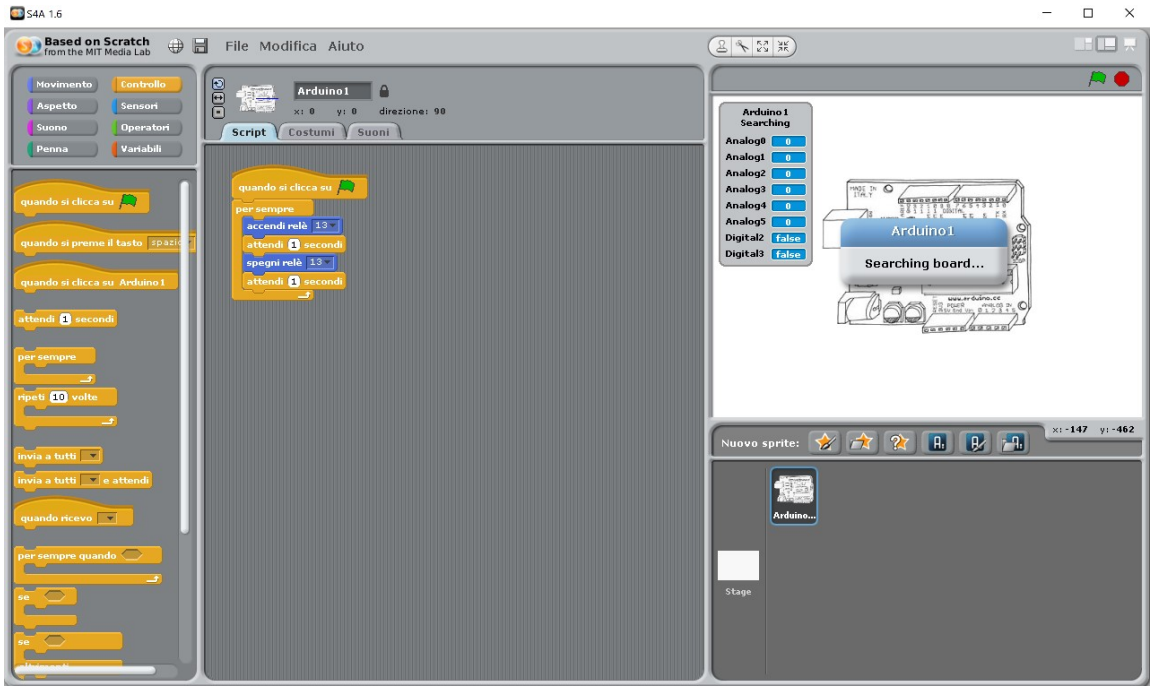
#define LED 13 //led collegato al pin 13
#define BUTTON 7 //pulsante collegato al pin 7
int ButtonStato; //dichiarazione della variabile stato del pulsante
bool stato= LOW; //dichiariamo lo stato del led basso
void setup()

```
{
  pinMode (LED, OUTPUT); //imposta il led come output
  pinMode (BUTTON, INPUT); //imposta il pulsante come input
}

void loop()
{
  ButtonStato= digitalRead (BUTTON); //leggi stato sul pin del pulsante
  if (ButtonStato==HIGH) //se il pulsante e' premuto

{
  if (stato==HIGH) stato=LOW;  //se il led e' gia' acceso spegnilo
  else stato=HIGH;  //altrimenti accendilo
}
digitalWrite(LED, stato);  //scrivi su pin LED il contenuto della variabile stato (1/0)
delay(100);  //aspetta un decimo di secondo
}
```

ITIS-LS "Francesco Giordani" Caserta
Anno scolastico: *2019/2020*
Classe 3^ sez.B spec. Informatica e telecomunicazioni*– articolazione Informatica*
Data:
Numero progressivo dell'esercizio: *es2*
Versione: *1.0*
Programmatore/i:
Sistema Operativo: *Windows 10*
Compilatore/Interprete: *Arduino 1.8.10*
Obiettivo didattico: *Programmare Arduino con Scratch*
Obiettivo del programma:
far illuminare un led mediante un pulsante



1. *Scaricare S4A dal sito http://s4a.cat/*
2. *Installare il programma S4A*
3. *Scaricare il firmware S4AFirmware16.ino*
4. *Aprire l'ambiente di programmazione Arduino, caricare S4AFirmware16.ino*
5. *Lanciare S4A ed attendere il riconoscimento della scheda arduino*
6. *Realizzare il ciurcuito in figura*
7. *Scrivere il programma Scratch in figura*
8. *Click sulla bandierina verde*



