
BE MAKER



BE MAKER

ELETTRONICA, ROBOTICA E CODING PER RAGAZZI... E NON SOLO !

**COURSE OF ELECTRONICS, ROBOTICS AND CODING FOR
CHILDREN... and not only!**

BASIC COURSE – lesson 5

Index

Warnings 3

Notes sul Copyright 3

THE SOUND 4

CURIOSITY: What is there sound? 4

Project 16 – The Vespa Teresa and the active buzzer. 8

The Passive Buzzer 11

Project 17 – The passive buzzer and the Piano-Man. 13

 Analysis of the sketch of Project 17 – The passive buzzer and the Piano-Man. 16

CURIOSITY: The Microphone and the Loudspeaker or Loudspeaker 21

Project 18 – What happens in that house? We use the microphone..... 22

Warnings

With regard to the safety aspects, since the projects are based on a very low voltage power supply supplied by the USB port of the PC or by support batteries or power supplies with a maximum of 9V output, there are no particular risks of an electrical nature. It is however necessary to specify that any short circuits caused during the exercise phase could produce damage to the PC, to the furnishings and in extreme cases even to burns, for this reason every time a circuit is assembled, or changes are made on it, it will be necessary to do so in the absence of power and at the end of the exercise it will be necessary to provide for the disconnection of the circuit by removing both the USB cable connecting to the PC and any batteries from the appropriate compartments or external power connectors. In addition, always for safety reasons, it is strongly recommended to carry out projects on insulating and heat-resistant carpets that can be purchased in any electronics store or even on specialized websites.

At the end of the exercises it is advisable to wash your hands, as the electronic components could have processing residues that could cause damage if ingested or if in contact with eyes, mouth, skin, etc. Although the individual projects have been tested and safe, those who decide to follow what is reported in this document, assume full responsibility for what could happen in the execution of the exercises provided for in the same. For younger children and / or the first experiences in the field of Electronics, it is advisable to perform the exercises with the help and in the presence of an adult.

Notes sul Copyright

All trademarks are the property of their respective owners; third-party trademarks, product names, trade names, corporate names and companies mentioned may be trademarks owned by their respective owners or registered trademarks of other companies and have been used for purely explanatory purposes and for the benefit of the owner, without any purpose of violation of the copyright rights in force. What is reported in this document is the property of Roberto Francavilla, Italian and European laws on copyright are applicable to it – any texts taken from other sources are also protected by the Copyright and property of the respective Owners. All the information and contents (texts, graphics and images, etc.) reported are, to the best of my knowledge, in the public domain. If, unintentionally, material subject to copyright or in violation of the law has been published, please notify info@bemaker.org by email and I will promptly remove it.

Roberto Francavilla

THE SOUND



We dedicate this lesson to the study of Sound, or rather to the generation and propagation of sound waves and the related Arduino sensors and modules that exploit this phenomenon to manage events.

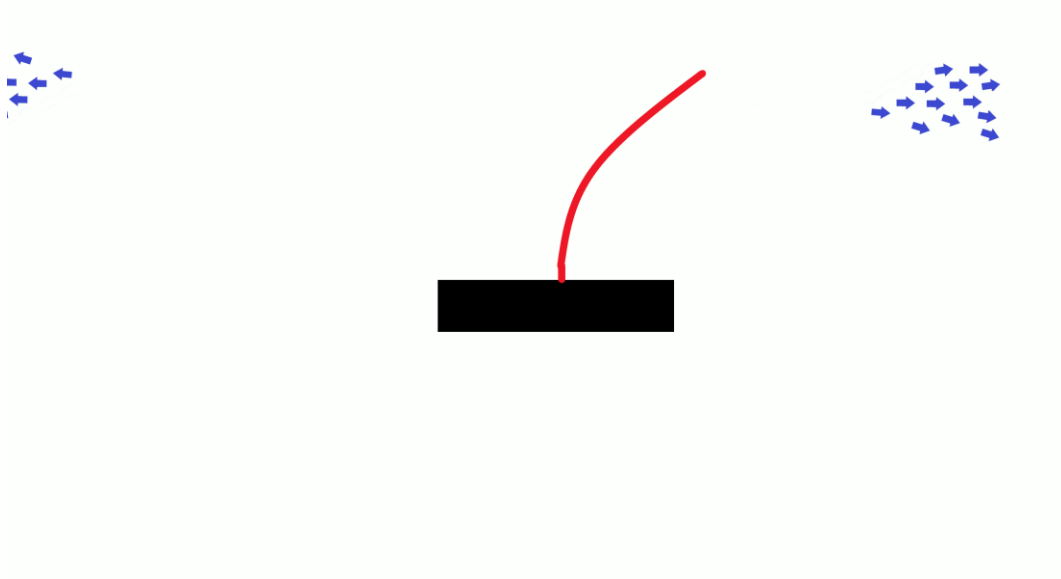
CURIOSITY: What is there sound?

Suppose you drop a stone into the water of a pond, you will see that from the point of fall of the stone waves expand, these waves move away from the point of fall and propagate in concentric circles.



If we vibrate a body, these vibrations of the body are transferred to what surrounds the body itself (for example: air, water, etc.) and this phenomenon occurs thanks to the action of expansion / compression of the molecules of the air or water in which the body that is vibrating is immersed. This is how you have the generation of sound.




The sound produced by the vibration of a body propagates through waves and in this regard, to give visibility to this phenomenon, we have given the example of a stone dropped into the water of a pond.



In the animated image above, the phenomenon of compression/expansion is shown with the example of a foil vibrating in the air; the light blue arrows indicate the air sucked into the depression zone created by the movement of the foil. The blue arrows indicate the overpressure created by the same movement in the opposite direction. The vibration, therefore, compresses and expands the air that surrounds the body, this compression and expansion is transmitted to the surrounding air thus producing waves called "sound waves" (sound: because at certain frequencies, to simplify ... at a certain speed of vibration, these waves can also be heard). In fact, these vibrations are then picked up by the ear that inside, thanks to the presence of the "eardrum" and thanks to the presence of "ossicles", transfer and transform the vibrations captured into electrical impulses that our brain translates into a sensation called sound.

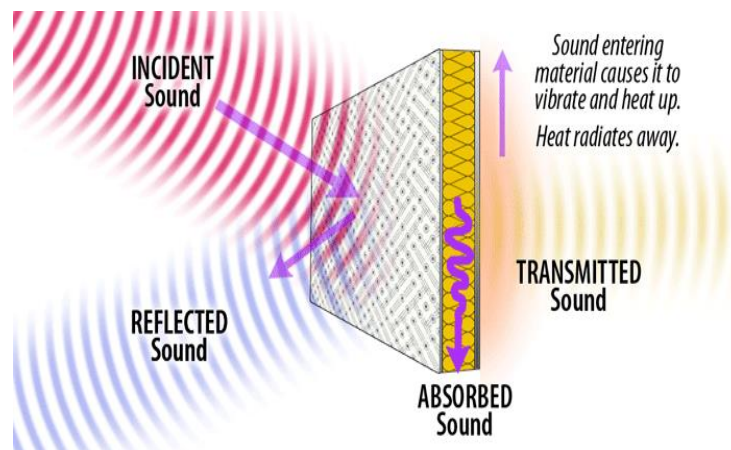


So **sound is a sensation** that we receive and comes from a vibration of a body. If such a feeling is pleasant, then we call it "sound", otherwise it is classified as "noise".
 Very important in the propagation of sound is the medium in which it propagates that also determines its speed.

