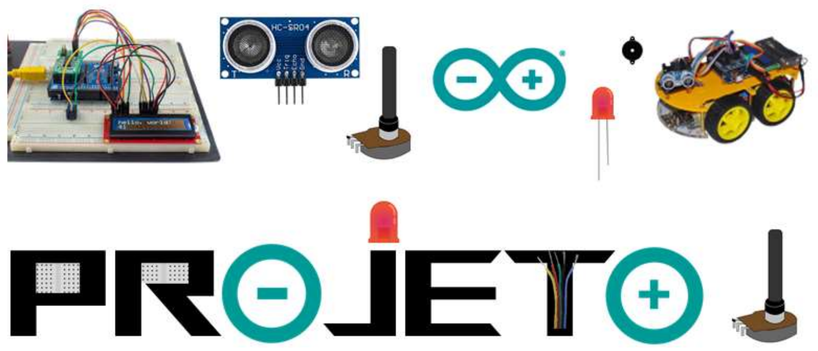


Licenciatura em Engenharia Eletrotécnica e de Computadores

Laboratorial assignment # 5

Piezo Sounder Melody Player

Components list:

- Arduino UNO
- 1 USB cable
- 1 white breadboard
- 1 piezo disc
- 8 LEDs + 8 220 Ohm resistors
- 8 Push Buttons
- 1 Potentiometer + 220 Ohm resistor
- Wires
- Software: Arduino IDE

1. Piezo Sounder Tone Player

Observe the circuit in **Figure 1**. Implement it and write a program for your UNO to control the sound that you hear from the piezo electric device. The operating mode should be as follows:

- Play the DO-RE-MI-… full scale up and down (continuously) according to the code illustrated in **Figure 2**. Please notice that the sequence of notes is defined by an array `notes[]`, each note having duration (*tempo*) defined by another array `beats[]` and variable `tempo`.

Upload and test your program.

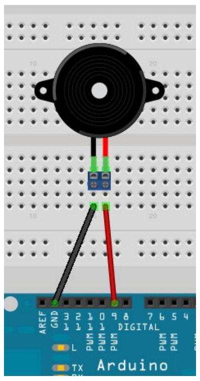


Figure 1. The piezo electric sound device should be connected to ground (GND) and to an output pin of the Arduino UNO.

```
int speakerPin = 9;

void playTone(int tone, int duration) {
 for (long i = 0; i < duration * 1000L; i += tone * 2) {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration) {
   int tone;
   switch (note) {
      case 'c': tone=1915; break;
      case 'd': tone=1700; break;
      case 'e': tone=1519; break;
      case 'f': tone=1432; break;
      case 'g': tone=1275; break;
      case 'a': tone=1136; break;
      case 'b': tone=1014; break;
      case 'C': tone= 956; break;
   }
   playTone(tone, duration);
}

void setup() {
   pinMode(speakerPin, OUTPUT);
}

void loop() {
   /******** DEFINE YOUR MELODY HERE **********/
   static String notes = "cdefgabCbagfedc"; // a space represents a rest
   static int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 4, 8, 4 };
   static int tempo = 300;

  for (int i = 0; i < notes.length(); i++) {
     if (notes[i] == ' ')
         delay(beats[i] * tempo); // rest
     else
         playNote(notes[i], beats[i] * tempo);

     delay(tempo/2); // pause between notes
  }
}
```

Figure 2. Example code.

2. Melody Mode

Adapt a well-known melody, and develop the corresponding source code, program the Arduino UNO and present the outcome to the professor. For the assignment to be considered fully successful, the professor should be able to identify the melody and you cannot use “Happy Birthday to you”!

After playing the melody, repeat it playing it backwards. Upload and test your program.

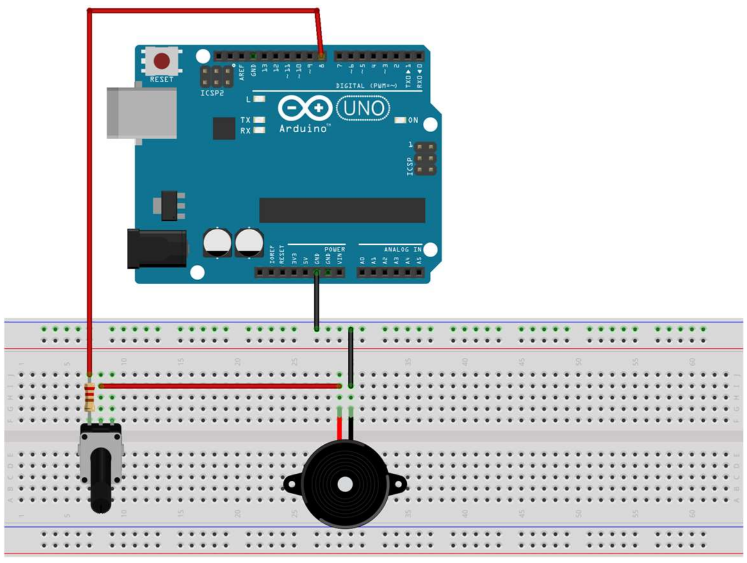
3. Synchronizing Lights and Melody

Mount eight 220 Ohm resistors and LEDs on the breadboard and change the source code, so that the set of lights may start blinking according to the previously developed melody (creativity is a plus: you can choose any other lightning effect).

Upload and test your program.

4. Controlling the Volume of your Melody

Add a potentiometer and a 220Ω resistor to your circuit and use it to control the volume being played according to the potentiometer’s input. Make sure that you are using the correct Speaker Pin.



5. Simple Piano

Now add 8 push buttons to your breadboard. Assign a note (`'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C'`) to each pair push button/LED. Upload the code to the board and play the simple piano.

Tip: Use 6 analog pins + 2 digital pins for the 8 push buttons, while using digital pins for all LEDs. You might also consider using two breadboards for this exercise.

Important: The tone should play even when the button is long pressed.

6. Sound Pattern Game

Change the previous setup to use only 6 push buttons and 6 LEDs, make sure that your 6 LEDs correspond to 3 Red LEDs and 3 Green LEDs (see **Figure 3**).

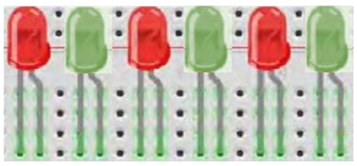


Figure 3. LEDs for Sound Pattern Game.

Tip: To better organize your circuit, you can use 6 analog pins for the push buttons and 6 digital pins for the LEDs.

Implement a game, which:

a. Initially generates a random sequence of 5 notes and plays it. The notes should have at least 1 second of delay between them, and the LED for that note should light up when that note is played.

b. After playing the random sequence, blink all LEDs for 3 seconds, and then turn them off.

c. With the push buttons, try to replicate the same random sequence.

d. If you push a wrong button, your Arduino should immediately detect that the sequence is not correct and blink all red LEDs for 3 seconds.

e. Otherwise, if the sequence of 5 notes is correct, blink all green LEDs for 3 seconds.

f. Restart the game.