

Mobile maker spaces for promoting maker education
in schools democratizing STEAM education and
innovation development for all the learners



Theremin “The sound creation”

Antonio Caserta & Renato Gatti

international@isisdavinci.eu



Erasmus+

Grant number: 2020-1-PL01-KA201-081698

THEREMIM “THE SOUND CREATION” - THE PHILOSOPHY OF THE PROJECT

Introducing the Theremin Musical Instrument

The Theremin is an electronic musical instrument controlled with no physical contact with the Thereminist (performer). It was invented about 90 years ago.

Original instrument's controlling section work in high frequency and it usually consists of two metal antennas that detect the relative positions of the Thereminist's hands and, based on the distance detected, they control the oscillators: one antenna controls the frequency and the other controls the amplitude (volume) of the output audio signal.

The output electric signals from the theremin are amplified and sent to a loudspeaker.

Our project will be little different. We will use HC-SR04 ultrasonic distance sensor to measure hand position and will translate the distance detected by the sensors into a sound that will have an amplitude and a frequency proportional to the distance detected by the two sensors. Also, sound will turn off when hand it too far from the sensors.

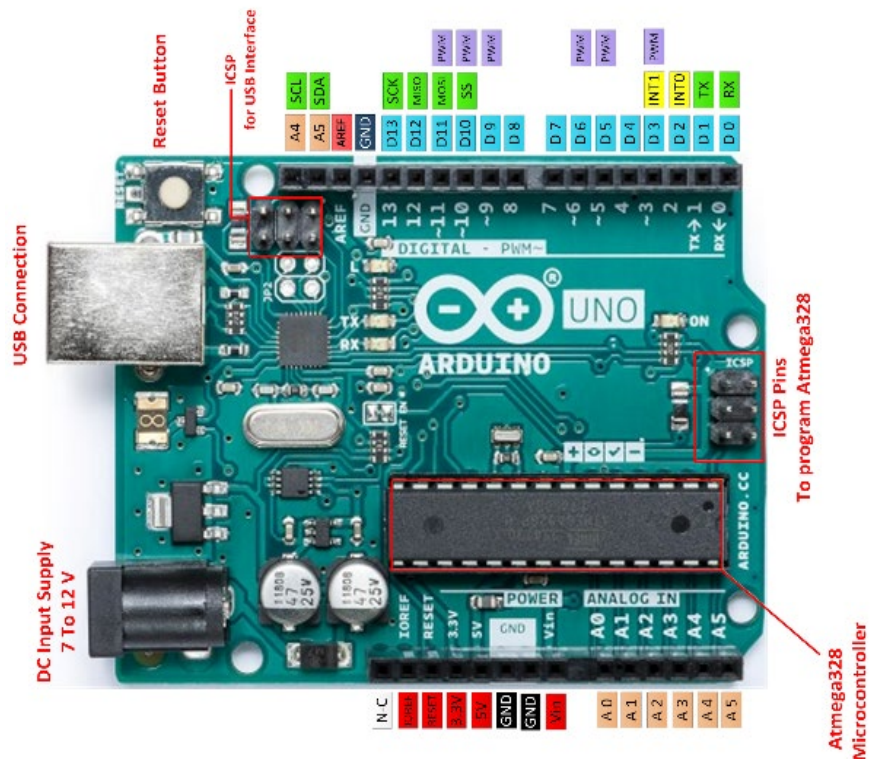
THEREMIN “THE SOUND CREATION” - THE PROJECT

