



# Projeto 06

## Controle de Tempo – Prática

Jan K. S. – [janks@puc-rio.br](mailto:janks@puc-rio.br)

ENG1419 – Programação de Microcontroladores

# Testes Iniciais

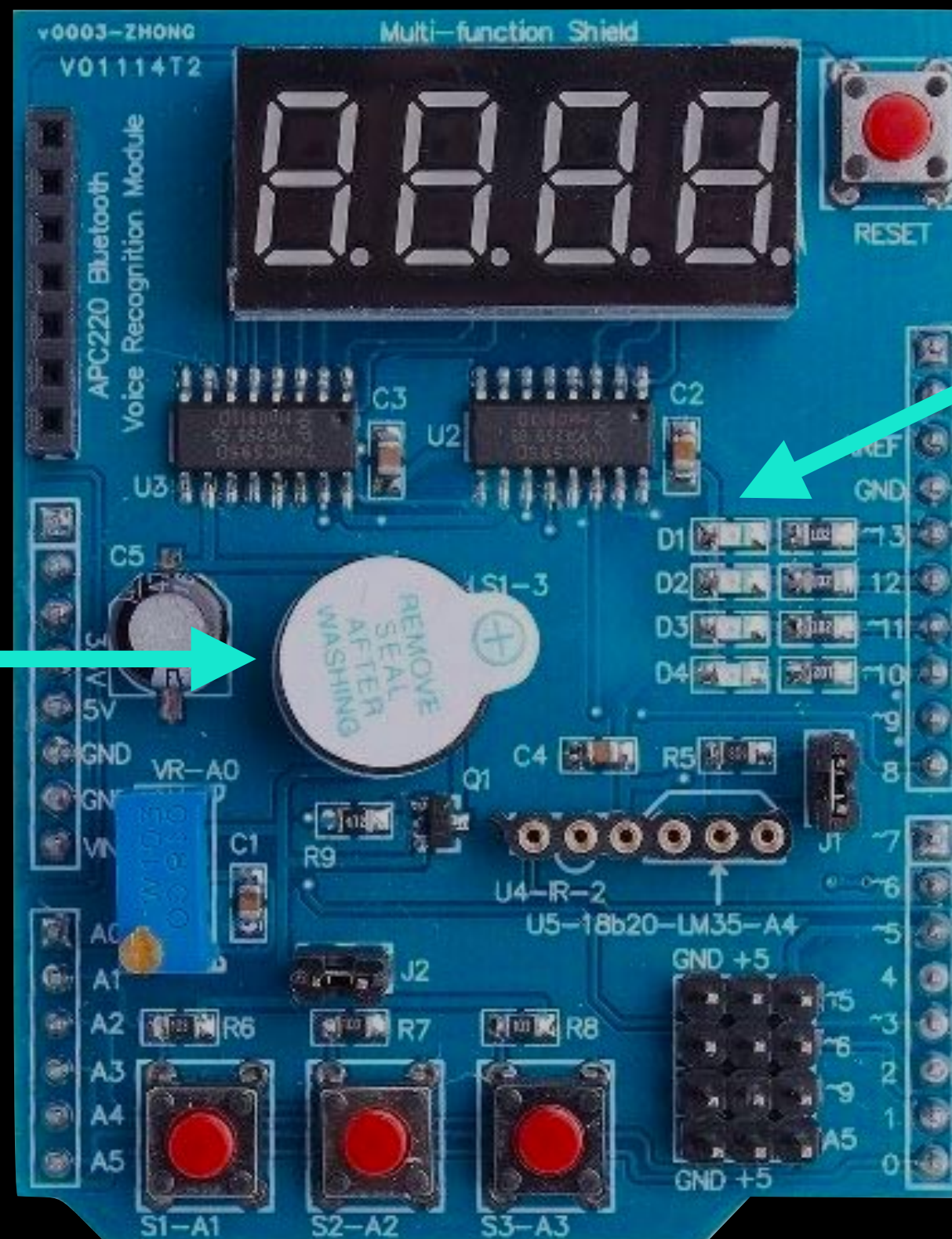
pinos 4, 7 e 8

pino 3

pinos 13, 12, 11 e 10

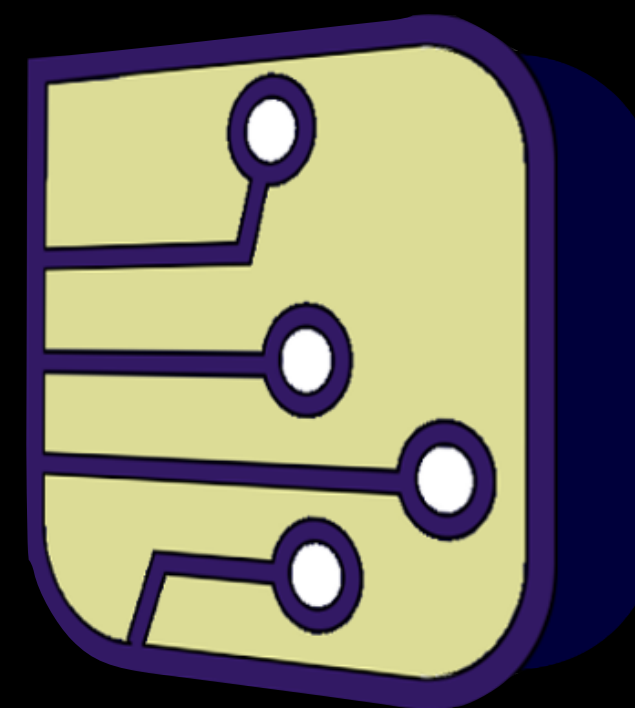
pinos A1, A2 e A3

Pinos Usados pelo Shield Multifunção





Edita o código  
Exporta binário compilado



Simula o código



# p06a\_testes\_iniciais.ino

```
p06a_testes_iniciais | Arduino 1.8.13
1 void setup() {
2   // put your setup code here, to run once:
3 }
4
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8 }
9 }
```

1 Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) em /dev/cu.usbmodem14101

