



**PDM: Projeto para Dispositivos Móveis**  
**Aula 09: Sensor Ultrassônico e Emissão de Sons no Arduino**

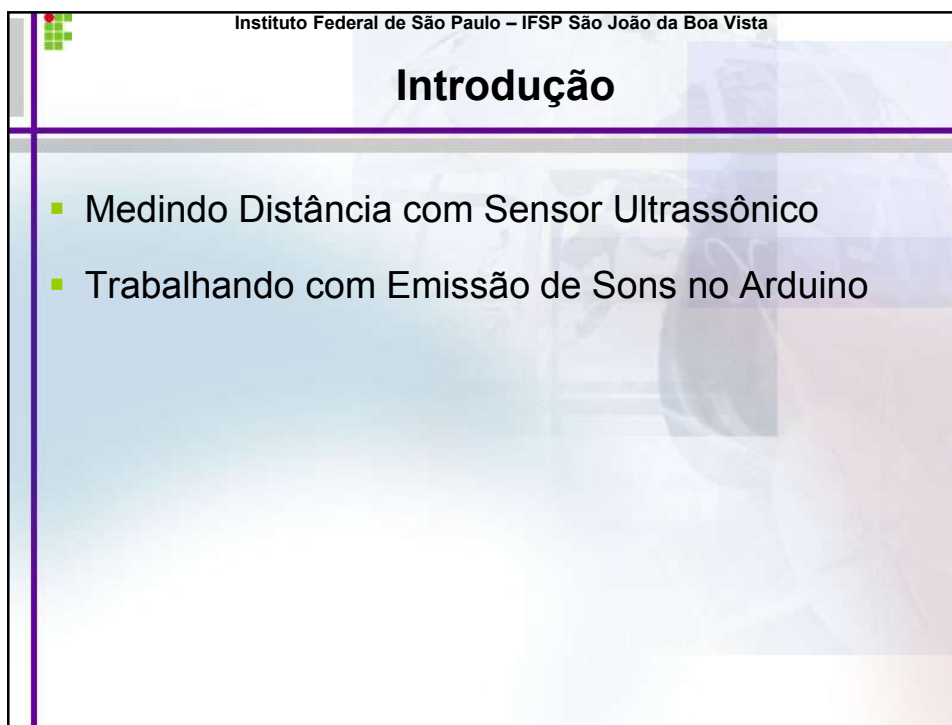
Breno Lisi Romano

<http://sites.google.com/site/blromano>

Instituto Federal de São Paulo – IFSP São João da Boa Vista  
Especialização em Desenvolvimento para Dispositivos Móveis



INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
SÃO PAULO  
Campus São João da Boa Vista



Instituto Federal de São Paulo – IFSP São João da Boa Vista

## Introdução

- Medindo Distância com Sensor Ultrassônico
- Trabalhando com Emissão de Sons no Arduino

## Medindo Distância com Sensor Ultrassônico



INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
SÃO PAULO  
Campus São João da Boa Vista

3

Instituto Federal de São Paulo – IFSP São João da Boa Vista

### ***Problem Based Learnd - PBL 01***

- **Problema:** Imprimir o valor da distância em centímetros medida ao se utilizar o Sensor Ultrassônico HC-SR04 integrado com o Arduino, realizando o cálculo matemático para encontrar a distância

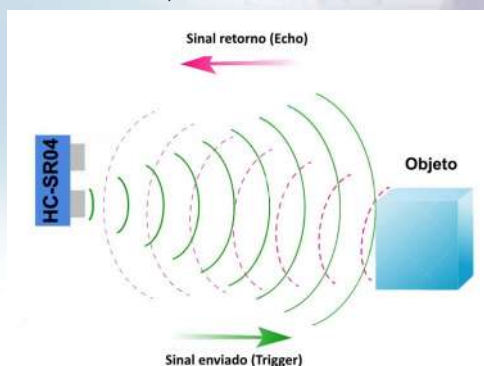


## Informações Necessárias (*Background*) (1)

- O Sensor Ultrassônico HC-SR04 é um componente muito comum em projetos com Arduino e permite que você faça leituras de distâncias entre 2 cm e 4 m, com precisão de 3 mm
- Pode ser utilizado para medir a distância entre o sensor e um objeto, como para acionar portas do microcontrolador, desviar um robô de obstáculos, acionar alarmes, etc.

## Informações Necessárias (*Background*) (2)

- O funcionamento do HC-SR04 se baseia no envio de sinais ultrassônicos pelo sensor (TRIGGER), que aguarda o retorno (ECHO) do sinal, e com base no tempo entre envio e retorno, calcula a distância entre o sensor e o objeto detectado



O Arduino envia um pulso para o Trigger no pino Trig e então “escuta” o retorno no pino Echo utilizando a função `pulseIn()`. A duração do pulso ECHO é igual ao tempo que o sinal ultrassônico viajou até um objeto e retornou ao sensor (em microssegundos). Isto pode ser calculado.