ITIS-LS "Francesco Giordani" Caserta

Anno scolastico: *2019/2020*

Classe 3^ sez.B spec. Informatica e telecomunicazioni*– articolazione Informatica*

Data:

Numero progressivo dell'esercizio: *es3*

Versione: *1.0*

Programmatore/i:

Sistema Operativo: *Windows 10*

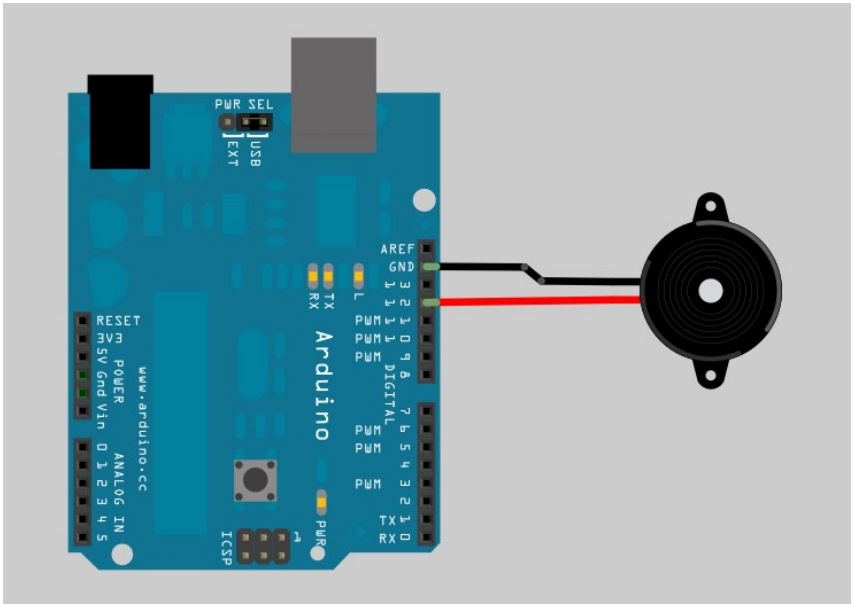
Compilatore/Interprete: *Arduino 1.8.10*

Obiettivo didattico: *Programmare Arduino*

Obiettivo del programma:

Utilizzare un buzzer

Il buzzer, anche conosciuto in Italia come "cicalino", venne introdotto alla fine degli anni ottanta del secolo scorso ed è tutt'oggi utilizzato su diversi dispositivi adoperati quotidianamente. In pratica è un componente elettronico di segnalazione audio: un piccolo altoparlante in grado di emettere toni a determinate frequenze (il classico "beep!"). Esistono buzzer attivi e passivi, la principale differenza tra essi consiste nel fatto che un buzzer di tipo "attivo" è in grado di generare toni in autonomia (ovviamente si parla di toni preimpostati dal produttore), mentre il buzzer "passivo" necessita di un microcontrollore che generi una forma d'onda in modo tale da far vibrare la membrana interna del componente emettendo così la frequenza desiderata. Arduino ben si presta per adempiere a questo scopo. I buzzer di tipo "attivo" sono dotati di ulteriore elettronica mentre quelli passivi ne risultano sprovvisti. Passiamo alla costruzione del circuito.
Abbiamo deciso di utilizzare il classico buzzer per la segnalazione degli errori delle schede madri poiché questo è sicuramente quello di uso più comunque e di conseguenza il più facile da reperire in qualsiasi rivendita. Per costruire il circuito sarà sufficiente collegare il polo negativo del buzzer a GND e quello positivo a all'uscita digitale numero 2 (D2). Ovviamente si sarebbe potuta utilizzare un'uscita digitale qualunque.



void setup()

{

```
pinMode(2,OUTPUT); //buzzer connesso al pin 2

}

void loop()

{

tone(2, 1000, 500); //suona una nota alla frequenza di 1000Hz per la durata di 500 ms

delay(1500); //attende 1500 ms

tone(2, 1000, 500);

delay(1500);

tone(2, 1000, 2000); //suona una nota alla frequenza di 1000Hz per la durata di 2 secondi

delay(2000); //attende 2 secondi

}
```

