



# Projeto 07

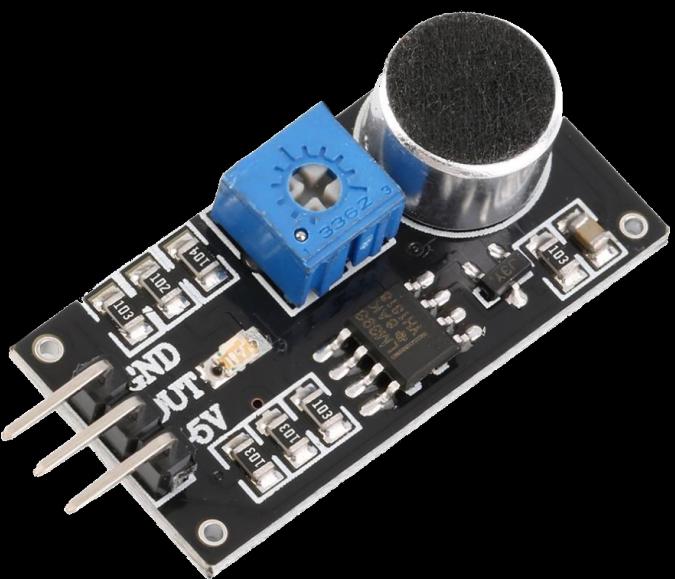
## Controle Sonoro – Prática

Jan K. S. – janks@puc-rio.br

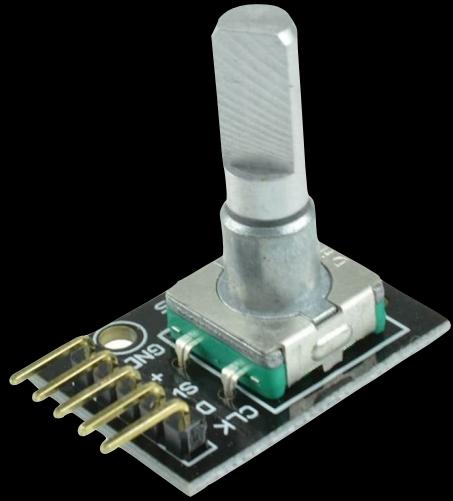
ENG1419 – Programação de Microcontroladores