Theremin “the sound creation” - The working phases

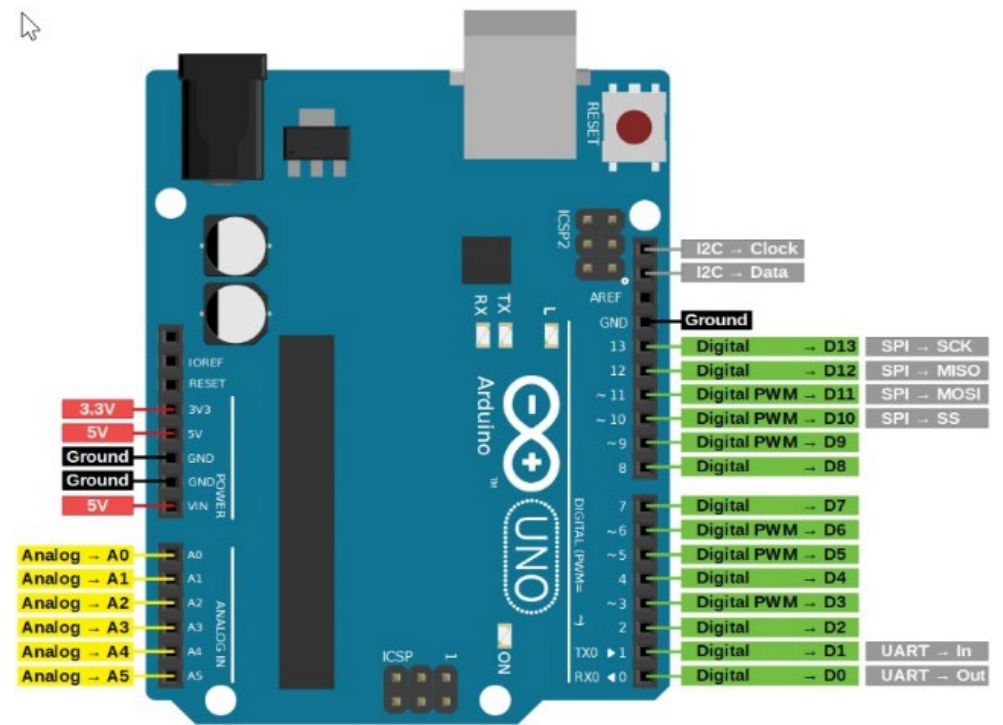
1. The initial phase will be the identification of the correct functioning of a Theremin as musical instrument. Particular attention must be paid to study the physical principle that governs the instrument. This can be done through a quick Internet search guided by the teacher.
2. Make all the necessary interconnections between the ARDUINO UNO board and the various external components
3. Create and transfer the program from the PC to the Arduino board and replace the generic commands and I/O, used for programming the system on the PC
4. Test the correct functioning of the system