## Informações Necessárias (*Background*) (3)

- Para obter a distância em centímetro, precisamos das seguintes informações:
  - Velocidade do Som: 340,29 m/s
- Convertendo a Velocidade do som para cm/s:
  - 1m – 100 cm
  - $340,29 \text{ m/s} \times 100 = 34029 \text{ cm/s}$  (Velocidade do som em cm/s)
- Quanto tempo o som demora para percorrer 1 cm?
  - $34029 \text{ cm} - 1\text{s}$
  - $1 \text{ cm} - t$
  - $t = 1/34029 = 0,000029387 \text{ cm/s}$

## Informações Necessárias (*Background*) (4)

- Mas o Sensor Ultrassônico HC-SR04 trabalha em microssegundos, ou seja, ele retorna o tempo percorrido em microssegundos, convertendo:
  - Tempo que demora para percorrer 1 cm: 0,000029387 cm/s
  - Tempo em microssegundos:
  - $0,000029387 \text{ cm/s} \times 1000000 = 29,387 \text{ micros/cm}$

Anos	$3,17 \times 10^{-8}$
Meses	$3,8 \times 10^{-7}$
Semanas	$1,65 \times 10^{-6}$
Dias	$1,16 \times 10^{-5}$
Horas	$2,78 \times 10^{-4}$
Minutos	0,02
Segundos	1
Millissegundo	1000
Microsegundos	1000000
Nanosegundos	1000000000

**Ou seja, a cada 29,387 microssegundos o som irá  
viajar um centímetro**

## Informações Necessárias (*Background*) (5)

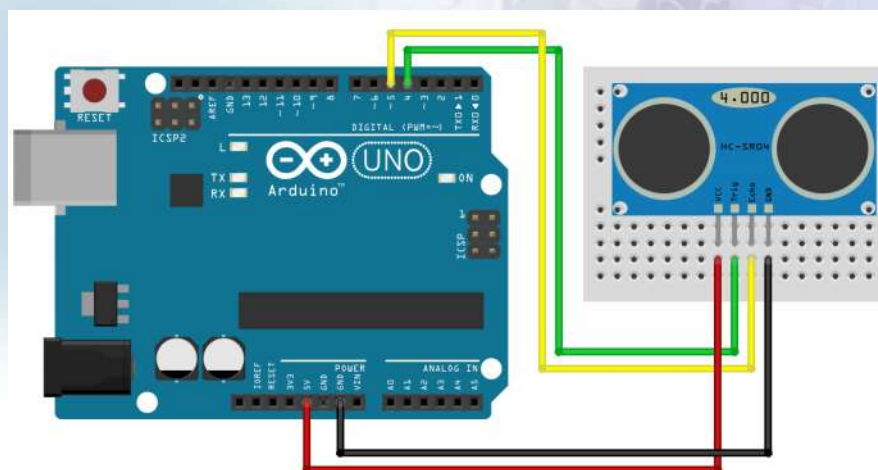
- Vamos supor que, ao utilizar o sensor HC-SR04 o tempo de retorno do sinal ECHO foi de 264,483 microssegundos. Quantos centímetros foram percorridos?
  - Sabemos que: 29,387 micros – 1 cm
  - Então:
    - 264,483 micros – D
    - 29,387 micros – 1 cm
    - $D = 264,483 / 29,387 = 9 \text{ cm}$
  - Mas, este é o tempo para ir, encontrar o objeto e retornar, ou seja, precisamos dividir por 2:
    - $d = D / 2 = 9 / 2 = 4,5 \text{ cm}$

**Fórmula para Encontrar a Distância em Centímetros:  $d = T / 29,387 / 2$**

## Componentes Utilizados

- Arduino UNO / Arduino MEGA
- Cabo USB
- Protoboard
- Sensor Ultrassônico HC-SR04
- Jumpers

## Construindo o Circuito no Protoboard



## Funções Necessárias para o Desenvolvimento do Sistema Embarcado

- **pulseIn(pino, valor, tempo):**
  - Parâmetros:
    - pino: o número do pino no qual você deseja ler o pulso (int)
    - valor: tipo de pulso a ler: tanto HIGH como LOW
    - tempo (opcional): o número de microsegundos a esperar para que o pulso comece; o padrão é um segundo (unsigned long)
  - Lê um pulso (tanto HIGH como LOW) em um pino
    - Por exemplo, se valor for HIGH, pulseIn() espera que o pino vá para HIGH, inicia a cronometragem, e então espera que o pino vá para LOW e para a cronometragem
    - Retorna a duração do pulso em microssegundos
    - Desiste e retorna 0 se nenhum pulso iniciar dentro de um tempo especificado
  - O tempo desta função foi determinado empiricamente e provavelmente dará erro em pulsos longos
    - Funciona com pulsos entre 10 microsegundos e 3 minutos

