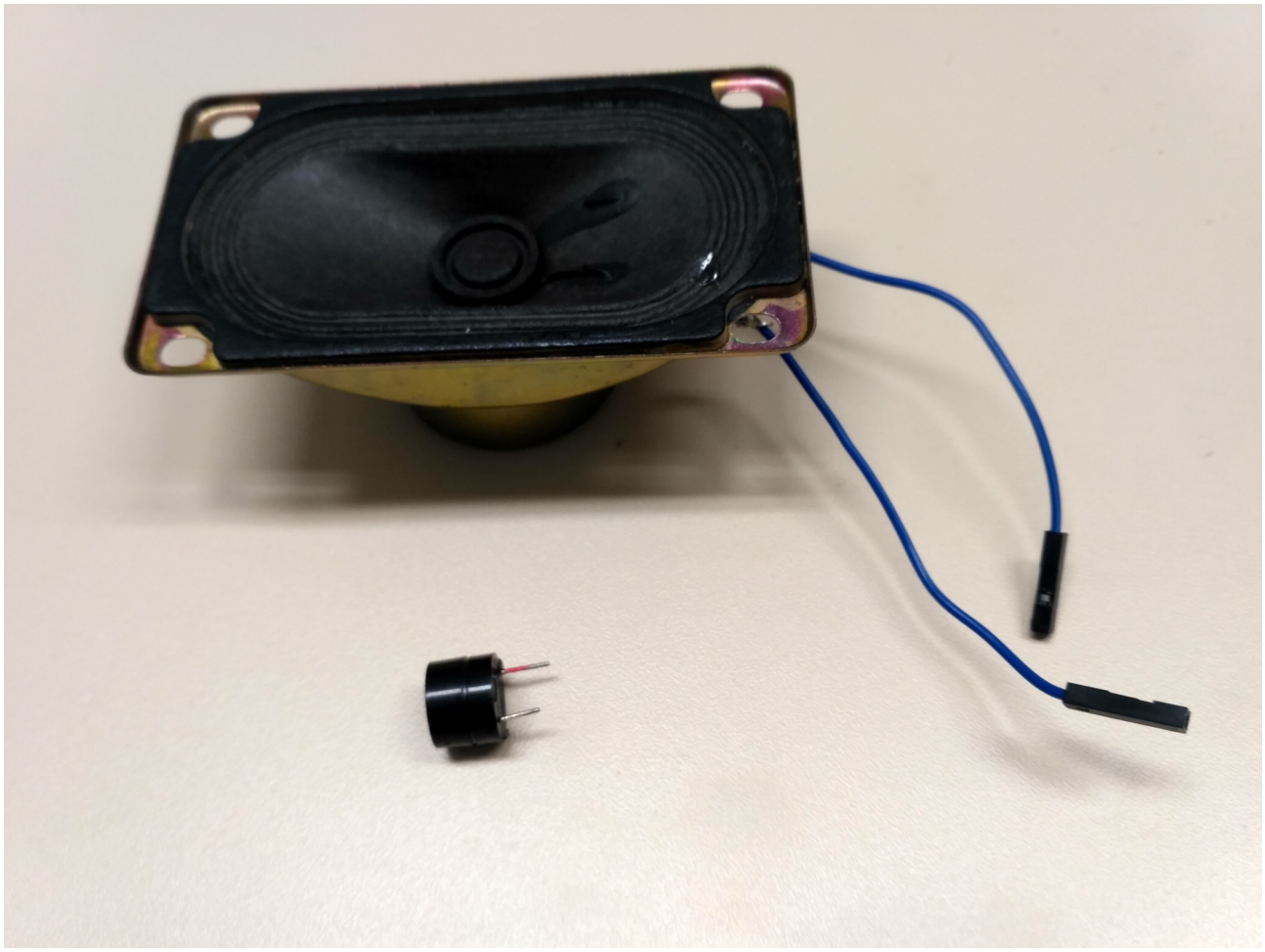
Trabalhando com sons

O som é uma vibração transmitida pelo ar. Para gerar uma vibração que corresponda ao que conhecemos como som, em frequência audível, precisamos gerar uma corrente alternada (que varia ao longo do tempo).