è possibile ottenere molto di più dal buzzer ed un Arduino, infatti è possibile ricreare vere e proprie musiche dato che ogni nota ha una specifica frequenza e un esatto periodo in millisecondi.
Il nostro sistema uditivo riesce a percepire sensazioni sonore in un determinato intervallo di frequenze che appartiene al range di 20 Hz - 20 kHz. L'infrasuono appartiene ad un range di frequenze inferiore alla soglia di 20 Hz come limite inferiore per l'orecchio umano. Lo studio della loro gamma è la stessa utilizzata dai sismografi per il monitoraggio degli eventi sismici. Questi segnali possono propagarsi per lunghe distanze e aggirare ostacoli con molta semplicità. Al di sopra dei 20 kHz ci sono gli ultrasuoni, ovvero tutto quello che è al di là del suono fisico udibile. Come per altri tipi di segnali, anche in questo caso, ogni suono può essere caratterizzato da parametri quali l'intensità e la frequenza, oltre ad essere soggetto a fenomeni di riflessione e rifrazione. I musicisti individuano per ogni nota musicale una frequenza particolare, quest'ultima usata come standard internazionale. Per calcolare la frequenza delle note si utilizza la seguente formula:

$f = 2N/12 * frif$

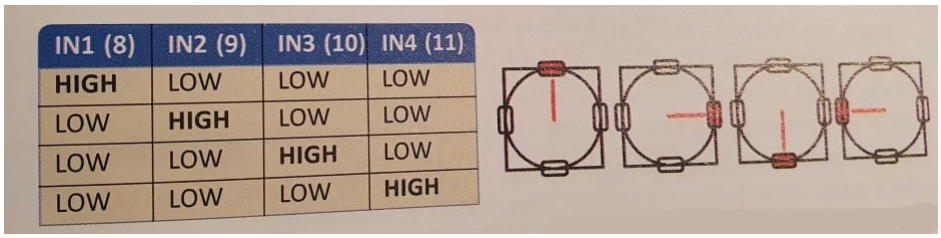
dove N è il numero di semitoni ed frif è la frequenza di riferimento pari a 440 Hz.

La funzione "tone" può essere richiamata nel seguente modo: tone(pin, frequency), oppure tone (pin, frequenza, duration). La variabile pin identifica l'uscita di Arduino (pin 2 nel costro caso) da collegare all'altoparlante; la variabile frequency invece esprime in Hz la frequenza del tono, infine duration è la durata del tono in millisecondi (opzionale) - di tipo unsigned long.
La funzione una volta richiamata inizia la riproduzione del suono e passa subito alla parte successiva del codice senza attendere che la riproduzione sia completata, è quindi necessario, in caso si voglia riprodurre una serie di note, inserire un opportuno delay uguale o maggiore alla durata del tono, affinché le note non si sovrappongano.

Il file di inclusione denominato pitches.h contiene tutti i valori di frequenza per le note (tipiche) del tono da riprodurre. Ad esempio, NOTE_C4 è do4, NOTE_FS4 è FA diesis, e così via in accordo alle frequenze delle note musicali. Così, invece di scrivere la frequenza nella funzione tone (), non resta che scrivere il nome della nota. C4, D4, E4, F4, G4, A4, B4 corrispondono a DO, RE, MI, FA, SOL ,LA, SI .

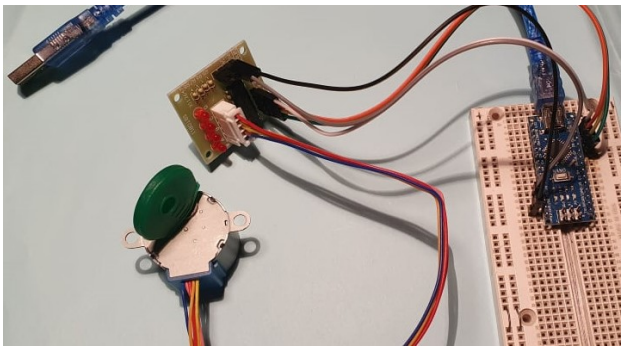
Il Motore passo-passo (Stepper)

