

# Campus La Camilla

## *Scuola di maker*

### Developer Academy

Wordpress  
Primo sito web

### Game Academy

Scratch 2.0  
Robot  
Constructor 2  
Coderdojo

### APP Academy

Sviluppastore iOS  
Prima APP



### Maker Academy

MakerLab  
Stampa 3D  
Arduino  
Raspberry PI

### Social Media Academy

Social media base

## Arduino

### *Le basi*

di Giacomo Bellazzi

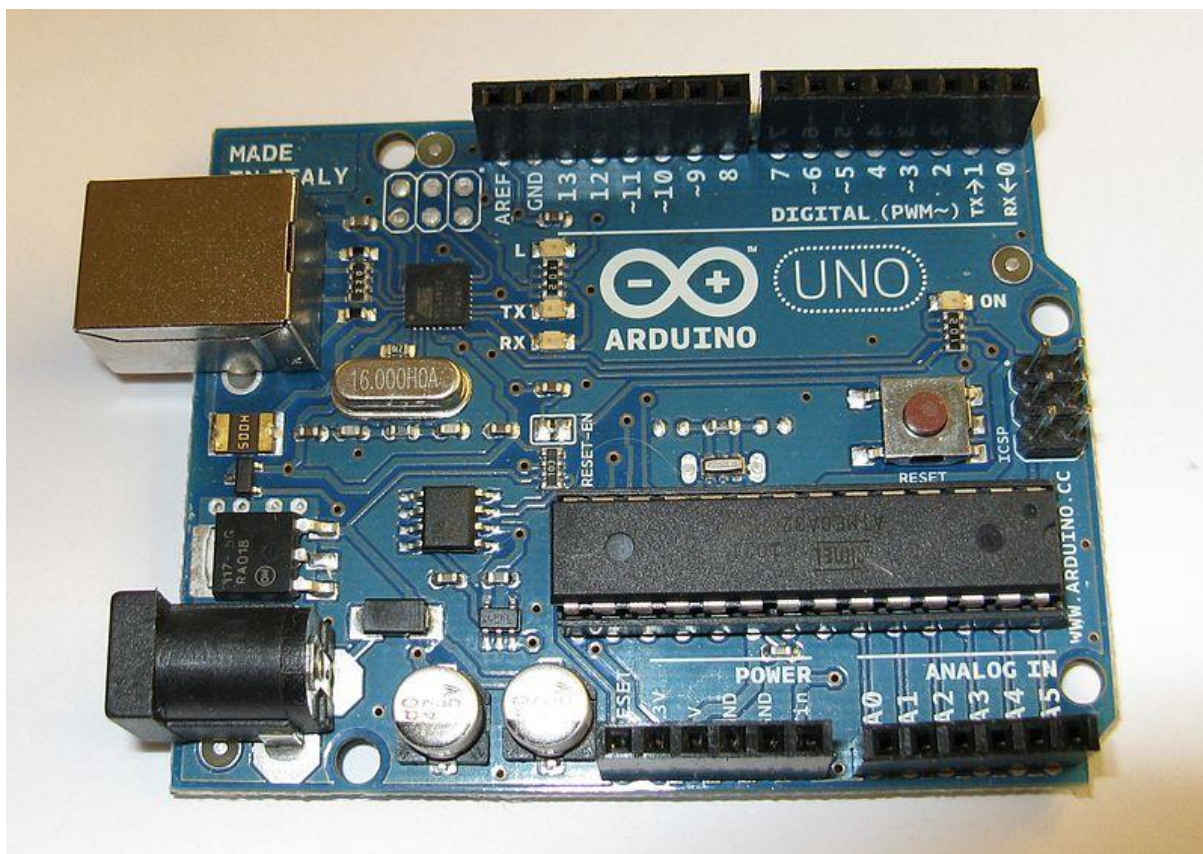
[www.campuslacamilla.it](http://www.campuslacamilla.it)

---

<b>CAPITOLO 1: DIVERTIAMOCI CON I LED</b>	<b>7</b>
<b>CAPITOLO 2: LEGGIAMO LA TEMPERATURA</b>	<b>13</b>
<b>CAPITOLO 3: SENSORI ULTRASUONI</b>	<b>20</b>
<b>CAPITOLO 4: UTILIZZIAMO IL DISPLAY</b>	<b>22</b>
<b>CAPITOLO 5: INTRODUZIONE AL RELÈ</b>	<b>32</b>
<b>CAPITOLO 6: I SENSORI DI PRESENZA PIR</b>	<b>37</b>
<b>CAPITOLO 7: SENSORE CHE RILEVA RUMORE</b>	<b>43</b>
<b>CAPITOLO 8: UTILIZZIAMO IL KEYPAD CON ARDUINO</b>	<b>49</b>
<b>CAPITOLO 9: COME ASSOCIARE UN MODULO RTC AD ARDUINO</b>	<b>66</b>
<b>CAPITOLO 10: SCOPRIAMO GLI INFRAROSSI</b>	<b>74</b>

## Introduzione

Oggi partirà una nuova serie di post che parlano del micro-controllore Arduino. Per chi non lo conoscesse Arduino è un framework open source, cioè permette di sfruttare tantissime librerie per realizzare i propri progetti. Questo dispositivo si basa su un circuito stampato che integra un microcontrollore con pin connessi alle porte I/O, un regolatore di tensione e quando necessario un'interfaccia USB che permette la comunicazione con il computer, attraverso il relativo programma. A questo hardware viene affiancato un ambiente di sviluppo integrato (IDE) multiplatforma (per Linux, Apple Macintosh e Windows). I grandi vantaggi di Arduino, sono quelli di permettere di completare progetti in poco tempo e di essere utilizzato da persone alle prime armi nella robotica, dal momento che il codice utilizzato per scrivere i programmi si basa su Wiring derivato da C e C++ chiamato , liberamente scaricabile e modificabile.



Sul sito ufficiale di Arduino <http://arduino.cc/>, oltre a poter trovare i rivenditori ufficiali, è possibile trovare tantissimi tutorial, che spiegano come utilizzare questo micro-controllore. Un'altra importante area del sito è il forum, dove è possibile trovare tantissime informazioni utili, oltre alla possibilità di discutere di questo interessantissimo dispositivo.

## Acquisti utili

Per poter utilizzare Arduino alla perfezione, sono necessari alcuni dispositivi, che vengono elencati qui sotto:



- Starter Kit

Costo 79 Euro e contiene già praticamente

l'indispensabile per iniziare ad utilizzare Arduino  
Oppure



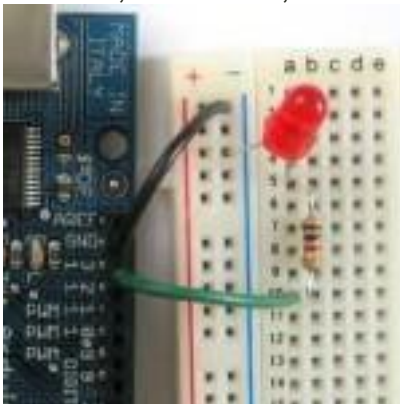
- Arduino Uno

Costo 20 Euro



- Cavi flessibili, resistenze, breadboard

Costo 15 Euro



- LED

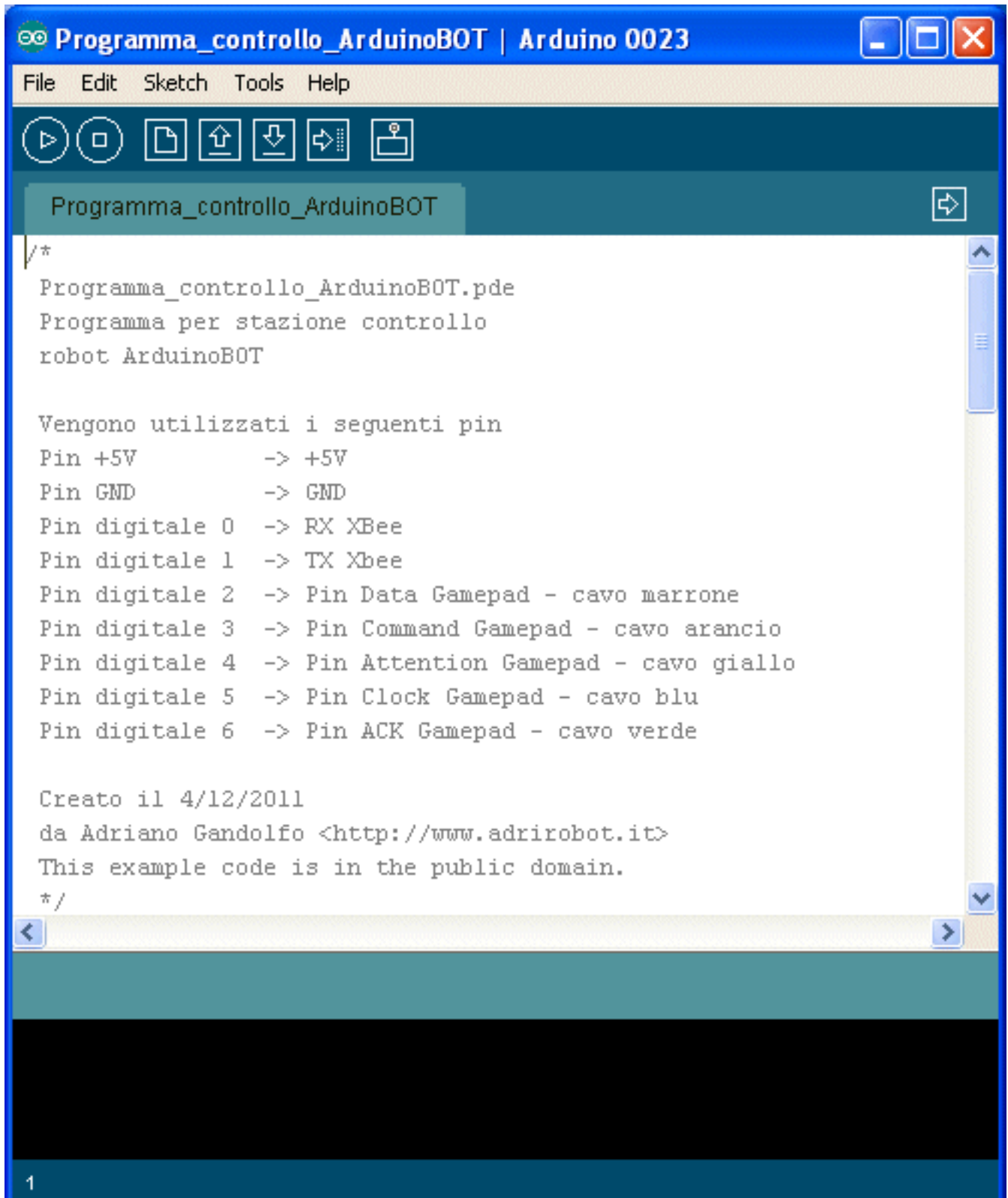
Costo 5-10 Euro a seconda della dimensione



- Tester Digitale

Costo 10 Euro, indispensabile per fare  
misure di tensione, resistenze, correnti e capacità

Dal punto di vista dei programmi ci basterà scaricare il software di programmazione dal seguente sito <https://code.google.com/p/arduino/>. Il programma è disponibile per le principali piattaforme, Windows, Mac e Linux.



```
Programma_controllo_ArduinoBOT | Arduino 0023
File Edit Sketch Tools Help
Programma_controllo_ArduinoBOT
/*
Programma_controllo_ArduinoBOT.pde
Programma per stazione controllo
robot ArduinoBOT

Vengono utilizzati i seguenti pin
Pin +5V          -> +5V
Pin GND          -> GND
Pin digitale 0  -> RX XBee
Pin digitale 1  -> TX XBee
Pin digitale 2  -> Pin Data Gamepad - cavo marrone
Pin digitale 3  -> Pin Command Gamepad - cavo arancio
Pin digitale 4  -> Pin Attention Gamepad - cavo giallo
Pin digitale 5  -> Pin Clock Gamepad - cavo blu
Pin digitale 6  -> Pin ACK Gamepad - cavo verde

Creato il 4/12/2011
da Adriano Gandolfo <http://www.adrirobot.it>
This example code is in the public domain.
*/
```

Come mostrato qui sopra, ci sono due possibili strade; la prima è quella di acquistare direttamente il Kit che contiene all'interno praticamente tutto l'indispensabile per iniziale ad utilizzare Arduino. In alternativa, è possibile prendere i vari pezzi separatamente. La scelta

dipende un po' da quello che vogliamo fare con Arduino. La scelta personale è per chi è alle prime armi conviene prendere il Kit, dal momento che contiene anche un interessante libro, con 15 tutorial pratici per iniziare a progettare.

## **Conclusione**

Arduino è un ottimo progetto tutto italiano, che permette di realizzare piccoli progetti anche per chi non dispone di conoscenze troppo particolari, dal momento che basta conoscere il linguaggio di programmazione C

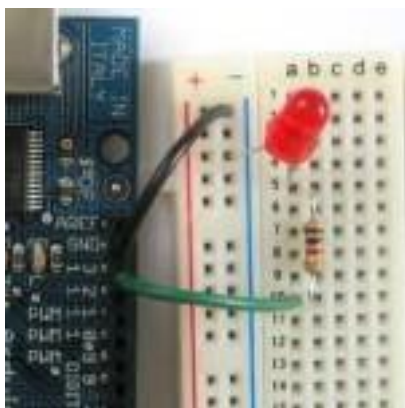
Nelle prossime puntate vedremo alcuni esempi pratici sull'utilizzo di Arduino, come accedere vari LED, utilizzare il display LCD per mostrare informazioni, utilizzare dei termistori per creare una piccola stazione meteo e tanto altro.



## Capitolo 1: Divertiamoci con i LED

In questa prima guida su Arduino, si parlerà di come creare piccoli programmi che permettono di accedere e spegnere vari LED. Attraverso questi semplici programmi, sarà possibile prendere dimestichezza con questo fantastico dispositivo, dalle grandi possibilità.

### Requisiti per questi programmi sui LED



- LED Costo 5-10 Euro a seconda della dimensione



- Arduino Uno

Costo 20 Euro



- Cavi flessibili, resistenze, breadboard

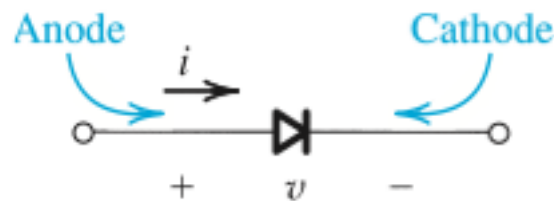
Costo 15 Euro

## Un po' di Elettronica sui LED

I light emitting diode o meglio noti con la sigla LED, non sono altre che dei diodi, che attraverso il passaggio di corrente, permettono creare una luce. Essi vengono utilizzati in



tantissimi dispositivi, come in macchina per segnalare eventuali anomalie oppure come luci diurne, nei PC. Il grande vantaggio dei LED è che scaldano davvero poco. I normali LED a due piedini, sono composti dall'anodo e dal catodo, come mostrato in figura:



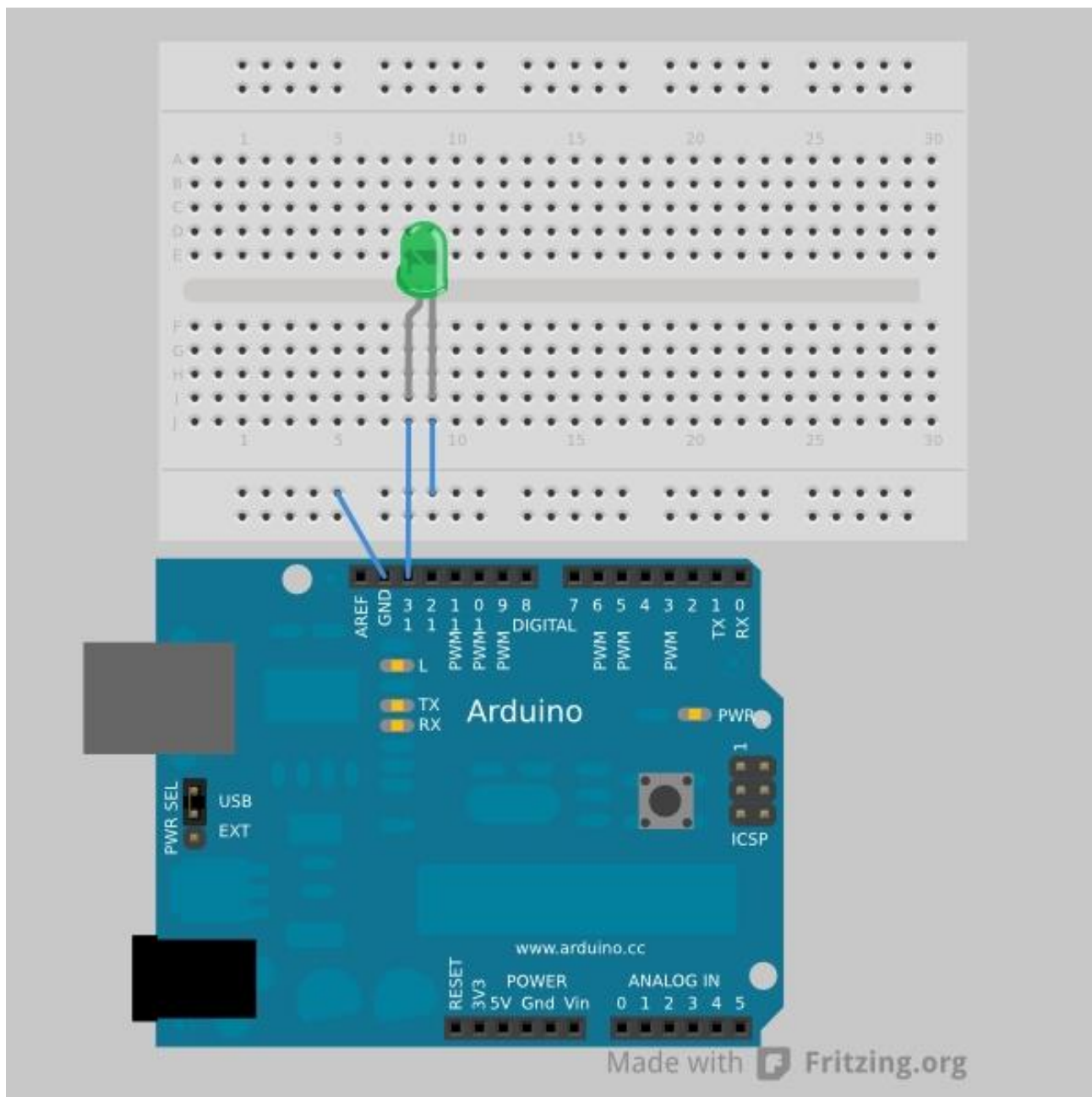
I diodi normali, come i Led, sono in grado di far passare corrente da Anodo verso Catodo quando si trovano nella zona di polarizzazione diretta, che avviene quando c'è una differenza di potenziale maggiore di circa 0.6 [V]. Tuttavia, ci sono alcuni diodi, noti con il nome di Zener, sono in grado di far passare corrente da Catodo all'Anodo, quando si trovano ad una differenza di potenziale di circa -4.3 [V] (dipende da dispositivo e tale zona si chiama "breakdown"). Tuttavia, sebbene sono abilitati per passare corrente i LED, bisogna evitare che passi troppa corrente. Per quanto di solito si mettono delle resistenze. Questo è dovuto alla più famosa legge dell'Elettronica: la Legge di Ohm.

$$V = R \cdot i$$

Il valore della corrente che dovrebbe passare nel LED affinché quest'ultimo non si bruci deve essere dell'ordine delle 10 di mA. Tuttavia, sfruttando le librerie presenti in Arduino, non sarà necessario inserire una resistenza in serie, dal momento che la tensione che verrà applicata al diodo sarà calibrata in modo da non bruciarlo.

