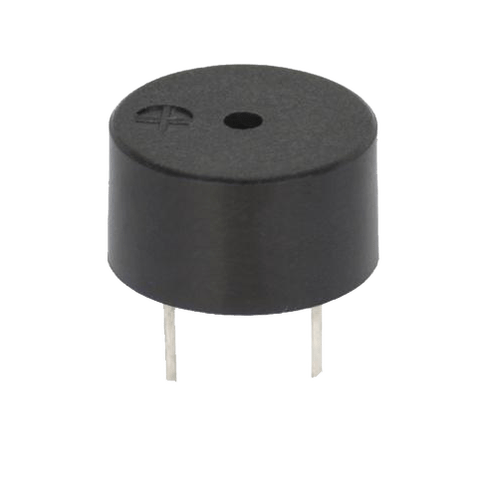
**GUIDA ARDUINO DIGISPARK**

**Ultrasuoni + Cicalino**



Mattia Ruberto & Matteo Ghilardini

SOMMARIO

[SOMMARIO 2](#_Toc536014718)

[Scopo 3](#_Toc536014719)

[Componenti 3](#_Toc536014720)

[Arduino Digispark 3](#_Toc536014721)

[Pulsante 4](#_Toc536014722)

[Led RGB 4](#_Toc536014723)

[Schema Elettrico 5](#_Toc536014724)

[Librerie 5](#_Toc536014725)

[Libreria Led 5](#_Toc536014726)

[Utilizzo 5](#_Toc536014727)

[Hardware 6](#_Toc536014728)

[Software (ogni codice dovrà essere mostrato) 6](#_Toc536014729)

Scopo

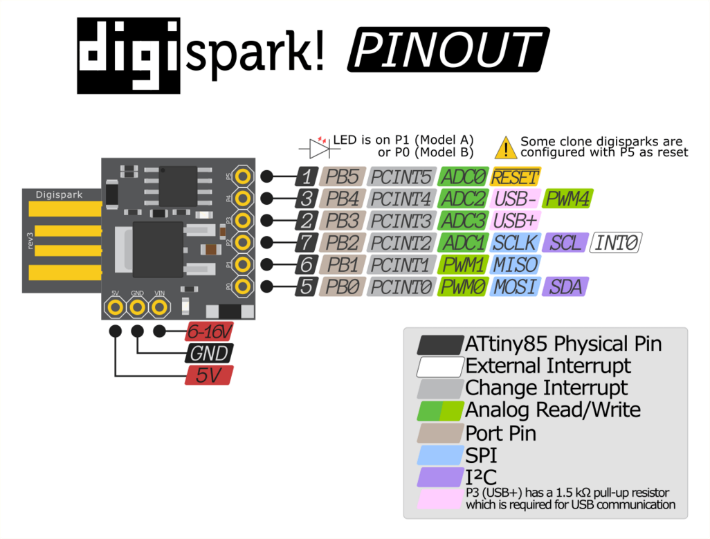
Lo scopo di questa guida è illustrare il funzionamento del circuito in modo che sia facilmente comprensibile anche agli utenti più inesperti. Illustreremo perciò ogni componente utilizzato e il funzionamento di essi singolarmente, così anche per il prodotto globale.

Componenti

## Arduino Digispark

Arduino Digispark, così come tutti gli altri componenti della famiglia Arduino, è una scheda elettronica dotata di un microcontrollore. La funzionalità principale di Arduino è quella di realizzare in maniera pressoché semplice dei dispositivi di controllo oppure degli automatismi (specialmente nel caso di Arduino Digispark). Uno dei punti di forza di Arduino è la sua convenienza economica dal momento che le schede programmabili hanno prezzi veramente bassi (per Digispark meno di 5 CHF) e inoltre il software e il linguaggio di programmazione utilizzato sono Open Source (ossia gratis).

Per collegare elementi esterni alle schede si utilizzando dei pin che possono venir saldati sulle apposite interfacce. L’alimentazione (ossia il +) è indicata da “5V”, mentre la terra (ossia il -) è indicata da “GND”, mentre gli altri pin (da P0 a P5) possono assumere diverse funzionalità seguendo il seguente modello:



Per poter utilizzare il software di Arduino col Digispark sono necessari alcuni accorgimenti, per poter installare le schede è necessaria una connessione a internet (preferibilmente senza proxy):