# Testes Iniciais

19



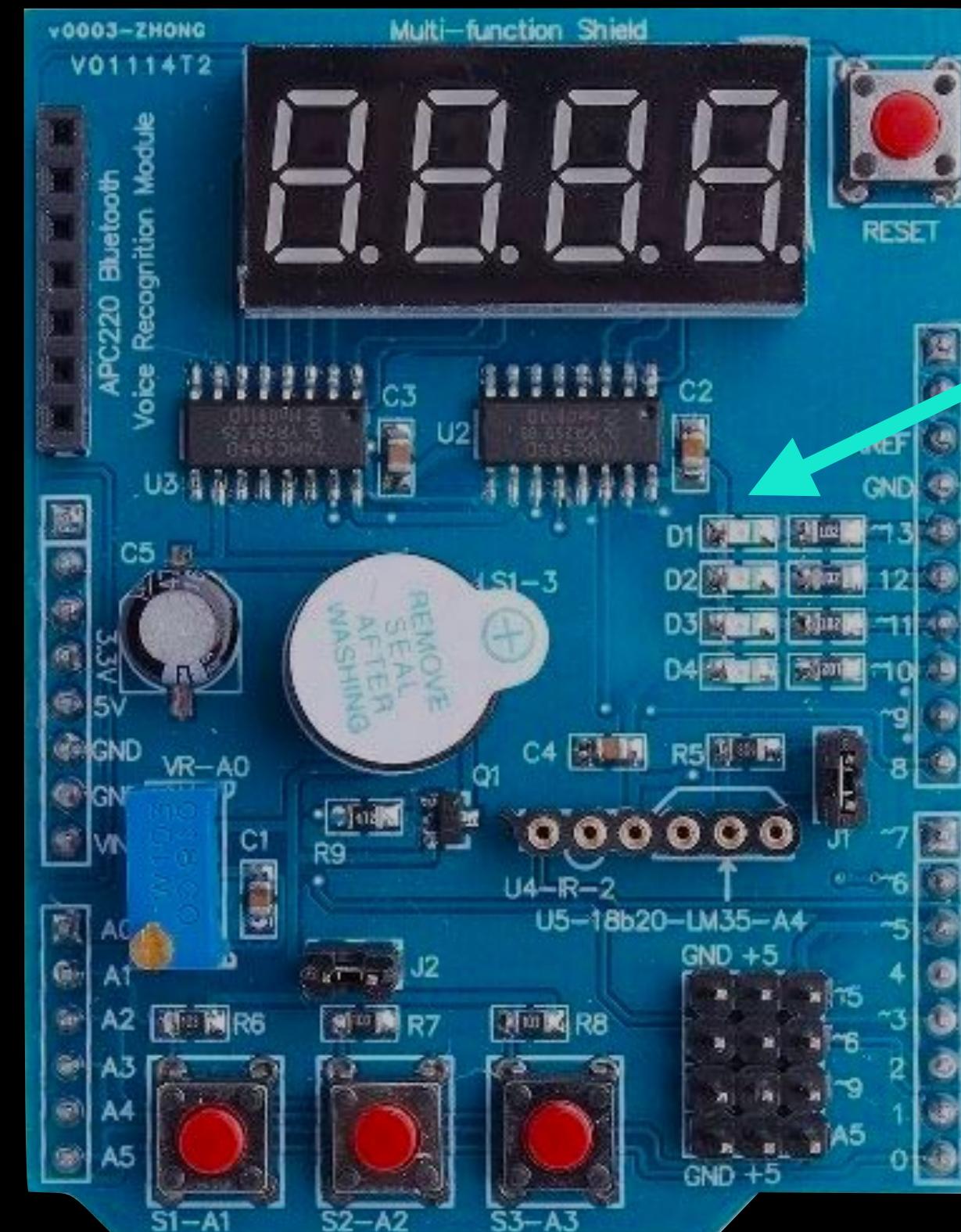
20,21



A5 5



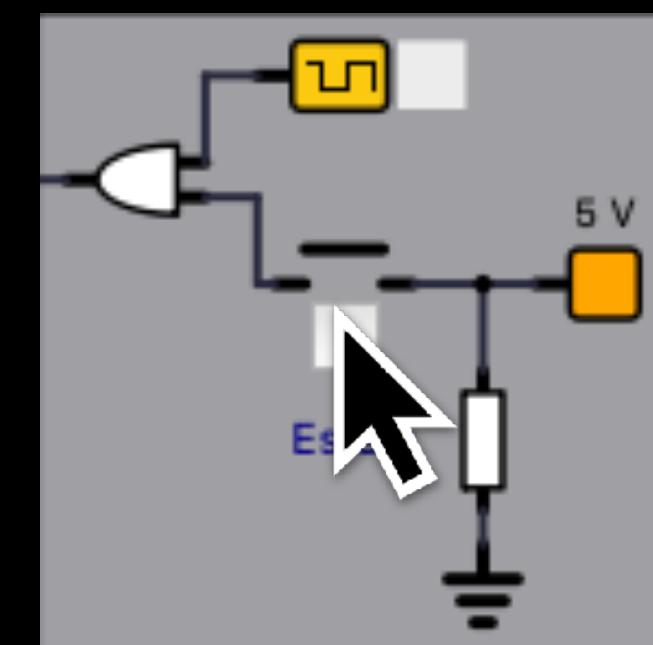
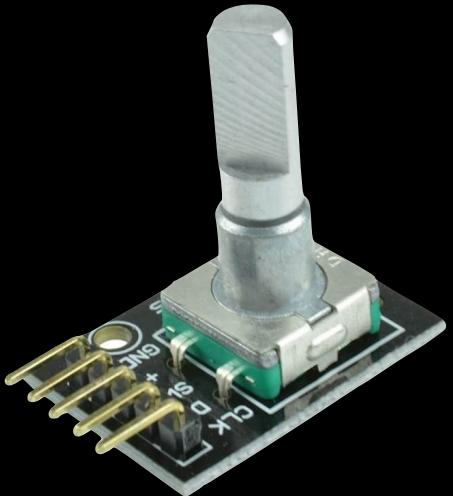
pinos 4,7 e 8



pinos 13,12,11 e 10

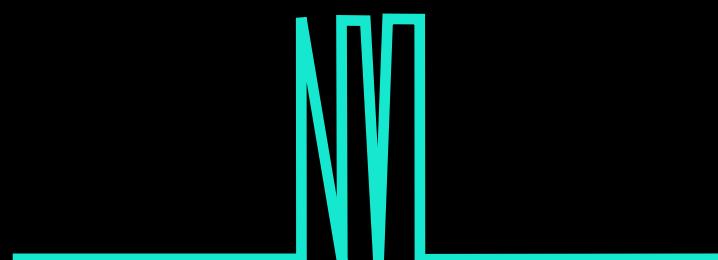
pinos A1, A2 e A3

Pinos Usados pelos Componentes



clique rápido

```
possível ajuste  
de tempo  
void somDetectado () {  
    unsigned long instanteAtual = millis(); ↴  
    if (instanteAtual > instanteAnterior + 50) {  
        Serial.println("som!");  
        instanteAnterior = instanteAtual;  
    }  
}
```