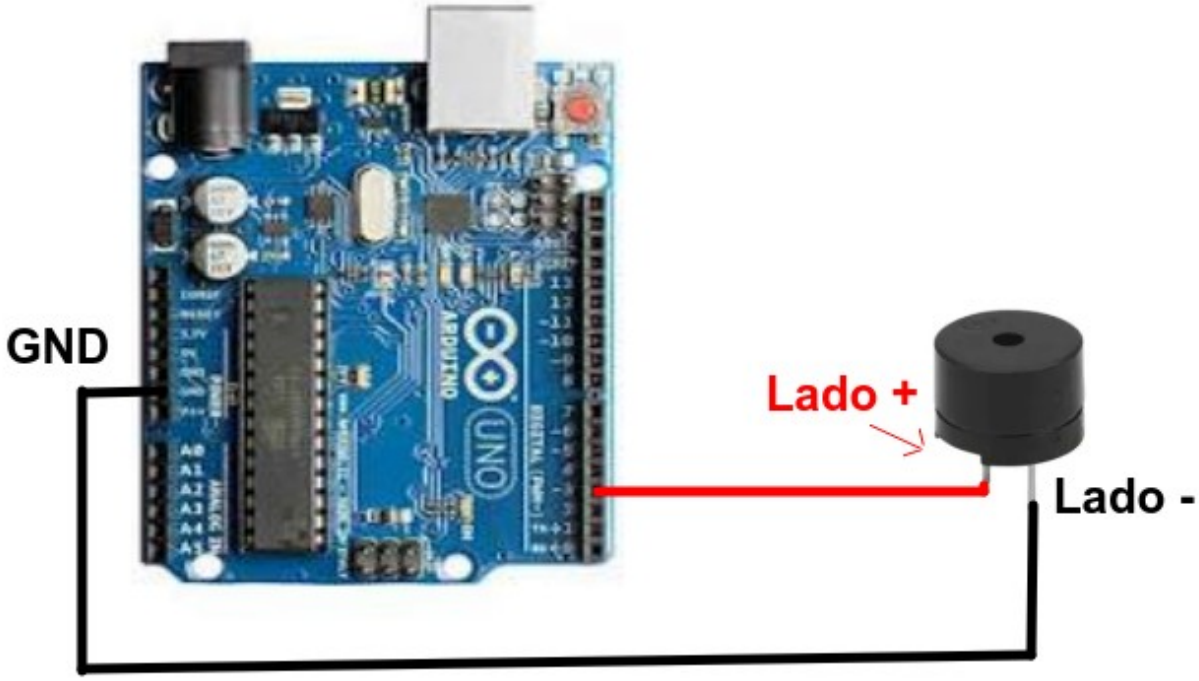
Para reproduzirmos esta corrente utilizamos um transdutor: um componente capaz de transformar a corrente recebida em vibrações. Os tipos usuais no mundo do Arduino são o *buzzer* e o alto falante:



O *buzzer* poderá ser polarizado (o lado + vem marcado no invólucro), respeite isso. O alto falante não é (se houver marcação + e – no caso do alto falante é somente para quando forem ligados mais de um eles fiquem com a mesma fase).