LET'S MAKE THE "THEREMIN" USING ARDUINO

Introducing the Arduino Uno board: layout and GPIOs



Arduino Uno Rev 3
Layout



Arduino Uno Rev 3
Pinout

LET'S MAKE THE "THEREMIN" USING ARDUINO AND THE BREAD BOARD

Connecting the Arduino Board to external devices

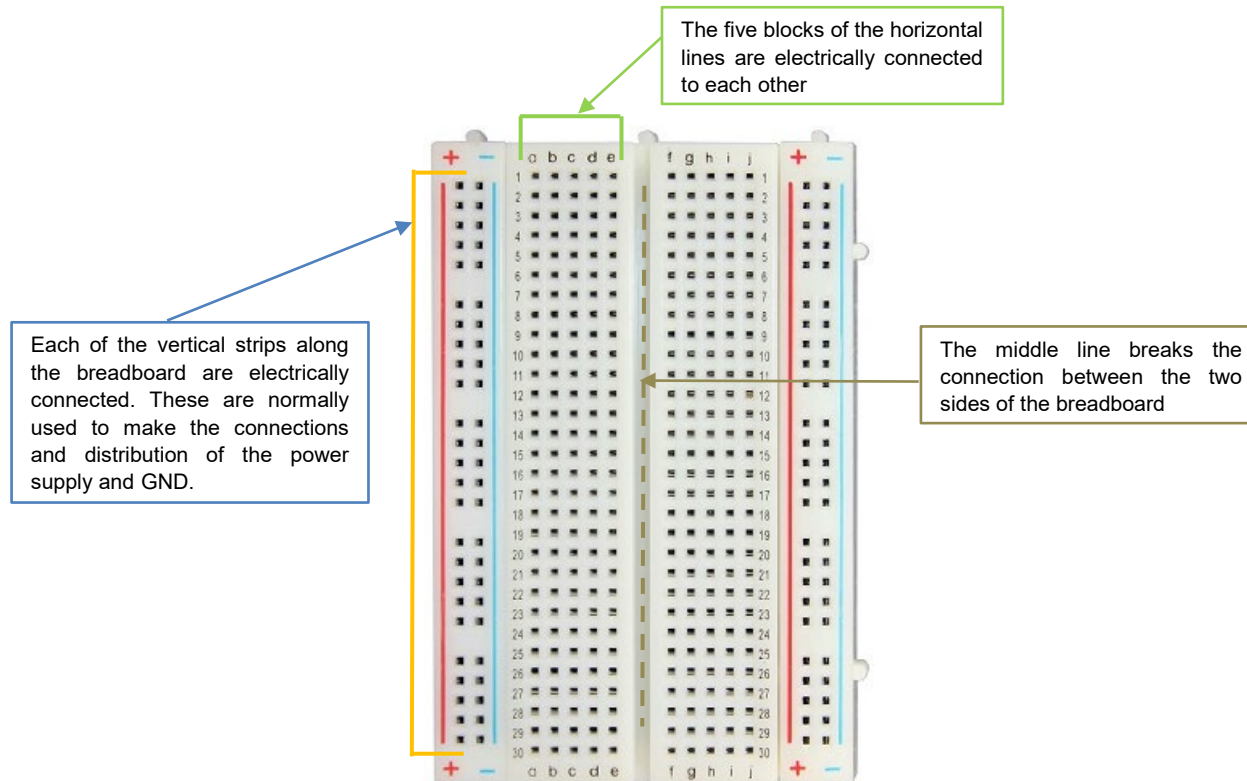
The first step is to connect the Arduino Board to the external devices, in our case two ultrasonic distance sensors, a loudspeaker and a LED RGB. To do this it is necessary to use the breadboard and the jumper wires, included in the Mobile Maker Space

It's convenient to make a spreadsheet with the interconnections between the Arduino GPIOs and the various traffic lights present in the respective breadboards, as shown below:

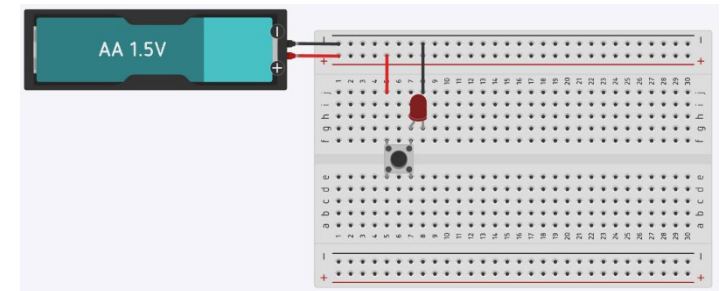
Arduino		Ultrasonic Sensor 1	Arduino		Loudspeaker 1
GPIO	PIN	Pin Name	GPIO	PIN	Cable/PIN Colour
+5V	5V via Breadbord	Vcc	D8	8	RED
D12	12	Trig	GND	GND via Breadbord	BLACK
D11	11	Echo			
GND	GND or 14	GND			
Arduino		Ultrasonic Sensor 2	Arduino		Loudspeaker 2 (optional)
GPIO	PIN	Pin Name	GPIO	PIN	Cable/PIN Colour
+5V	5V	Vcc	D9	9	RED
D7	7	Trig	GND	GND via Breadbord	BLACK
D6	6	Echo			
GND	GND via Breadbord	GND			

LET'S MAKE THE "THEREMIN" USING ARDUINO AND THE BREAD BOARD

The Breadboard is a way of constructing electronics without having to use a soldering iron. Components are pushed into the sockets (holes) on the breadboard and then extra 'jumper' wires are used to make connections between the various components