		
<p>In water the speed is 1480 m/s. That is, in a second the sound wave travels about 1.5 km.</p>	<p>In the air, since the molecules are more rarefied (they are more distant) the speed of propagation of sound is about 340 m / s, that is, in a second the sound travels 340 m.</p>	<p>In iron the propagation speed is 5130 m/s, that is, in one second the sound travels about 5 km. This is because the crystalline lenses are much closer to each other and therefore the transmission of vibration induced by the compression / expansion action is more immediate.</p>

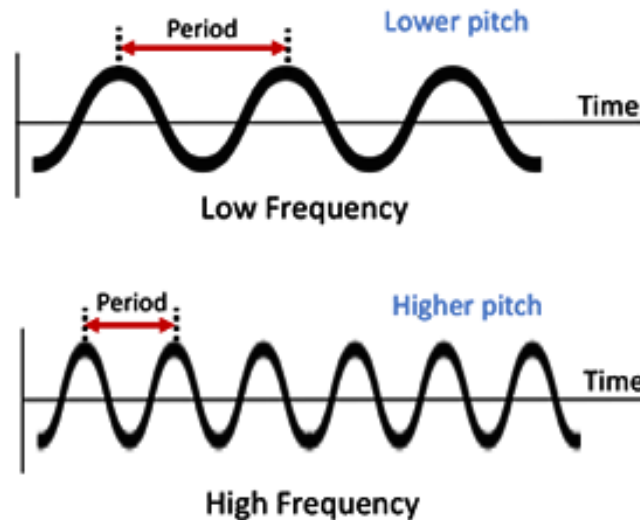
Is it in a vacuum? Obviously in a vacuum, since there is no compression / expansion of molecules, no sound propagates.

The sound wave in its path can encounter an obstacle, usually a more "dense" body. For example, a sound wave that is propagating in the air, when it meets a rock wall, is reflected on it and a "reflected wave" is generated and a part of the sound is also reproduced in the rock.

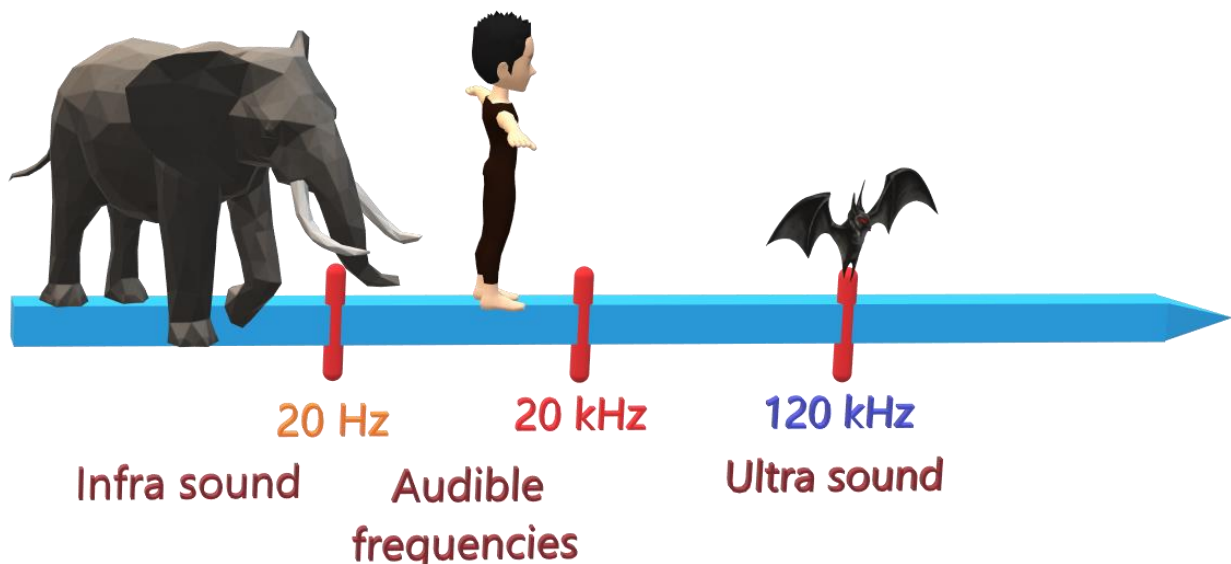


When the reflected wave returns in the same direction as the source, there is the phenomenon that we call "**echo**".

The frequency, which is measured in Hz (pronounced erz), or the proximity between the sound waves produced (or rather: the number of sound waves transmitted in a second), classifies the sound into: grave sounds (what we call "bass") or high-pitched sounds (which we call "highs").

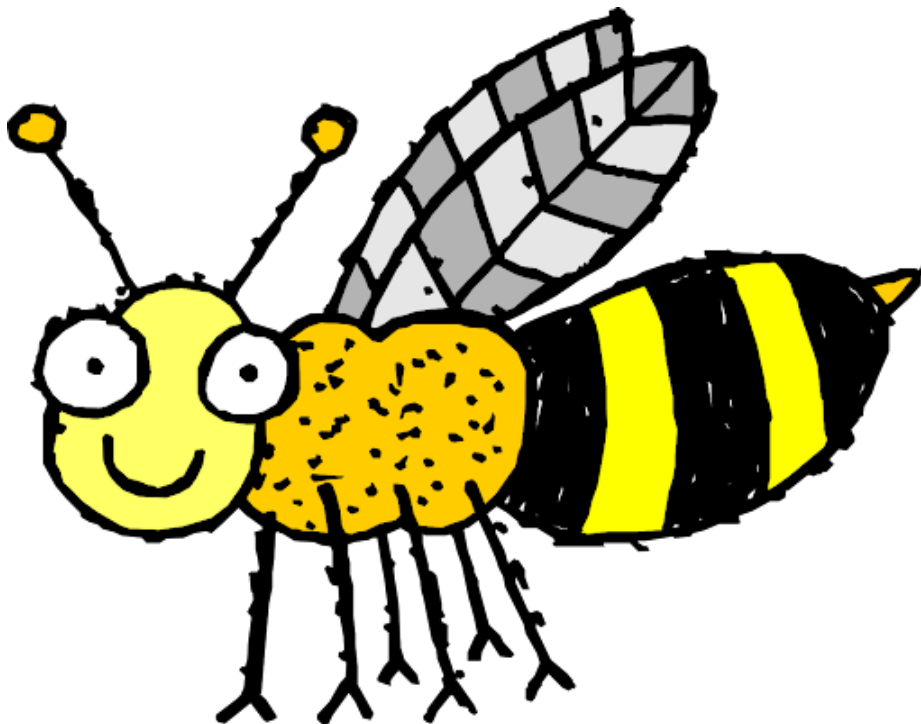


Only certain sound frequencies are audible to the human ear, those that have a frequency between 20 Hz and 20,000 Hz (or even written 20 kHz). Elephants can perceive sounds even below 20 Hz, while bats can perceive sounds of up to 120 kHz.


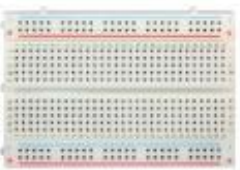



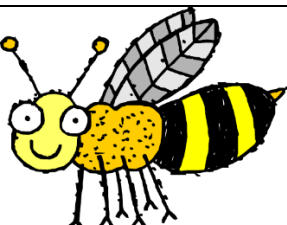


After the theoretical part, which gave us the opportunity to learn more about the sound, we move on to the practical part with a nice project, very funny.