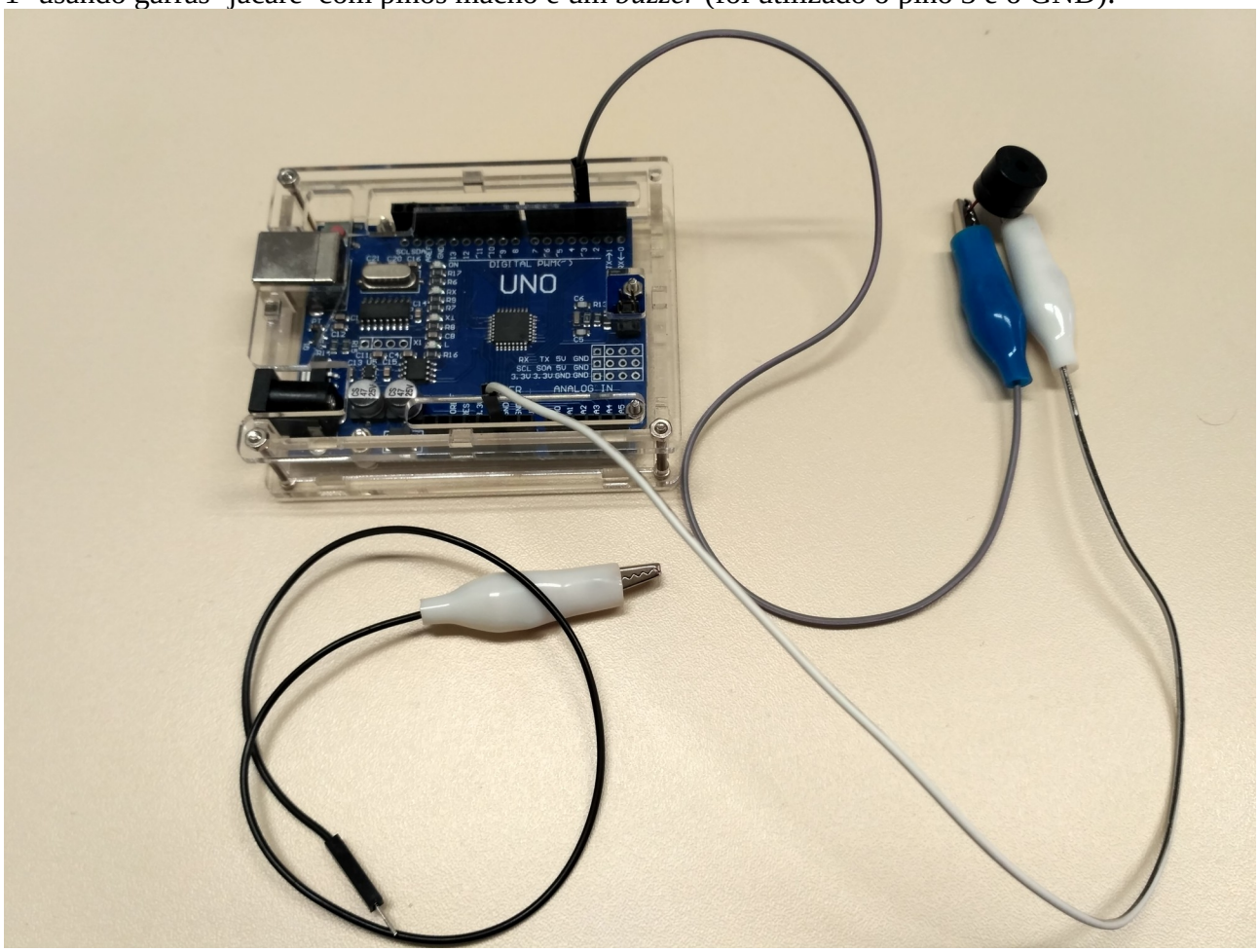
Ligações

Escolhemos um pino do Arduino (neste exemplo escolhi o pino 3) e efetuamos a conexão do Arduino com o transdutor escolhido.