Below an example of how to connect a LED with a battery and a button



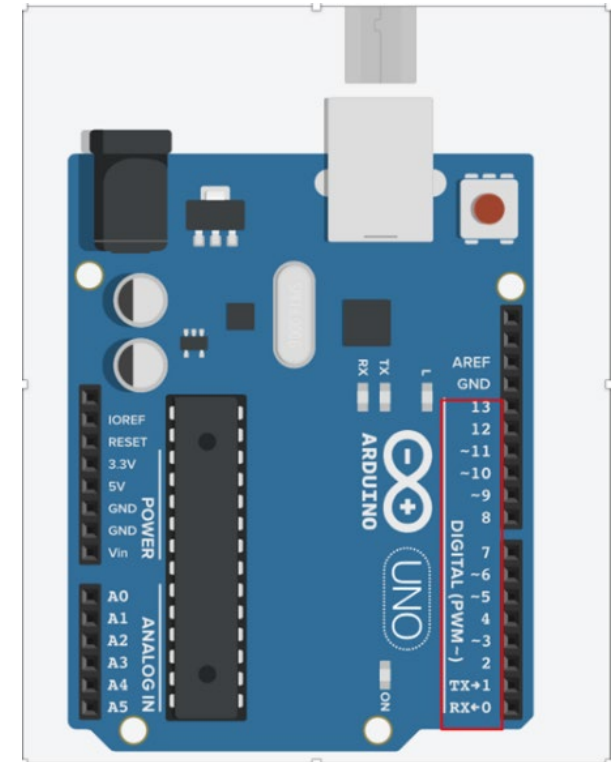
LET'S MAKE THE "THEREMIN" USING ARDUINO AND THE ULTRASONIC SENSORS HC-SR04

The Arduino Board digital ports

The digital ports of the Arduino boards are shown in the figure on the right:

The HC-SR04 ultrasonic distance sensors

The digital ports of HC-SR04 are shown in the figure below



LET'S MAKE THE "THEREMIN" USING ARDUINO AND THE ULTRASONIC SENSORS

How does the ultrasonic sensor HC-SR04 work?

Ultrasonic sensors do not directly measure the distance between the object and the sensor itself, but measure the time taken by the signal they emit to reach the object and return back to the sensor, i.e. they measure the echo, also called "flight time".

The HC-SR04 sensor has 4 pins: Vcc (+ 5V), Trigger, Echo, GND.

To activate the sensor, a "high level" pulse signal is sent on the Trigger pin for at least 10 microseconds: at this point the sensor will send an ultrasonic wave, called "ping", and wait for the return of the reflected waves; the sensor will respond on the Echo pin with a high pulse of the duration corresponding to the "travel" time of the sound waves.

The duration of this return-back signal is used by the program to define the related output sound parameter.

A SIMULATION OF OUR THEREMIN USING TINKERCAD

TinkerCad is an application developed by Autodesk that allows you to carry out numerous simulations with various electronic devices, including Arduino.

It is necessary to register and create an account on the following website: <https://www.tinkercad.com>