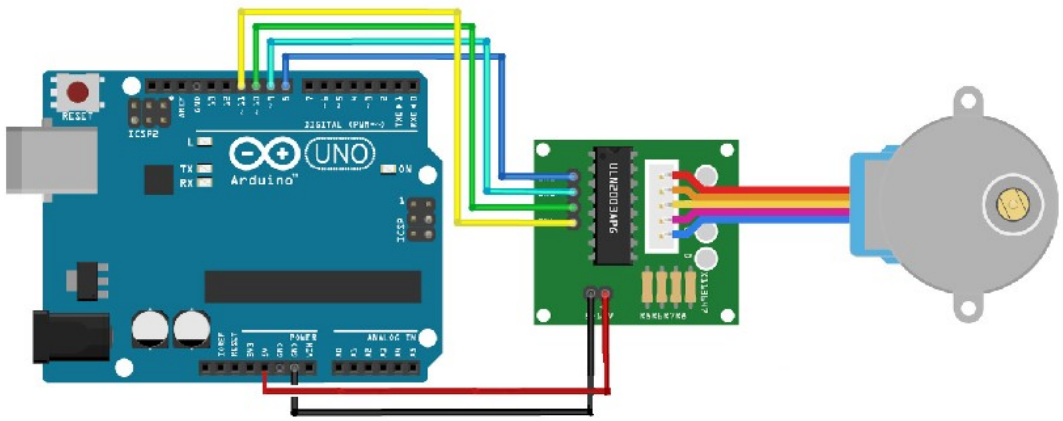
Il motore passo-passo, detto anche stepper motors, è un motore elettrico sincrono in corrente continua, senza spazzole, che permette la suddivisione della rotazione in piccoli angoli detti step.
E' un motore molto preciso e veloce e facilmente controllabile tramite una scheda elettronica, denominata "driver".
Il nome passo-passo deriva dalla possibilità di compiere una rotazione completa attraverso un certo numero di micromovimenti detti passi(o step).
Il rotore è un magnete permanente a forma cilindrica; lo statore è composto da 4 avvolgimenti.
Il movimento avviene inviando corrente alternativamente ai vari avvolgimenti, come si vede nella seguente figura (ad ogni avvolgimento corrisponde un pin del driver):



IN1 (8)	IN2 (9)	IN3 (10)	IN4 (11)
HIGH	LOW	LOW	LOW
LOW	**HIGH**	LOW	LOW
LOW	LOW	**HIGH**	LOW
LOW	LOW	LOW	**HIGH**

In1, in2, in3,in4 sono i pin del driver
8,9,10,11 sono i pin di arduino
Nella foto sottostante si vedono:
Arduino NANO
Il motore 28BY-48 5 v e il driver SBT0811





C Arduino

Esercitazione Stepper 1

```
byte sbt0811_in[4] = {8,9,10,11};   //IN1,IN2,IN3,IN4 driver->arduino
void muoviMotore(int direzionePar)
{
   if (direzionePar==1)
   {
    digitalWrite(sbt0811_in[0],HIGH);// situazione avvolgimenti 1000
    digitalWrite(sbt0811_in[1],LOW);
    digitalWrite(sbt0811_in[2],LOW);
    digitalWrite(sbt0811_in[3],LOW);
    delay(10);
    digitalWrite(sbt0811_in[0],LOW); //situazione Avvolgimenti 0100
    digitalWrite(sbt0811_in[1],HIGH);
    digitalWrite(sbt0811_in[2],LOW);
    digitalWrite(sbt0811_in[3],LOW);
    delay(10);
    digitalWrite(sbt0811_in[0],LOW); //situazione avvolgimenti 0010
    digitalWrite(sbt0811_in[1],LOW);
    digitalWrite(sbt0811_in[2],HIGH);
    digitalWrite(sbt0811_in[3],LOW);
    delay(10);
    digitalWrite(sbt0811_in[0],LOW); //situazione avvolgimenti 0001
    digitalWrite(sbt0811_in[1],LOW);
    digitalWrite(sbt0811_in[2],LOW);
    digitalWrite(sbt0811_in[3],HIGH);
    delay(10);
   }
   else
   {
    digitalWrite(sbt0811_in[0],LOW); //situazione avvolgimenti 0001
    digitalWrite(sbt0811_in[1],LOW);
    digitalWrite(sbt0811_in[2],LOW);
    digitalWrite(sbt0811_in[3],HIGH);
    delay(10);
    digitalWrite(sbt0811_in[0],LOW); //situazione avvolgimenti 0010
    digitalWrite(sbt0811_in[1],LOW);
    digitalWrite(sbt0811_in[2],HIGH);
    digitalWrite(sbt0811_in[3],LOW);
    delay(10);
```

```
    digitalWrite(sbt0811_in[0],LOW); //situazione avvolgimenti 0100
    digitalWrite(sbt0811_in[1],HIGH);
    digitalWrite(sbt0811_in[2],LOW);
    digitalWrite(sbt0811_in[3],LOW);
    delay(10);
    digitalWrite(sbt0811_in[0],HIGH);// situazione avvolgimenti 1000
    digitalWrite(sbt0811_in[1],LOW);
    digitalWrite(sbt0811_in[2],LOW);
    digitalWrite(sbt0811_in[3],LOW);
    delay(10);
  }
}
void setup()
{
   for (byte i = 0; i <= 3; i++)
   {
      pinMode(sbt0811_in[i], OUTPUT);
   }
}
void loop()
{
  for (int i=0; i<100;i++)
  {
   muoviMotore(1);
  }
  for (int i=0; i<100;i++)
  {
    muoviMotore(0);
  }
}
```