Limitações da Simulação



## Testes Iniciais

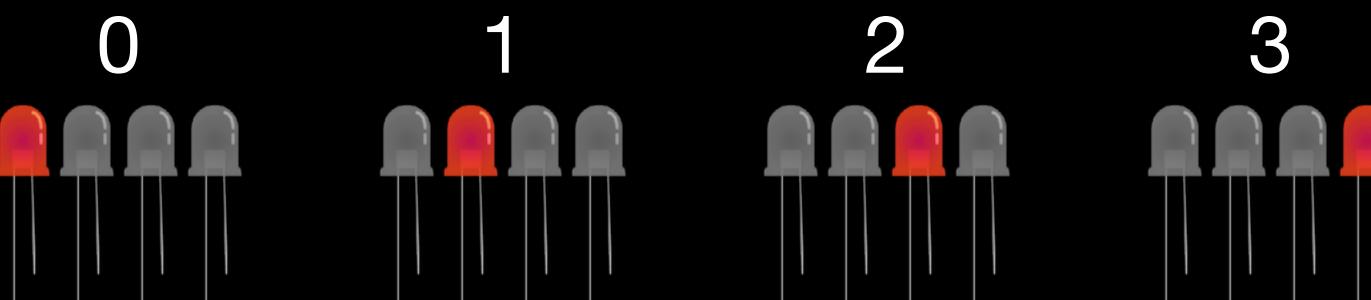
Ao apertar o Botão 1, **toque a campainha 1 vez durante 500ms**, com uma frequência 440Hz.  
↪ DICA: use a GButton e não se esqueça de colocar o pino A5 (terra da campainha) no LOW.

Ao apertar o Botão 2, **toque a campainha com uma frequência 220Hz continuamente**. Ao soltar, **pare de tocar a campainha**.  
↪ DICA: use a GButton e a função noTone.

Conte quantas vezes houve um estalar de dedos, e **exiba essa contagem no display de 7 segmentos**.  
↪ DICA: use a receita com attachInterrupt e millis.

**Imprima a contagem** do item anterior via serial **a cada 500ms**, usando a função millis dentro do loop.

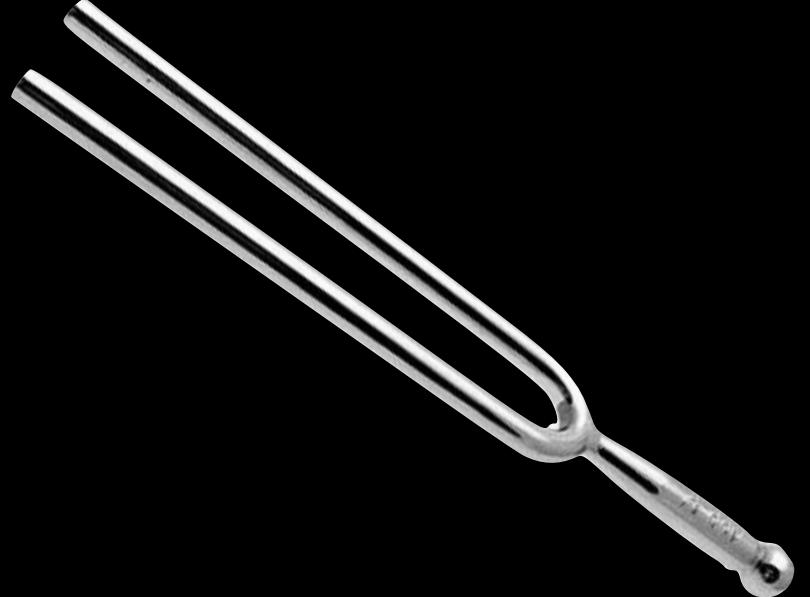
Acenda os LEDs de acordo com o valor do **módulo do resto da divisão por 4 da posição do encoder**.  
↪ DICA: use getPosition( ), % e abs( ) no loop.



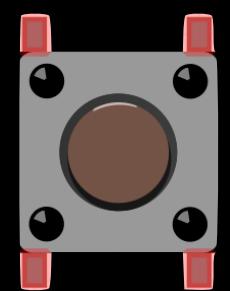
# Implementação



Controle Sonoro Musical



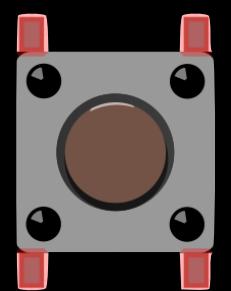
Botão 1  
modo Afinador



modo = 1;