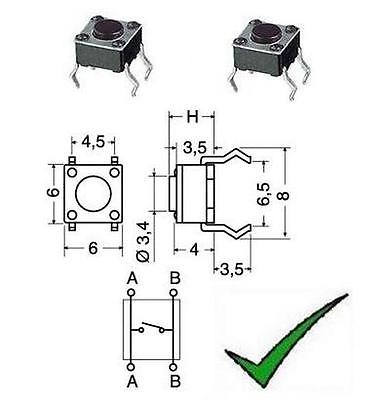
* Nelle impostazioni di arduino (File🡪Impostazioni), nel campo “URL aggiuntive per il Gestore schede:” inserire l’URL <http://digistump.com/package_digistump_index.json>;
* Riavviare il software (se procedendo qualcosa non va, riavviare il pc);
* Cliccando “Gestore schede” (Strumenti 🡪 Scheda) verrà aperta una schermata nella quale è presente una barra di ricerca, scriveteci “Digistump” e verrà mostrata una possibilità come quella da immagine:

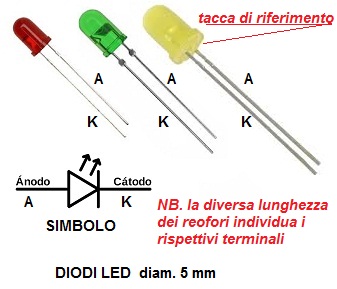
nell’angolo in basso a detra di questa sarà presente il pulsante Installa (premerlo);

* Riavviare il software (se procedendo qualcosa non va, riavviare il pc);
* Nella selezione delle schede cercare e selezionare “Digispark (Default – 16.5 MHz)”;
* Per quanto riguarda la selezione della porta (COM…) dipende dal vostro computer e da quale porta usb utilizzerete per inserire il digispark.

## Sensore a Ultrasuoni

Un pulsante si comporta come se fosse un cavo che viene collegato e scollegato. La funzione corrispondente al fatto che è collegato, sarebbe quando viene premuto il pulsante, mentre quando viene rilasciato il circuito viene aperto (e quindi scollegato).

Esistono numerosi tipi diversi di pulsanti, ma quelli più comuni e più utilizzati sono quelli a 4 pin come quello mostrato nelle foto. I pin sono collegati a coppie e perciò per collegarli in modo da rilevare la pressione del pulsante bisogna seguire lo schema a sinistra (collegare un polo A, con un polo B).

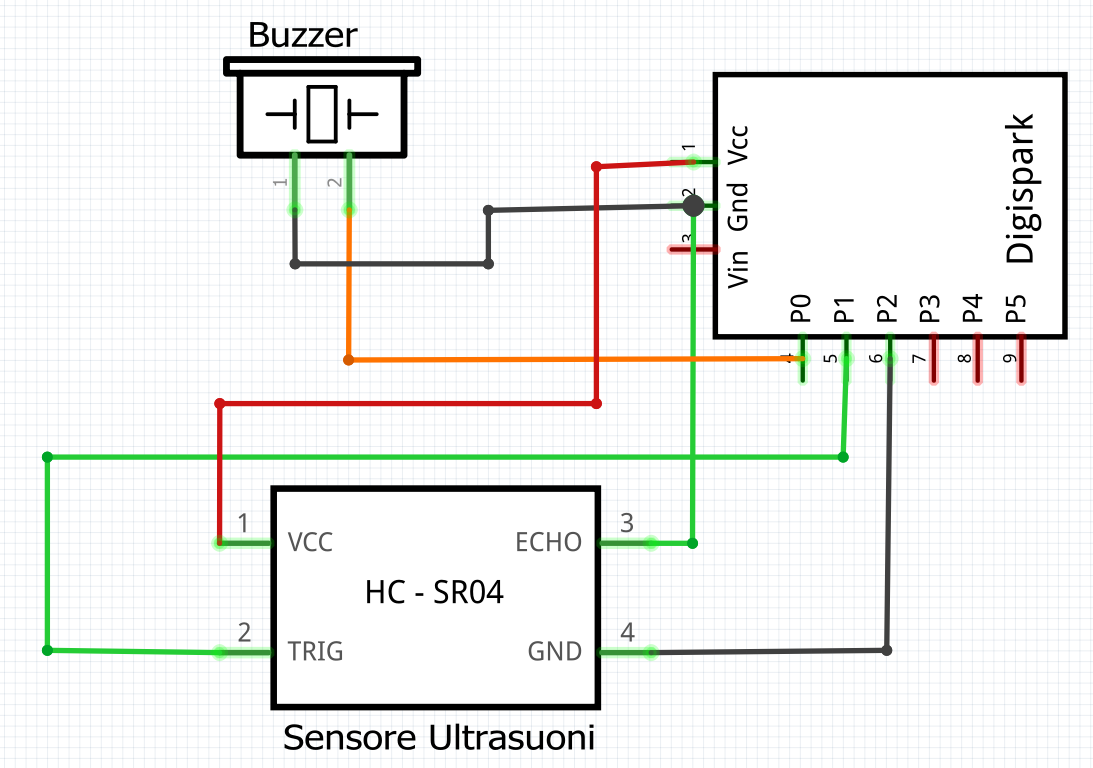
I pulsanti non hanno nessun circuito sensibile al loro interno, quindi non è necessario prestare attenzione ai voltaggi che gli vengono impressi o alla loro polarità. Questo perché, come già detto, i pulsanti sono esattamente come se fossero due cavi che vengono collegati e scollegati a seconda del fatto che sia stato premuto o meno il pulsante.