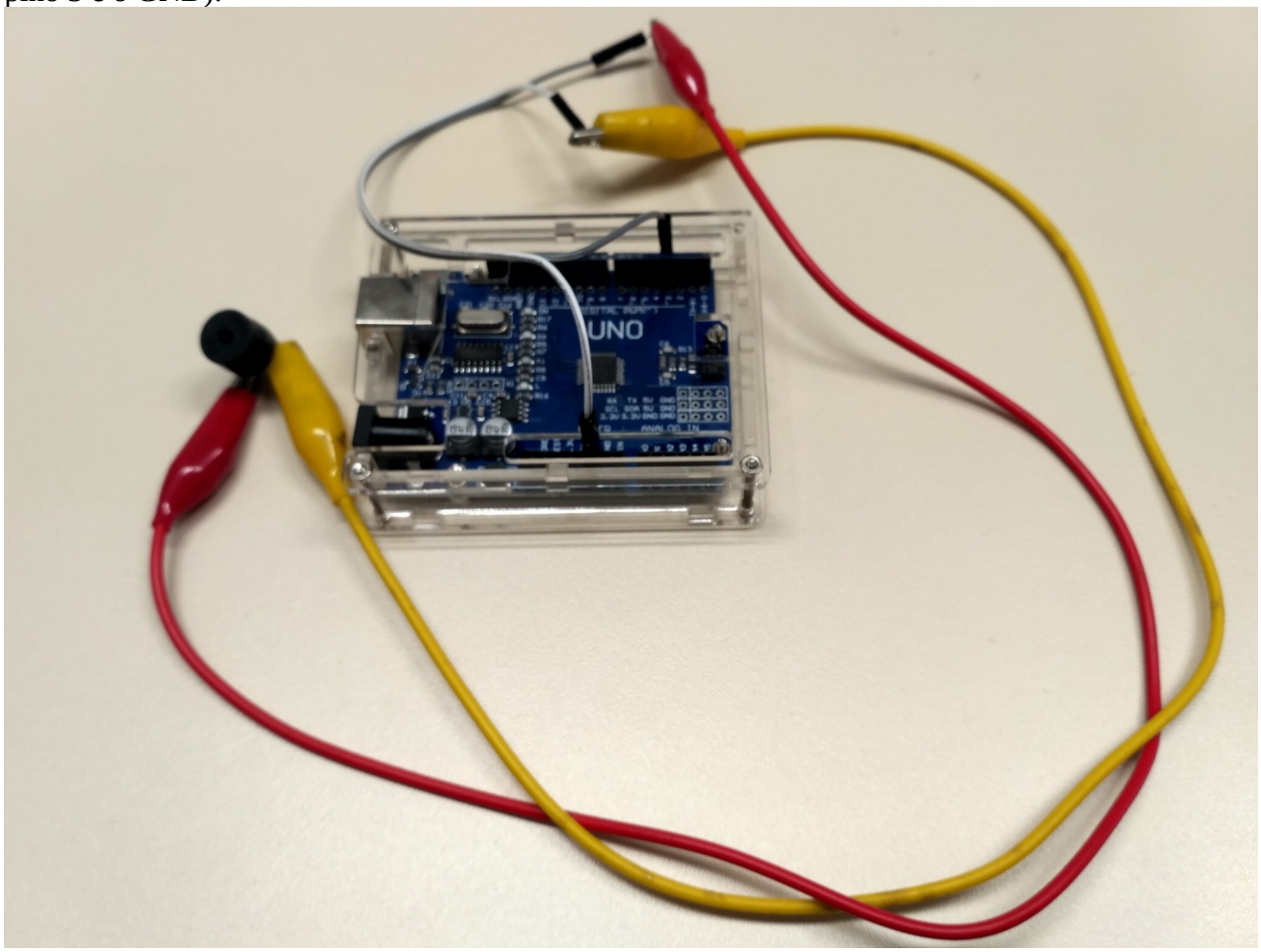


Veja exemplos:

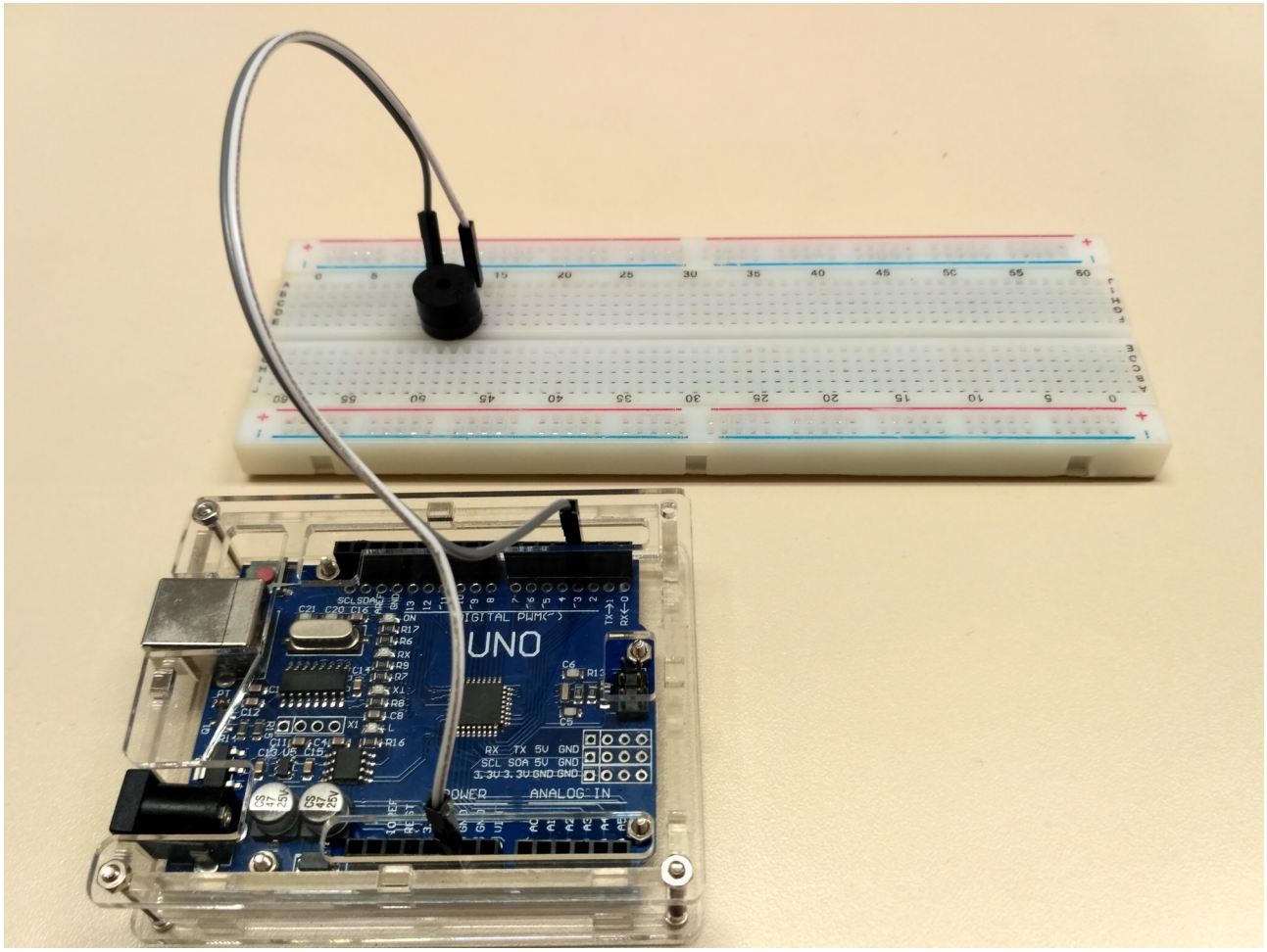
1- usando garras ‘jacaré’ com pinos macho e um *buzzer* (foi utilizado o pino 3 e o GND):