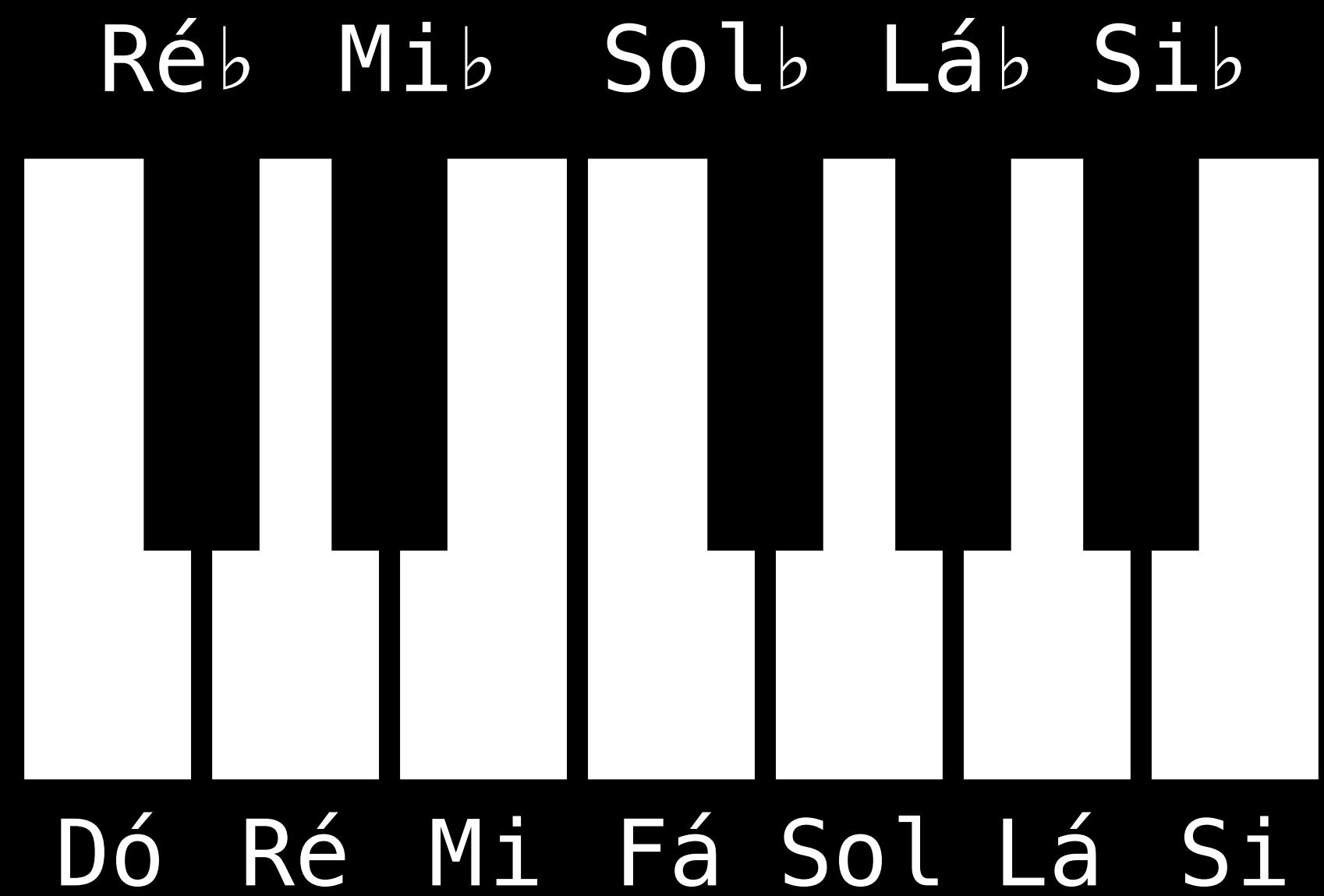


Botão 2  
modo Metrônomo



modo = 2;

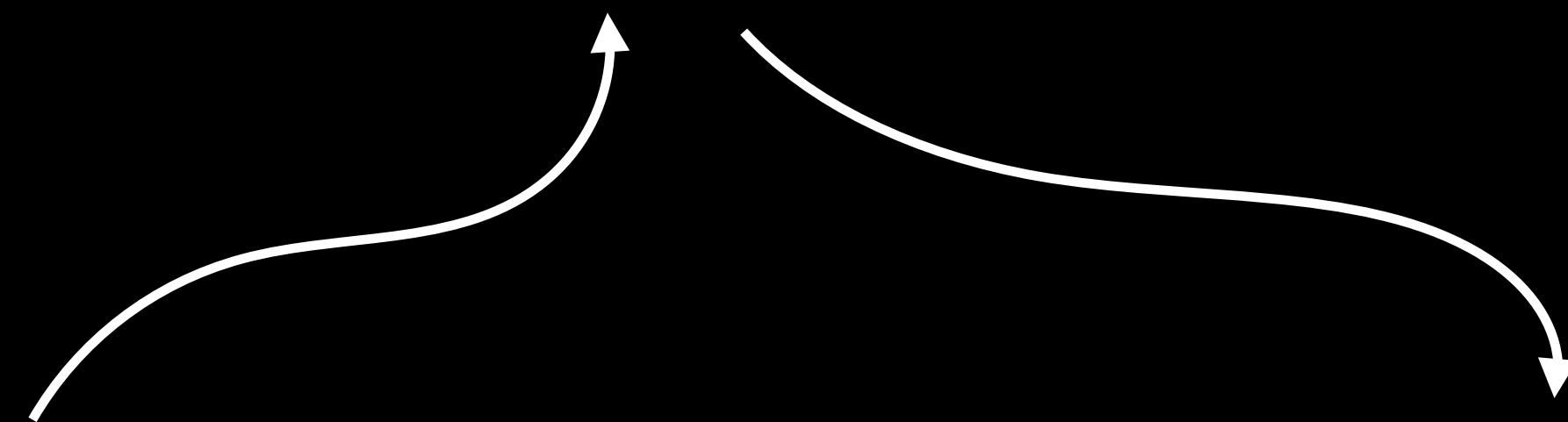
Afinador e Metrônomo



```
char* nomeDasNotas[] = {"D0 ", "REb", "RE ", ...};  
int frequencias[] = {131, 139, 147, ...};
```

```
char* nomeDasNotas[] = {"D0 ", "REb", "RE ", ...};
```

```
int frequencias[] = {131, 139, 147, ...};
```



```
TocarEMostrarNota(0, 100);  
índice duração
```