## Código Fonte para Solução

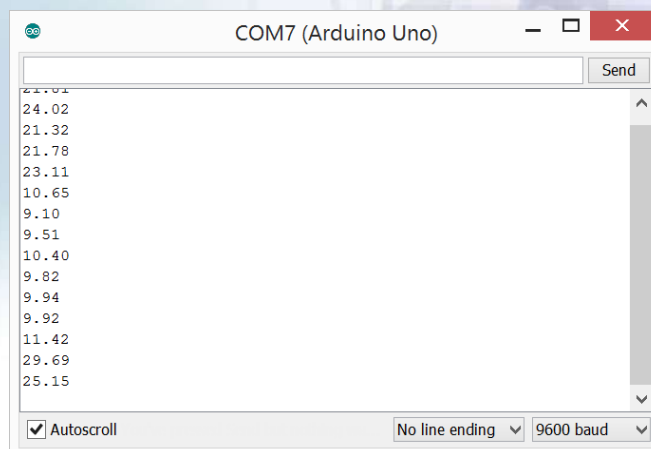
```

1 // the pins connected to the Ultrasonic sensor
2 int trigPin = 4;
3 int echoPin = 5;
4
5 void setup()
6 {
7     // set up serial
8     Serial.begin(9600);
9
10    pinMode(trigPin, OUTPUT);
11    pinMode(echoPin, INPUT);
12 }
13
14 void loop()
15 {
16     float distanceCentimeters;
17     int pulseLenMicroseconds;
18
19     digitalWrite(trigPin, LOW);
20     delayMicroseconds(20);
21
22     digitalWrite(trigPin, HIGH);
23     delayMicroseconds(100);
24
25     digitalWrite(trigPin, LOW);
26
27     // measure the pulse length from the echo pin
28     pulseLenMicroseconds = pulseIn(echoPin, HIGH);
29
30     // calculate the distance using the speed of sound
31     distanceCentimeters = pulseLenMicroseconds / 29.387 / 2;
32
33     // print it out over serial
34     Serial.println(distanceCentimeters);
35
36     delay(1000);
37 }

```

## Verificação e Simulação do Código Desenvolvido

- Embarcar o código-fonte na aplicação





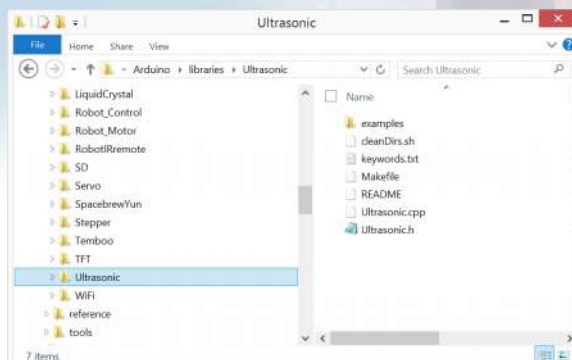
## Problem Based Learnd - PBL 02

- **Problema:** Imprimir o valor da distância em centímetros e em polegadas medida ao se utilizar o Sensor Ultrassônico HC-SR04 integrado com o Arduino, utilizando uma Biblioteca Externa



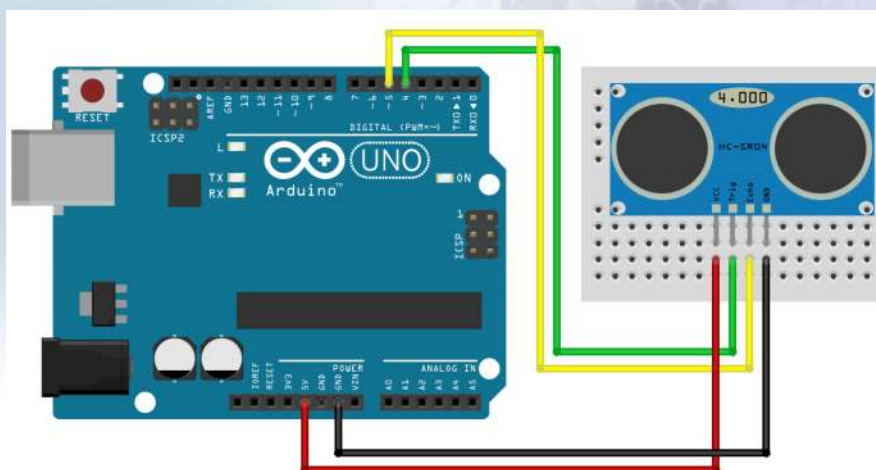
## Importação de Biblioteca do Sensor Ultrassônico HC-SR04