## Cicalino

Led non possono essere collegati direttamente al polo positivo o negativo della corrente perché subirebbero un voltaggio troppo alto rispetto a quello supportato, per questo dobbiamo utilizzare delle resistenze. Il minimo per il led che utilizziamo noi è una resistenza da 330 Ω.

Per distinguere il pin positivo e quello negativo è sufficiente guardare la lunghezza del suddetto pin e la posizione della tacca di riferimento (Vedi immagine a fianco). Il pin più lungo rappresenta il polo positivo, quindi il più corto quello negativo. Il polo positivo è identificabile anche dalla presenza della tacca.

Schema Elettrico



Librerie

*Tutte le librerie realizzate per questo progetto sono state realizzate nel linguaggio di C++, come d’altronde anche il software di Arduino.*

Tutte le nostre librerie sono composte da un’interfaccia (chiamata nel linguaggio specifico di “C” *Header*, ed è un file con estensione “.h”) e la libreria in sé (con estensione “.cpp”) che estende l’interfaccia.

Sia l’Header, che la libreria devono includere l’interfaccia “Arduino.h”.

## Libreria Buzzer (Cicalino)

L’Header contiene:

* 1 attributo: corrisponde al pin al quale è collegato il cicalino;
* 3 metodi:
  + setPinBuzzer(int buzzerPort);
  + setTone(int frequency);
  + powerOff();

La libreria contiene tutti i metodi dell’Header dichiarandoli in questo modo:

void <NomeHeader>::<metodo>(<tipoParametro> <parametro>){};

Ecco cosa fa nello specifico ogni metodo:

* **setPinBuzzer**: in base al parametro che riceve, attribuisce tale valore all’attributo che indica il pin del cicalino;
* **setTone**: fa emettere al cicalino dei suoni a seconda della frequenza ricevuta come parametro;
* **powerOff**: spegne il cicalino.

## Libreria UltraSound (Sensore a Ultrasuoni)

L’Header contiene:

* 2 attributi: corrispondenti ai pin di emissione e ricezione del sensore;
* 3 metodi:
  + setUltraSoundPin(int pinTrigPort, int pinEchoPort);
  + getDistance();

La libreria contiene tutti i metodi dell’Header dichiarandoli in questo modo:

void <NomeHeader>::<metodo>(<tipoParametro> <parametro>){};

Ecco cosa fa nello specifico ogni metodo:

* **setUltraSoundPin**: in base ai parametri che riceve, attribuisce tali valori agli attributi che indicano i pin di emissione e ricezione del sensore;
* **getDistance**: ritorna la distanza misurata dal sensore in cm;

Utilizzo

# Hardware

I componenti da utilizzare per questo progetto sono i 3 citati più e più volte all’interno di questa guida:

* Arduino Digispark;
* Sensore a Ultrasuoni;
* Cicalino (o Buzzer);

Per costruire il circuito dobbiamo fissare il sensore e il cicalino su di una breadboard (circuito provvisorio) o su una veroboard (circuito definitivo), non è necessario metterli in uno schema preciso a patto che abbia un senso.

 **Fare attenzione alle piste delle board per evitare cortocircuiti**  ***(In caso di dubbio, eccoti un’immagine che mostra com’è fatta una breadboard di Arduino al suo interno)*** 

Per costruire il circuito dobbiamo fissare il pulsante in modo che le 2 coppie di pin non siano in contatto fra loro. Il led allo stesso modo può essere montato in qualunque modo a patto che i 2 pin non siano connessi.

Per costruire il circuito dobbiamo fissare il pulsante in modo che le 2 coppie di pin non siano in contatto fra loro. Il led allo stesso modo può essere montato in qualunque modo a patto che i 2 pin non siano connessi.

Per costruire il circuito dobbiamo fissare il pulsante in modo che le 2 coppie di pin non siano in contatto fra loro. Il led allo stesso modo può essere montato in qualunque modo a patto che i 2 pin non siano connessi.

# Software

* Il primo esempio di codice che abbiamo realizzato con questi componenti si occupa di suonare sempre più velocemente il cicalino man mano che la distanza registrata diminuisce.