♪  
131 Hz  
100 ms

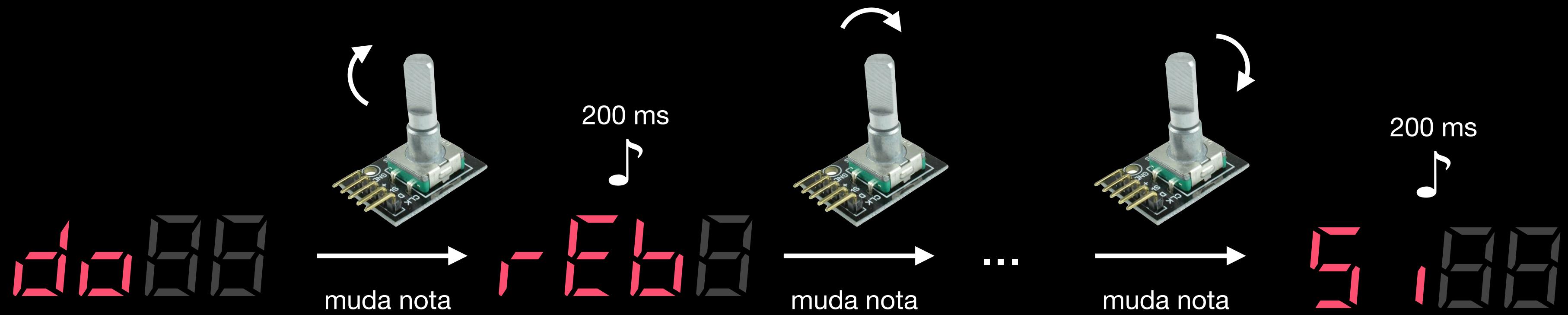
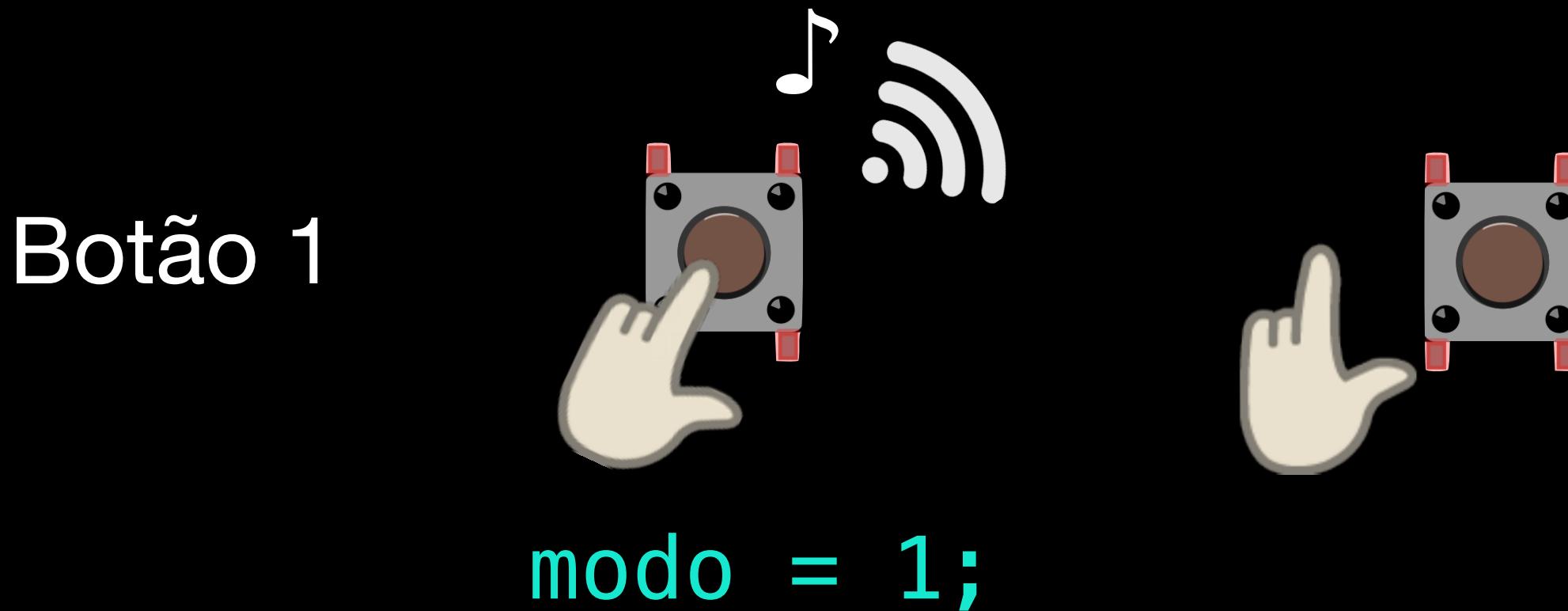
*do 88*

```
TocarEMostrarNota(2, -1);
```