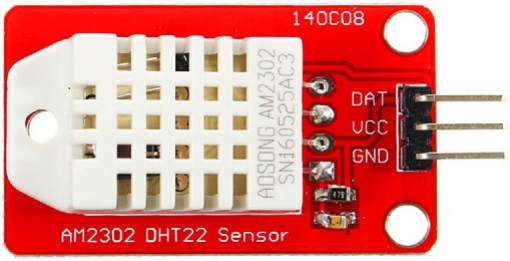
Esercitazione Stepper 2

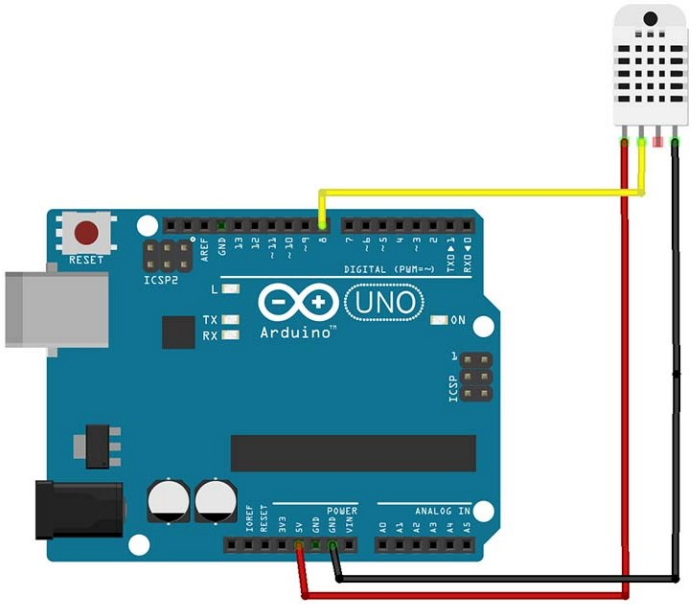
```
byte sbt0811_in[4] = {8,9,10,11};   //IN1,IN2,IN3,IN4 driver->arduino
boolean avvolgimenti[4][4] =
{
   { 1,0,0,0 },
   { 0,1,0,0 },
   { 0,0,1,0 },
   { 0,0,0,1 },
};
```

```
void muoviMotoreConMatrice(int direzionePar)
{
   if (direzionePar==1)
   {
    for (int rigaCorrente=0;rigaCorrente<=3; rigaCorrente++)
      {
       for (int i=0; i<=3; i++)
         {
           digitalWrite(sbt0811_in[i], avvolgimenti[rigaCorrente][i]);
         }
       delay(10);
      }
   }
   else
   {
    for (int rigaCorrente=3;rigaCorrente>=0; rigaCorrente--)
      {
       for (int i=0; i<=3; i++)
         {
           digitalWrite(sbt0811_in[i], avvolgimenti[rigaCorrente][i]);
         }
       delay(10);
      }
   }
}
void setup()
{
   for (byte i = 0; i <= 3; i++)
   {
      pinMode(sbt0811_in[i], OUTPUT);
   }
}
void loop()
{
  for (int i=0; i<150;i++)
  {   muoviMotoreConMatrice(1);   }
  for (int i=0; i<50;i++)
  {    muoviMotoreConMatrice(0);   }
}
```

Sensore di temperatura e umidità DHT22

Il sensore digitale DHT22 rileva temperature che vanno da -40°C a 80°C con una precisione di 0.5°C. La lettura di temperatura ed umidità viene campionata ogni 2 secondi. Può lavorare sia ad una tensione di 3V che di 5V e il picco massimo di assorbimento è di 2.5mA (durante la lettura dei dati).