## 1° Programma: Accendiamo un LED

In questo primo programma, vedremo come far accedere ad intermittenza un singolo LED, messo sulla breadboard collegata ad Arduino. Per costruire tale circuito, occorre collegare il pin 13 di Arduino, con l'anodo del nostro LED e mettere a massa il catodo.



Il programma da far girare sul micro-ctrllore è il seguente oppure è possibile trovare il file .ino da caricare nel compilatore:

```
1 /* Programma che fa accedere e spegnere un LED*/  
2 int led = 13; // Viene incato a che PIN è collegato l'anodo del LED  
3
```

```

4 int ritardo= 1000; //Questo valore rappresenta la durata di accensione di ci
5 void setup() {
6
7 pinMode(led, OUTPUT); // Viene inizializzata il PIN 13, per far uscire una t
8
9 // Serve qualora fosse necessario utilizzare la console
10 Serial.begin(9600);
11
12 }
13
14 // La funzione loop permette di continuare ad effettuare le varie operazioni
15 void loop() {
16
17 digitalWrite(led, HIGH); // Viene accesso LED
18 delay(ritardo);
19 digitalWrite(led, LOW); // Viene spento LED
20 delay(ritardo);
21
22 }

```



**LED\_base.ino** Come si può vedere dal main del programma, il ruolo di accedere e spegnere il LED viene fatta dalla funzione digitalWrite.

## 2° Programma: Accendiamo più LED

Ora che abbiamo capito come funziona il metodo di accedere un LED, possiamo creare un programma che permetta di accedere più LED. Per quanto riguarda il collegamento dei fili, occorre collegare per ciascun LED, l'anodo con i PIN 13,12 e 11 di Arduino e mettere a massa il catodo. Per far questo conviene mettere un file tra GND e la file dei – della breadboard e da qui far partire i tre cavi che vanno ai catodi. Tale programma è riportato qui sotto:

## 3° Programma: Semaforo con i LED

Per creare questo programma è necessario disporre di 3 LED: 1 Verde, 1 Giallo e 1 Rosso. In questo modo potremmo creare un piccolo semaforo. Nel mio caso ho utilizzato per il colore verde e rosso i LED da 10mm, mentre per quello arancione, un LED da 5mm. qualora la luminosità di quelli grossi fosse troppo elevata, è possibile inserire una resistenza in serie da 1 k \$\$ ohm \$\$\$. Il codice del programma è riportato qui:

```

1 /* Questo programma simula un semaforo, attraverso i 3 LED colorati */

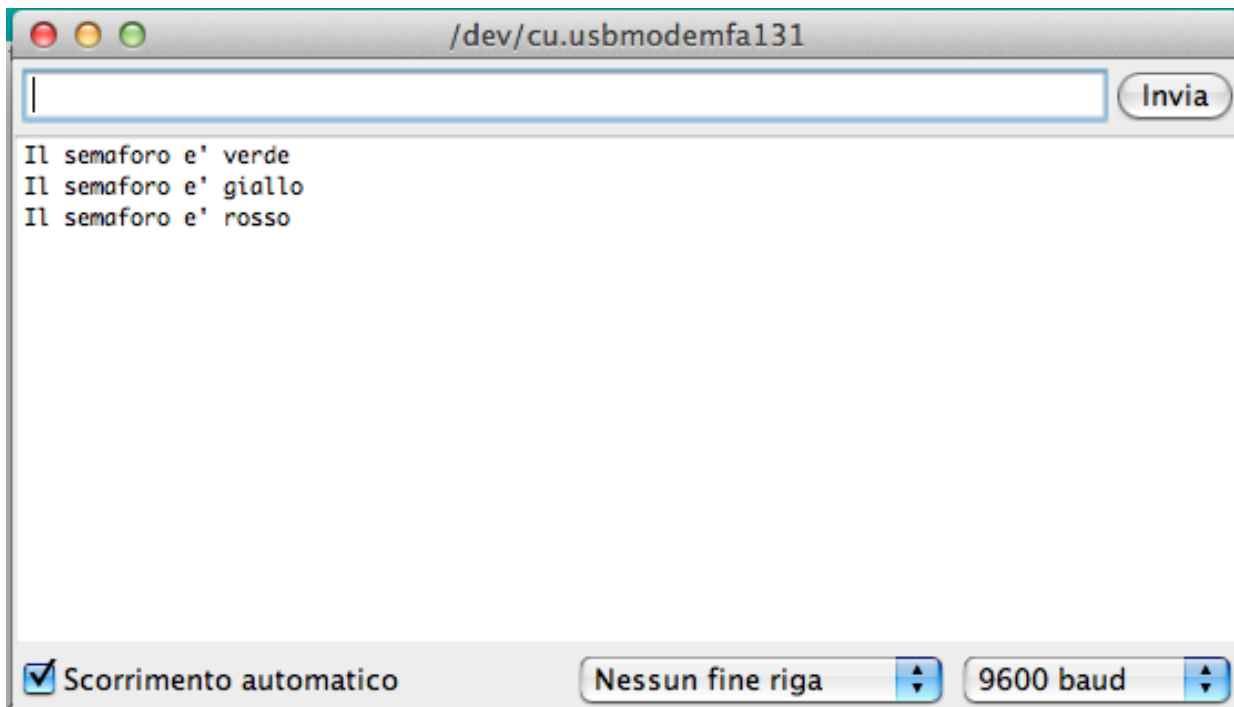
```

```
2 int led_green = 13;
3 int led_yellow = 12;
4 int led_red = 11;
5 int tempo_verde = 5000;
6 int tempo_giallo = 4000;
7 int tempo_rosso = 3000;
8
9 // the setup routine runs once when you press reset:
10 void setup() {
11   // initialize the digital pin as an output.
12   pinMode(led_green, OUTPUT);
13   pinMode(led_yellow, OUTPUT);
14   pinMode(led_red, OUTPUT);
15   Serial.begin(9600);
16
17 }
18
19 // the loop routine runs over and over again forever:
20 void loop() {
21   digitalWrite(led_green, HIGH); // turn the LED on (HIGH is the voltage level)
22   Serial.print("Il semaforo e' verde n");
23   delay(tempo_verde);           // wait for a second
24   digitalWrite(led_green, LOW);
25   digitalWrite(led_yellow, HIGH); // turn the LED on by making the voltage HIGH
26   Serial.print("Il semaforo e' giallo n");
27   delay(tempo_giallo);
28   digitalWrite(led_yellow, LOW);
29   digitalWrite(led_red, HIGH);
30   Serial.print("Il semaforo e' rosso n");
31   delay(tempo_rosso);
32   digitalWrite(led_red, LOW);
33
34 }
```

1 Per la prima volta, si è fatto uso della funzione Serial.print. Questa funzione

<a

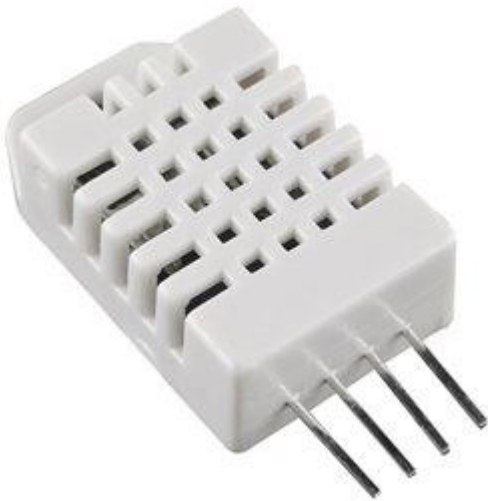
href="https://www.dropbox.com/s/pyh1gn5ulnx2t7b/Semaforo.ino"></a>



## Capitolo 2: Leggiamo la temperatura

In questa seconda puntata, vedremo come utilizzare Arduino per leggere la temperatura di casa, attraverso un sensore. Prossimamente vedremo come fare una vera e propria stazione meteo, che permetterà di leggere la temperatura di casa, oppure quella fuori anche in giro, sfruttando una connessione ad Internet. Il funzionamento dei sensori di temperatura è abbastanza semplice; al variare della temperatura, varia la resistenza presente e attraverso una semplice lettura della corrente che passa in tale elemento, è possibile capire l'attuale temperatura, con un errore di 0.5°.

### Requisiti per leggere la temperatura



- DHT22
- Resistenza da 10 K

Costo 10 Euro

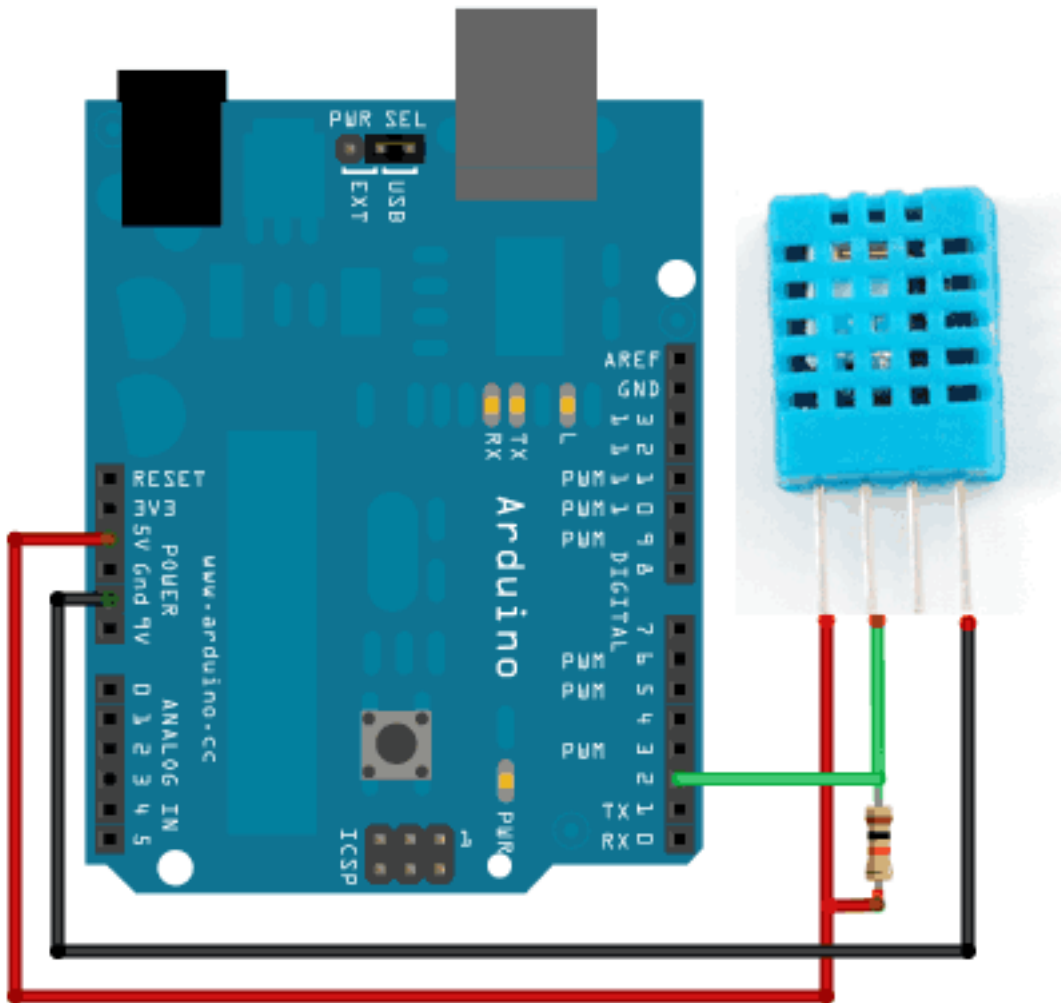


Costo meno di 1 Euro

### **1° Programma: Leggiamo la temperatura e l'umidità di casa**

Per utilizzare questo programma, è necessario scaricare alcune librerie, che contengono le funzioni per tradurre il valore di tensione in temperatura, dal seguente sito. Tale cartella dovrà essere inserita in Arduino/Libraries, in modo che il compilatore, possa leggere correttamente. Per quanto riguarda i collegamenti, basta seguire l'immagine riportati qui sotto.





Ora possiamo finalmente caricare il programma sul nostro Arduino !

```

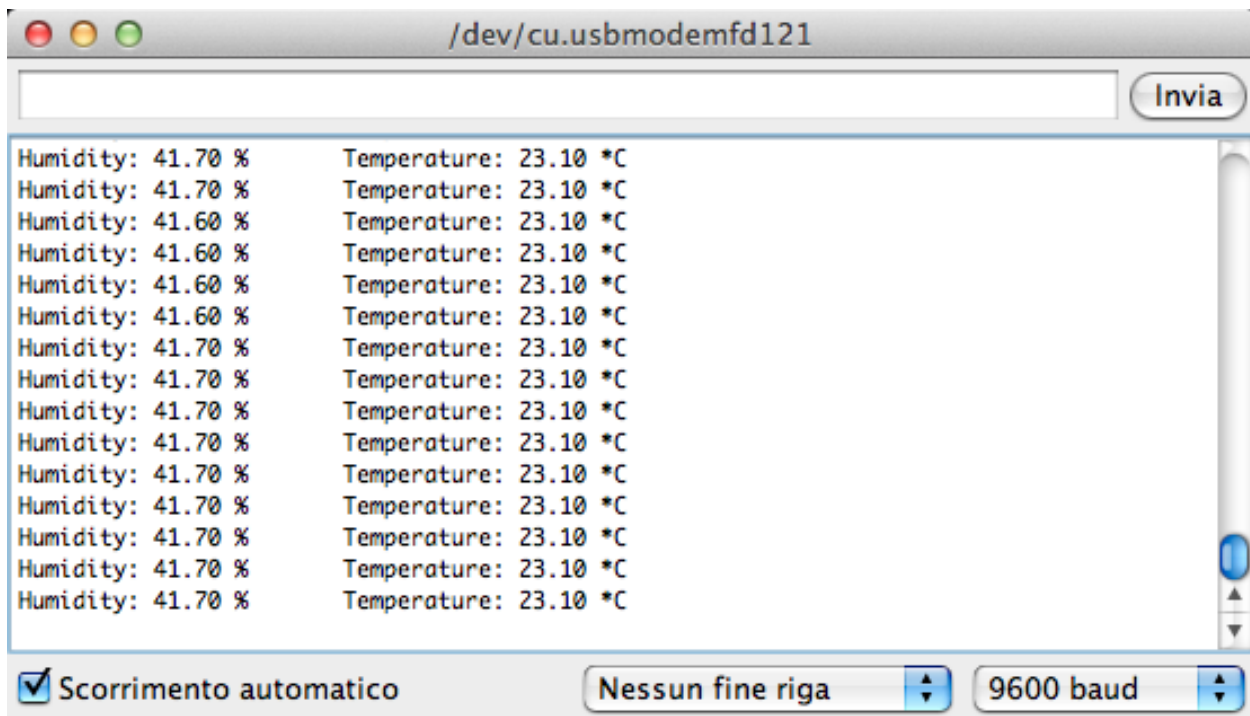
1 // Programma che permette di leggere la temperatura, attraverso un sensore
2
3 #include "DHT.h"
4
5 #define DHTPIN 2 // Il pin a cui è collegato il sensore
6
7 // Togli il commento al sensore che vuoi usare
8 // #define DHTTYPE DHT11 // DHT 11
9 #define DHTTYPE DHT22 // DHT 22 (AM2302)
10 // #define DHTTYPE DHT21 // DHT 21 (AM2301)
11
12 // Connettere il pin 1 (a sinistra) a +5V
13 // Connettere il pin 2 del sensore alla porta 2
14 // Connettere il pin 4 (a destra) del sensore a GROUND
15 // Connettere una resistenza da 10K tra il pin 2 (data) e il pin 1 (power) d
    
```

```
16
17 DHT dht(DHTPIN, DHTTYPE);
18
19 void setup() {
20   Serial.begin(9600);
21   Serial.println("DHTxx test!");
22
23   dht.begin();
24 }
25
26 void loop() {
27   // Legge la temperatura e l'umidità ogni 250 millisecondi!
28   // Vengono inizializzate le variabili in cui vengono scritti i valori letti
29   float h = dht.readHumidity();
30   float t = dht.readTemperature();
31
32   // Controlla se la lettura è andata a buon fine
33   if (isnan(t) || isnan(h)) {
34     Serial.println("Failed to read from DHT");
35   } else {
36     Serial.print("Humidity: ");
37     Serial.print(h); // Stampa nel seriale la percentuale dell'umidità
38     Serial.print(" %t");
39     Serial.print("Temperature: ");
40     Serial.print(t); // Stampa nel seriale il valore della temperatura
41     Serial.println(" *C");
42   }
43 }
```



DHT22\_Lettura.ino

Il valore della temperatura verrà visualizzato nella seriale di Arduino.