♪  
147 Hz  
contínuo

*r E 88*

Função para Tocar e Mostrar Nota



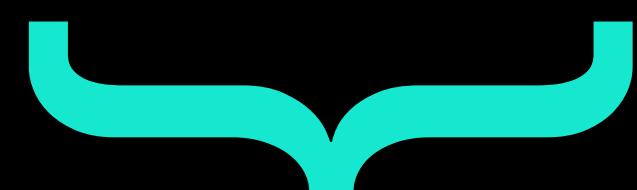
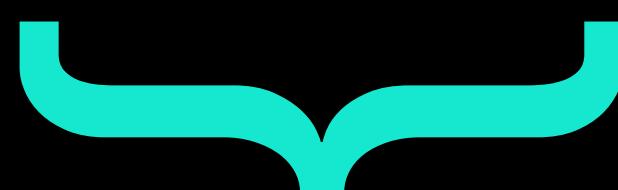
Controle do Afinador via Botão 1 e Encoder

```
int notaAtual
```

0 → 1 → 2 → 3 → ... → 10 → 11

```
int posicao = encoder.getPosition();
```

... → -2 → -1 → 0 → 1 → 2 → ... → 10 → 11 → 12 → 13 → ...



```
encoder.setPosition(0);
```

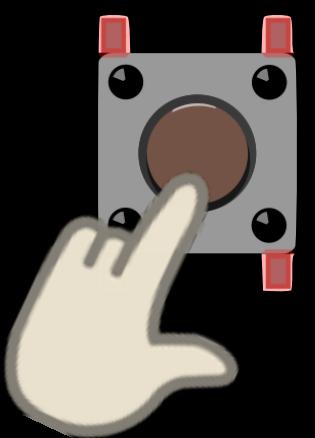
```
encoder.setPosition(11);
```

# Correção da Posição do Encoder para os Limites da Nota

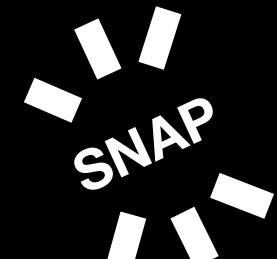


Metrônomo para Marcação de Ritmo

Botão 2



modo = 2;



instante1



intervalo



instante2

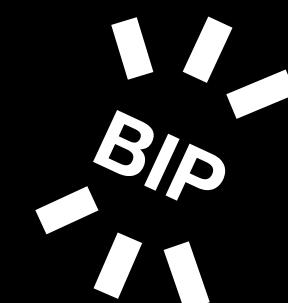
unsigned long intervalo = instante2 - instante1;

int batidasPorMinuto = 60000 / intervalo;

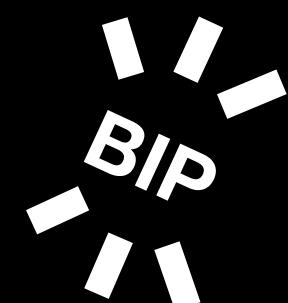
loop:

8875

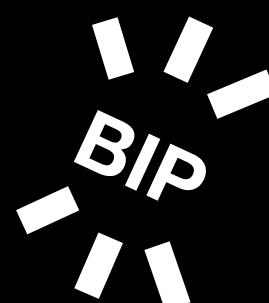
batidasPorMinuto



intervalo



intervalo



...

Metrônomo Digital com Base nos Estalos



## Implementação

Crie uma função que receba o índice da nota e uma duração em ms. Toque a nota com essa duração ou toque continuamente se esta for menor que zero. Exiba o nome da nota no display.