- Para facilitar, já existe uma biblioteca para o Sensor Ultrassônico HC-SR04 que pode ser baixada no Portal da Disciplina
  - Após o download, descompacte o arquivo .zip e mova-o para a pasta C:\Program Files (x86)\Arduino\libraries e reinicie a IDE do Arduino





## Construindo o Circuito no Protoboard



## Código Fonte para Solução

```

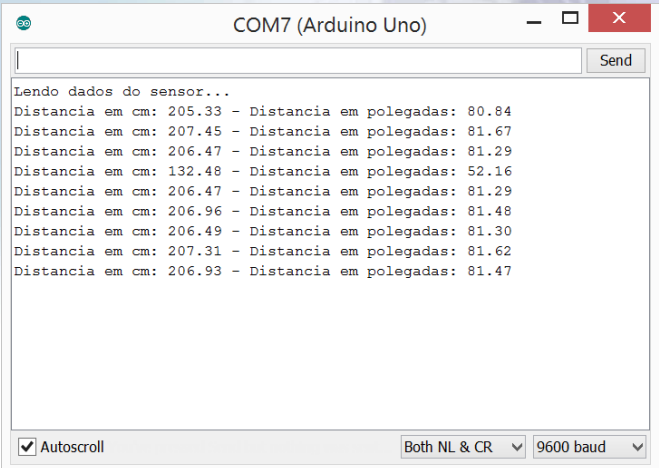
1 //Carrega a biblioteca do sensor ultrassônico
2 #include <Ultrasonic.h>
3
4 //Define os pinos para o trigger e echo
5 #define pino_trigger 4
6 #define pino_echo 5
7
8 //Inicializa o sensor nos pinos definidos acima
9 Ultrasonic ultrasonic(pino_trigger, pino_echo);
10
11 void setup()
12 {
13   Serial.begin(9600);
14   Serial.println("Lendo dados do sensor...");
15 }
16
17 void loop()
18 {
19   //Le as informações do sensor, em cm e pol
20   float cmMsec, inMsec;
21   long microsec = ultrasonic.timing();
22   cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
23   inMsec = ultrasonic.convert(microsec, Ultrasonic::IN);
24
25   //Exibe informações no serial monitor
26   Serial.print("Distancia em cm: ");
27   Serial.print(cmMsec);
28   Serial.print(" - Distancia em polegadas: ");
29   Serial.println(inMsec);
30   delay(1000);
31 }

```

Instituto Federal de São Paulo – IFSP São João da Boa Vista

## Verificação e Simulação do Código Desenvolvido

- Embarcar o código-fonte na aplicação



## Trabalhando com Emissão de Sons no Arduino



INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
SÃO PAULO  
Campus São João da Boa Vista

20

## Informações Necessárias (1)

- Será utilizado um Piezo Elétrico (Buzzer) para produzir som
  - Piezoelectricidade é a capacidade de alguns cristais gerarem tensão elétrica por resposta a uma pressão mecânica
- Para isso, pode-se colocar uma onda de tensão na faixa de frequência audível, assim o Piezo vibrará na mesma frequência, emitindo som



Transdutor piezo elétrico típico



Buzzer TMB 12A05: Piezo Encapsulado

## Informações Necessárias (2)

- Com a função **Tone()**, pode-se tocar facilmente uma Nota quando se conecta um Piezo em uma Pino Digital
  - `tone(pino, frequência)`
  - `tone(pino, frequência, duração)`

## ***Problem Based Learnd - PBL 03***

- **Problema:** Fazer uma melodia utilizando o Buzzer TMB 12A05 utilizando uma matriz para armazenar as notas musicais da melodia

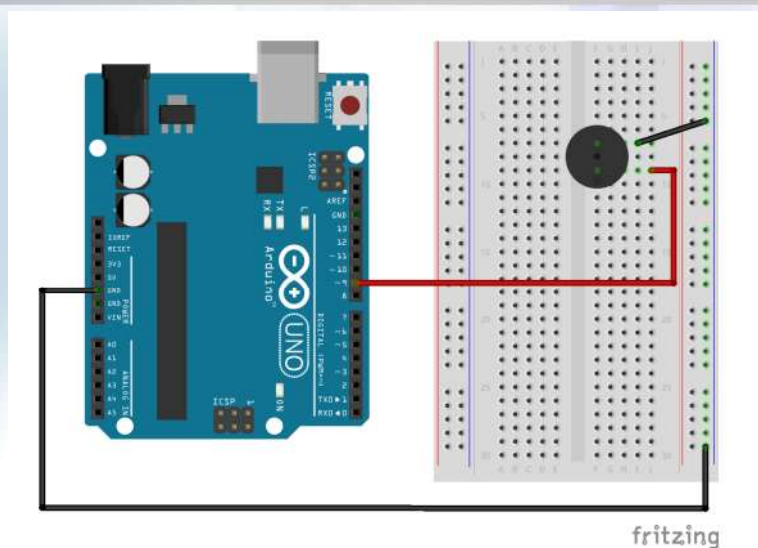


Buzzer TMB 12A05: Piezo Encapsulado

## **Componentes Utilizados**

- Arduino UNO / Arduino MEGA
- Cabo USB
- Protoboard
- Buzzer TMB 12A05
- Jumpers