Program to measure the distance with HC-SR04

```
/*
 *
 * Complete Guide for Ultrasonic Sensor HC-SR04
 *
 Ultrasonic sensor Pins:
 VCC: +5VDC
 Trig : Trigger (INPUT) - Pin11
 Echo: Echo (OUTPUT) - Pin 12
 GND: GND
 */

int trigPin = 11; //Trig - green Jumper
int echoPin = 12; //Echo - yellow Jumper
long duration; // duration is in microsec
float distance; // distance is in cm

void setup() {
  //Serial Port begin
  Serial.begin (9600);
  //Define inputs and outputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop()
{
  // The sensor is triggered by a HIGH pulse of 10 or more microsec
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

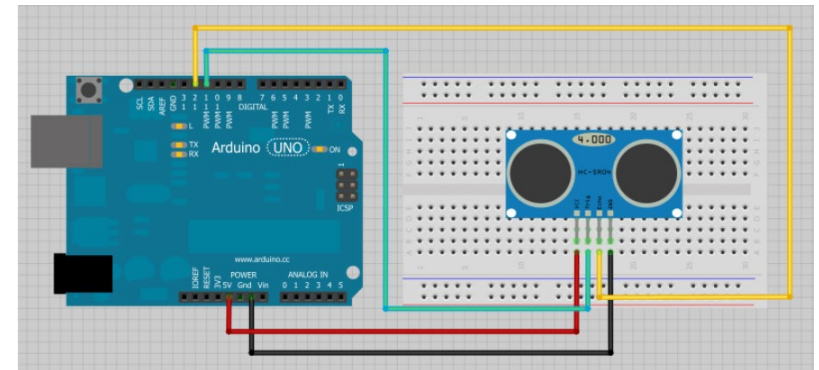
  // Read the signal from the sensor: a HIGH pulse whose
  // duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  duration = pulseIn(echoPin, HIGH);

  // convert the time into a distance
  distance = (duration/2.0 * 0.0343; // or, which is the same, /29.15

  Serial.print(distance);
  Serial.print("cm");
  Serial.println();

  delay(250);
}
```