## 2° Programma: Utilizziamo i LED per capire la temperatura

Ora che abbiamo capito come poter leggere la temperatura con Arduino, possiamo utilizzare i LED, che ci permettono di capire l'attuale temperatura, senza dover accedere il computer. Per esempio possiamo scegliere di far accedere il LED rosso, qualora la temperatura fosse superiore a 30°, e far accedere il LED blu, qualora la temperatura scendesse sotto i 15°. Per quanto riguarda il collegamento, basterà connettere i LED alle porte 7 e 6 di Arduino (l'anodo) mentre il catodo a massa. Nulla varia per quanto riguarda il sensore di temperatura. Si può mettere una resistenza di poche centinaia di Ohm ai LED, per ridurre la corrente.

```

1 // Programma che permette di leggere la temperatura, attraverso un sensore
2
3 #include "DHT.h"
4
5 #define DHTPIN 2 // Il pin a cui è collegato il sensore
6
7 // Togli il commento al sensore che vuoi usare
8 // #define DHTTYPE DHT11 // DHT 11
9 #define DHTTYPE DHT22 // DHT 22 (AM2302)
10 // #define DHTTYPE DHT21 // DHT 21 (AM2301)
11
12 // Connettere il pin 1 (a sinistra) a +5V
13 // Connettere il pin 2 del sensore alla porta 2
14 // Connettere il pin 4 (a destra) del sensore a GROUND
15 // Connettere una resistenza da 10K tra il pin 2 (data) e il pin 1 (power) d
16
17 DHT dht(DHTPIN, DHTTYPE);
18

```

```
19 // Vengono indicate le temperature per cui i LED si devono accendere
20 int temp_max = 30;
21 int temp_min = 15;
22
23 // Vengono inizializzati i pin a cui sono connessi i LED
24 int led_red = 7;
25 int led_blue = 6;
26 int ritardo = 1000;
27
28 void setup() {
29   Serial.begin(9600);
30   Serial.println("DHTxx test!");
31   pinMode(led_red, OUTPUT);
32   pinMode(led_blue, OUTPUT);
33   dht.begin();
34 }
35
36 void loop() {
37   // Legge la temperatura e l'umidità ogni 250 millisecondi!
38   // Vengono inizializzate le variabili in cui vengono scritti i valori letti
39   float h = dht.readHumidity();
40   float t = dht.readTemperature();
41
42   // Controlla se la lettura è andata a buon fine
43   if (isnan(t) || isnan(h)) {
44     Serial.println("Failed to read from DHT");
45   } else {
46     Serial.print("Humidity: ");
47     Serial.print(h); // Stampa nel seriale la percentuale dell'umidità
48     Serial.print(" %t");
49     Serial.print("Temperature: ");
50     Serial.print(t); // Stampa nel seriale il valore della temperatura
51     Serial.println(" *C");
52   }
53
54   // I prossimi due IF fanno accedere i LED, a seconda della temperatura
55   if (t >= temp_max) {
56     digitalWrite(led_red, HIGH); // Viene acceso il LED rosso
57     delay(ritardo);
58     digitalWrite(led_red, LOW); // Viene spento il LED rosso
59   }
60
61   if (t <= temp_min) {
62     digitalWrite(led_blue, HIGH); // Viene acceso il LED blu
63     delay(ritardo);
64     digitalWrite(led_blue, LOW); // Viene spento il LED blu
65   }
66
67 }
```



**DHT22\_LED.ino**

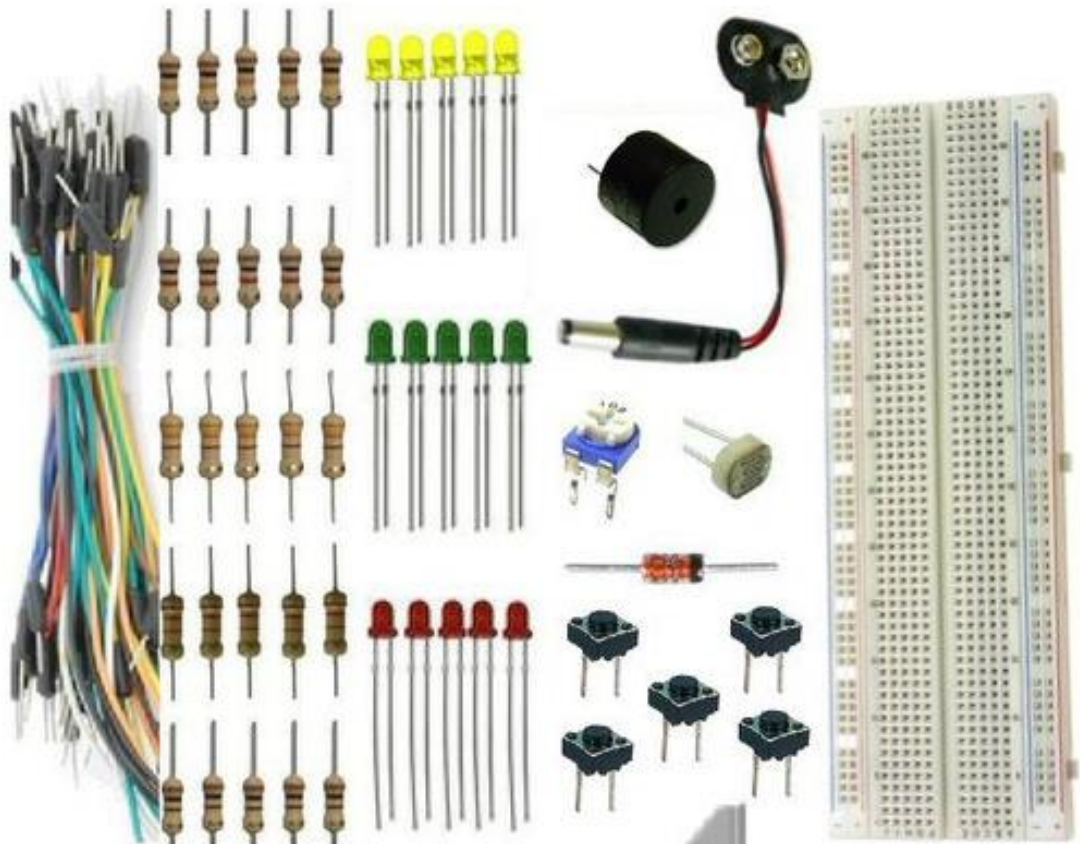
Ora non ci resterà altro che verificare il corretto funzionamento. Per far variare i valori di temperatura per cui i LED si accendono, basterà modificare i valori di temp\_max e temp\_min.

## Capitolo 3: Sensori ultrasuoni

In questa terza lezione su Arduino, vedremo come utilizzare alcuni sensori, che ci permetteranno di fare delle misure, sugli oggetti che si avvicinano al sensore. Il meccanismo della lettura della distanza è abbastanza semplice: il sensore manda una onda sonora, di cui si conosce la velocità e attraverso il calcolo del tempo impiegato dall'onda a ritornare indietro, è possibile calcolare la distanza dell'oggetto.

### Requisiti per la lettura della distanza

- LED



Costo pochi euro

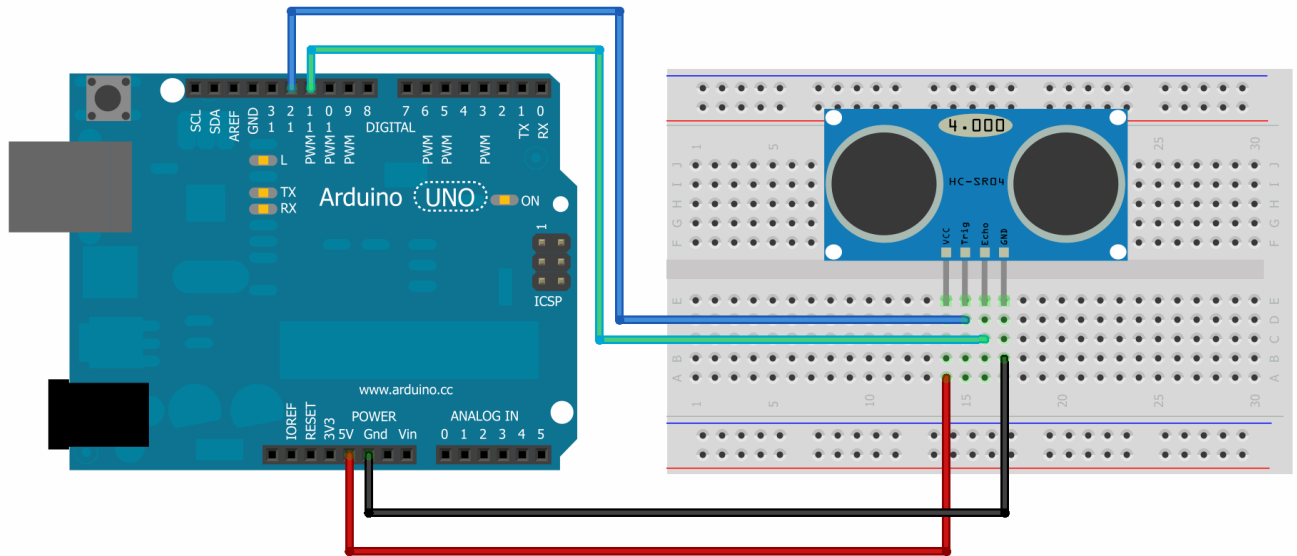


- Ultrasonic Sensore SR04

Costo 5 euro ed è acquistabile su Ebay

### 1° Programma: Leggiamo la distanza dell'oggetto dal sensore e accendiamo un LED

In questo programma, sfrutteremo il sensore ad ultrasuoni per calcolare la distanza di un oggetto in movimento e faremo accedere un LED, qualora questo oggetto si trovi ad una distanza inferiore ad un determinato valore. Per quanto riguarda i collegamenti, basta seguire quelli mostrati nella seguente figura. Inoltre sarà necessario inserire un LED al PIN 13, con l'anodo in quest'ultimo pin e il catodo collegato a massa.



```

1  /* Programma che fa accedere un LED rosso, quando c'è un oggetto ad una certa
2  HC-SR04 sensore di distanza
3  VCC al arduino 5v
4  GND al arduino GND
5  Echo al Arduino pin 7
6  Trig al Arduino pin 8
7  */
8
9  #define echoPin 7 // Echo Pin
10 #define trigPin 8 // Trigger Pin
11 #define LEDPin 13 // LED
12
13 int maximumRange = 200; // Maximum range
14 int minimumRange = 10; // Minimum range
15 long duration, distance;
16
17 void setup() {
18   Serial.begin (9600);
19   pinMode(trigPin, OUTPUT);
20   pinMode(echoPin, INPUT);
21   pinMode(LEDPin, OUTPUT); // Inizializza il PIN del LED
22 }
23
24 void loop() {
25  /* La seguente funzione permette di ricavare la distanza a cui si trova un og

```



```
26 digitalWrite(trigPin, LOW);
27 delayMicroseconds(2);
28
29 digitalWrite(trigPin, HIGH);
30 delayMicroseconds(10);
31
32 digitalWrite(trigPin, LOW);
33 duration = pulseIn(echoPin, HIGH);
34
35 /* Calcola la distanza in centimetri dall'oggetto*/
36 distance = duration/58.2;
37
38 if (distance <= minimumRange){
39 /* Si accende il LED rosso, se l'oggetto si trova a meno di una certa dista
40 Serial.println("-1");
41 digitalWrite(LEDpin, HIGH);
42 }
43 else {
44 /* Viene mostrata la distanza nel seriale */
45 Serial.println(distance);
46 digitalWrite(LEDpin, LOW);
47 }
48
49 delay(50);
50 }
```



Distance\_LED.ino

## Capitolo 4: Utilizziamo il display

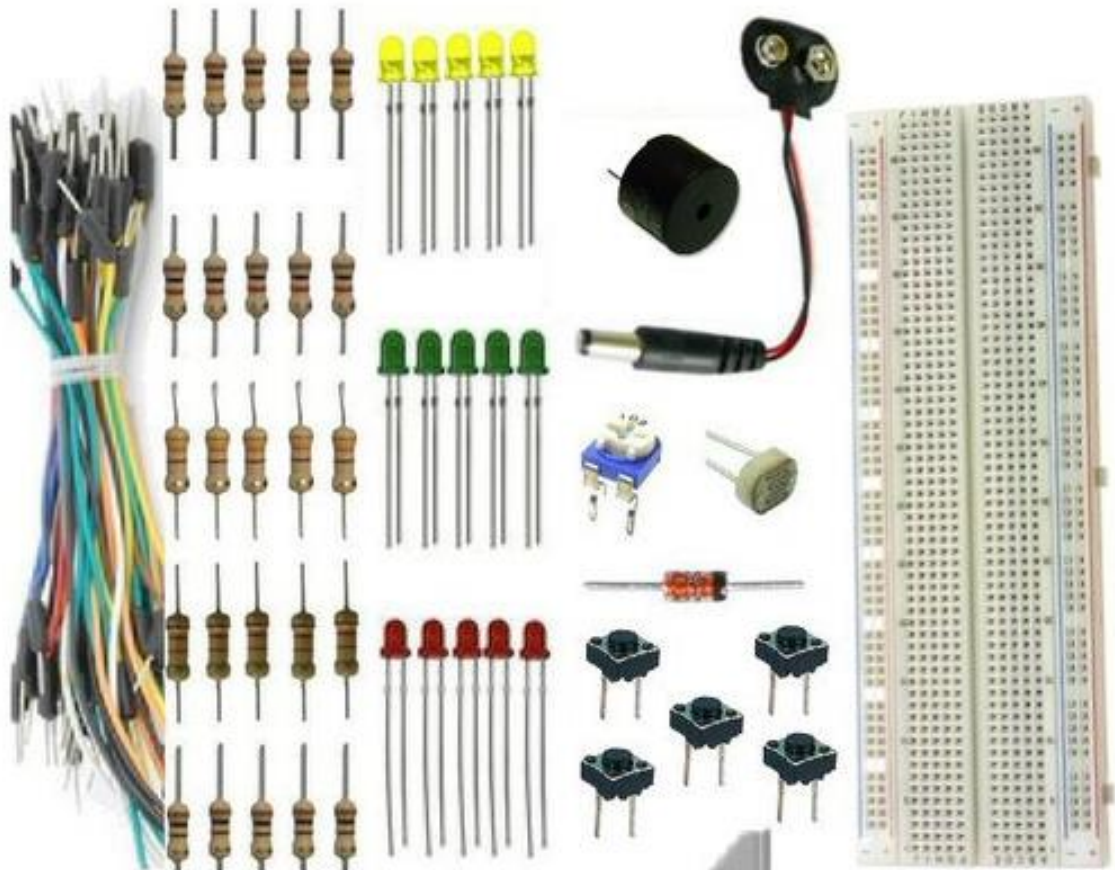
In questa quarta lezione su Arduino, scopriremo un interessante strumento, utile per interfacciare il mondo digitale con quello umano: il display a cristalli liquidi. Con questo strumento, sarà possibile leggere unità fisiche, come l'attuale temperatura in casa, l'ora, lo stato di accensione di un LED e tanto altro. Vedremo questi aspetti dal punto di vista

pratico, nei vari esempi proposti.