Project 16 – The Vespa Teresa and the active buzzer.



For this project we need:

			
Arduino Uno R3 or compatible	Breadboard	Dupont cables male - male	USB connection cable
			
Active Buzzer	Printing the drawing		

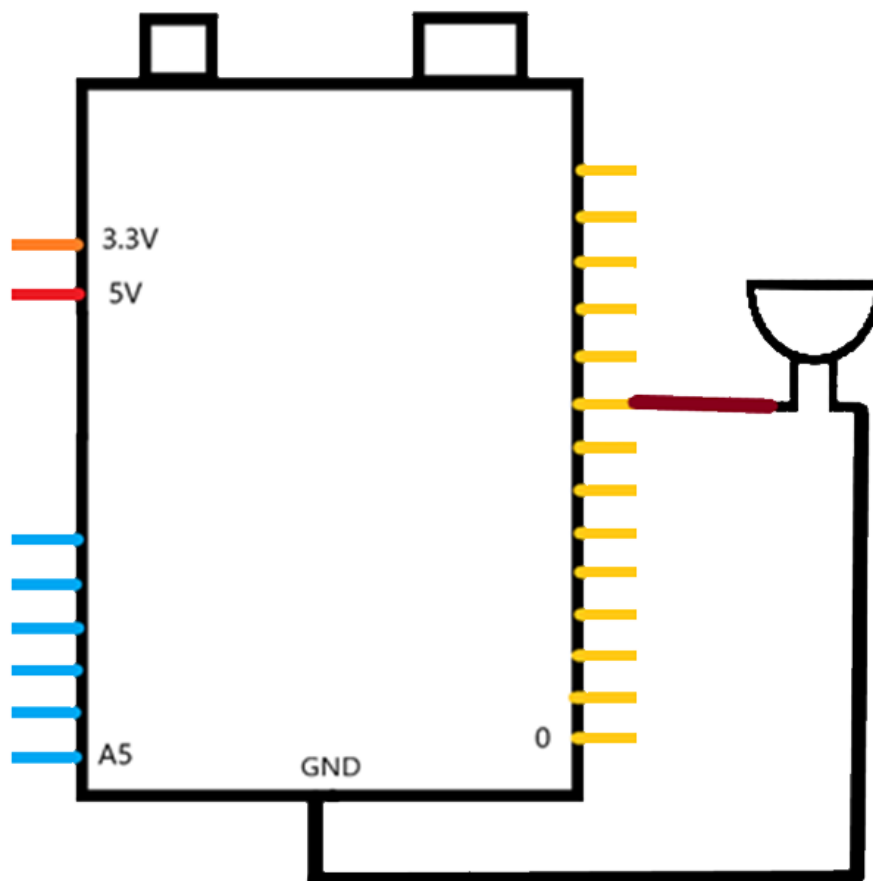
Before moving on to mounting the circuit it is good to know that the active buzzer has two PINs, one of which is longer. The longest one is the positive PIN " + ", the shortest one is the negative " — ". Generally the passive Buzzer also has a sticker where the positive PIN is indicated.



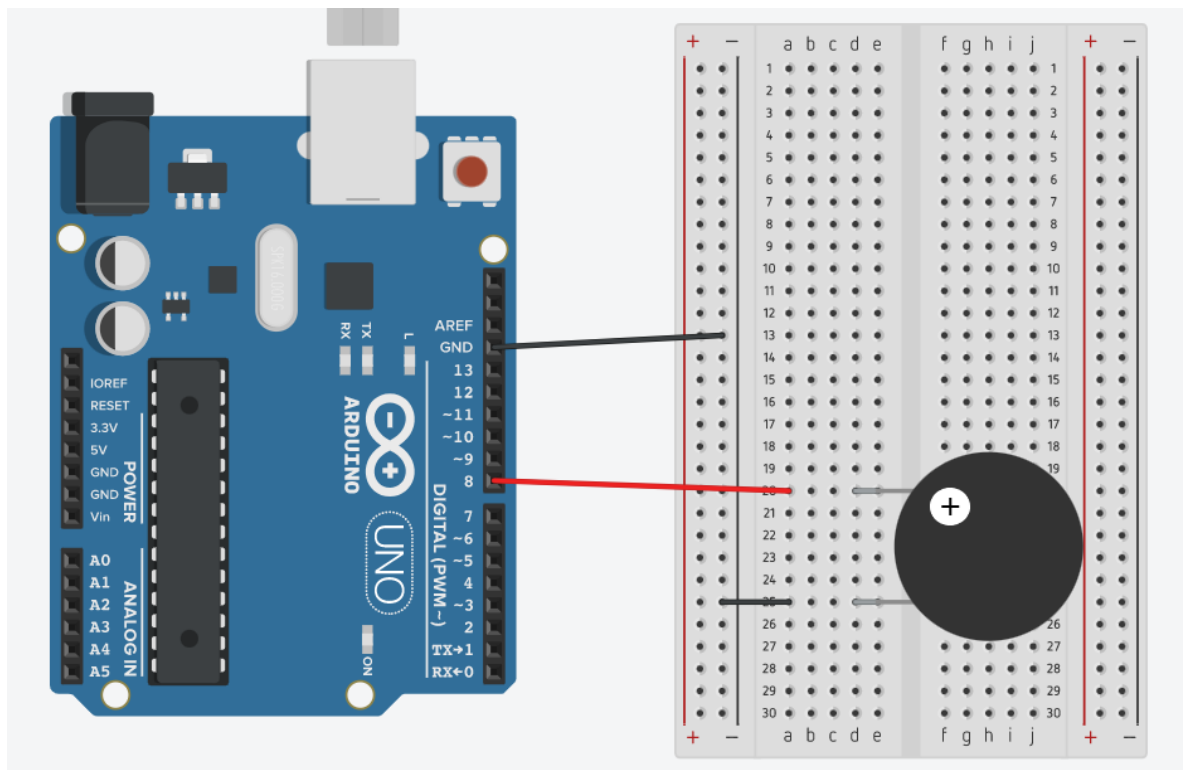
The electric symbol of the active buzzer is:



The wiring diagram to be made is as follows:



The mounting scheme to be made is as follows:



Note: The positive PIN of the active buzzer is linked to the Arduino digital PIN 8, while the negative PIN to the GND.

Once the circuit is mounted (and the drawing is placed on the buzzer), the sketch to be written and loaded on Arduino is:

```
/*
BE MAKER - STEM - Project n. 16 - Vespa Teresa
*/
int buzzerPin = 8; //I assign the PIN 8 the positive PIN of the buzzer

void setup ()
{
    pinMode (buzzerPin, OUTPUT); //I declare in OUTPUT pin 8
}
void loop ()
{
    digitalWrite (buzzerPin, HIGH); //feed pin 8 to 5 V
    delay (500); //I leave PIN 8 powered for half a second
    digitalWrite (buzzerPin, LOW); //I turn off PIN 8

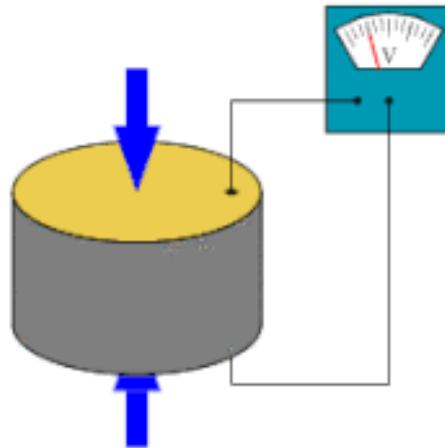
    delay (500); //I leave PIN 8 off for half a second
}
```

Once the sketch is loaded, the buzzer will sound intermittently, you can change the time of on/off of the buzzer, going to change the time indicated in the delay of the sketch.