Sketch Ferramentas Ajuda

Autoformatação ⌘T  
Arquivar Sketch  
Corrigir codificação e recarregar  
Gerenciar Bibliotecas... ⇧⌘I  
Monitor serial ⇧⌘M  
Plotter serial ⇧⌘L  
WiFi101 / WiFinINA Firmware Updater  
Placa: "Arduino/Genuino Mega or Mega 2560" ▶  
Processador: "ATmega2560 (Mega 2560)" ▶  
Porta ▶  
Obter informações da Placa  
Programador: "AVRISP mkII" ▶  
Gravar Bootloader

Editar Sketch Ferramentas Ajuda

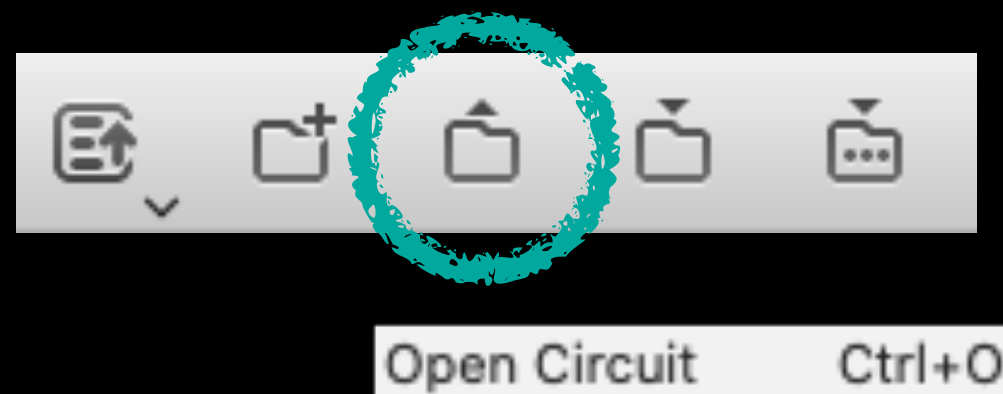
Verificar/Compilar ⌘R  
Carregar ⌘U  
Carregar usando programador ⇧⌘U  
Exportar Binário compilado ⌘⌘S  
Mostrar a página do Sketch ⌘K  
Incluir Biblioteca ▶  
Adicionar Arquivo...