### Requisiti per i programmi



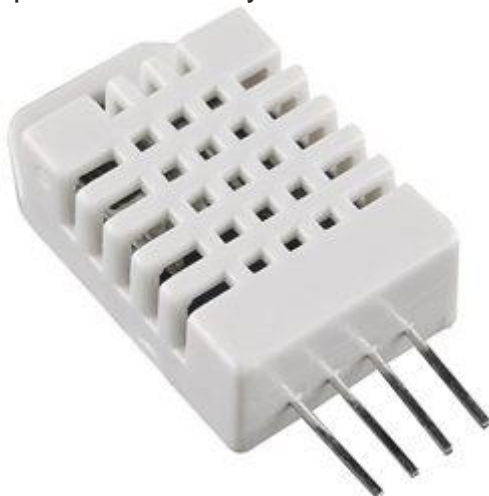
- Arduino UNO
- Cavi flessibil, LED, Resistenze



- Display



Acquistabile su Ebay al costo di 10 Euro

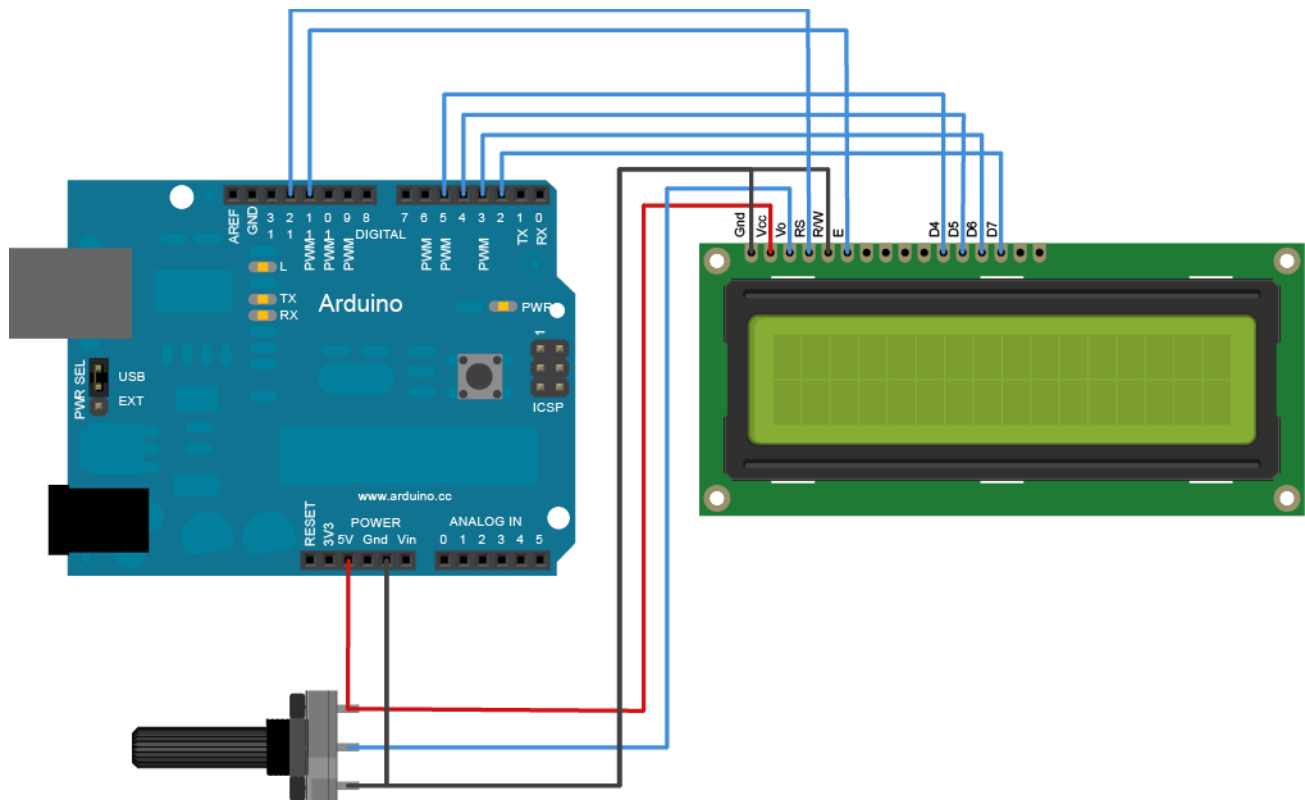


- DHT22



- Sensore ad ultrasuoni

Il collegamento dei fili, per quanto riguarda il Display, è leggermente più complesso rispetto agli esempi precedenti. Tuttavia basta collegare i vari cavi, come riportati nella seguente figura.



L'elemento in basso a sinistra non è altro che un potenziometro. Esso non fa altro che far variare la resistenza al suo interno, a seconda della posizione del manopola. Questo permetterà di regolare l'intensità dello schermo.

### 1° Programma Hello World sul Display

Quando si impara a programma, il primo esempio mostrato è il famoso Hello World, in cui si mostra a video la scritta "Ciao Mondo". Allo stesso modo, faremo mostrato sul piccolo schermo, la parola "Ciao Mondo", sfruttando Arduino.

```

1  /*
2  Questo programma mostra come usare il Display LCD per mostrare la parola "
3
4  Il Circuito:
5  * LCD RS pin to digital pin 12
6  * LCD Enable pin to digital pin 11
7  * LCD D4 pin to digital pin 5

```

```

 8  * LCD D5 pin to digital pin 4
 9  * LCD D6 pin to digital pin 3
10  * LCD D7 pin to digital pin 2
11  * LCD R/W pin to ground
12  * 10K resistor:
13  * ends to +5V and ground
14  * wiper to LCD V0 pin (pin 3)
15
16  Questo esempio è tratto dal sito di Arduino
17
18  http://www.arduino.cc/en/Tutorial/LiquidCrystal
19
20  */
21
22  // include the library code:
23  #include <LiquidCrystal.h>
24
25  // Viene inizializzata la libreria del Display con i relativi PIN a cui è con
26  LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
27
28  void setup() {
29    // set up the LCD's number of columns and rows:
30    lcd.begin(16, 2);
31    // Stampa a video la parola "hello, world!"
32    lcd.print("hello, world!");
33  }
34
35  void loop() {
36    // Imposta il cursore alla colonna 0, riga 1
37    // (nota: line 1 è la seconda riga, dal momento che il conteggio inizia da
38    lcd.setCursor(0, 1);
39    // Stampa a video i secondi passati dal momento del reset
40    lcd.print(millis()/1000);
41  }

```



Display\_Hello\_Worl  
d.ino

Ecco il circuito del primo programma

## 2° Programma Mostriamo la temperatura di casa sul Display

In questo secondo esempio, vedremo come mostrare la temperatura di casa sul nostro Display, sfruttando il sensore DHT22, che è già stato trattato nella seconda Lezione su Arduino

```
1  /*
2  Questo programma mostra come usare il Display LCD per mostrare la temperatura
3
4  Il Circuito:
5  * LCD RS pin to digital pin 12
6  * LCD Enable pin to digital pin 11
7  * LCD D4 pin to digital pin 5
8  * LCD D5 pin to digital pin 4
9  * LCD D6 pin to digital pin 3
10 * LCD D7 pin to digital pin 2
11 * LCD R/W pin to ground
12 * 10K resistor:
13 * ends to +5V and ground
14 * wiper to LCD V0 pin (pin 3)
15
16 Questo esempio è tratto dal sito di Arduino
17
18 http://www.arduino.cc/en/Tutorial/LiquidCrystal
19
20 */
21
22 // include the library code:
23 #include <LiquidCrystal.h>
24 #include "DHT.h"
25
26 #define DHTPIN 8      // Il pin a cui è collegato il sensore
27
28 // Togli il commento al sensore che vuoi usare
29 // #define DHTTYPE DHT11 // DHT 11
30 #define DHTTYPE DHT22 // DHT 22 (AM2302)
31 // #define DHTTYPE DHT21 // DHT 21 (AM2301)
32
33 // Connettere il pin 1 (a sinistra) a +5V
34 // Connettere il pin 2 del sensore alla porta 8
35 // Connettere il pin 4 (a destra) del sensore a GROUND
36 // Connettere una resistenza da 10K tra il pin 2 (data) e il pin 1 (power) del sensore
37
38 DHT dht(DHTPIN, DHTTYPE);
39
40 // Viene inizializzata la libreria del Display con i relativi PIN a cui è connesso
41 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
42
43 void setup() {
```

```

44  lcd.begin(16, 2);
45  dht.begin();
46  }
47
48  void loop() {
49    // Legge la temperatura e l'umidità ogni 250 millisecondi!
50    // Vengono inizializzate le variabili in cui vengono scritti i valori letti
51    float h = dht.readHumidity();
52    float t = dht.readTemperature();
53    //Imposto la colonna 0, riga 0
54    lcd.setCursor(0,0);
55    // print the humidity
56    lcd.print("Humidita' ");
57    lcd.print(h);
58    lcd.print(" % ");
59    //Imposto la colonna 0, riga 1
60    lcd.setCursor(0,1);
61    lcd.print("Temp. ");
62    lcd.print(t);
63    lcd.print(" *C");
64
65    delay(1000);
66  }

```



Display\_temperatur  
a.ino

### 3° Programma Mostriamo la distanza di un oggetto sul display e accendiamo un LED (una sorta di sensori di parcheggio)

In questo terzo esempio, mostriamo come utilizzare Arduino, con il display, con il sensore ad ultrasuoni, per visualizzare la distanza di un oggetto e utilizzeremo un LED rosso, per indicare che l'oggetto è troppo vicino.

```

1  /*
2  Questo programma mostra come usare il Display LCD per mostrare la parola '
3

```



```
4   Il Circuito:
5   * LCD RS pin to digital pin 12
6   * LCD Enable pin to digital pin 11
7   * LCD D4 pin to digital pin 5
8   * LCD D5 pin to digital pin 4
9   * LCD D6 pin to digital pin 3
10  * LCD D7 pin to digital pin 2
11  * LCD R/W pin to ground
12  * 10K resistor:
13  * ends to +5V and ground
14  * wiper to LCD V0 pin (pin 3)
15
16  Questo esempio è tratto dal sito di Arduino
17
18  http://www.arduino.cc/en/Tutorial/LiquidCrystal
19
20  */
21
22  // include the library code:
23  #include <LiquidCrystal.h>
24
25  #define echoPin 7 // Echo Pin
26  #define trigPin 8 // Trigger Pin
27  #define LEDPin 13 // LED
28
29  int green = 30;
30  int orange = 15;
31  int red = 14;
32  long duration, distance;
33
34  // Inizializziamo i 3 LED
35
36  int ledgreen = 10;
37  int ledorange = 9;
38  int ledred = 13;
39
40  // Viene inizializzata la libreria del Display con i relativi PIN a cui è con
41  LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
42
43  void setup() {
44    // set up the LCD's number of columns and rows:
45    lcd.begin(16, 2);
46
47    pinMode(trigPin, OUTPUT);
48    pinMode(echoPin, INPUT);
49    // Iniziliazzo i 3 LED
50    pinMode(ledgreen, OUTPUT);
51    pinMode(ledorange, OUTPUT);
52    pinMode(ledred, OUTPUT);
```

```
53 }
54
55 void loop() {
56     /* La seguente funzione permette di ricavare la distanza a cui si trova un
57     digitalWrite(trigPin, LOW);
58     delayMicroseconds(2);
59
60     digitalWrite(trigPin, HIGH);
61     delayMicroseconds(10);
62
63     digitalWrite(trigPin, LOW);
64     duration = pulseIn(echoPin, HIGH);
65
66     /* Calcola la distanza in centimetri dall'oggetto*/
67     distance = duration/58.2;
68     delay(500);
69
70     if (distance>=green) {
71         green_on();
72         print_lcd();
73     }
74
75     if (green<=distance<green) {
76         orange_on();
77         print_lcd();
78
79     }
80     if (distance<=red) {
81         red_on();
82         print_lcd();
83     }
84
85 }
86
87 // Funziona che stampa sul display la distanza
88 void print_lcd () {
89
90     // Imposta il cursore alla colonna 0, riga 1
91     // (nota: line 1 è la seconda riga, dal momento che il conteggio inizia da 0)
92     lcd.setCursor(0, 0);
93     // Stampa a video i secondi passati dal momento del reset
94     lcd.print("Distanza ");
95     lcd.print(distance);
96     lcd.print(" cm ");
97 }
98
99 // Accende il LED verde
100 void green_on() {
101     digitalWrite(ledgreen, HIGH);
102     digitalWrite(ledorange, LOW);
```

```
103     digitalWrite(ledred, LOW);
104 }
105
106 // Accende il LED arancione
107 void orange_on () {
108     digitalWrite(ledgreen, LOW);
109     digitalWrite(ledorange, HIGH);
110     digitalWrite(ledred, LOW);
111 }
112
113 // Accende il LED rosso
114 void red_on () {
115     digitalWrite(ledgreen, LOW);
116     digitalWrite(ledorange, LOW);
117     digitalWrite(ledred, HIGH);
118 }
```



Display\_distance.in

o

Ecco il circuito del terzo programma.

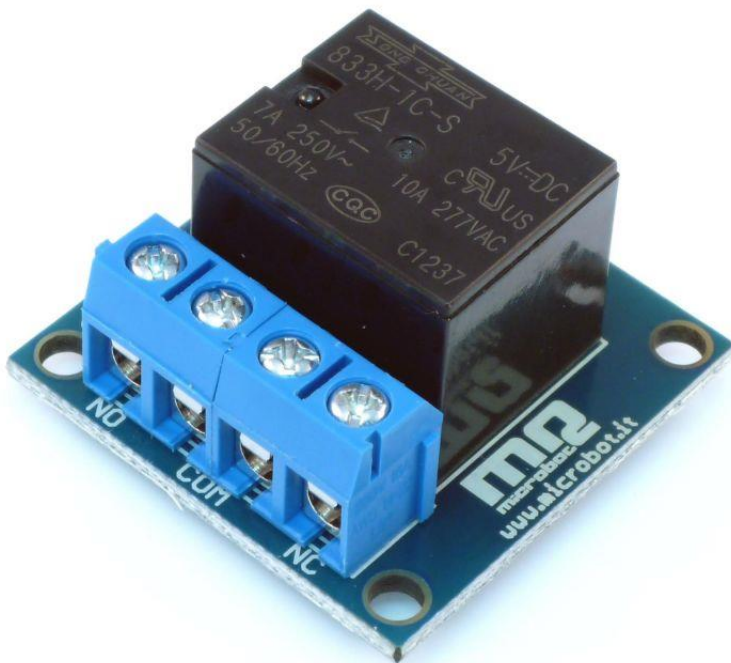
## Capitolo 5: Introduzione al relè

In questa quinta puntata su Arduino, vedremo l'utilizzo di un nuovo dispositivo: il Relè.

Ecco una breve descrizione tratta da Wikipedia

Il **relè** è un dispositivo elettrico comandato dalle variazioni di corrente per influenzare le condizioni di un altro circuito. In sostanza il relè è un interruttore che non viene azionato a mano ma da un elettromagnete. Un relè può azionare un circuito sia se è a riposo, non attraversato da corrente (in quel caso tale circuito va collegato ai terminali centrale e di sinistra del relè nella foto, quindi normalmente chiusi), sia se è attivo, attraversato da corrente (in questo caso il circuito va collegato ai terminali centrale e di destra, quindi normalmente aperti).

In sostanza il relè non è altro che un interruttore che può essere azionato attraverso un segnale digitale, (alto, oppure basso). Ci sono due tipi di configurazioni **NO** e **NC**. Stanno ad indicare "normally open" and "normally close". Questo vuole dire che per far passare corrente nella prima configurazione, occorre inviare un segnale alto, mentre per la seconda configurazione è l'opposto.



**Avviso importantissimo !**



Nei prossimi programmi presenti in questo articolo, si farà uso del relay, che permetterà di comandare apparecchi elettrici che funzionano alla tensione 220 V con una corrente massima di 10 A. Qualora non siate esperti elettricisti, **NON** provate a manovrare questi apparecchi. Anche il più piccolo errore nel collegare i fili, potrebbe essere fatale per voi ! Infatti, se per caso doveste toccare la parte di rame del cavo collegato all'alimentazione, la corrente passerà nel vostro corpo prima che tocchi terra ! Fate attenzione quindi e non mi assumo nessuna responsabilità in caso di incidenti.

### 1° Programma: Accedere una lampada attraverso Arduino ad intervalli regolari

In questo programma vedremo come accedere una lampada di casa, ad intervalli regolari con Arduino. Prima di passare alla parte software, è necessario configurare correttamente il Relè.

Il primo passo è quello di acquistare un cavo a 3 poli, 1 presa femmina e 1 presa maschio. Il passo successivo è

```

1  /*Questo programma permette l'accessione di una lampada ad intervalli regolari
2  int relay1 = 4;           // PIN a cui è connesso il relay ad Arduino
3  int t = 5000; // Imposto intervallo di tempo con cui si accende e spegne la
4
5  void setup()
6  {
7    pinMode(relay1, OUTPUT); // Imposta l'uscita del PIN
8    Serial.begin(9600);
9  }
10
11 void loop()
12 {
13   digitalWrite(relay1, HIGH); // Chiudo l'interruttore del relè
14   Serial.println("HIGH n");
15   delay(t); // Tempo di attesa
16   digitalWrite(relay1, LOW); // Apro l'interruttore del relè
17   Serial.println("LOW n");
18   delay(t); // Tempo di attesa
19 }

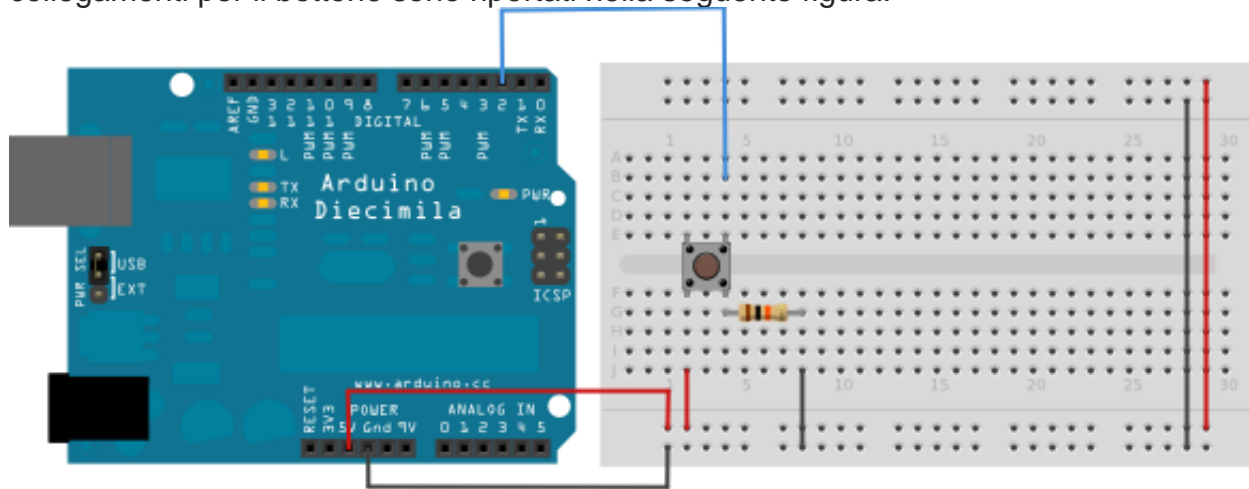
```



Relay\_easy.ino

## 2° Programma: Accendiamo la lampada attraverso un “push button”

In questo secondo programma, vedremo come accedere una lampada con Arduino per un certo periodo, semplicemente premendo un bottone posto nel nostro circuito. I collegamenti per il bottone sono riportati nella seguente figura.