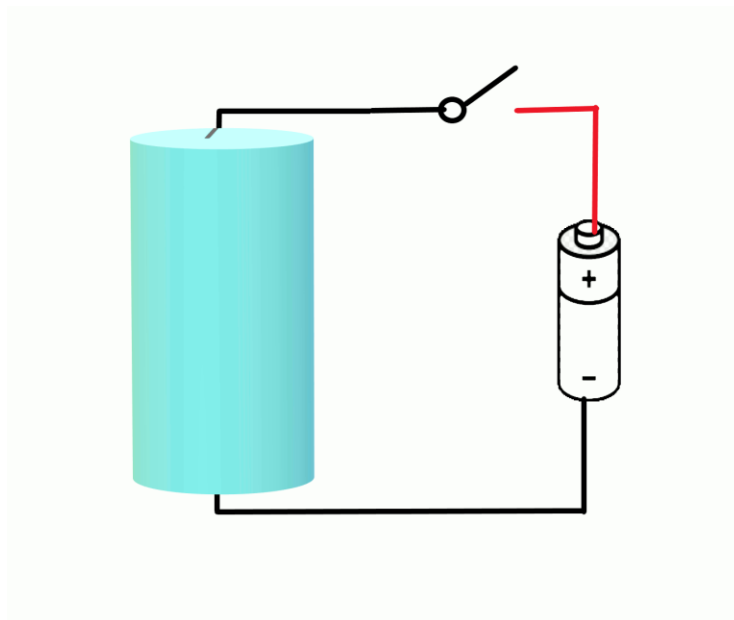
[For the video-project click [here](#)]

The Passive Buzzer

The Buzzer is a piezoelectric device. Piezoelectricity is the property of some crystalline materials to polarize generating a potential difference when they are subjected to mechanical deformation and at the same time to deform elastically when subjected to an electrical voltage.



The GIF above shows how a Deformation of the piezoelectric material generates a Voltage



The GIF above shows how a Voltage applied to the piezoelectric generates a Deformation

Buzzers can be **active**, if they have inside them an oscillator (a quartz with a predetermined and fixed frequency) that once powered by electric voltage stresses the piezoelectric material at a certain frequency and therefore, this vibration of the piezoelectric material (usually a membrane), produces the sound at a fixed frequency (this is the so-called Monotonous Sound).

Passive Buzzers, not having an oscillator, in order to produce a sound, need a variable supply voltage over time. Then the variable voltage with a certain frequency is given from the outside. To do this we can take advantage of the ability of our Arduino to produce a voltage variable over time thanks to PWM (Pulse Width Modulation), that is, a Voltage with a width of modulated pulses. But at this point the question arises spontaneously, what if we vary the frequency? You will get different sounds and at certain frequencies, you can also play musical notes. So, the conclusion is that with an Arduino and a passive Buzzer we can play melodies.



The buzzer is used in alarms, countdowns, for the confirmation of keyboard input, etc ...

The Passive Buzzer is the one shown in the picture below:




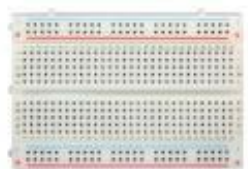




In the table there is the correspondence between the main notes and the frequency.

NOME DELLA NOTA	SIGLA DELLA NOTA	FREQUENZA
DO4	C4	262 Hz
DO#4	C#4	277 Hz
RE4	D4	294 Hz
RE#4	D#4	311 Hz
MI4	E4	330 Hz
FA4	F4	349 Hz
FA#4	F#4	370 Hz
SOL4	G4	392 Hz
SOL#4	G#4	415 Hz
LA4	A4	440 Hz
LA#4	A#4	466 Hz
SI4	B4	494 Hz

Project 17 – The passive buzzer and the Piano-Man.



For this project we need:

			
Arduino Uno R3 or compatible	Breadboard	Dupont cables male - male	USB connection cable
			
Passive buzzer	220 Ohm resistance		

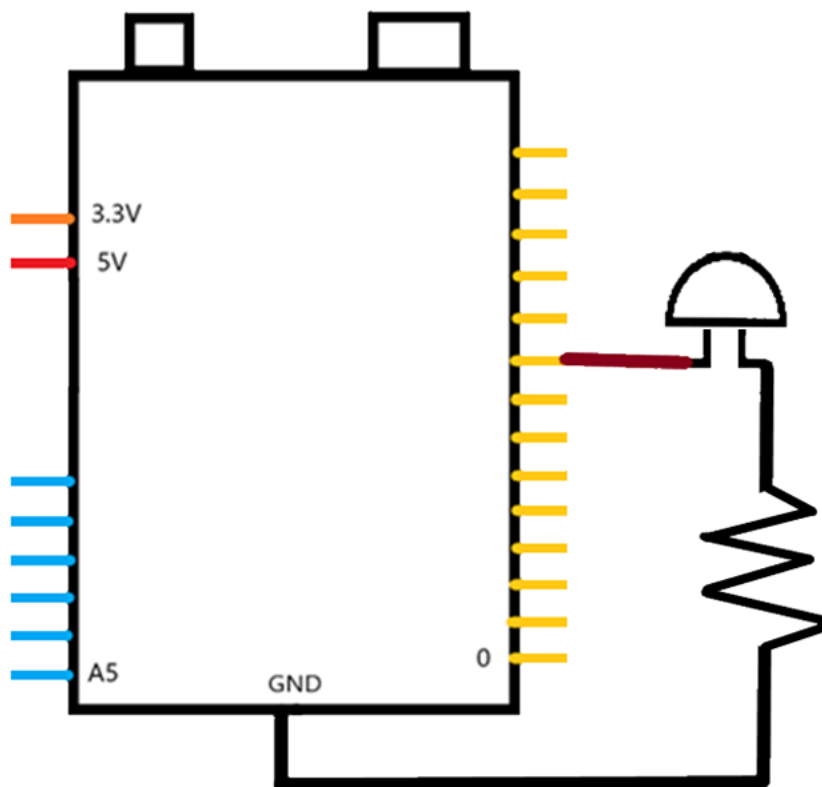
Before moving on to the assembly of the circuit it is good to know that the passive buzzer has a low resistance, that's why it is used to put in series to the passive buzzer a resistance of 220 Ohms, moreover, since we will feed the buzzer with a "square half-wave", to have a cleaner

emitted sound, it would be useful to couple to the buzzer a filter (basically a capacitor), but these aspects, concerning applied electronics,. we will see them in a dedicated course, for the moment, let's limit ourselves to inserting a resistance and for our educational purposes.