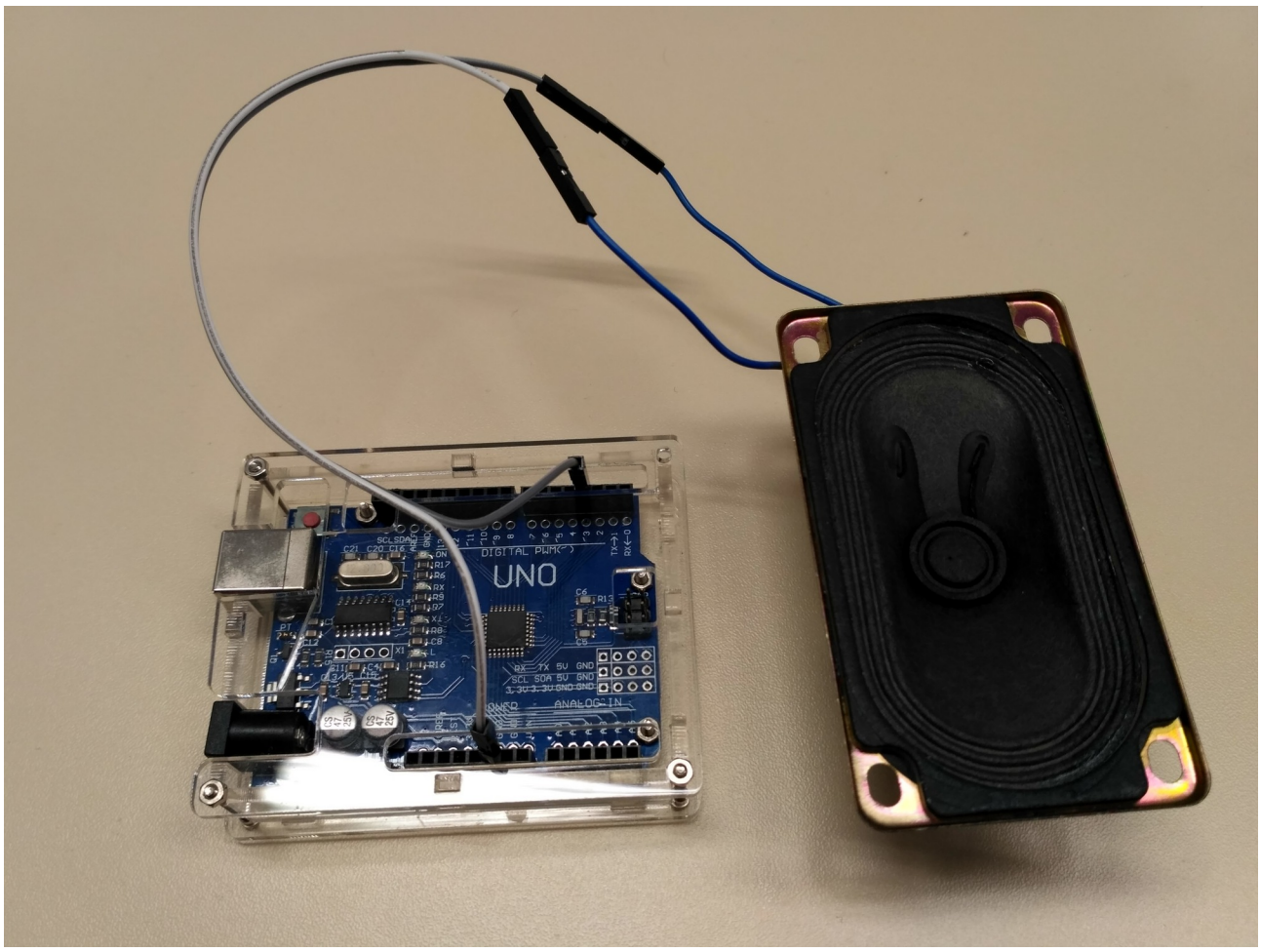
2- usando garras ‘jacaré’/’jacaré’ e conectores com pinos macho/macho e um *buzzer* (foi utilizado o pino 3 e o GND):