La resistenza presente nel circuito è da 10 K  $\Omega$ .

```

1  /* Questo programma permette di accendere una lampada connessa al relè attra
2  This example code is in the public domain.
3
4  http://www.arduino.cc/en/Tutorial/Button
5
6  */
7
8  const int buttonPin = 2;    // Il PIN a cui è connesso il bottone
9  const int relay = 13;      // Il PIN a cui è connesso il relè
10 int t = 5000;
11
12 // Inizializzo la variabile che indica lo stato del bottone
13 int buttonState = 0;
14
15 void setup() {
16   // Inizializza l'uscita del relè
17   pinMode(relay, OUTPUT);
18   // Inizializza l'ingresso del bottone
19   pinMode(buttonPin, INPUT);
20 }
21
22 void loop(){
23   // Leggo lo stato del bottone
24   buttonState = digitalRead(buttonPin);
25

```

```
26 // Controllo se viene premuto il bottone
27 // Se lo stato è HIGH, allora viene premuto
28 if (buttonState == HIGH) {
29
30     // Chiudo l'interruttore per un periodo t
31     digitalWrite(relay, HIGH);
32     delay(t);
33 }
34 else {
35     // Apro l'interruttore
36     digitalWrite(relay, LOW);
37 }
38 }
```



Relay\_push\_button.  
ino

In questo articolo, si è fatta una breve introduzione sul relè, elemento importante nei circuiti elettrici. Nelle prossime puntate, vedremo progetti molti più complessi e divertenti, che permetteranno, per esempio, di accedere una lampada da remoto attraverso l'iPhone, di accedere il condizionatore di casa e tanto altro. Con questo elemento, sarà possibile fare cose molto più fantasiose, come aprire il garage di casa attraverso una interfaccia web e tantissimo altre. Tutte le idee sull'utilizzo del relè con Arduino, sono ben accette attraverso i commenti !



## Capitolo 6: I sensori di presenza PIR

In questa sesta lezione su Arduino, parleremo di un interessante trasduttore, che può essere utile per tantissimi progetti: il sensore di presenza. Ecco una breve descrizione tratta da Wikipedia:

I **sensori di prossimità** (chiamati anche **proximity**) sono dei sensori in grado di rilevare la presenza di oggetti nelle immediate vicinanze del “lato sensibile” del sensore stesso, senza che vi sia un effettivo contatto fisico. La distanza entro cui questi sensori rilevano oggetti è definita portata nominale (o campo sensibile). Alcuni modelli dispongono di un sistema di regolazione per poter calibrare la distanza di rilevazione. L'assenza di meccanismi d'attuazione meccanica, e di un contatto fisico tra sensore e oggetto, fa sì che questi sensori presentino un'affidabilità elevata.

Dal punto di vista circuitale, un sensore di presenza, ha 3 PIN.

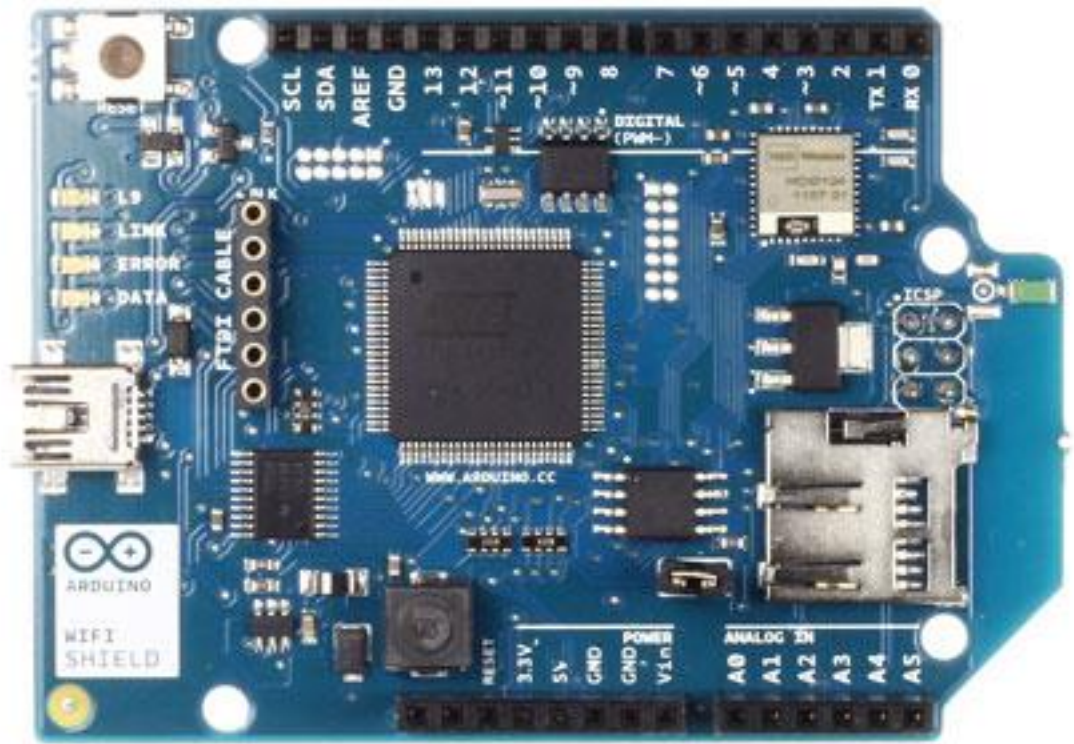
- VCC: Per l'alimentazione
- OUT: Ha un valore 1 o 0 logico, a seconda della presenza di un oggetto in movimento
- Ground

Nei prossimi programmi, vedremo come utilizzare il sensore PIR per monitorare la presenza di oggetti in movimento, creare un piccolo sistema d'allarme oppure accendere una luce.

### Requisiti per i programmi



- Arduino Uno
- Shield Ethernet



Serve per ottenere l'ora



- Sensore di presenza PIR HC-SR501 acquistabile su Ebay

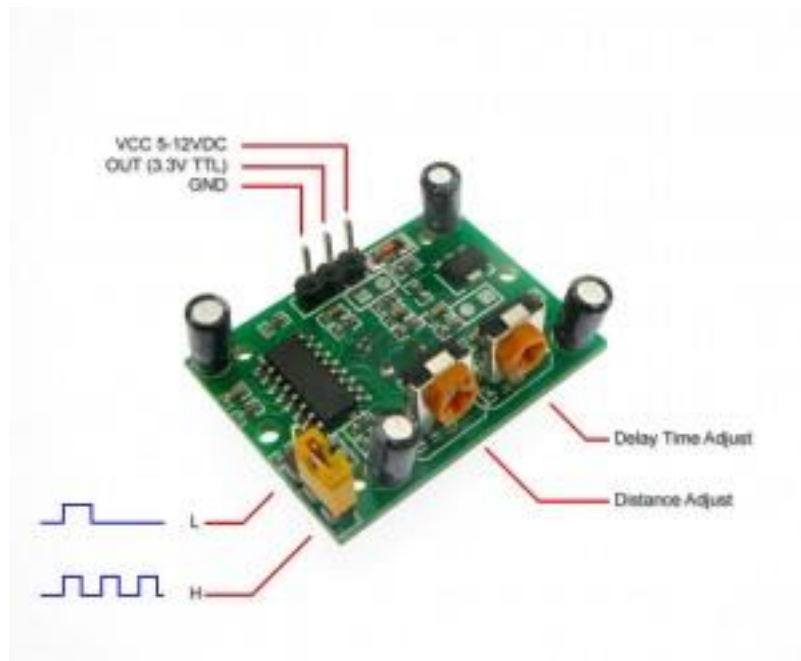
Costo 4-5 Euro,



- Relè  
**1° Programma: Rileviamo la presenza di un oggetto in movimento e accendiamo un LED**

I collegamenti, per questo programma sono abbastanza semplici. Basta collegare i cavi

del sensore PIR, come riportati in figura.



Da notare, una interessante caratteristica, è quella di poter aumentare il periodo in cui l'uscita sia ad un livello logico alto, regolando la manopola di destra. In questo caso, non si fa altro che variare una resistenza, che fa aumentare il tempo in cui l'uscita sia ad 1 logico.

```

1  /* Programam che utilizza il sensore di presenza PIR HC-SR501, per rilevare
2
3  // Tempo di calibrazione del sensore
4  int calibrationTime = 30;
5
6  //Il tempo in cui l'uscita sia bassa
7  long unsigned int lowIn;
8
9  // valore di millisecondi, per cui si ritiene che ci sia "quiete"
10 long unsigned int pause = 5000;
11
12 boolean lockLow = true;
13 boolean takeLowTime;
14
15 int pirPin = 3;    //il PIN di Arduino a cui è collegato il sensore
16 int ledPin = 13;  //il PIN a cui è connesso il LED
17
18 // Impostazione del sensore
19 void setup(){
20   Serial.begin(9600);

```

```

21  pinMode(pirPin, INPUT);
22  pinMode(ledPin, OUTPUT);
23  digitalWrite(pirPin, LOW);
24
25  //Fase di calibrazione
26  Serial.print("calibrating sensor ");
27  for(int i = 0; i < calibrationTime; i++){
28      Serial.print(".");
29      delay(1000);
30  }
31  Serial.println(" done");
32  Serial.println("SENSOR ACTIVE");
33  delay(50);
34  }
35
36  void loop(){
37
38      // Questo IF permette di stabilire se il sensore rileva un oggetto in mov
39      if(digitalRead(pirPin) == HIGH){
40          digitalWrite(ledPin, HIGH);  /Accendiamo il LED
41          if(lockLow){
42
43              lockLow = false;
44              Serial.println("---");
45              Serial.print("motion detected at ");
46              Serial.print(millis()/1000);
47              Serial.println(" sec");
48              delay(50);
49          }
50          takeLowTime = true;
51      }
52      // Questo IF permette di stabilire se non c'è più nessun movimento
53      if(digitalRead(pirPin) == LOW){
54          digitalWrite(ledPin, LOW);  //Si spegne il LED
55
56          if(takeLowTime){
57              lowIn = millis();
58              takeLowTime = false;
59          }
60
61          if(!lockLow && millis() - lowIn > pause){
62
63              lockLow = true;
64              Serial.print("motion ended at ");  //output
65              Serial.print((millis() - pause)/1000);
66              Serial.println(" sec");
67              delay(50);
68          }
69      }
70  }

```



Sensore\_PIR\_LED.in

o

## 2°Programma: Creiamo un un piccolo sistema di allarme con il sensore PIR

Questo programma permette di creare un piccolo sistema d'allarme, grazie ad Buzzer, LED e al sensore PIR. I collegamenti sono indicati nel codice del programma.

```
1  /* Programma che fa emettere un suono e accedere un LED, quando c'è un oggetto
2
3  int pirPin = 2;
4  int pinSpeaker= 10;
5  int ledPin = 13;
6  void setup(){
7    Serial.begin(9600);
8    pinMode(pirPin, INPUT);
9    pinMode(ledPin, OUTPUT);
10   pinMode(pinSpeaker, OUTPUT);
11  }
12  void loop(){
13   int pirVal = digitalRead(pirPin);
14   Serial.println("starting to read PIR");
15   analogWrite(ledPin, 0);
16   delay(2000);
17   if(pirVal == LOW) {
18     analogWrite(ledPin, 255);
19     Serial.println("Motion Detected");
20     tone(10, 700, 100);
21     delay(2000);
22   }
23  }
24  void tone(long duration, int freq) {
25   duration *= 1000;
26   int period = (1.0 / freq) * 1000000;
27   long elapsed_time = 0;
28   while (elapsed_time < duration) {
29     digitalWrite(pinSpeaker,HIGH);
30     delayMicroseconds(period / 2);
31     digitalWrite(pinSpeaker, LOW);
32     delayMicroseconds(period / 2);
33     elapsed_time += (period);
```

```
34 }  
35 }
```



Sensore\_PIR\_Alarm.  
ino

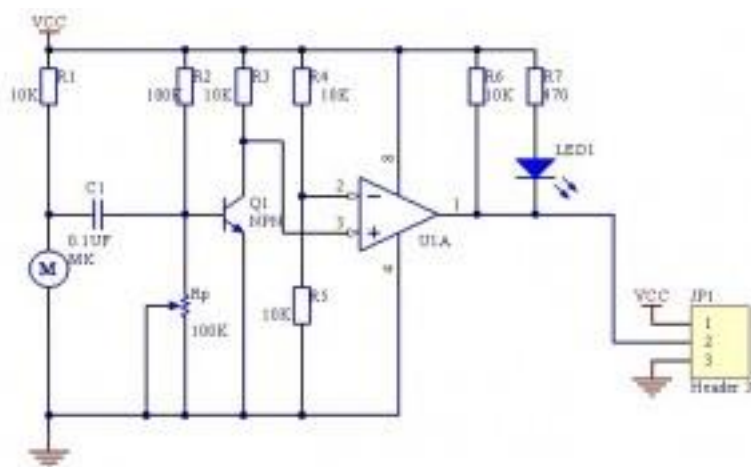
## Capitolo 7: Sensore che rileva rumore

In questa settima lezione su Arduino, vediamo come utilizzare il sensore LM393, che permette di rilevare i suoni che sono presenti in prossimità dell'apparecchio. In pratica è un convertitore analogico/digitale, che converte il livello di rumore, con un'uscita, che varia, nel caso di Arduino, da 0 a 1023.



Ecco il datasheet e il circuito del dispositivo:

LM393





L'utilizzo di questo semplicissimo device è il più ampio possibile. Per esempio è possibile utilizzarlo insieme ad un sensore PIR come allarme, oppure utilizzarlo per accedere e spegnere una lampada, applaudendo. Dal punto di vista logico, il sensore acustico funziona in questo modo. Quando rileva un rumore, il livello di tensione arriva fino a 0 [V], che rappresenta, quindi, lo 0. Mentre se non c'è rumore, il livello logico è 1.

### 1° Programma: Accendiamo una lampada attraverso un applauso

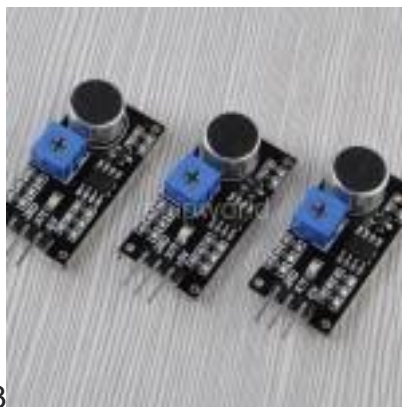
#### Requisiti



- Arduino Uno

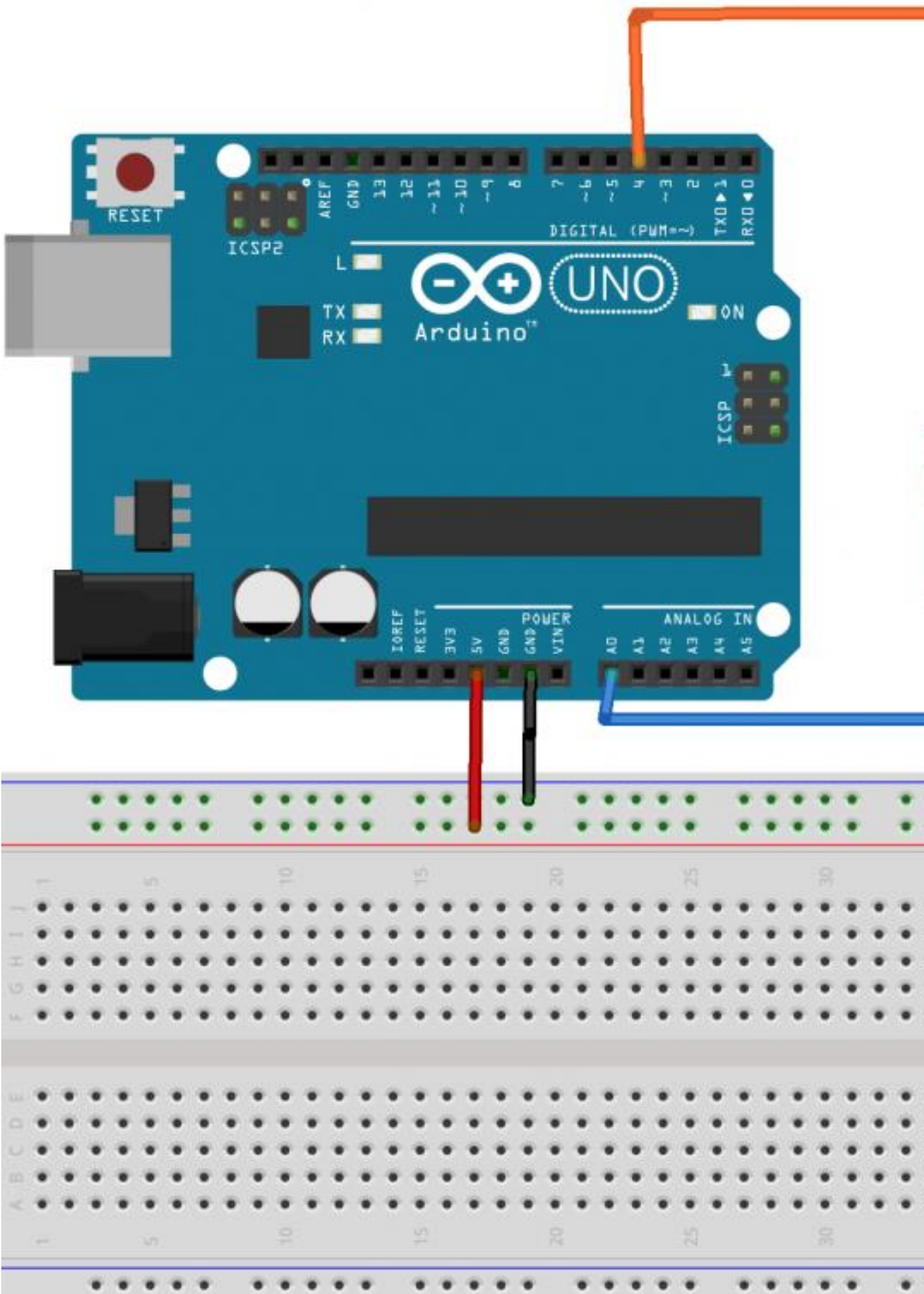


- Relè



- Sensore acustico LM393
- Ecco lo schema del circuito del programma:







Sound\_Sensor\_PCB\_bb



File Fritzing PCB

Questo programma permette di accedere una lampada, controllata attraverso un relè (NO), sfruttando il LM393

```
1 // Questo programma permette di utilizzare il sensore acustico LM393 per accendere una lampada
2 int Relay = 4; // Pin a cui è connesso il relè
3
4 void setup()
5 {
```

```
6   pinMode(Relay, OUTPUT);      // Viene imposta il relè
7   pinMode(A0, INPUT);         // Viene inizializzato il sensore LM393
8   digitalWrite(Relay, LOW);    // Il relè è aperto e quindi non passa corren
9 }
10
11 void loop()
12 {
13   // Questa variabile permette di capire lo stato del relè
14   int on = 0;
15   // Se viene rilevato del rumore il relè si chiude
16   if (digitalRead(A0) == LOW ) {
17     on = !on;
18     digitalWrite(Relay, on ? HIGH : LOW);
19   }
20 }
21
22 }
```



LUCE\_applausi.ino

Il codice, come per tutti gli altri, è possibile scaricarli gratuitamente da questo



link



## Capitolo 8: Utilizziamo il KeyPad con Arduino

In questa ottava puntata su Arduino, introdurremo un nuovo strumento, utile per tanti diversi progetti: il KeyPad.