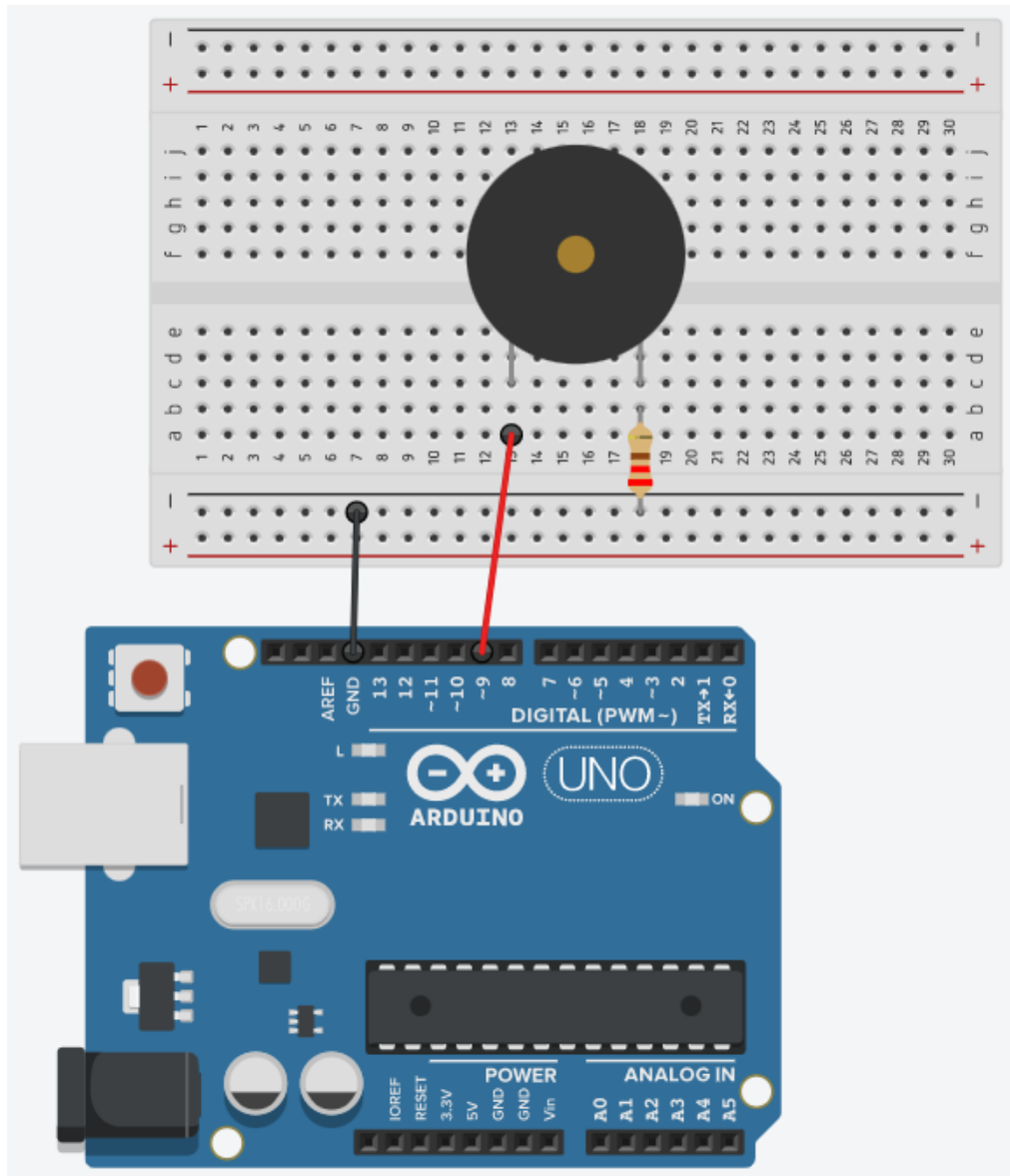
The electric symbol of the passive buzzer is:



The wiring diagram to be made is as follows:



The mounting scheme to be made is as follows:



Once the circuit is mounted, the sketch to be written and loaded on Arduino is:

```
/*
Be Maker STEM - Passive Buzzer Piano_Man
*/

#define BUZZER_PIN 9 //connect the Buzzer to PIN 9

#define NOTE_C5 523
#define NOTE_D5 587
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_G5 784
#define NOTE_A5 880
#define NOTE_B5 988
#define NOTE_C6 1047
```

You construct a vector (variable that contains multiple values) with the Notes:

```
unsigned int melody[] = {  
  NOTE_C5, NOTE_D5, NOTE_E5, NOTE_F5, NOTE_G5, NOTE_A5, NOTE_B5, NOTE_C6};
```

```
unsigned int duration = 500; Duration of the note in milliseconds
```

```
void setup() {  
  pinMode(BUZZER_PIN, OUTPUT); We set the PIN 9 output  
}
```

```
void loop() {  
  for (int thisNote = 0; thisNote < 8; thisNote++) {  
    tone(BUZZER_PIN, melody[thisNote], duration); see more  
    delay(1000); //Changes note after one second  
  }  
  delay(2000); //After two seconds the cycle starts again  
}
```

Once the sketch is loaded, the buzzer will play a melody formed by the Note scale.

[For the video-project click [here](#)]

Analysis of the sketch of Project 17 – The passive buzzer and the Piano-Man.

In the sketch you can see that to activate the passive Buzzer, the instruction to be given is "tone" with the following syntax:

tone (num_pin, frequenza_note, durata_note);

This statement generates a PWM signal with the specified frequency and with the same frequency maintaining the same value as indicated.

While for the activation of the active Buzzer it was sufficient to put the Buzzer PIN in a high state with a simple instruction of

digitalWrite (buzzerPin, HIGH);

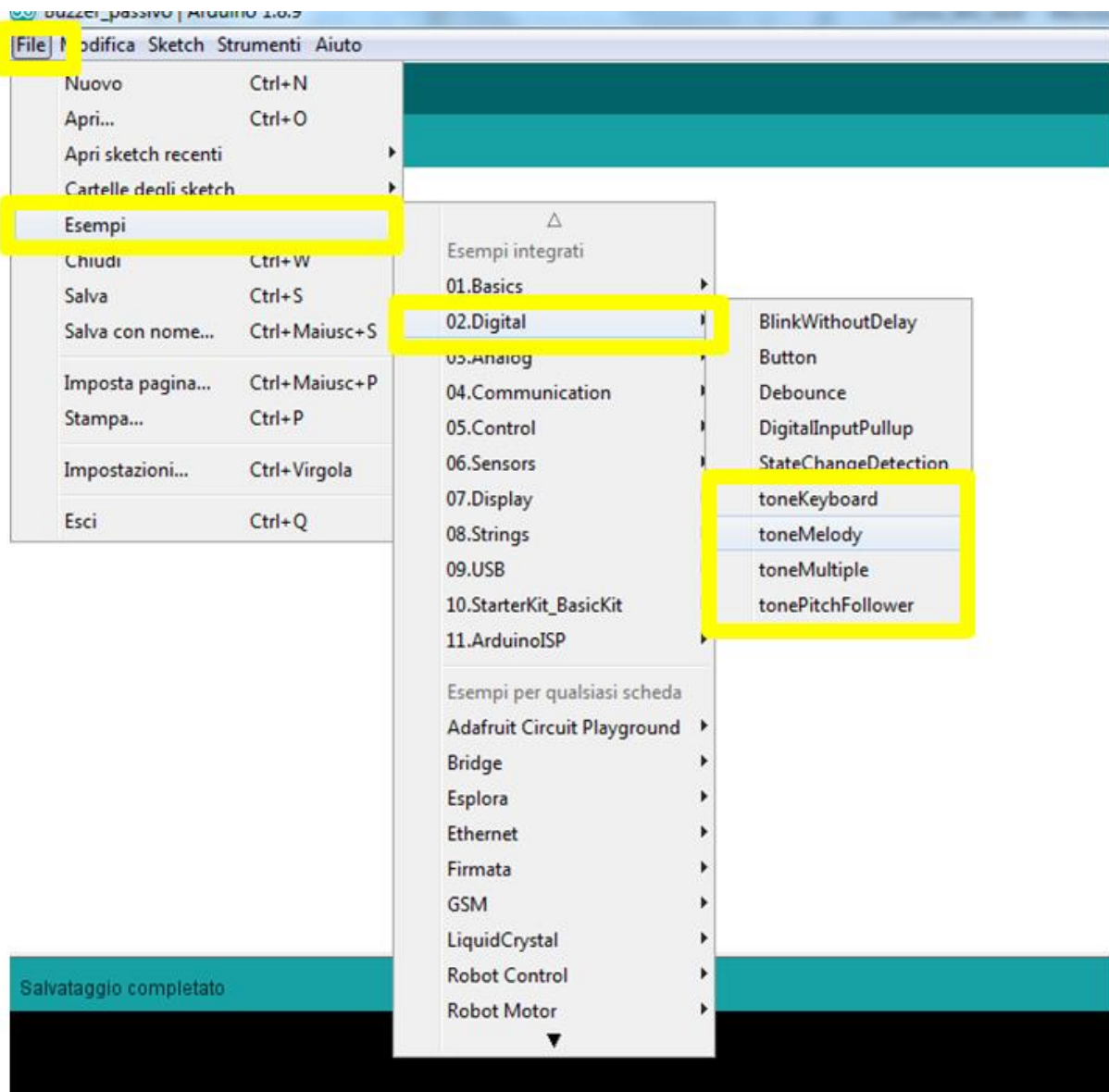
and to turn it off, of course:

digitalWrite (buzzerPin, LOW);

Another thing I want to point out is the definition of the vector, that is, that particular variable that we have called "melody", which as you can see contains more values. The individual values are placed neatly from position 0 to position "n-1". They can be invoked by writing in the sketch the name of the variable and inside the square brackets, the position of the value we want to read:

melody [thisNote]

It is possible to experiment with new melodies and practice with the Tone statement, using the examples in the IDE. To open them:



You can also try this melody that I found on the internet (just copy-paste):

```
/*
Be Maker STEM - melody
*/
```

```
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
```

```
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
```

```

#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
REST #define 0
int time=114; change this to make the song slower or faster
int buzzer = 9;// change this to whichever pin you want to use
notes of the melody followed by the duration.
a 4 means a quarter note, 8 an eighteenth , 16 sixteenth, so on
// !! negative numbers are used to represent dotted notes,
so -4 means a dotted quarter note, that is, a quarter Plus an eighteenth!!
int melody[] = {
    NOTE_E4.4, NOTE_E4.4, NOTE_F4.4, NOTE_G4.4,/1
    NOTE_G4.4, NOTE_F4.4, NOTE_E4.4, NOTE_D4.4,
    NOTE_C4.4, NOTE_C4.4, NOTE_D4.4, NOTE_E4.4,
    NOTE_E4,-4, NOTE_D4,8, NOTE_D4,2,
    NOTE_E4.4, NOTE_E4.4, NOTE_F4.4, NOTE_G4.4,/4
    NOTE_G4.4, NOTE_F4.4, NOTE_E4.4, NOTE_D4.4,
    NOTE_C4.4, NOTE_C4.4, NOTE_D4.4, NOTE_E4.4,
    NOTE_D4,-4, NOTE_C4,8, NOTE_C4,2,
    NOTE_D4.4, NOTE_D4.4, NOTE_E4.4, NOTE_C4.4,/8
    NOTE_D4.4, NOTE_E4,8, NOTE_F4,8, NOTE_E4,4, NOTE_C4,4,
    NOTE_D4.4, NOTE_E4,8, NOTE_F4,8, NOTE_E4.4, NOTE_D4.4,
    NOTE_C4,4, NOTE_D4,4, NOTE_G3,2,
    NOTE_E4.4, NOTE_E4.4, NOTE_F4.4, NOTE_G4.4,//12
    NOTE_G4.4, NOTE_F4.4, NOTE_E4.4, NOTE_D4.4,
    NOTE_C4,4, NOTE_C4,4, NOTE_D4,4, NOTE_E4,4,
    NOTE_D4,-4, NOTE_C4,8, NOTE_C4,2
};
sizeof gives the number of bytes, each int value is composed of two bytes (16 bits)
there are two values per note (pitch and duration), so for each note there are four bytes

```

```

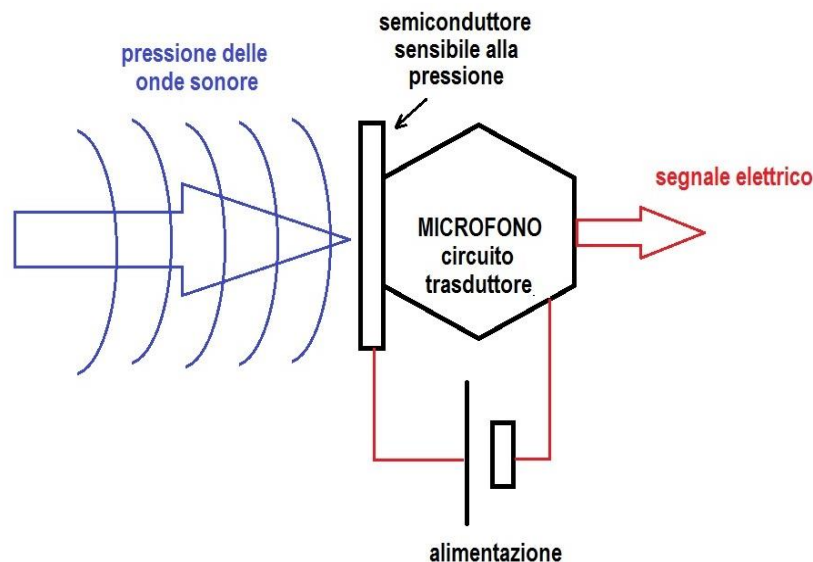
int notes=sizeof(melody)/sizeof(melody[0])/2;
this calculates the duration of a whole note in ms (60s/tempo)*4 beats
int wholenote = (60000 * 4) / time;
int divider = 0, noteDuration = 0;
void setup() {
  iterate over the notes of the melody.
  Remember, the array is twice the number of notes (notes + durations)
  for (int thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {
    calculates the duration of each note
    divider = melody[thisNote + 1];
    if (divider > 0) {
      noteDuration = (wholenote) / divider; regular note, just proceed
    } else if (divider < 0) {
      dotted notes are represented with negative durations!!
      noteDuration = (wholenote) / abs(divider);
      noteDuration *= 1.5; increases the duration in half for dotted notes
    }
    we only play the note for 90% of the duration, leaving 10% as a pause
    tone(buzzer, melody[thisNote], noteDuration*0.9);
    Wait for the specief duration before playing the next note.
    delay(noteDuration);
    noTone(buzzer); stop the waveform generation before the next note.
  }
}
void loop() {
  if you want to repeat the song forever, paste the setup code here .
}