## Construindo o Circuito no Protoboard



## Código Fonte para Solução (1)

```

1 // include our list of note pitches
2 #include "pitches.h"
3
4 // the pin the speaker is attached to
5 int speakerPin = 9;
6
7 // the notes in our melody and their duration in fractions of a second
8 // e.g. quarter note = 4, eighth note = 8, etc.
9 const int melody[][2] =
10 {
11     {NOTE_C4, 4},
12     {NOTE_G3, 8},
13     {NOTE_G3, 8},
14     {NOTE_A3, 4},
15     {NOTE_G3, 4},
16     {NOTE_BLANK, 4},
17     {NOTE_B3, 4},
18     {NOTE_C4, 4}
19 };

```

Incluir o arquivo "pitches.h", disponível no portal da disciplina, dentro da pasta do Sketch desenvolvido. Este arquivo contém a lista de notas musicais e suas respectivas frequências

## Código Fonte para Solução (2)

```

21 void setup()
22 {
23     // figure out the number of notes in our melody
24     int numberOfNotes = sizeof(melody) / sizeof(melody[0]);
25
26     // iterate over the notes of the melody
27     for (int thisNote = 0; thisNote < numberOfNotes; thisNote++)
28     {
29         // grab our note and note duration from our array
30         int thisNoteTone = melody[thisNote][0];
31         int thisNoteDuration = melody[thisNote][1];
32
33         // to calculate the note duration in ms
34         int noteDurationMS = 1000 / thisNoteDuration;
35
36         // play the note
37         tone(speakerPin, thisNoteTone, noteDurationMS);
38
39         // to distinguish the notes, set a minimum time between them.
40         // the note's duration + 30% seems to work well:
41         delay(noteDurationMS * 1.30);
42     }
43 }
44
45 void loop()
46 {
47     // no need to repeat the melody.
48     // do nothing
49 }

```

## Verificação e Simulação do Código Desenvolvido

- Embarcar o código-fonte na aplicação
- Verificar se a melodia é tocada
  - Para tocá-la novamente, basta apertar o reset do Arduino, pois o código que toca a melodia está no setup() do código