Il meccanismo di funzionamento è davvero semplice. Esso è formato da 7 cavi, che rappresentano 4 righe e 3 colonne. Attraverso la pressione di un tasto, c'è una variazione di tensione sul Pin collegato ad Arduino e quest'ultimo è in grado di rilevare il relativo valore.

Attraverso questo device è davvero possibile fare tantissimi progetti, come ad esempio una simulazione di un piccolo Bancomat, proteggere una porta d'entrata e tanto altro. Vediamo qualche esempio per Arduino.

### Esempi di programmazione del Keypad

Requisiti Hardware:



- Arduino
- Keypad









- LED



- Relay



- Servo

Requisiti Software:

- Libreria keypad

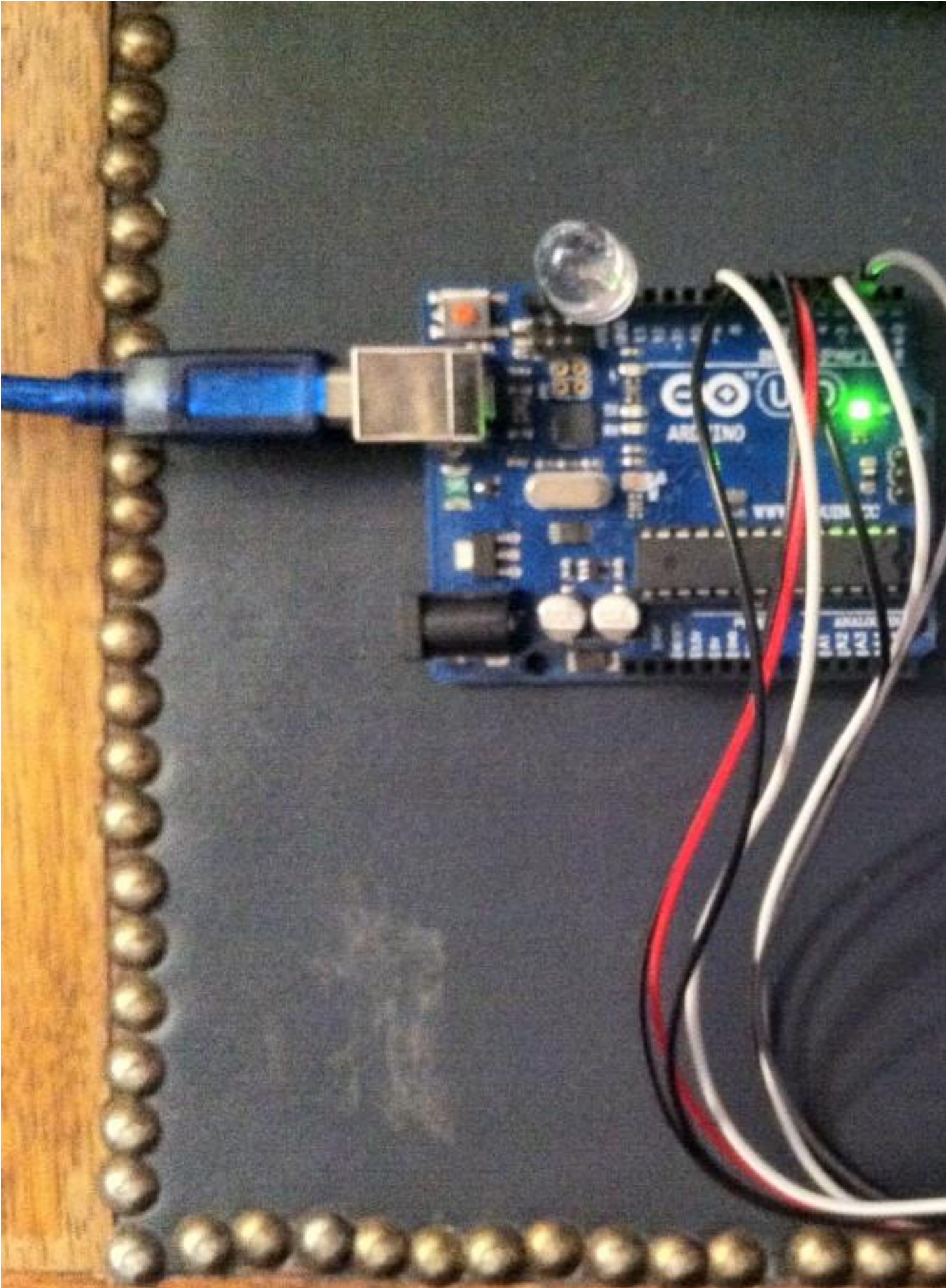
### 1° Programma: Hello Keypad

In questo piccolo programma, vedremo come mostrare su seriale il tasto premuto sul Keypad

```
1 /* @file HelloKeypad.pde
2 || @version 1.0
3 || @author Alexander Brevig
4 || @contact alexanderbrevig@gmail.com
5 ||
6 || @description
7 || | Demonstrates the simplest use of the matrix Keypad library.
8 || #
9 */
```

```
10 #include <Keypad.h>
11
12 const byte ROWS = 4; //four rows
13 const byte COLS = 3; //three columns
14 char keys[ROWS][COLS] = {
15     {'1','2','3'},
16     {'4','5','6'},
17     {'7','8','9'},
18     {'*','0','#'}
19 };
20 byte rowPins[ROWS] = {8,7,6,5}; //connect to the row pinouts of the keypad
21 byte colPins[COLS] = {4,3,2}; //connect to the column pinouts of the keypad
22
23 Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
24
25 void setup(){
26     Serial.begin(9600);
27 }
28
29 void loop(){
30     char key = keypad.getKey();
31
32     if (key){
33         Serial.println(key);
34     }
35 }
```

Il funzionamento del programma dal punto di vista hardware è già stato spiegato precedentemente. Dal punto di vista software, la situazione è davvero semplice. Sfruttando le librerie già presenti sul sito di Arduino, il loop() è davvero immediato da capire: una volta che viene premuto qualcosa, si entra in if, che stampa sul seriale il relativo valore, semplice no ? Il collegamento del KeyPad con Arduino è semplice, basta guardare la seguente foto.



In pratica, prendendo il Keypad dal lato frontale, basta collegare i cavi in modo parallelo, partendo dal Pin 8 fino al Pin 2.

Ecco il video che mostra il Keypad in azione con Arduino:

## 2° Programma: Keypad che permette di proteggere qualcosa

Questo secondo programma può essere modificato dall'utente. In pratica, questo codice, non fa altro che permette di inserire un codice con il Keypad, verificarlo tramite Arduino e se tale codice è corretto, accende un LED. In realtà è possibile fare qualsiasi cosa, come ad esempio accendere una lampada, aprire una porta con un servo. Lascio questo codice, in modo che chiunque possa implementarlo a suo piacimento.

```

1  /**
2  Questo programma permette di far inserire da un utente un codice con il Keypad
3  Per inviare il codice, una volta digitato interamente è necessario premere *
4  Autore Giacomo Bellazzi
5  Versione 1.0
6  */
7  #include <Keypad.h>
8  #define LED 13
9  const byte ROWS = 4; //quattro righe
10 const byte COLS = 3; //tre colonne
11 char keyInsert[6];
12 // Queste variabili servono come verifica del corretto inserimento del codice
13 int i = 0;
14 int j = 0;
15 int s = 0;
16 int x = 0;
17 // Codice segreto
18 char code[7]= "112233";
19 char keys[ROWS][COLS] = {
20   {'1','2','3'},
21   {'4','5','6'},
22   {'7','8','9'},
23   {'*','0','#'}
24 };
25 byte rowPins[ROWS] = {8,7,6,5}; //i Pin a cui sono connesse le righe del Keypad
26 byte colPins[COLS] = {4,3,2}; // i Pin a cui sono connesse le colonne del Keypad
27
28 Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
29
30 void setup(){
31   Serial.begin(9600);
32   pinMode(LED,OUTPUT);
33 }
34
35 void loop(){

```



```
36 char key = keypad.getKey();
37 if (i==0){
38     Serial.println("Insert PIN to verify...");
39     i++;
40 }
41 if (key != NO_KEY && j<6){
42     Serial.print("*");
43     //Serial.println(key);
44     keyInsert[j]=key;
45     j++;
46 }
47 if(key == '*') {
48     Serial.println();
49     Serial.println("Verifyng the code...");
50     delay(1000);
51     for(s=0; s<6;s++){
52         if(keyInsert[s]==code[s]){
53             x++;
54         }
55     }
56     if(x==6){
57         Serial.println("The code is correct");
58         digitalWrite(LED,HIGH);
59         //TODO possibili ulteriori implementazioni
60     }else{
61         Serial.println("The code is incorrect, please retry");
62         delay(2000);
63         x=0;
64         i=0;
65         j=0;
66     }
67 }
68 }
69 if(key == '#'){
70     x=0;
71     i=0;
72     j=0;
73     digitalWrite(LED,LOW);
74 }
75 }
```

Il codice risultato essere leggermente un po' più complesso del primo, tuttavia attraverso i commenti presenti è facile intuire il meccanismo di funzionamento dei vari blocchi di codice. In pratica vengono letti i valori inseriti attraverso il Keypad. Una volta terminata l'immissione dei 6 caratteri, si preme il comando \* e Arduino valuta se il codice inserito è lo stesso di quello presente in memoria. Questo avviene attraverso un ciclo for e un if che verificano se i 6 valori sono uguali.

Ecco il video che mostra il funzionamento del programma :

### 3°Programma: Utilizziamo il OTP e il KeyPad

Questo programma è stato realizzato da Luca Dentella, un appassionato di Arduino. Questo programma si basa sull'utilizzo del metodo di sicurezza OTP, che viene usato da chi ha un Online Banking. Per chi non sapesse cosa sia l'OTP, ecco una definizione tratta da Wikiepdia:

Una **One-Time Password** (password usata una sola volta) è una password che è valida solo per una singola sessione di accesso o una transazione. La OTP evita una serie di carenze associate all'uso della tradizionale password (statica). Il più importante problema che viene risolto da OTP è che, al contrario della password statica, esso non è vulnerabile agli attacchi con replica. Ciò significa che, se un potenziale intruso riesce ad intercettare una OTP che è stata già utilizzata per accedere a un servizio o eseguire una transazione, non sarà in grado di riutilizzarla, in quanto non sarà più valida. D'altra parte, una OTP non può essere memorizzata da una persona. Essa richiede quindi una tecnologia supplementare per poter essere utilizzata.[1]

Come vengono generate le password OTP ?

Gli algoritmi di generazione delle OTP in genere fanno uso di numeri casuali. Ciò è necessario perché altrimenti sarebbe facile prevedere l'OTP futuro osservando i precedenti. Gli algoritmi OTP che sono stati realizzati sono abbastanza diversi tra loro. I vari approcci per la generazione di OTP sono elencati di seguito.

- Algoritmi basati sulla **sincronizzazione temporale** tra server di autenticazione e client che fornisce la password (le OTP sono valide solo per un breve periodo di tempo)
- Algoritmi matematici che generano una nuova **password in base alla password precedente** (le OTP sono, di fatto, una catena di password legate tra loro, e devono essere utilizzate in un ordine predefinito)
- Algoritmi matematici dove la password è **basata su una sfida** (per esempio, un numero casuale scelto dal server di autenticazione o dai dettagli della transazione) e/o su un contatore.

Ci sono anche diversi modi per rendere note all'utente le successive OTP da usare. Alcuni sistemi elettronici prevedono l'uso di speciali token che l'utente porta con sé, che generano le OTP e le mostrano utilizzando un piccolo display. Altri sistemi sono costituiti da un software che gira sul telefono cellulare dell'utente. Altri sistemi generano le OTP sul lato server e le trasmettono all'utente su un canale fuori banda, come ad esempio un canale di messaggistica SMS. Infine, in alcuni sistemi le OTP sono stampate su carta, che l'utente è tenuto a portare con sé.

In pratica l'OTP si basa su un calcolo di codice, che è formato da una password one time e da un algoritmo che proviene dall'attuale ora.

Per il calcolo del codice, viene incontro una semplice applicazione di Google. Per il download della parte di Arduino, trovato tutto il materiale sul post dell'autore del codice <http://www.lucadentella.it/2013/09/14/serratura-otp/2/>.

Per prima cosa, è necessario inserire hmackey all'interno del codice di Arduino; questo valore si ricava inserendo la password di default, all'interno di questo script, realizzato sempre da Luca Dentella <http://www.lucadentella.it/OTP>

## OTP Tool for Arduino and Google Authenticator

Choose an account name:	<input type="text" value="my door"/>
Insert your secret (10 characters):	<input type="text" value="testpwd123"/> <input type="button" value="GO"/>
Arduino HEX array:	<input style="background-color: #e0e0e0;" type="text" value="{0x74, 0x65, 0x73, 0x74, 0x70, 0x77, 0x64, 0x31, 0x32, 0x33}"/>
Google Authenticator code:	<input style="background-color: #e0e0e0;" type="text" value="ORSXG5DQO5SDCMRT"/>
QRCode:	

Una volta copiato il codice nello Sketch per Arduino, basterà compilare e per quanto riguarda la parte legata ad Arduino, abbiamo finito. Ora è necessario installare il programma per il calcolo del codice OTP sul nostro Smartphone: Google Authenticator.

Ora che abbiamo installato il programma lo apriamo e attraverso la fotocamera, puntiamo lo smartphone sul codice QR che è stato generato in precedenza. Ora abbiamo finito e possiamo testare il programma, attraverso il seriale con Arduino.