Circuit to measure the distance with TinkerCAD

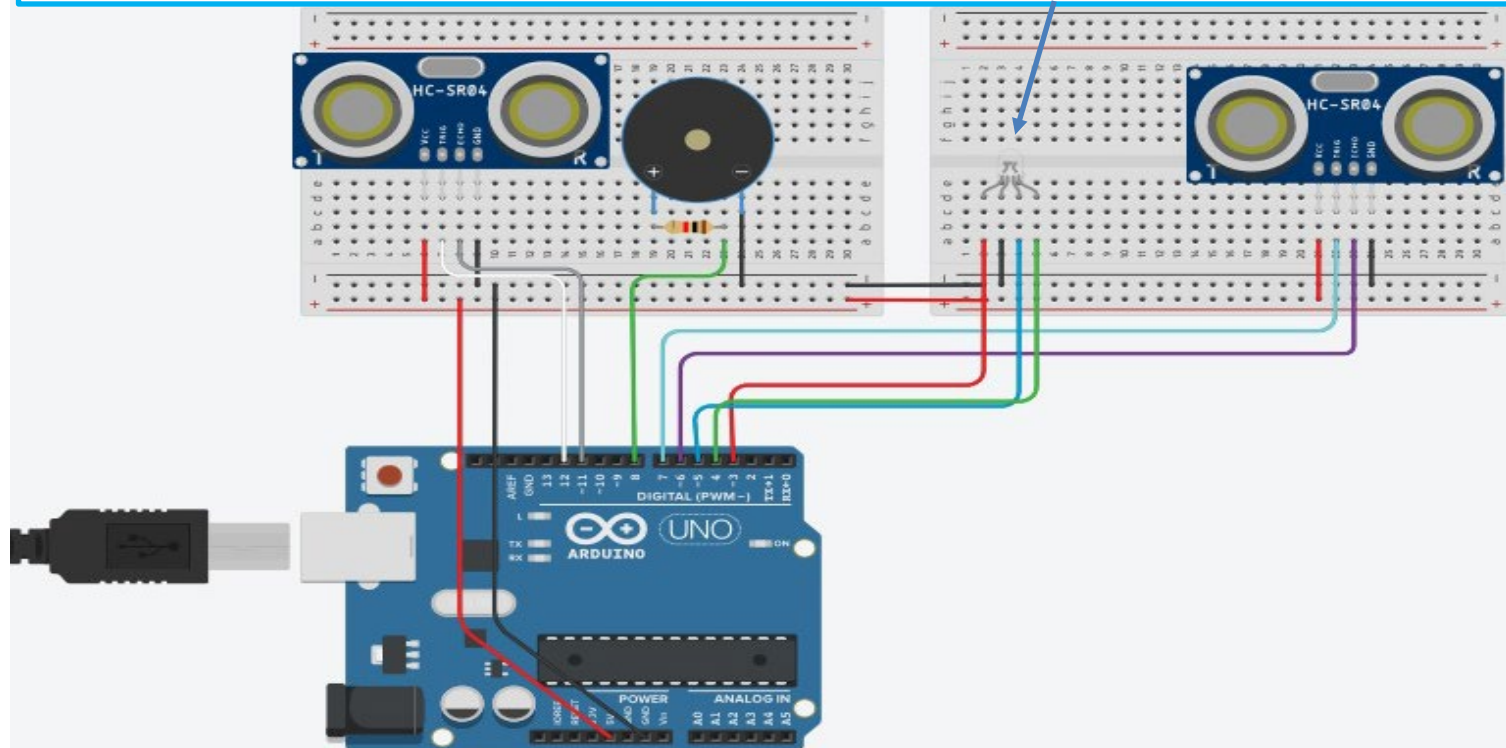


A SIMULATION OF OUR THEREMIN USING TINKERCAD

Theremin with two HC-SR04 ultrasonic distance sensors and RGB LED

To implement the "Theremin" system we connect the two ultrasonic sensors as shown in the next figure. Then we connect the RGB LED respecting the pinout, with the shorter pin (cathode) connected to ground, finally we connect a loudspeaker to pin 8.

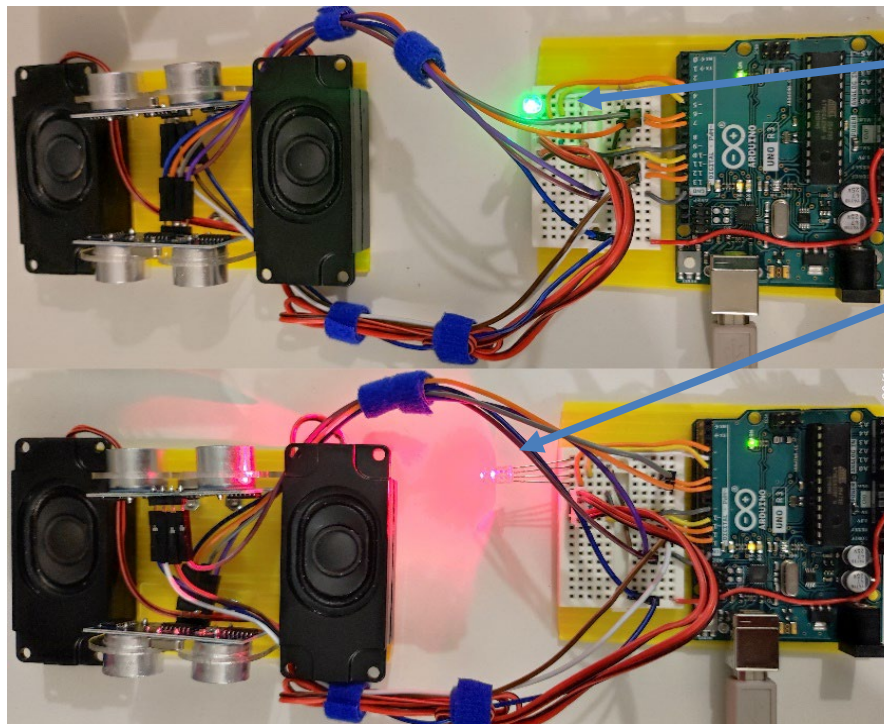
We will also use a multicolour LED (RGB) which will change colour, based on the frequency emitted by the Theremin speaker



A SIMULATION OF OUR THEREMIN USING TINKERCAD

Theremin with two HC-SR04 ultrasonic distance sensors and RGB LED

The RGB LED, that we have included in the project, will change colour in a directly proportional way to the frequency of the sound produced. This will make it possible to the teacher to explain the correlation between the audio frequency spectrum and the electromagnetic frequency spectrum in general. For example, the visible spectrum could be introduced, which is between red (lowest visible frequency) and violet (highest visible frequency), observing the changes in the colour of the LED as the frequency of the emitted sound changes.