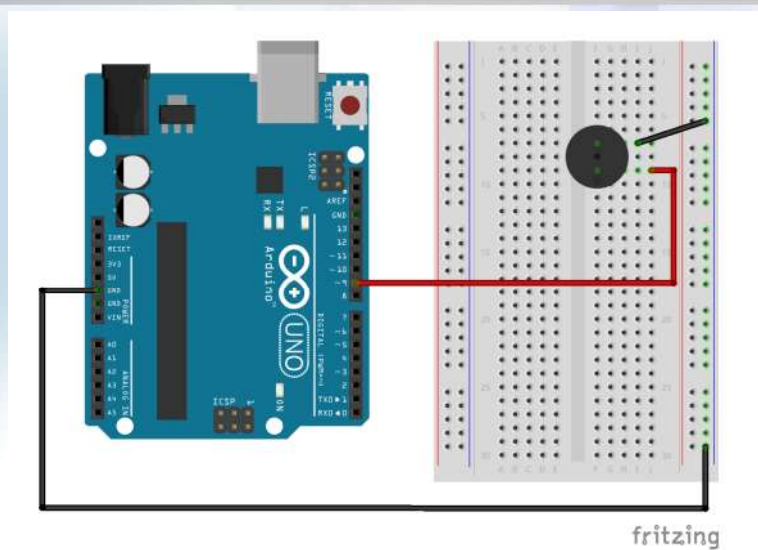
## Problem Based Learnd - PBL 04

- **Problema:** Criar o som de uma sirene utilizando o o Buzzer TMB 12A05



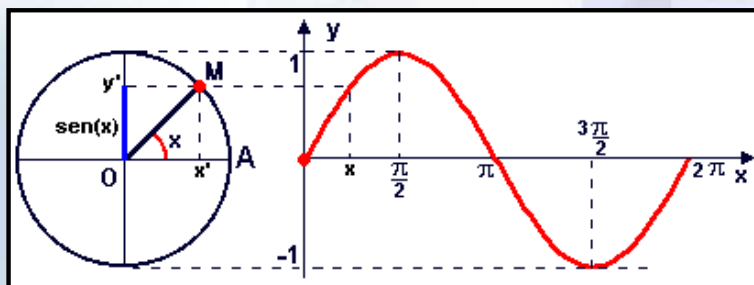
Buzzer TMB 12A05: Piezo Encapsulado

## Construindo o Circuito no Protoboard



Instituto Federal de São Paulo – IFSP São João da Boa Vista

## Fundamentos Matemáticos para Reproduzir o Som da Sirene



Radianos	Graus
$2\pi$ rad	$360^\circ$
$\pi$ rad	$180^\circ$
$\pi/2$ rad	$90^\circ$
$\pi/3$ rad	$60^\circ$
$\pi/4$ rad	$45^\circ$
$\pi/6$ rad	$30^\circ$

Para converter Graus em Radianos, basta Multiplicarmos o número por  $\pi/180$ :  
 Num Radianos = Num em Graus \*  $\pi/180$

Instituto Federal de São Paulo – IFSP São João da Boa Vista

## Código Fonte para Solução

```

1 float seno;           //armazena o valor do seno que faz o som se elevar ou diminuir
2 int frequencia;       //utiliza o valor do seno e converte para uma frequencia
3
4 void setup() {
5     //define o pino 9 como saída
6     pinMode(9,OUTPUT);
7 }
8
9 void loop() {
10    for(int x=0;x<180;x++){ //garante que o valor do seno não fique negativo
11        //converte graus para radiando e depois obtém o valor do seno
12        seno=(sin(x*3.1416/180));
13
14        //gera uma frequência a partir do valor do seno
15        frequencia = 2000+(int(seno*1000));
16        tone(9, frequencia);
17        delay(2);
18    }
19 }
20

```

Alterando os valores de 2.000 e 1.000 no cálculo da frequência, pode-se gerar sons diferentes para o alarme

Instituto Federal de São Paulo – IFSP São João da Boa Vista

## Verificação e Simulação do Código Desenvolvido

- Embarcar o código-fonte na aplicação
- Depois do upload do código, haverá uma pequena espera e seu piezo começará a emitir sons
  - Se tudo estiver funcionando como planejado, você ouvirá um alarme do tipo sirene, como um alarme de carro

Instituto Federal de São Paulo – IFSP São João da Boa Vista

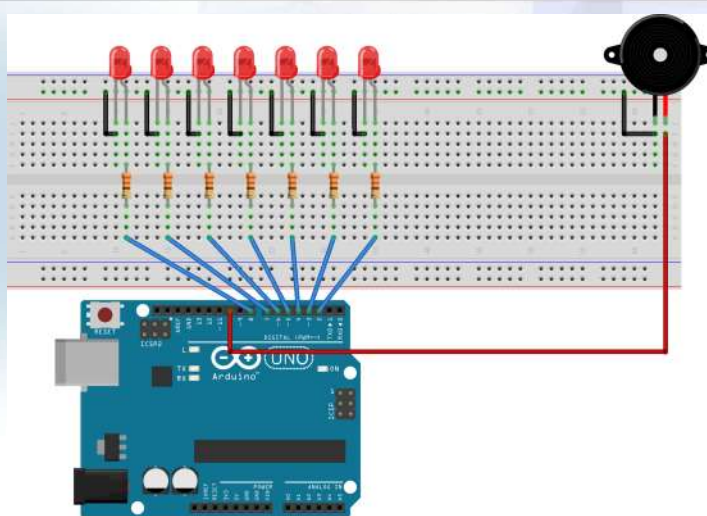
## *Problem Based Learnd - PBL 05*

- **Problema:** Criar um teclado musical em que as notas Dó, Ré, Mi, Fá, Sol, Lá e Si podem ser entradas pelo usuário através do teclado e a nota será emitida pelo Piezo TMB 12A05 e o LED correspondente da Nota Musical deverá acender.

## Componentes Utilizados

- Arduino UNO / Arduino MEGA
- Cabo USB
- Protoboard
- Buzzer TMB 12A05
- 07 resistor de 330 ohms
- 07 LED Vermelho
- Jumpers