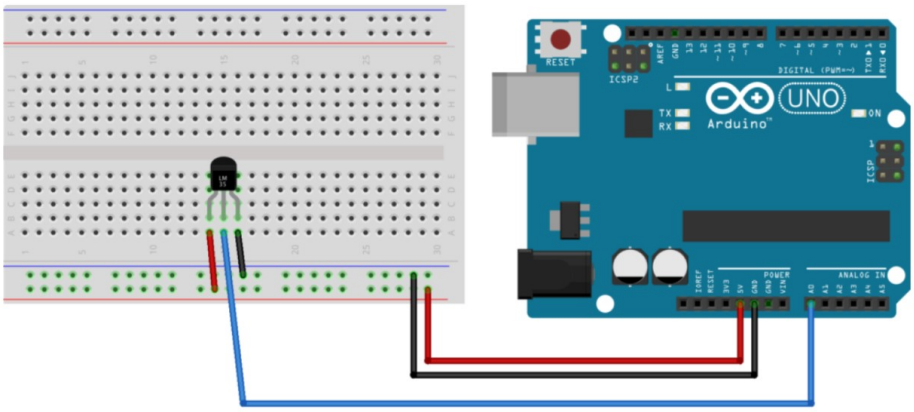





```
#include <DHT.h> //DHT sensor library versione 12
#define DHTPIN 2   //Pin a cui è connesso il sensore
#define DHTTYPE DHT22   //Tipo di sensore che stiamo utilizzando (DHT22)
DHT dht(DHTPIN, DHTTYPE);
 //Inizializza oggetto chiamato "dht", parametri: pin a cui è connesso il sensore, tipo di dht 11/22
int chk;
float hum;  //Variabile in cui verrà inserita la % di umidità
float temp; //Variabile in cui verrà inserita la temperatura
void setup()
{
  Serial.begin(9600);
  dht.begin();
}
```

```
void loop()
{
  delay(2000);
  //Ritardo di 2 secondi. (E' possibile leggere dal sensore massimo una volta ogni 2 secondi)
  //Leggi i dati e salvali nelle variabili hum e temp
  hum = dht.readHumidity();
  temp= dht.readTemperature();
  //Stampa umidità e temperatura tramite monitor seriale
  Serial.print("Umidità: ");
  Serial.print(hum);
  Serial.print(" %, Temp: ");
  Serial.print(temp);
  Serial.println(" Celsius");
}
```

Sensore LM35

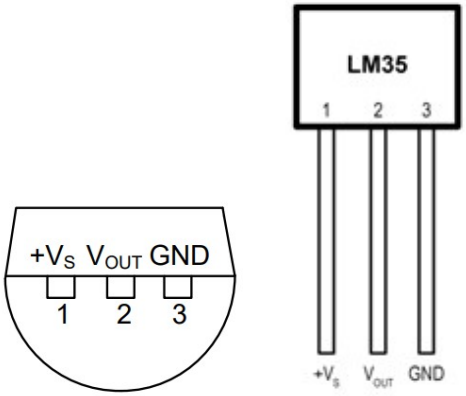


Il sensore LM35 ha tre terminali - V_s- 7_{cambio} e GND. Collegheremo il sensore come segue:

$+V_s$ a +5v sulla scheda Arduino.

V_{out} ad Analog0 o A0 su scheda Arduino.

GND con GND su Arduino.



Il sensore è

- Calibrato direttamente in Gradi Celsius (Centigradi)
- Fattore di scala lineare + 10 mV/°C
- Precisione garantita di 0,5 °C (a 25 °C)
- Nominale per l'intervallo completo da -55 °C a 150 °C
- Adatto per applicazioni remote

```
float temp;
int tempPin = 0;
void setup()
{
  Serial.begin(9600);
}
void loop()
 {
  temp = analogRead(tempPin);
  // read analog volt from sensor and save to variable temp
  temp = temp * 0.48828125;
  // convert the analog volt to its temperature equivalent
  Serial.print("TEMPERATURE = ");
  Serial.print(temp); // display temperature value
  Serial.print("*C");
  Serial.println();
  delay(1000); // update sensor reading each one second
}
```

Secondo esempio

```
#define sensorPin A0
void setup()
 {
Serial.begin(9600);
}
void loop()
 {
  int reading = analogRead(sensorPin);  // Convert the reading into voltage:
  float voltage = reading * (5000 / 1024.0);  // Convert the voltage into the temperature Celsius:
  float temperature = voltage / 10;
  Serial.print(temperature);
  Serial.print(" \xC2\xB0"); // shows degree symbol
  Serial.println("C");
  delay(1000);
}
```