Ecco un video che mostra l'utilizzo del programma:

### KeyPad 4x4







Sul web è possibile trovare una versione del Keypad con 16 caratteri, cioè con i numeri dallo zero al nove e le lettere A,B,C,D. Il meccanismo di funzionamento è lo stesso, il codice varia leggermente. Ecco due esempi come mostrano l'utilizzo del Keypad nella sua versione "più grande":

#### 4° Programma con Keypad 4x4: HelloKeypad 4x4

```

1  /*Questo programma mostra come usare la tastiera Keypad 4x4 con Arduino Uno.
2  Il testo che viene premuto sulla Keypad viene mostrato nel seriale
3  */
4  #include <Keypad.h>
5
6  const byte ROWS = 4; //quattro righe
7  const byte COLS = 4; //quattro colonne
8  byte colPins[4] = {5,4,3,2}; // Pin a cui sono connesse le colonne
9  byte rowPins[4] = {9,8,7,6}; // Pin a cui sono connesse le righe
10 char Keys[4][4]= //creo la matrice dei tasti della tastiera.
11 {
12 {'1','2','3','A'} ,
13 {'4','5','6','B'},
14 {'7','8','9','C'},
15 {'*','0','#','D'}
16 };
17 Keypad keyp = Keypad(makeKeymap(Keys), rowPins, colPins,4,4);
18
19 void setup(){
20   Serial.begin(9600);
21 }
22
23 void loop(){
24   char key = keyp.getKey();
25   if (key){
26     Serial.println(key);
27   }
28 }

```

#### 5°Programma con Keypad 4x4: Codice Segreto con lettere

```

1  /**
2  Questo programma permette di far inserire da un utente un codice con il Keypad
3  Per inviare il codice, una volta digitato interamente è necessario premere *
4  Autore Giacomo Bellazzi
5  Versione 1.0
6  */
7  #include <Keypad.h>
8  #define LED 13
9  const byte ROWS = 4; //quattro righe
10 const byte COLS = 4; //quattro colonne
11 char keyInsert[6];
12 // Queste variabili servono come verifica del corretto inserimento del codice

```

```

13 int i = 0;
14 int j = 0;
15 int s = 0;
16 int x = 0;
17 // Codice segreto
18 char code[7]= "11ABCD";
19 char Keys[ROWS][COLS]= //creo la matrice dei tasti della tastiera.
20 {
21 {'1','2','3','A'} ,
22 {'4','5','6','B'},
23 {'7','8','9','C'},
24 {'*','0','#','D'}
25 };
26 byte colPins[4] = {5,4,3,2}; // Pin a cui sono connesse le colonne
27 byte rowPins[4] = {9,8,7,6}; // Pin a cui sono connesse le righe
28
29 Keypad keypad = Keypad( makeKeymap(Keys), rowPins, colPins, ROWS, COLS);
30
31 void setup(){
32   Serial.begin(9600);
33   pinMode(LED,OUTPUT);
34 }
35
36 void loop(){
37   char key = keypad.getKey();
38   if (i==0){
39     Serial.println("Insert PIN to verify...");
40     i++;
41   }
42   if (key != NO_KEY && j<6){
43     Serial.print("*");
44     //Serial.println(key);
45     keyInsert[j]=key;
46     j++;
47   }
48   if(key == '*') {
49     Serial.println();
50     Serial.println("Verifying the code...");
51     delay(1000);
52     for(s=0; s<6;s++){
53       if(keyInsert[s]==code[s]){
54         x++;
55       }
56     }
57     if(x==6){
58       Serial.println("The code is correct");
59       digitalWrite(LED,HIGH);
60       //TODO possibili ulteriori implementazioni
61     }else{
62       Serial.println("The code is incorrect, please retry");

```

```
63         delay(2000);
64         x=0;
65         i=0;
66         j=0;
67     }
68 }
69 if(key == '#'){
70     x=0;
71     i=0;
72     j=0;
73     digitalWrite(LED,LOW);
74 }
75 }
```

### Conclusione

Spero che i programmi presentati in questo post, siano di facile lettura e soprattutto, siano possibili di ulteriori implementazioni.





## Capitolo 9: Come associare un modulo RTC ad Arduino

Per chi ha avuto modo di usare Arduino, si sarà accorto che non è presente un modulo RTC all'interno della scheda. Questo significa che Arduino non è in grado "da solo" di sapere l'ora attuale; l'unica operazione che è in grado di fare è quello di contare i millisecondi che sono passati dall'accensione del dispositivo, ma non niente di più.

Come fare per poter utilizzare l'ora, nei nostri progetti con il dispositivo ?

A questa domanda ci sono ben 3 soluzioni:

- Utilizzare delle librerie "speciali" di Arduino, che permettono di sfruttare la funzione di calcolo dei millisecondi presenti nel microcontrollore
- Utilizzare un modulo separato RTC, in grado di essere supportato da Arduino
- Sfruttare il web, con i relativi server NTC, per ricevere l'ora attuale

In questo post vedremo le prime due soluzioni, mentre la terza verrà mostrata, quando introdurremo lo shield per connettere Arduino al web.

### 1° Metodo: Librerie software per Arduino

Per il download della libreria, occorre andare al seguente link

<https://github.com/leomil72/swRTC>

```

1  /* This file is part of swRTC library.
2     Please check the README file and the notes
3     inside the swRTC.h file to get more info
4
5     This example will print the time every second
6     to the computer through the serial port using the
7     format HH:MM or HH:MM:SS
8
9     Written by Leonardo Miliani <leonardo AT leonardomiliani DOT com>
10
11    This library is free software; you can redistribute it and/or
12    modify it under the terms of the GNU General Public
13    License as published by the Free Software Foundation; either
14    version 3.0 of the License, or (at your option) any later version.
15 */
16
17 #include <swRTC.h>
18 swRTC rtc; //create a new instance of the lib
19 const byte WITHOUT_SECONDS = 0;
20 const byte WITH_SECONDS = 1;
21
22 void setup() {
23     rtc.stopRTC(); //stop the RTC

```

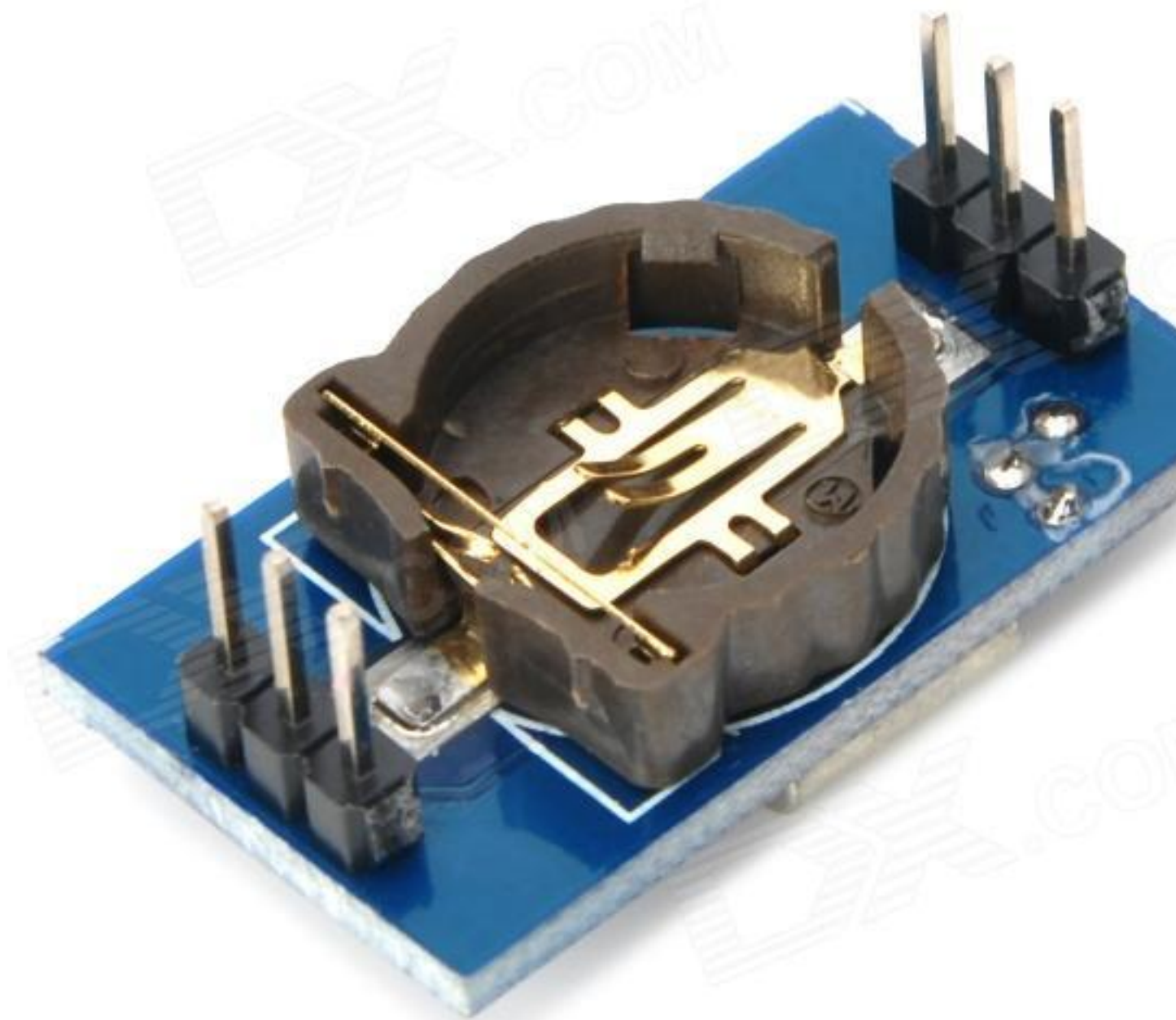
```
24     rtc.setTime(12,0,0); //set the time here
25     rtc.setDate(4,6,2012); //set the date here
26     rtc.startRTC(); //start the RTC
27     Serial.begin(19200); //choose the serial speed here
28     delay(2000); //delay to let the user opens the serial monitor
29 }
30
31 void loop() {
32     printTime(WITHOUT_SECONDS);
33     printTime(WITH_SECONDS);
34     Serial.println("");
35     delay(1000);
36 }
37
38 void printTime(byte withSeconds) {
39     sendNumber(rtc.getHours());
40     Serial.print(":");
41     sendNumber(rtc.getMinutes());
42     if (withSeconds) {
43         Serial.print(":");
44         sendNumber(rtc.getSeconds());
45     }
46     Serial.println("");
47 }
48
49 void sendNumber(byte number) {
50     byte temp;
51     if (number>9) {
52         temp=int(number/10);
53         Serial.print(char(temp+48));
54         number-=(temp*10);
55     } else {
56         Serial.print("0");
57     }
58     Serial.print(char(number+48));
59 }
```

Come si può notare dal codice del programma, all'interno della funzione void setup(), è possibile impostare la data attuale, in modo che Arduino si in grado di calcolarsela, sfruttando il contatore presente nel dispositivo. Ovviamente, qualora Arduino venisse spento, durante la fase di riavvio, verrebbe ricaricato il programma presente in memoria, con l'ora che non è più aggiornata, quindi occorre prestare attenzione per questa soluzione.

## 2° Metodo : Utilizzare un RTC hardware esterno

Per questa seconda soluzione è necessario utilizzare un dispositivo esterno Arduino. Sul web se ne trovano tanti disponibili. Per questo programma ho scelto il DS1302.





Ecco la documentazione del dispositivo:



Ecco il codice per utilizzare questo modulo RTC con Arduino:

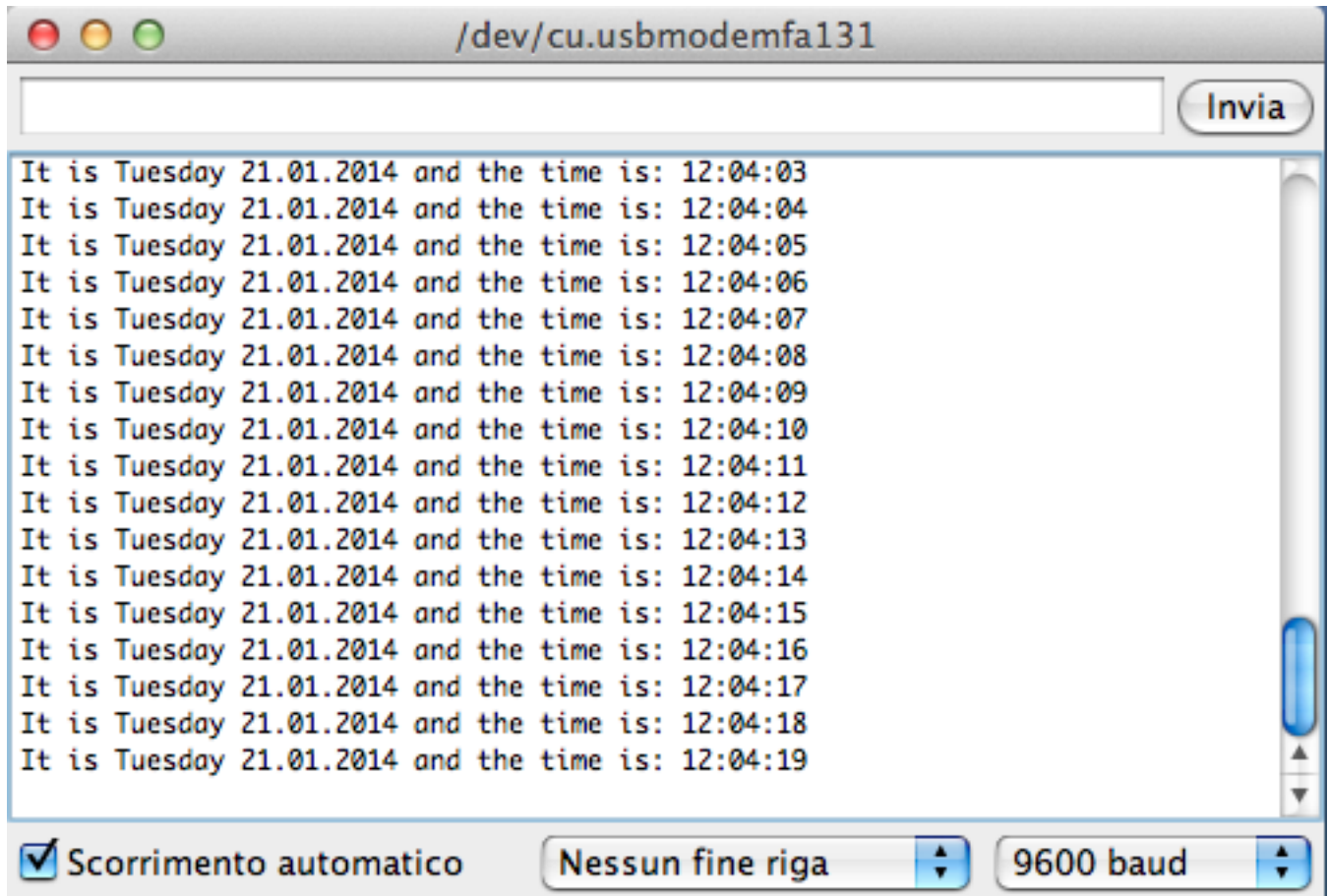
```

1  /*
2  Questo programma permette di utilizzare il modulo RTC DS1302 con Arduino, in
3  */
4  // Vengono definiti i PIN a cui è connesso il modulo
5  #define SCK_PIN 4
6  #define IO_PIN 3
7  #define RST_PIN 2
8  #include <DS1302.h>
9
10 // Viene inizializzata la libreria
11 DS1302 rtc(RST_PIN, IO_PIN, SCK_PIN);
12
13 void setup()
14 {
15     rtc.halt(false);
16     // Viene disattiva la protezione alla scrittura
17     rtc.writeProtect(false);
18     Serial.begin(9600);
19 }
20
21 /* Main program */
22 void loop()
23 {
24
25     /* Qui è necesario impostare l'ora e la data.
26     Durante la fase di impostazione occorre togliere via i commenti laterali e
27     Successivamente occorre commentare le tre righe di codice e compilare.
    
```

```
28   In questo modo qualora Arduino dovesse riavviarsi, non verranno impostati i
29   */
30   // rtc.setDOW(TUESDAY);
31   //rtc.setTime(12,03,0);
32   //rtc.setDate(21, 01, 2014);
33
34   /* Read the time and date once every second */
35   while(1)
36   {
37     Serial.print("It is ");
38     Serial.print(rtc.getDOWStr());
39     Serial.print(" ");
40     Serial.print(rtc.getDateStr());
41     Serial.print(" ");
42     Serial.print("and the time is: ");
43     Serial.println(rtc.getTimeStr());
44     delay (1000);
45   }
46 }
```

Per un corretto funzionamento del programma, è necessario leggere quanto spiegato tra i commenti. Riassumendo occorre impostare una volta i dati relativi all'ora e alla data, compilare. Dopo aver fatto ciò, occorre commentare le righe di codice che permettono di impostare i dati e compilare ancora. In questo modo, un eventuale riavvio di Arduino, non farà impostare i dati sbagliati !

Ed ecco il risultato finale:







Come sempre i codici dei programmi riportati in questo post sono scaricabili dal seguente link.

## Capitolo 10: Scopriamo gli infrarossi

In questa lezione, vedremo come vengono usati in Arduino i segnali infrarossi. Chi di noi non ha un telecomando in casa, come ad esempio quello per accendere e spegnere la TV ? Bene oggi capiremo come il nostro TV è in grado di stabilire il tasto premuto dal nostro telecomando.

I supporti hardware richiesti per questi programmi sono davvero pochi. Infatti oltre allo stesso Arduino, dovremo procurarci su Ebay un ricevente IR, con il relativo telecomando e un LED ad infrarossi, in grado di inviare i segnali. Ecco una lista di possibili prodotti, acquistabili su ebay:







Il costo di questi dispositivi è davvero poco e si aggira intorno ai 5 Euro.

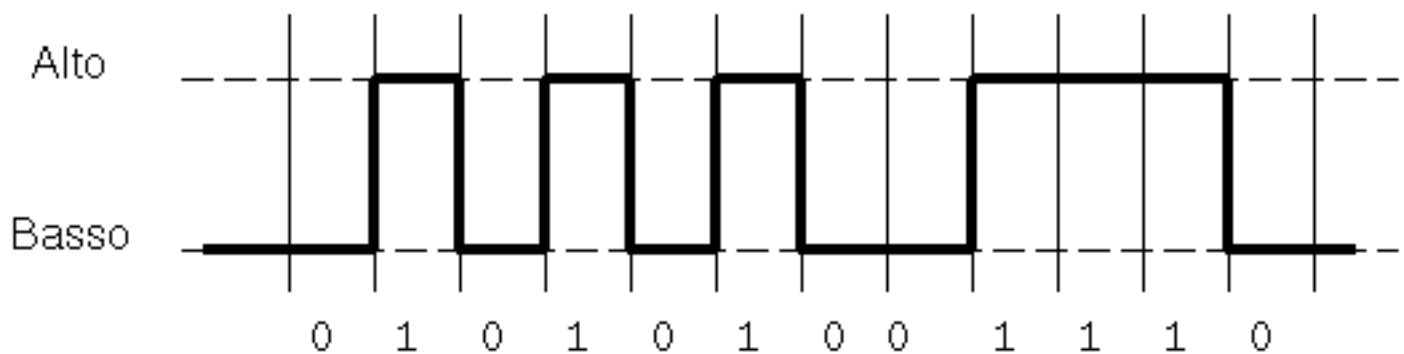
Dal punto di vista software è necessario scaricare la libreria ufficiale per Arduino, da questo link:



### Come funziona una trasmissione ad infrarossi ?

Il meccanismo di funzionamento di una trasmissione ad infrarossi è abbastanza “semplice”; un fascio luminoso, ad elevata frequenza 38 KHz (per i telecomandi tipici per la TV) viene proiettato contro un ricevente. Questo fascio luminoso non è altro che una sequenza di bit (0 e 1) che vengono codificati dal ricevente, attraverso la polarizzazione di photo-transistor. Facciamo questo semplice caso; nel nostro telecomando, vengono codificati i vari pulsanti con varie sequenze bit e per esempio il tasto on/off vale:

**010101001110**



Il ricevente avrà un segnale in ricezione molto simile, se non uguale a quello inviato e in base alla corrispondenza che ha il memoria, sarà in grado di svolgere la richiesta effettiva, come ad esempio accendere la TV ! Il fascio luminoso che viene inviato dal telecomando, non è possibile vederlo ad occhio nudo perché ha una frequenza troppo veloce per noi !