## Construindo o Circuito no Protoboard



fritzing

## Código Fonte para Solução (1)

```

1 #define NOTA_DO 262
2 #define NOTA_RE 294
3 #define NOTA_MI 330
4 #define NOTA_FA 349
5 #define NOTA_SOL 392
6 #define NOTA_LA 440
7 #define NOTA_SI 494
8
9 int ledNOTA_DO = 2;
10 int ledNOTA_RE = 3;
11 int ledNOTA_MI = 4;
12 int ledNOTA_FA = 5;
13 int ledNOTA_SOL = 6;
14 int ledNOTA_LA = 7;
15 int ledNOTA_SI = 8;
16
17 int pinoAudio = 10;
18 int nota;
19
20 void setup() {
21   Serial.begin(9600);
22   pinMode(pinoAudio, OUTPUT);
23   pinMode(ledNOTA_DO, OUTPUT);
24   pinMode(ledNOTA_RE, OUTPUT);
25   pinMode(ledNOTA_MI, OUTPUT);
26   pinMode(ledNOTA_FA, OUTPUT);
27   pinMode(ledNOTA_SOL, OUTPUT);
28   pinMode(ledNOTA_LA, OUTPUT);
29   pinMode(ledNOTA_SI, OUTPUT);
30 }

```

## Código Fonte para Solução (2)

```

32 void loop() {
33   if (Serial.available() > 0) {
34     nota = Serial.read();
35
36     if (nota == 49) { /* nota sera emitida quando a tecla 1 for apertada */
37       Serial.println("DO");
38       tone(pinoAudio, NOTA_DO);
39       digitalWrite(ledNOTA_DO, HIGH);
40       delay(250);
41       digitalWrite(ledNOTA_DO, LOW);
42       noTone(pinoAudio); //parar o som da nota
43     }
44
45     if (nota == 50) { /* nota sera emitida quando a tecla 2 for apertada */
46       Serial.println("RE");
47       tone(pinoAudio, NOTA_RE);
48       digitalWrite(ledNOTA_RE, HIGH);
49       delay(250);
50       digitalWrite(ledNOTA_RE, LOW);
51       noTone(pinoAudio);
52     }
53
54     if (nota == 51) { /* nota sera emitida quando a tecla 3 for apertada */
55       Serial.println("MI");
56       tone(pinoAudio, NOTA_MI);
57       digitalWrite(ledNOTA_MI, HIGH);
58       delay(250);
59       digitalWrite(ledNOTA_MI, LOW);
60       noTone(pinoAudio);
61     }
62   }
63 }

```

## Código Fonte para Solução (3)

```
63 if (nota==52){ /* nota sera emitida quando a tecla 4 for apertada */
64     Serial.println("FA");
65     tone(pinoAudio, NOTA_FA);
66     digitalWrite(ledNOTA_FA, HIGH);
67     delay(250);
68     digitalWrite(ledNOTA_FA, LOW);
69     noTone(pinoAudio);
70 }
71
72 if (nota==53){ /* nota sera emitida quando a tecla 5 for apertada */
73     Serial.println("SOL");
74     tone(pinoAudio, NOTA_SOL);
75     digitalWrite(ledNOTA_SOL, HIGH);
76     delay(250);
77     digitalWrite(ledNOTA_SOL, LOW);
78     noTone(pinoAudio);
79 }
80
81 if (nota==54){ /* nota sera emitida quando a tecla 6 for apertada */
82     Serial.println("LA");
83     tone(pinoAudio, NOTA_LA);
84     digitalWrite(ledNOTA_LA, HIGH);
85     delay(250);
86     digitalWrite(ledNOTA_LA, LOW);
87     noTone(pinoAudio);
88 }
```

## Código Fonte para Solução (4)

```
90 if (nota==55){ /* nota sera emitida quando a tecla 7 for apertada */
91     Serial.println("SI");
92     tone(pinoAudio, NOTA_SI);
93     digitalWrite(ledNOTA_SI, HIGH);
94     delay(250);
95     digitalWrite(ledNOTA_SI, LOW);
96     noTone(pinoAudio);
97 }
98 }
99 }
```



Instituto Federal de São Paulo – IFSP São João da Boa Vista

## Verificação e Simulação do Código Desenvolvido

- Embarcar o código-fonte na aplicação
- Verificar se as notas musicais são tocadas e os LEDs acesos quando as teclas são apertadas no Serial Monitor
  - 1 – Dó
  - 2 – Ré
  - 3 – Mi
  - 4 – Fá
  - 5 – Sol
  - 6 – Lá
  - 7 – Si

Instituto Federal de São Paulo – IFSP São João da Boa Vista

## *Problem Based Learnd - PBL 06*

- **Problema:** Utilizar o Piezo TMB 12A05 para detectar a vibração em um objeto sólido e acender um LED quando for detectado esta vibração
  - Por exemplo, uma batida na porta faz com que a Luz do LED pisque