3- usando conectores com pinos macho/macho e uma placa de experimentação *breadboard* e um *buzzer* (foi utilizado o pino 3 e o GND):



4- usando conectores com pinos macho/macho e um alto falante (foi utilizado o pino 3 e o GND):



Nos exemplos fizemos as ligações diretamente, por simplicidade. Podemos melhorar a qualidade do som inserindo um resistor e um capacitor em série como alto falante (e com o *buzzer*), mas deixaremos as sofisticações para depois. Vamos ver o código.

(Se quiser, baixe os códigos utilizados nos exemplos aqui:
https://tiaplicada.ufpr.br/wp-content/uploads/2024/08/arquivossomsimples.zip)

transdutorSimples §

```
1  #define transdutor 3 //pino conectado ao transdutor
2
3  void setup(){
4    pinMode(transdutor, OUTPUT); // seta o pino como saída
5  }
6  void loop(){
7    tone(transdutor, 1000);
8    delay(2000);
9    noTone(transdutor);
10   delay(3000);
11 }
```

No código, na linha 1 foi definido que o transdutor (nome que eu escolhi) terá o valor 3 (que é a porta digital que eu escolhi para ligação no Arduino – se você mudar a porta em sua ligação elétrica, mude este número).

Na função **setup** (entre linhas 3 e 5), foi definido na linha 4 que nosso transdutor será uma saída. Isto foi realizado com a função que controla o modo do pino (**pinMode**) para a qual são passados dois parâmetros: qual pino e qual o modo. No exemplo, o pino será o 'transdutor' (portanto, o pino 3, definido anteriormente); e o modo será de OUTPUT (que quer dizer saída; ou seja, nosso transdutor vai realizar uma interface do Arduino para o mundo. Se fosse um sensor de calor poderia ser uma entrada: do mundo para o Arduino).

Na função **loop** (entre as linhas 6 e 11), que repete-se enquanto houver energia, temos o controle do som que será gerado. Isto é realizado por meio da função **tone**. Na declaração da linha 7 temos dois parâmetros passados para a função: a porta de ligação e o valor da frequência a ser enviada. No exemplo a porta é o 'transdutor' (nossa porta 3); e a frequência será de 1000 Hz (Hertz, unidade de frequência).

Uma vez que o som seja gerado ele continuará se repetindo. Esta repetição foi 'quebrada' com o uso de duas novas declarações: na linha 8, a função **delay**(2000) que faz com que o Arduino espere 2 segundos (2000 milissegundos), e na linha 9 a declaração **noTone** (não ou sem tom), que desliga a emissão do som na porta indicada (que foi a 'transdutor', a 3). Na linha 10 foi solicitada uma espera de 3 segundos antes do reinício.

Teste !

Depois de testar, faça as seguintes alterações (se quiser, salve o arquivo com outro nome…):

transdutorSimplesFrequenciaTempo

```
1 #define transdutor 3 //pino conectado ao transdutor
2
3 void setup(){
4   pinMode(transdutor, OUTPUT); // seta o pino como saída
5 }
6 void loop(){
7   tone(transdutor, 1000, 250);
8   delay(2000);
9 }
```

Agora estamos utilizando a mesma função **tone** com três parâmetros: porta, frequência e duração). As linhas 9 e 10 originais foram apagadas. Todo o controle agora é realizado pela função **tone** (linha 7) e pelo **delay** (linha 9). O terceiro parâmetro (a duração do tom) que foi inserido na função **tone**, combinado com a frequência, permite a criação de uma infinidade de sons (pergunte ao seu professor de música).

Teste com valores diferentes.

Agora teste o seguinte:

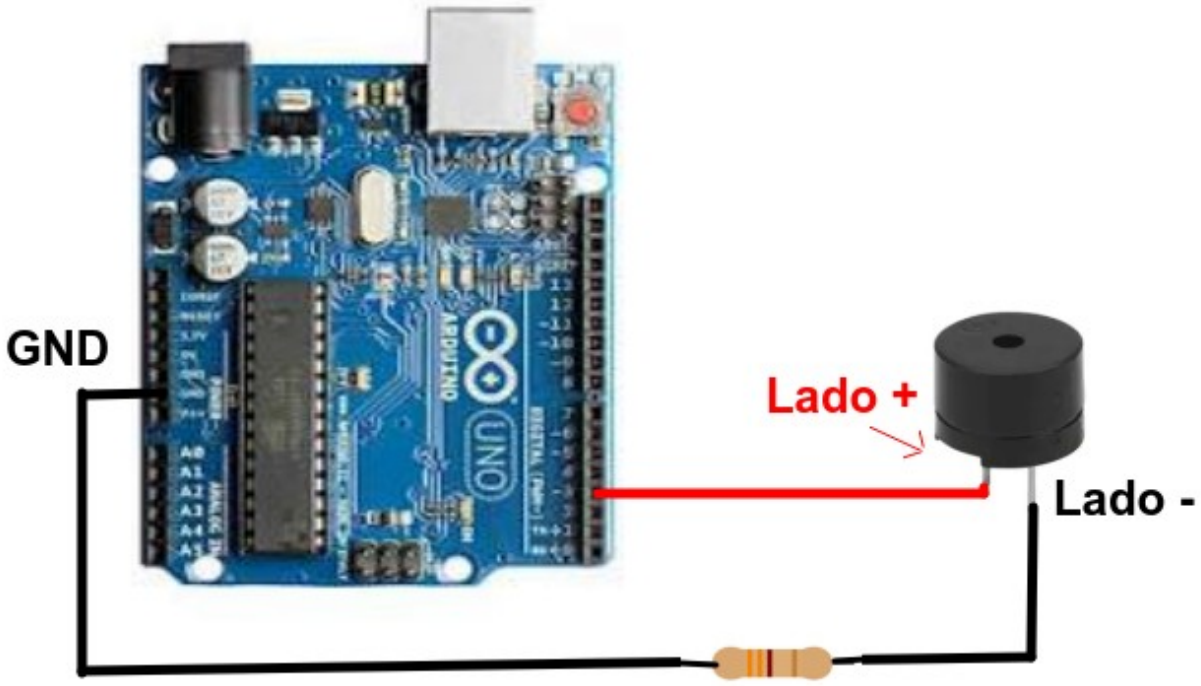
transdutorSimplesSirene

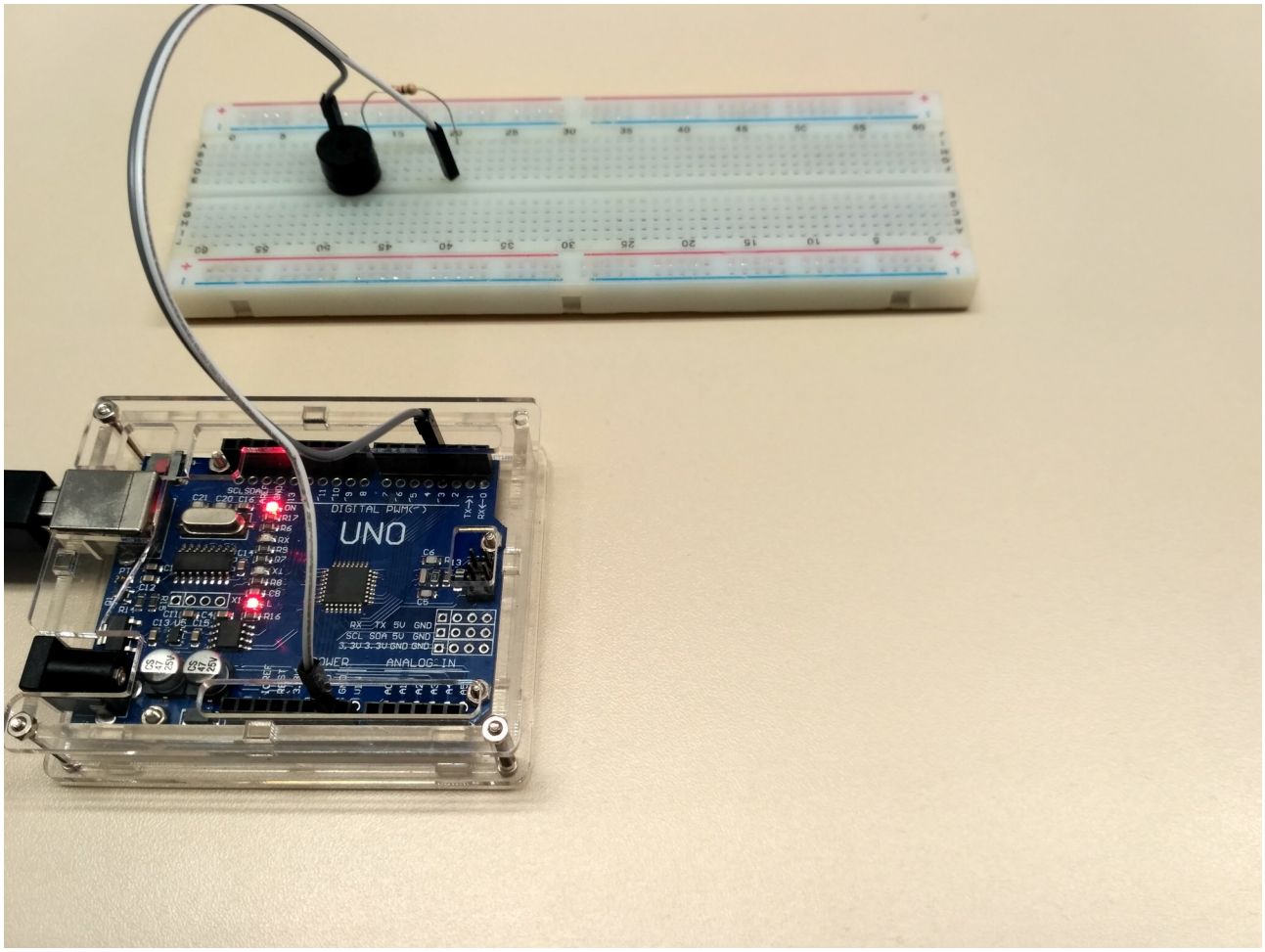
```
#define transdutor 3 //pino conectado ao transdutor

void setup(){
  pinMode(transdutor, OUTPUT); // seta o pino como saída
}
void loop(){
  tone(transdutor, 1350);
  delay(150);
  tone(transdutor, 1050);
  delay(150);
}
```