Here you can see the RGB LED changing color based on the output audio frequency

BUILD THE PROGRAM USING ARDUINO IDE ON THE PC

In this phase the code to implement the Theremin system is written using the Arduino IDE platform. Below is shown the complete program code of Theremin made by Arduino IDE

```
Theremin.ino
1  #include <NewPing.h>
2  #include <toneAC.h>
3
4  #define DEBUG      false // Set to true to enable Serial debug
5  #define TONE_PIN   8
6  // #define TONE_VOLUME  10 // 1-20
7  #define TRIGGER_PIN 12 // Board pin tied to trigger pin on the ultrasonic sensor.
8  #define ECHO_PIN   11 // Board pin tied to echo pin on the ultrasonic sensor.
9  #define MAX_DISTANCE 200 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-
10 int portarossa=3;
11 int portaverde=4;
12 int portablu=5;
13 NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
14 NewPing sonar1(7,6,200);
15
16 void colore (unsigned char rosso, unsigned char verde, unsigned char blu)
17 {
18   analogWrite(portarossa, rosso); //activates the red LED with the intensity defined in the rosso variable
19   analogWrite(portablu, blu); //activates the blue LED with the intensity defined in the blu variable
20   analogWrite(portaverde, verde); //activates the green LED with the intensity defined in the verde variable
21 }
22
23 void setup() {
24   Serial.begin(9600);
25   //Serial.println("Theremin starting");
26   pinMode(portarossa, OUTPUT); // declares port 3 as the output port
27   pinMode(portaverde, OUTPUT); // declares port 4 as the output port
28   pinMode(portablu, OUTPUT); // declares port 5 as the output port
29 }
30
31 void loop() {
32   delay(30); // Wait 30ms between pings (about 33 pings/sec). 29ms should be the shortest delay between pings.
33   unsigned long uS = sonar.ping(); // Send ping, get ping time in microseconds (uS).
34   unsigned long uS1= sonar1.ping();
35   int distance=sonar1.ping_cm();
36   int TONE_VOLUME=distance;
37   //Serial.println(uS);
38   //Serial.println(uS1);
39   //Serial.println(distance);
40   if (uS > 2000) { // Range is about 0-30 cm from sensor
41     toneAC(0); // Turn sound off when not in range
42     //if (DEBUG)
43     //Serial.println("No tone");
44   } else {
45     int freq = 2000 - uS / 1.5; // Get sound frequency
46     toneAC(freq, TONE_VOLUME); // Play it!
47     //if (DEBUG)
48     Serial.println(freq);
49     if (freq>675 && freq<=875)
50     {colore(255, 0, 0); // launches the color routine, with the red parameter at 255, the green at 0
51     // and blue at 0 "lights up" red
52     delay(10);}
53     if (freq>876 && freq<=996)
54     {colore(237,109,0); // "lights up" orange (237 of red and 109 of green)
55     delay(10);}
56     if (freq>997 && freq <=1117)
57     {colore(255,215,0); // "lights up" yellow (255 of red and 215 of green)
58     delay(10);}
59     if (freq>1118 && freq<=1398)
60     {colore(0,255, 0); // launches yhe color routine and "lights up" green
61     delay(10);}
62     if (freq>1399 && freq<=1639)
63     {colore(0, 0, 255); // "lights up" blue
64     delay(10);}
65     //if (freq>1100 && freq<=1200)
66     //{colore(0,46,90); // "lights up" indigo (46 of green and 90 of blue)
67     //delay(10);}
68     if (freq>1639 && freq<=2000)
69     {colore(128,0,128); // "lights up" violet (128 of red and 128 of blue)
70     delay(10);}
71   }
72 }
--
```

STEAM



Erasmus+