```

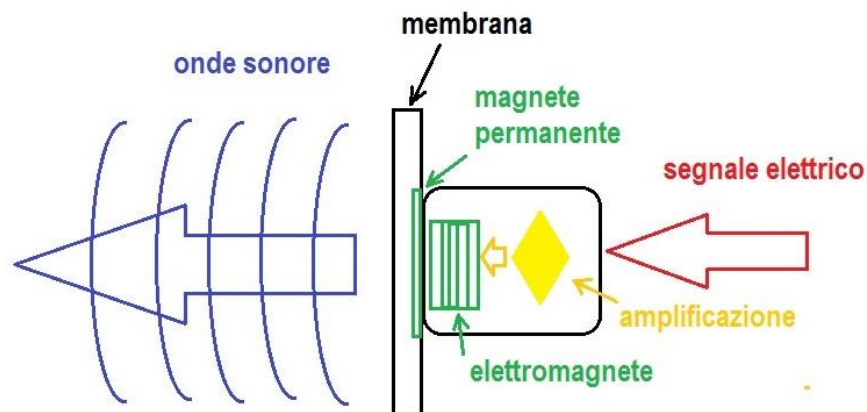

CURIOSITY: The Microphone and the Loudspeaker or Loudspeaker

At this point, with the knowledge gained on Sound and Piezoelectric materials, we are able to understand the operation of the microphone quite well.

The microphone is nothing more than a "Transducer" of the pressure that sound waves exert on a membrane, in electrical impulses. This "transduction" (or more simply, this translation of a vibration into a sequence of electrical impulses) takes place thanks to the help of a particular material called piezoelectric. In fact, the piezoelectric material as we have seen is able to produce a tension when it is stressed to a compression or expansion, that is, when it undergoes a deformation.



In the loudspeaker there is essentially the reverse process to the microphone, or the electrical impulse, suitably amplified, activates an electromagnet that attracts to itself a permanent magnet fixed to a stretched membrane. In this way the movement of the magnet generates the vibrations of the membrane and therefore reproduces the sound waves with the same frequency indicated by the movement of the magnets.


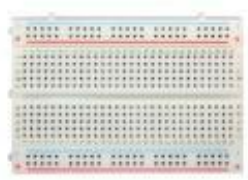







So with the use of microphones and speakers we are able to equip our Robot with hearing and speech.

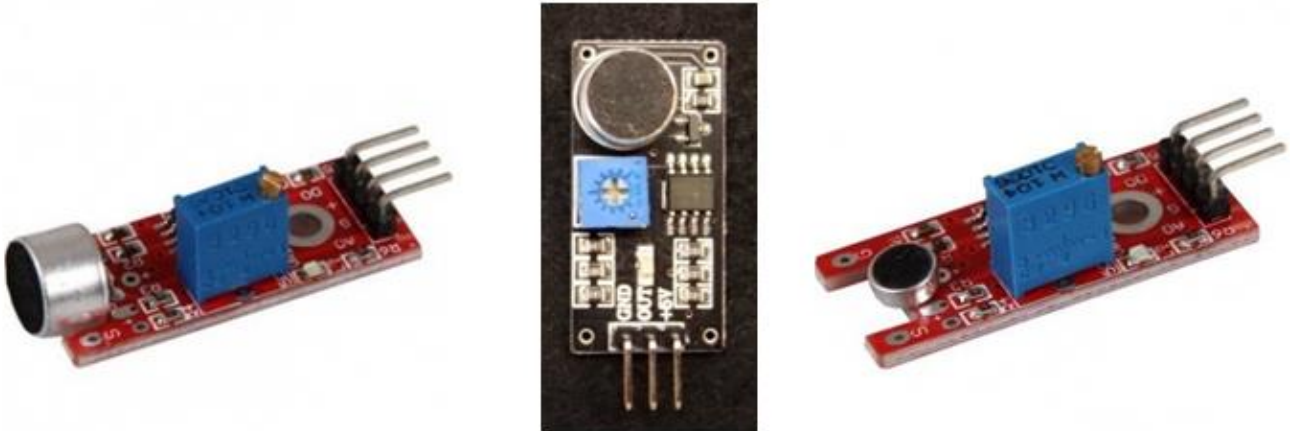
Project 18 – What happens in that house? We use the microphone.



For this project we need:

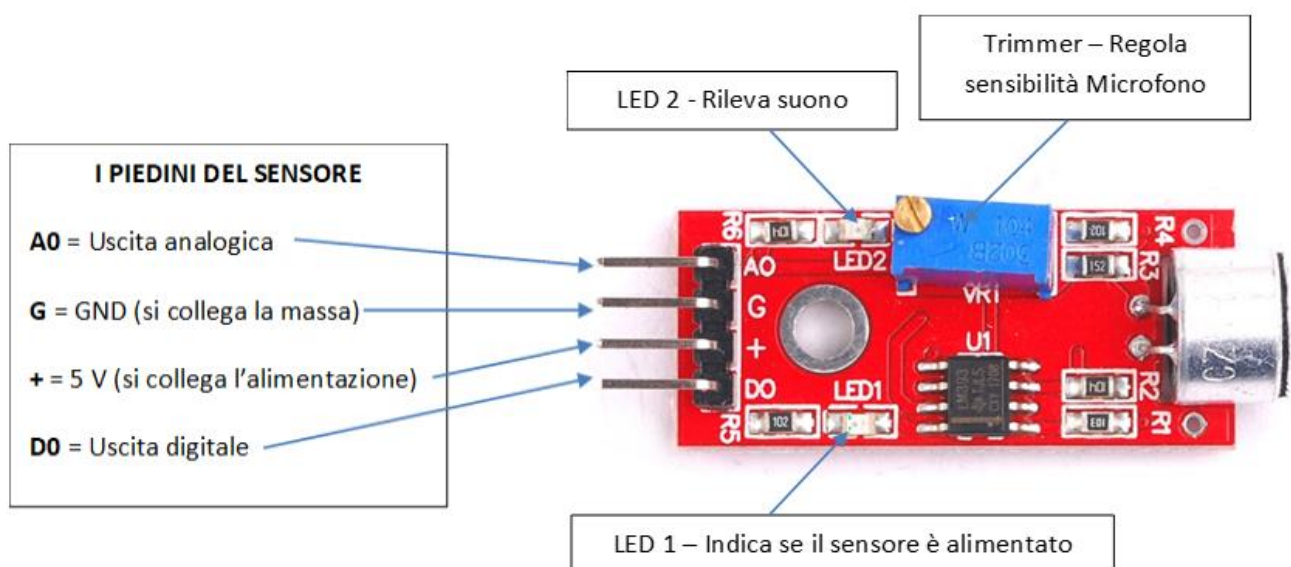
			
<i>Arduino Uno R3 or compatible</i>	<i>Breadboard</i>	<i>Dupont cables male - male</i>	<i>USB connection cable</i>
			
<i>Microphone module</i>	<i>220 Ohm resistance</i>	<i>Leds of any color</i>	

Before moving on to the assembly of the circuit it is good to know better the Sound sensor or microphone module that we will use for educational purposes, in fact, on the market there are different types of sound sensor modules, from the most sophisticated ones that also have an amplification circuit and very sensitive microphones, and those that have a pure educational purpose, less sensitive but equally functional. However, the operation is basically the same.