Ao apertar o Botão 1, entre no modo 1 (afinador) e toque a nota atual continuamente. Ao soltar, pare de tocar.

↪ DICA: use uma variável global para indicar o índice da nota atual nos vetores.

Ao mudar a posição do encoder, mude o índice da nota atual e toque-a por 200 ms. Mantenha os valores sempre entre 0 e 11.

↪ DICA: use a setPosition para restringir os valores da posição, como explicado uns slides atrás.

Ao apertar o Botão 2, entre no modo 2 (metrônomo). Na função do sensor, detecte dois estalos consecutivos e exiba o período entre eles (em batidas por minutos) no display.

↪ DICA: use uma segunda variável global para salvar o instante do primeiro estalo.

Após os 2 estalos, toque um bip rápido (220Hz em 200ms) periodicamente (usando o intervalo entre os dois estalos). Pare de tocar ao apertar o Botão 1 ou o 2.

↪ DICA: use a millis dentro do loop para contar o tempo entre os bips.

# Aperfeiçoamento



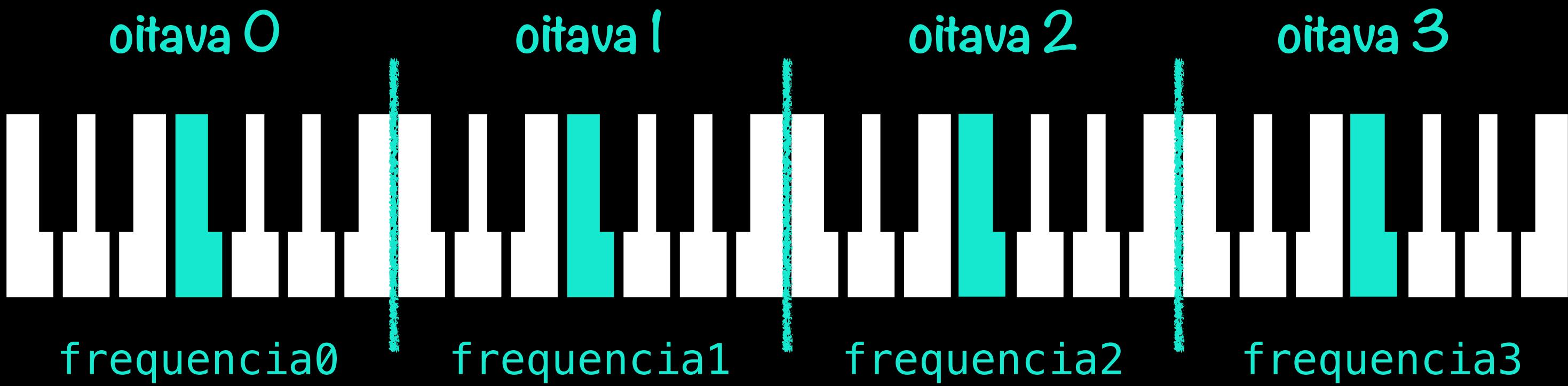
p07b\_implementacao.ino

cópia  
----->



p07c\_aperfeicoamento.ino

Cópia do Código da Implementação para o Aperfeiçoamento



$$\text{frequencia}_N = \text{frequencia}_0 \times 2^N$$



```
frequencia = frequencias[indice] * pow(2, oitava);
```

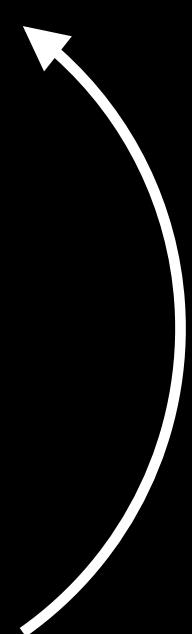
Posição do Encoder	Índice da Nota	Oitava	Display
0	0	0	do80
1	1	0	rE60
2	2	0	rE80
...	...	...	...
10	10	0	5_ib0
11	11	0	5_ib0
12	0	1	do81
13	1	1	rE61
14	2	1	rE81
...	...	...	...
22	10	1	5_ib1
23	11	1	5_ib1
24	0	2	do82
25	1	2	rE62
26	2	2	rE82
...	...	...	...
34	10	2	5_ib2
35	11	2	5_ib2

Mudança da Oitava

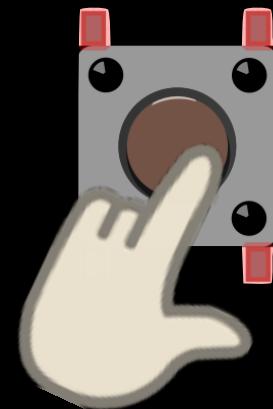


Caixinha de Música no Botão 3

```
char* nomeDasNotas[] = {"D0 ", "REb", "RE ", ...};  
int frequencias[] = {131, 139, 147, ...};  
  
int indicesDeNotaDaMusica[] = {7, 2, 0, 11...};  
int oitavasDaMusica[] = {0, 1, 1, 0, ...};  
int intervalosEntreNotas[] = {1000, 1000, 167, 167...};
```



Botão3



modo = 3

toque a primeira nota  
registre o instante

loop do Arduino:

se já passou o tempo esperado desde a nota anterior:  
incremente o índice da nota atual  
toque a nota atual

Pseudo-Algoritmo para Toque da Música dentro do Loop Principal



## Aperfeiçoamento

Modifique a função que toca a nota para receber também a oitava, e use-a para calcular o valor da frequência.