O que fizemos foi repetir a função tom com valores diferentes em um intervalo de tempo curto (o delay agora é de 150ms = 0,15 segundos).

Você deve ter notado que o som é ‘alto’. Podemos reduzir seu volume colocando um resistor em série com o *buzzer* (se você tiver um resistor variável, como um *trimpot* ou um potenciômetro, poderá ajustar o valor do volume – seu rádio ou seu amplificador, por exemplo, funcionam assim).





Em meu caso utilizei um resistor de 330 Ω (330 ohms, cores: laranja, laranja, marrom), que era o que eu tinha à mão. Se você utilizar um resistor de valor menor, por exemplo de 100 Ω (100 ohms, cores: marrom, preto, marrom), o som ficará mais alto. Valores maiores no resistor diminuem o som, e podem até impedir que ele seja reproduzido e o valor do resistor for muito alto.

Vamos ver mais combinações de frequências e durações para a função de geração de tons.

transdutorSimplesDoReMiFa

```
#define transdutor 3 //pino conectado ao transdutor

#define Do 262
#define Re 294
#define Mi 330
#define Fa 349
#define Sol 392
#define La 440
#define Si 494
#define Pausa 0

void setup(){
  pinMode(transdutor, OUTPUT); // seta o pino como saída
}

```

Note que, entre as linhas 3 e 10, foram incluídas definições de frequências das notas musicais (de novo, quem domina isso é seu professor de música…), para as quais foram dados os nomes usuais (por exemplo, La tem 440 Hz).

A função **setup** não mudou (só ficou em linhas diferentes).

A função **loop** ficou bem maior:

```
16 void loop(){
17   tone(transdutor, La,500);
18   delay(500);
19   tone(transdutor, Re,300);
20   delay(500);
21   tone(transdutor, Fa,250);
22   delay(500);
23   tone(transdutor, Sol, 250);
24   delay(500);
25   tone(transdutor, La, 250);
26   delay(500);
27   tone(transdutor, Re, 300);
28   delay(500);
29   tone(transdutor, Fa, 200);
30   delay(500);
31   tone(transdutor, Sol, 200);
32   delay(500);
33   tone(transdutor, Mi, 700);
34   delay(500);
35   tone(transdutor, Pausa, 200);
36   delay(500);
37   tone(transdutor, Sol, 500);
38   delay(500);
39   tone(transdutor, Do, 500);
40   delay(500);
41   tone(transdutor, Fa, 200);
42   delay(500);
43   tone(transdutor, Mi,200);
44   delay(500);
45   tone(transdutor, Sol,200);
46   delay(500);
47   tone(transdutor, Do,500);
48   delay(500);
49   tone(transdutor, Fa,200);
50   delay(500);
51   tone(transdutor, Mi,200);
52   delay(500);
53   tone(transdutor, Re,500);
54   delay(2000);
55 }
```

Note que, embora a função tenha ficado maior, os comandos são os mesmos de antes: **tone** e **delay**.

Teste com outras sequências e com outros valores.

Procure na internet: você vai achar formas bem sofisticadas de programar seu Arduino para tocar músicas. Divirta-se.