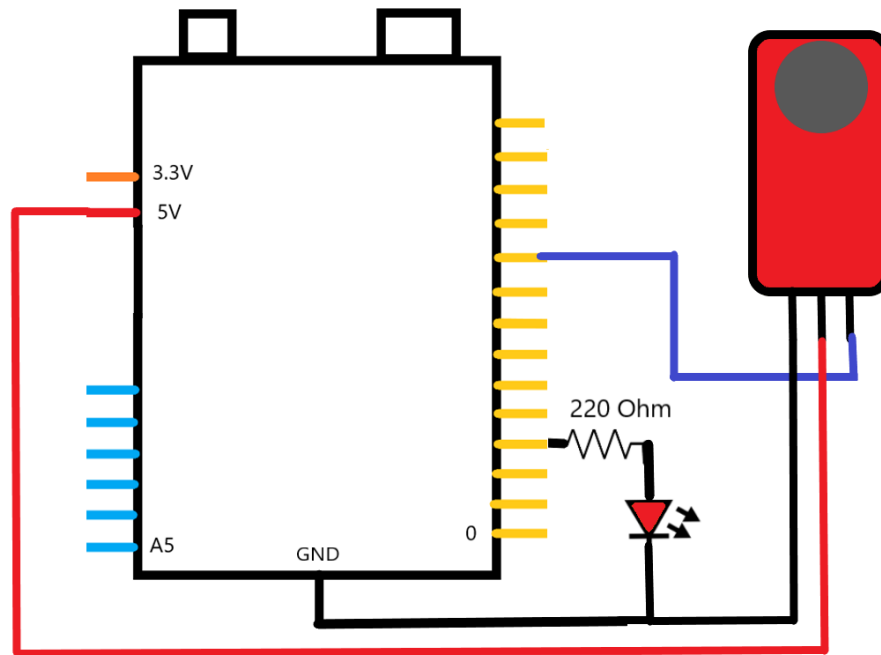


We will use the sensor modules that on the market are found with the initials KY-037 and with the initials KY-038 which is the older brother (slightly more sensitive), these in turn can have 4 PINs because they make available both a digital and analog output signal, those with 3 PIN make only the digital signal available.

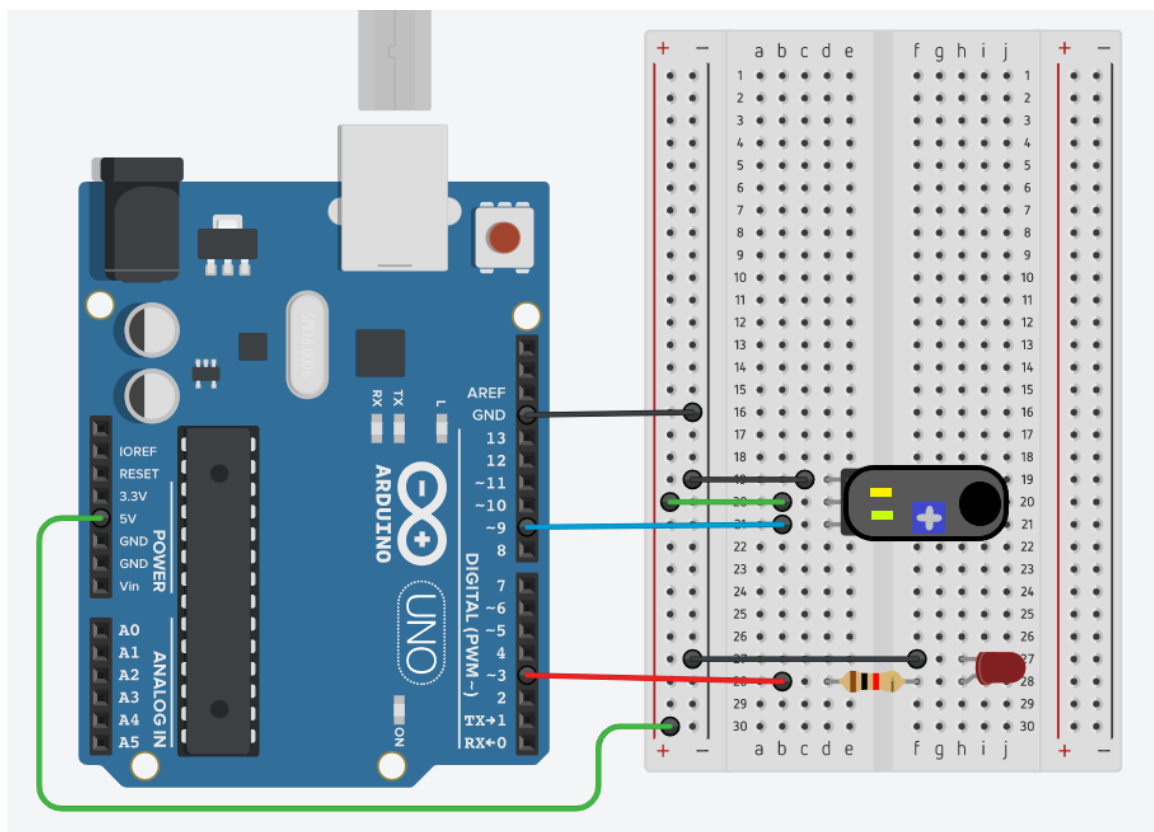
They can also have one or two LEDs, at least one LED is used to verify the correct adjustment of the sensitivity, while the second, not always present, indicates the power supply of the Module. Finally, these modules always have a trimmer (i.e. a small potentiometer) that by adjusting it using a screwdriver it is possible to establish the threshold sensitivity that the microphone must have to make the output digital signal available at the low or HIGH state to the relative PIN indicated with D0 or with OUT (in the case of a 3-PIN module).



At this point we move on to the wiring diagram:

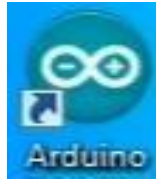


(Note: Use the digital PIN of the D0 Sound Sensor Module or also indicated with OUT for the 3-PIN Module). The assembly scheme is instead:



After the links we move on to write the sketch.

Connect Arduino to the PC via the USB cable and launch the Arduino IDE application by clicking twice on the relevant icon.



An empty window opens, or you need to open a new empty one and write the sketch below:

```
/*  
Be Maker STEM – Project 18-Sensore_Digitale_Rumore  
*/  
  
int digitalPin = 9;  Sound Sensor Module Digital      Signal PIN  
int digitalValue = 0; Variable that retains a value of 0 or 1 if it exceeds the threshold  
                      Sensor calibration required to create threshold value  
int ledPIN= 3;      PIN to which the LED is connected  
  
void setup ()  
{  
  pinMode (digitalPin, INPUT); We set PIN          9 as INPUT  
  pinMode (ledPIN, OUTPUT);  We set PIN 3 as OUTPUT  
  Serial.begin (9600);      We set the serial communication port  
}  
  
void loop ()  
{  
  digitalValue = digitalRead (digitalPin); reads the value at PIN 9  
  if (digitalValue > 0) {  
    digitalWrite (ledPIN, HIGH);  
    delay(3000);  keeps the LED active for 3 sec.  
  }  
  else {  
    digitalWrite (ledPIN, LOW);  
  }  
}
```

Attention: once the sketch is loaded on Arduino, the sensor module must be calibrated according to the background noise present in your environment. So if the LED connected to PIN 3 remains on, it means that the background noise is exceeding the calibration threshold, turn the screw on the trimmer of the module, using a flat head screwdriver, as long as the LED does not turn off. Conversely, if the LED is turned off even with loud noises, adjusted as long as it does not turn on. Place the right screw at the limit so that the LED is off and lights up at the slightest noise slightly higher than that present in the environment.

[For the video-project click [here](#)]