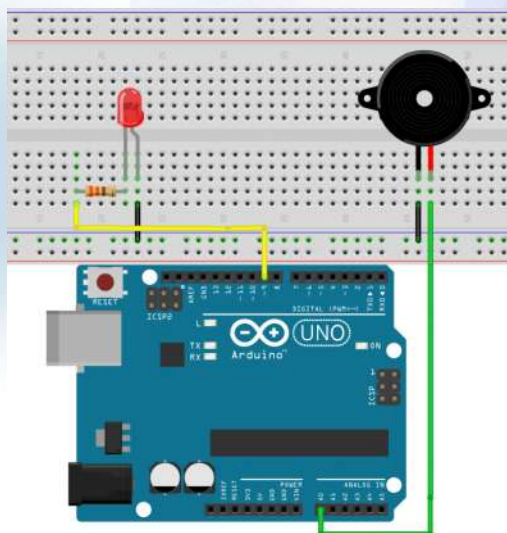
## Informações Necessárias

- Um Buzzer/Piezo funciona quando uma corrente elétrica passa pelo material cerâmico do disco, fazendo com que ele mude de forma e produza um som
- O disco também funciona de forma inversa: quando se bate nele ou ele sofre algum tipo de pressão, a força no material provoca a geração de uma corrente elétrica
  - Pode-se ler essa corrente utilizando o Arduino, e é justamente isso que será feito agora, com a criação de um sensor de batida
  - É uma medida analógica, pois depende da intensidade da batida
  - É muito sensível

## Componentes Utilizados

- Arduino UNO / Arduino MEGA
- Cabo USB
- Protoboard
- Buzzer TMB 12A05
- 01 resistor de 330 ohms
- 1 LED Vermelho
- Jumpers

## Construindo o Circuito no Protoboard



## Código Fonte para Solução

```

1 int ledPin = 9; // LED no pino digital 9
2 int piezoPin = A0; // Piezo no pino analógico 5
3 int threshold = 200; // O valor do sensor a ser atingido antes da ativação
4 int sensorValue = 0; // Uma variável para armazenar o valor lido do sensor
5
6 void setup() {
7   pinMode(ledPin, OUTPUT); // Define o ledPin como OUTPUT
8   // Pisca o LED duas vezes, para mostrar que o programa iniciou
9   digitalWrite(ledPin, HIGH); delay(150); digitalWrite(ledPin, LOW); delay(150);
10  digitalWrite(ledPin, HIGH); delay(150); digitalWrite(ledPin, LOW); delay(150);
11 }
12 void loop() {
13   sensorValue = analogRead(piezoPin); // Lê o valor do sensor
14   if (sensorValue >= threshold) { // Se uma batida for detectada, defina o brilho como máximo
15     digitalWrite(ledPin, HIGH);
16     delay(150);
17     digitalWrite(ledPin, LOW);
18   }
19 }

```

Instituto Federal de São Paulo – IFSP São João da Boa Vista

## Verificação e Simulação do Código Desenvolvido

- Embarcar o código-fonte na aplicação
- Verificar se, ao bater a mão na mesa próximo do Buzzer, a luz do LED pisca

Instituto Federal de São Paulo – IFSP São João da Boa Vista

## Miniprojeto 16

- Construir um detector de presenças utilizando o Sensor Ultrassônico HC-SR04 que irá emitir um sinal de sirene com o Piezo TMB 12A05 e ficar acendendo e apagando um LED Vermelho.
  - A distância inicialmente definida para se detectar a presença é de 50 cms
  - Só irá emitir o sinal sonoro se a distância for menor que 50 cms
- Desenvolver um aplicativo Android que permita ativar ou desativar o detector de presenças bem como reconfigurar a distância de detecção inicial, em centímetros

## Miniprojeto 17

- Desenvolver um Aplicativo Android que possua um teclado virtual musical que, conforme for sendo selecionado, as notas Dó, Ré, Mi, Fá, Sol, Lá e Si serão enviadas para o Arduino para que seja emitida pelo Piezo TMB 12A05, acendendo também o LED correspondente da Nota Musical
  - Realizar a integração utilizando o Módulo Bluetooth HC-05

## Miniprojeto 18: DESAFIO GENIUS (Opcional)

- Um jogo muito famoso na década de 1980, que buscava estimular a memória do jogador através de cores e sons, é o Genius



- Construir um circuito no Arduino que simule o funcionamento do Genius, acendendo 4 LEDs e emitindo sinais sonoros para cada um
- A sequência lógica que os LEDs acendem vai aumentando a cada etapa (Primeiro acende apenas 1, depois 2, depois 3, etc... ) até o usuário errar
- O usuário deverá enviar as ordens corretas das sequências emitidas pelo GENIUS através de uma aplicação Android utilizando conexão Bluetooth

# **PDM: Projeto para Dispositivos Móveis**

## **Aula 09: Sensor Ultrassônico e Emissão de Sons no Arduino**

Breno Lisi Romano

**Obrigado!**

**Instituto Federal de São Paulo – IFSP São João da Boa Vista**  
**Especialização em Desenvolvimento para Dispositivos Móveis**



INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
SÃO PAULO  
Campus São João da Boa Vista