↪ DICA: comece passando a oitava 0 ao chamar esta função.

Mostre o número da oitava ao lado da nota no display.

↪ DICA: pesquise sobre a função `sprintf` para concatenar uma string com um número inteiro.

Aumente o total de notas mudando o limite do encoder de 11 para 35. Em seguida, use a posição do encoder para calcular o índice e a oitava da nota.

↪ DICA: use o resto de divisão para circular o índice do array de notas. Crie uma outra variável global para guardar o valor da oitava atual, e use a divisão.

Ao apertar o Botão 3, entre no modo 3 (caixinha musical), toque a primeira nota da música (durante o intervalo de tempo do vetor) e registre esse instante de tempo numa variável global.

No loop do Arduino, caso já tenha passado o tempo esperado desde a nota anterior, toque a próxima nota.

↪ DICA: use uma segunda contagem de tempo com função `millis` dentro do loop. Verifique também se a música terminou.

# Desafio Extra



p07c\_aperfeicoamento.ino

cópia  
----->

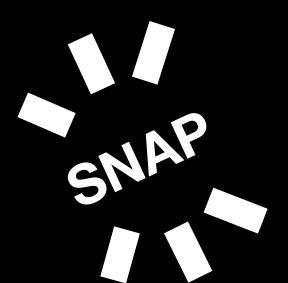
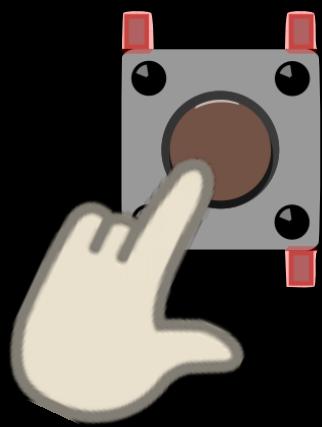


p07d\_desafio.ino

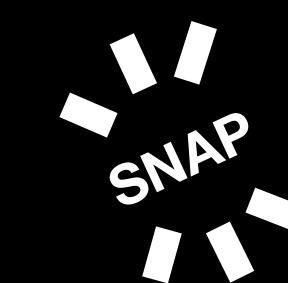
Cópia do Código do Aperfeiçoamento para o Desafio

## gravação

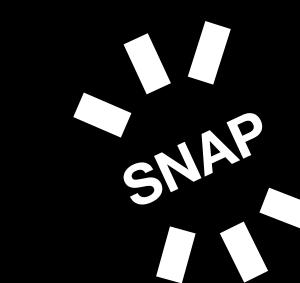
Botão 2



0.7s

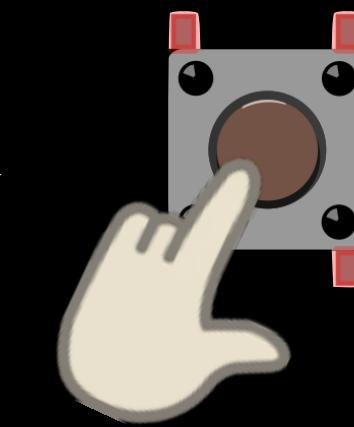


1.1s



0.3s

Botão 2



## reprodução



0.7s



1.1s



0.3s

```
unsigned long intervalos[100];
```

```
int totalDeEstalos;
```

```
int indiceDoEstaloAtual;
```

Gravação e Reprodução de Padrão de Estalos



## Desafio Extra

Ao apertar o Botão 2, **detecte uma sequencia de estalos** e armazene o intervalo de tempo entre cada um. Ao apertar novamente o Botão 2, **imprima esses intervalos**.

↪ DICA: armazene os valores em um vetor bem grande, e conte quantos aconteceram. O último valor deverá ser o tempo entre o último estalo e o segundo pressionamento do botão.

Após apertar o Botão 2 pela segunda vez, **reproduza continuamente essa sequencia de estalos com beeps da campainha** (440 Hz durando 50ms) **dentro do loop**.

↪ DICA: modifique a contagem de tempo entre os beeps feita na Implementação.



[janks.link/micro/projeto07.zip](https://janks.link/micro/projeto07.zip)

Material do Projeto 07