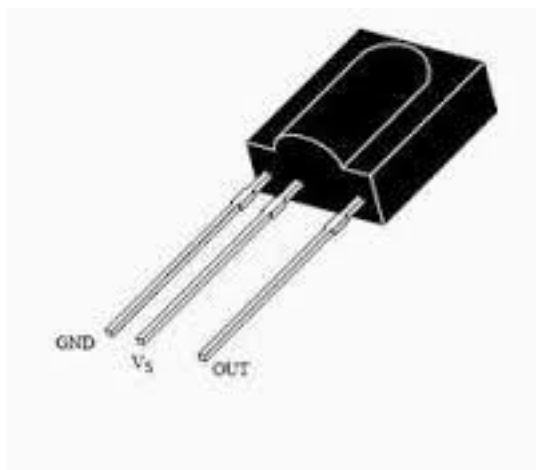
Tuttavia, ci sono diverse codifiche; tipicamente ogni produttore di dispositivi tecnologici, ha una propria codifica, come ad esempio Sony, NEC, Samsung etc..

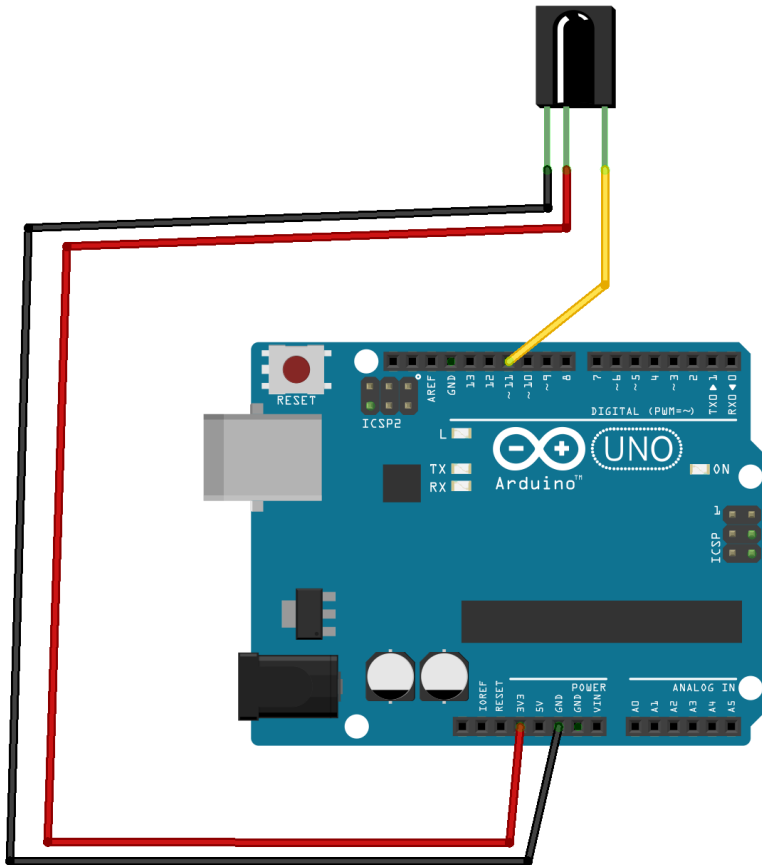
Tuttavia, è possibile stabilire la sequenza di bit di qualsiasi segnale ad infrarosso, calcolando il tempo in cui il segnale è alto e il tempo in cui il segnale è basso. Questo tipo di codifica si chiama RAW.

### Passiamo ai codici per Arduino

Come per ogni libreria nuova in Arduino, dobbiamo installarla, copiandola semplicemente nella cartella Libraries, che tipicamente si trova nei propri documenti.

Qualora utilizzassimo il dispositivo TSOP1738, il collegamento è davvero semplice; basterà alimentare il segnale con i 5 V di Arduino, collegarlo a massa e collegare il pin di output al pin 11.





Use irremote library:  
<https://github.com/shirriff/Arduino-IRremote>  
 Use IRrecvDump demo to display IR codes

Use irremote library:  
<https://github.com/shirriff/Arduino-IRremote>  
 Anode to signal

## TSOP1738 Datasheet

Per quanto riguarda altri dispositivi presenti su Ebay, basterà collegare il pin di output sempre al pin 11 di Arduino.

### 1° Programma decodifichiamo un generico segnale

Nome programma Decode\_generic\_IR

```

1  /*
2
3  * Modified by Chris Targett
4  * Now includes more protocols
5  * Novemeber 2011
6
7  * IRremote: IRrecvDump - dump details of IR codes with IRrecv
8  * An IR detector/demodulator must be connected to the input RECV_PIN.
9  * Version 0.1 July, 2009
10 * Copyright 2009 Ken Shirriff

```

```
11  * http://arcfn.com
12  *
13  * Modified by Chris Targett to speed up the process of collecting
14  * IR (HEX and DEC) codes from a remote (to put into and .h file)
15  *
16  */
17
18 #include <IRremote.h>
19
20 int RECV_PIN = 11;
21
22 IRrecv irrecv(RECV_PIN);
23
24 decode_results results;
25
26 void setup()
27 {
28   Serial.begin(9600);
29   irrecv.enableIRIn(); // Start the receiver
30 }
31
32 // Dumps out the decode_results structure.
33 // Call this after IRrecv::decode()
34 // void * to work around compiler issue
35 //void dump(void *v) {
36 //  decode_results *results = (decode_results *)v
37 void dump(decode_results *results) {
38   int count = results->rawlen;
39   if (results->decode_type == UNKNOWN) {
40     Serial.print("Unknown encoding: ");
41   }
42   else if (results->decode_type == NEC) {
43     Serial.print("Decoded NEC: ");
44   }
45   else if (results->decode_type == SONY) {
46     Serial.print("Decoded SONY: ");
47   }
48   else if (results->decode_type == RC5) {
49     Serial.print("Decoded RC5: ");
50   }
51   else if (results->decode_type == RC6) {
52     Serial.print("Decoded RC6: ");
53   }
54   else if (results->decode_type == SAMSUNG) {
55     Serial.print("Decoded SAMSUNG: ");
56   }
57   else if (results->decode_type == JVC) {
58     Serial.print("Decoded JVC: ");
59   }
}
```

```

60  else if (results->decode_type == PANASONIC) {
61      Serial.print("Decoded Panasonic: ");
62  }
63  Serial.print(results->value, HEX);
64  Serial.print("(");
65  Serial.print(results->bits, DEC);
66  Serial.println(" bits)");
67  Serial.print("#define Something_DEC ");
68  Serial.println(results->value, DEC);
69  Serial.print("#define Something_HEX ");
70  Serial.println(results->value, HEX);
71  Serial.print("Raw (");
72  Serial.print(count, DEC);
73  Serial.print("): ");
74  for (int i = 0; i < count; i++) {
75      if ((i % 2) == 1) {
76          Serial.print(results->rawbuf[i]*USECPERTICK, DEC);
77      }
78      else {
79          Serial.print(-(int)results->rawbuf[i]*USECPERTICK, DEC);
80      }
81      Serial.print(" ");
82  }
83  Serial.println("");
84  }
85
86  void loop() {
87      if (irrecv.decode(&results)) {
88          dump(&results);
89          irrecv.resume(); // Receive the next value
90      }
91  }

```

Il programma qui presente mostra sul seriale la codifica del segnale ad infrarosso che viene ricevuto dal ricevitore, collegato al pin 11 di Arduino. Il meccanismo è davvero abbastanza chiara, grazie ai commenti presenti nel codice, tratto dalla libreria ufficiale.

## 2° Programma per Arduino: Accendiamo una lampada con qualsiasi segnale ad Infrarosso

```

1  /**
2  Questo programme permette di accendere una lampada, attraverso un relay, con
3  telecomando ad infrarossi. Questa versione non permette di modificare il cod
4  a run time
5  Autore Giacomo Bellazzi
6  */
7
8  #include <IRremote.h>
9

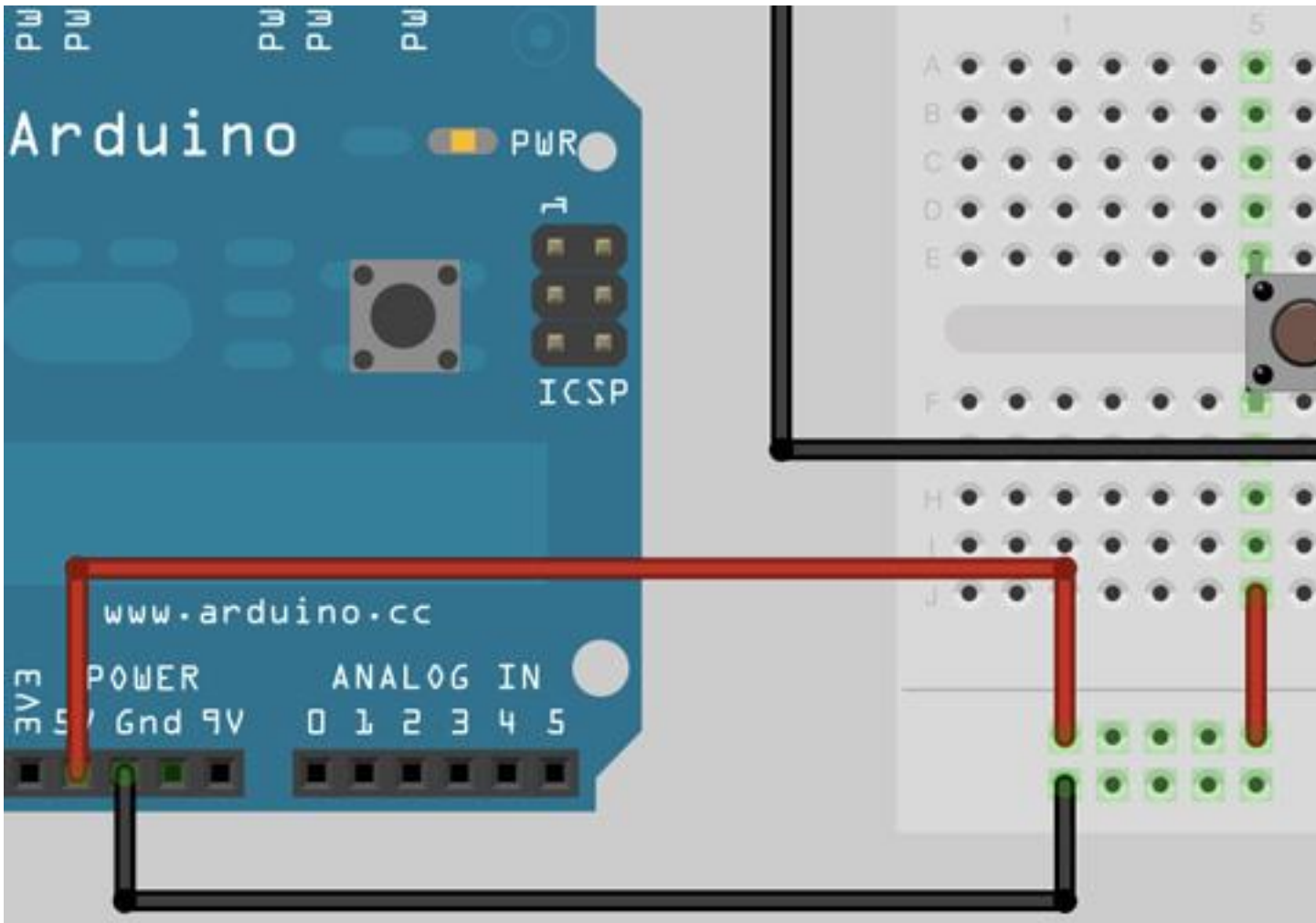
```

```
10 int RECV_PIN = 11;
11 int RELAY_PIN = 13;
12 int codice = 16738455; // codice dec del telecomando
13
14 IRrecv irrecv(RECV_PIN);
15 decode_results results;
16
17 void setup()
18 {
19   pinMode(RELAY_PIN, OUTPUT);
20
21   Serial.begin(9600);
22   irrecv.enableIRIn(); // Start the receiver
23 }
24
25 int on = 0;
26 unsigned long last = millis();
27
28 void loop() {
29
30   if (irrecv.decode(&results)){
31     // If it's been at least 1/4 second since the last
32     // IR received, toggle the relay
33     if (millis() - last > 250 && results.value==codice ) {
34       on = !on;
35       digitalWrite(RELAY_PIN, on ? HIGH : LOW);
36       //dump(&results);
37     }
38     last = millis();
39     irrecv.resume(); // Receive the next value
40   }
41 }
```

Questo codice permette di accendere una lampada connessa ad un relè, premendo un telecomando ad infrarossi. La codifica del telecomando che si vuole usare, può essere ottenuta con il codice del primo programma e inserita all'inizio di questo codice. Tuttavia, il codice non è modificabile a run time.

### 3° Programma: Accendiamo una lampada con un qualsiasi telecomando





```

1  /* Questo progetto permette di utilizzare Arduino e un telecomando IR, per ac
2  Inoltre è possibile modificare il code del telecomando, attraverso il pushbu
3  AUTHOR: Giacomo Bellazzi
4  VERSION: 1.0
5  */
6
7  #include <IRremote.h>
8
9  int RECV_PIN = 11;
10 int RELAY_PIN = 13;
11 int pushButton = 3;
12 unsigned long code;
13 IRrecv irrecv(RECV_PIN);
14 decode_results results;
15
16 void setup()
17 {
18   pinMode(RELAY_PIN, OUTPUT);
19   pinMode(pushButton, INPUT);
20   irrecv.enableIRIn();
21   Serial.begin(9600);
22 }

```

```
23
24 int on = 0;
25 unsigned long last = millis();
26
27 void loop() {
28     int buttonState = digitalRead(pushButton);
29     delay(100);
30     if ( buttonState == HIGH) {
31         Serial.println("Settings...");
32         if (irrecv.decode(&results) && millis() - last > 250){
33             code = results.value;
34             Serial.println("New code setting...");
35             Serial.println(code);
36             Serial.println(results.value);
37             last = millis();
38             irrecv.resume();
39         }
40     }
41     if (irrecv.decode(&results)){
42         if (millis() - last > 250 && results.value==code ) {
43             on = !on;
44             digitalWrite(RELAY_PIN, on ? HIGH : LOW);
45         }
46         last = millis();
47         irrecv.resume();
48     }
49 }
```

Questo programma, a differenza di quello precedente, permette di modificare il codice IR che permette di azionare un relè, anche in fase di run time, impostandolo attraverso un push button collegato ad Arduino. L'operazione è davvero semplice ed è mostrata nel video:

#### 4° Programma: accendiamo la TV Samsung di casa con Arduino

Questo programma permette di accendere una TV Samsung con il nostro piccolo Arduino. Il primo passaggio è quello di ricavarsi il codice RAW del segnale emesso dal nostro telecomando, attraverso il programma che è stato precedentemente discusso precedentemente,

```

/dev/cu.usbmodemfa131 (Arduino Uno)

Decoded SAMSUNG: E0E040BF(32 bits)
#define Something_DEC 3772793023
#define Something_HEX E0E040BF
Raw (68): -30498 4450 -4400 600 -1650 600 -1600 600 -1650 600 -500 600 -550 550 -550 600

 Scorrimento automatico
Nessun fine riga

```

Ora dobbiamo prendere il codice RAW, cancellare il primo valore, convertire i segni – con degli spazi e mettere le virgole tra i vari valori.

```

30498,4450,4400,600,1650,600,1600,600,1650,600,500,600,550,550,550,600,500,600,55
0,550,1600,650,1600,600,1650,600,550,550,550,550,600,550,550,550,600,500,600,1
600,600,550,600,550,550,550,550,600,500,600,550,600,1600,600,550,550,1650,600,
1600,600,1650,600,1600,600,1650,600,1600,600,

```

Il codice per Arduino che serve per inviare il segnale alla TV è il seguente:

```

1  #include "IRremote.h"
2
3  IRsend irsend;
4
5  void setup()
6  {
7    Serial.begin(9600);
8  }
9
10 unsigned int ON[68]={4450,4400,600,1650,600,1600,600,1650,600,500,600,550,55
11 0,550,1600,650,1600,600,1650,600,550,550,550,550,600,550,550,550,600,500,600,1
12 600,600,550,600,550,550,550,550,600,500,600,1600,600,550,600,550,550,550,550,600,
13 500,600,550,600,1600,600,550,550,1650,600,1600,600,1650,600,1600,600,1650,600,
14
15 void loop() {
16   irsend.sendRaw(ON,68,38);
17   Serial.println("Accendo la TV Samsung");

```

```
18 delay(10000);  
19 }
```

Il LED ad infrarossi va collegato al pin 3, con una resistenza di almeno 100 Ohm in serie. Il valore della resistenza è proporzionale al raggio d'azione del telecomando di Arduino.

Come per le lezioni precedenti, i codici di questi programmi sono presenti nella mia repo di Github.