Le prime linee di codice servono ad includere le librerie necessarie al programma (solo quelle che abbiamo creato noi perché quella base “Arduino.h” viene inclusa automaticamente):

|  |
| --- |
| #include <LibraryUltraSound.h> #include <LibraryBuzzer.h> |

Le linee successive le utilizziamo per dichiarare le istanze delle nostre librerie:

|  |
| --- |
| LibraryUltraSound libraryUltraSound;  LibraryBuzzer libraryBuzzer; |

Nel metodo setup richiamiamo i metodi che abbiamo creato nelle librerie che si occupano di attribuire ai pin di emissione e ricezione del sensore a ultrasuoni e al pin del Buzzer un pin di Digispark:

|  |
| --- |
| void setup() {  libraryBuzzer.setPinBuzzer(0);  libraryUltraSound.setUltraSoundPin(1, 2);  } |

Adesso entriamo nella parte più sostanziosa del programma: il metodo loop.

La prima cosa che facciamo nel loop è memorizzare la distanza registrata dal sensore:

|  |
| --- |
| int distance = libraryUltraSound.getDistance(); |

A questo punto, dopo aver controllato che la distanza registrata sia minore di 1 metro, mappiamo tale distanza in modo da ottenere l’intervallo fra un suono e l’altro del cicalino. Al termine di tutto impostiamo un delay di 50 millisecondi per evitare interferenze:

|  |
| --- |
| if(distance <= 100){  int risultato = map(distance, 3, 100, 20, 500);  libraryBuzzer.setTone(100);  delay(risultato);  libraryBuzzer.powerOff();  }  delay(50); |

* Il nostro secondo esempio di codice invece utilizza la frequenza, ossia la distanza letta dal sensore e la frequenza del cicalino sono direttamente proporzionali.

Le prime linee di codice servono ad includere le librerie necessarie al programma (solo quelle che abbiamo creato noi perché quella base “Arduino.h” viene inclusa automaticamente):

|  |
| --- |
| #include <LibraryUltraSound.h> #include <LibraryBuzzer.h> |

Le linee successive le utilizziamo per dichiarare le istanze delle nostre librerie:

|  |
| --- |
| LibraryUltraSound libraryUltraSound;  LibraryBuzzer libraryBuzzer; |

Nel metodo setup richiamiamo i metodi che abbiamo creato nelle librerie che si occupano di attribuire ai pin di emissione e ricezione del sensore a ultrasuoni e al pin del Buzzer un pin di Digispark:

|  |
| --- |
| void setup() {  libraryBuzzer.setPinBuzzer(0);  libraryUltraSound.setUltraSoundPin(1, 2);  } |

Adesso entriamo nella parte più sostanziosa del programma: il metodo loop.

La prima cosa che facciamo nel loop è memorizzare la distanza registrata dal sensore:

|  |
| --- |
| int distance = libraryUltraSound.getDistance(); |

A questo punto mappiamo tale distanza in modo da ottenere la frequenza dei suoni con un delay di 100 millisecondi fra un suono e l’altro:

|  |
| --- |
| int distance = libraryUltraSound.getDistance();  int frequenza = map(distance, 0, 100, 0, 1000);  libraryBuzzer.setTone(frequenza);  delay(100); |

* Il terzo esempio che abbiamo pensato fa in modo che il cicalino cominci a suonare a partire da una certa distanza.

Le prime linee di codice servono ad includere le librerie necessarie al programma (solo quelle che abbiamo creato noi perché quella base “Arduino.h” viene inclusa automaticamente):

|  |
| --- |
| #include <LibraryUltraSound.h> #include <LibraryBuzzer.h> |

Le linee successive le utilizziamo per dichiarare le istanze delle nostre librerie:

|  |
| --- |
| LibraryUltraSound libraryUltraSound;  LibraryBuzzer libraryBuzzer; |

Nel metodo setup richiamiamo i metodi che abbiamo creato nelle librerie che si occupano di attribuire ai pin di emissione e ricezione del sensore a ultrasuoni e al pin del Buzzer un pin di Digispark:

|  |
| --- |
| void setup() {  libraryBuzzer.setPinBuzzer(0);  libraryUltraSound.setUltraSoundPin(1, 2);  } |

Adesso entriamo nella parte più sostanziosa del programma: il metodo loop.

La prima cosa che facciamo nel loop è memorizzare la distanza registrata dal sensore:

|  |
| --- |
| int distance = libraryUltraSound.getDistance(); |

Se la distanza registrata è inferiore (o pari) a 20, il buzzer comincia a suonare pressoché ininterrottamente (intervallo di 10 millisecondi). In caso contrario, si spegne:

|  |
| --- |
| if(distance <= 20){  libraryBuzzer.setTone(100);  delay(10);  }else{  libraryBuzzer.powerOff();  } |