Exportação do Binário Compilado no Arduino IDE

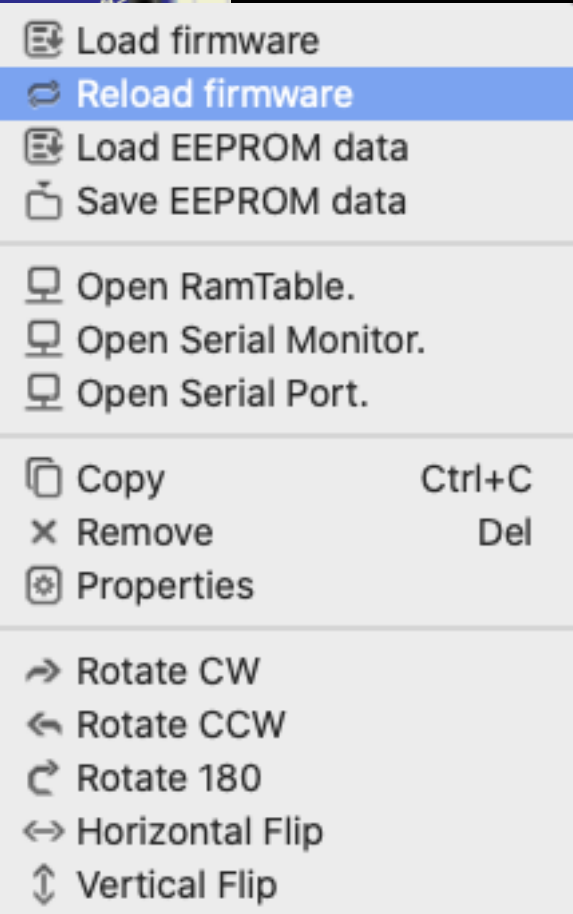
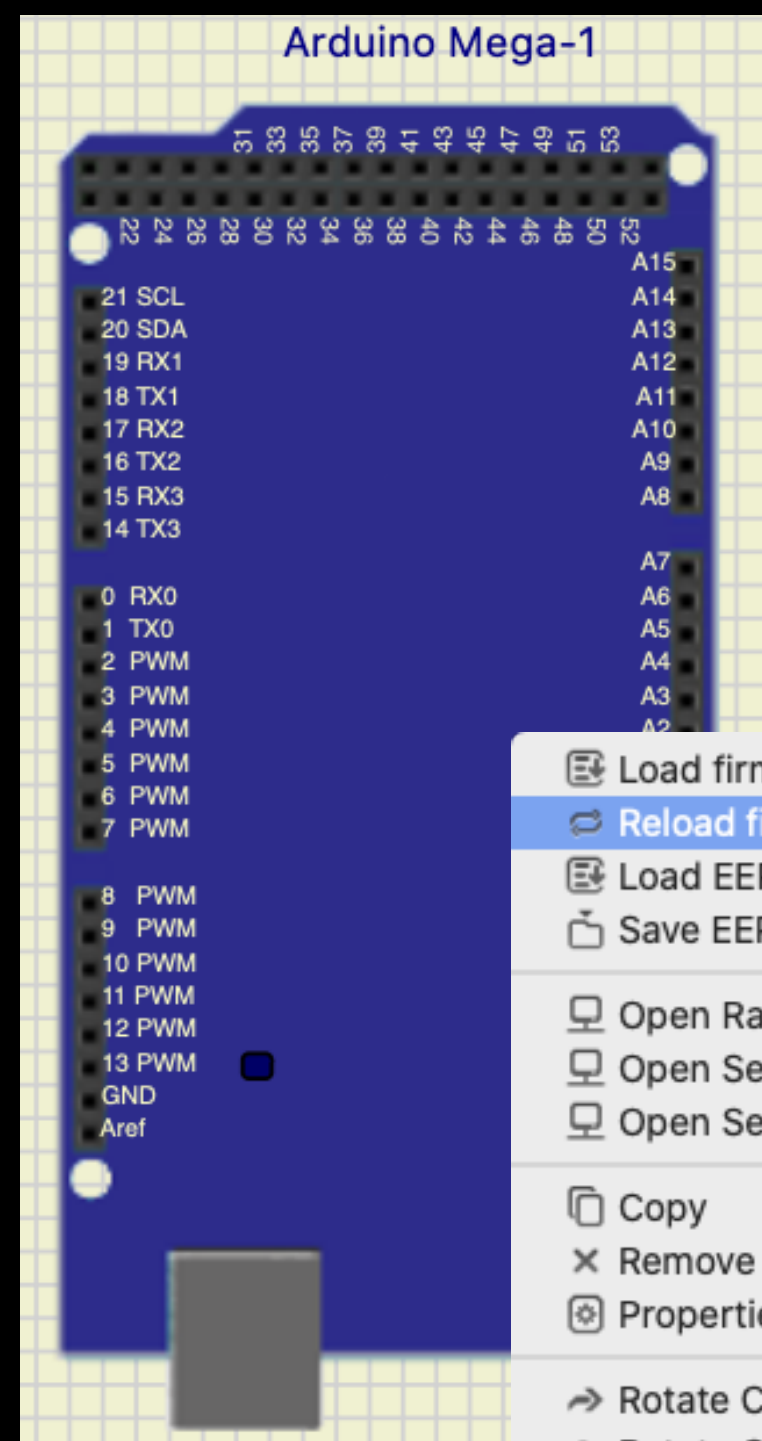




## Simulação do Circuito no SimulIDE



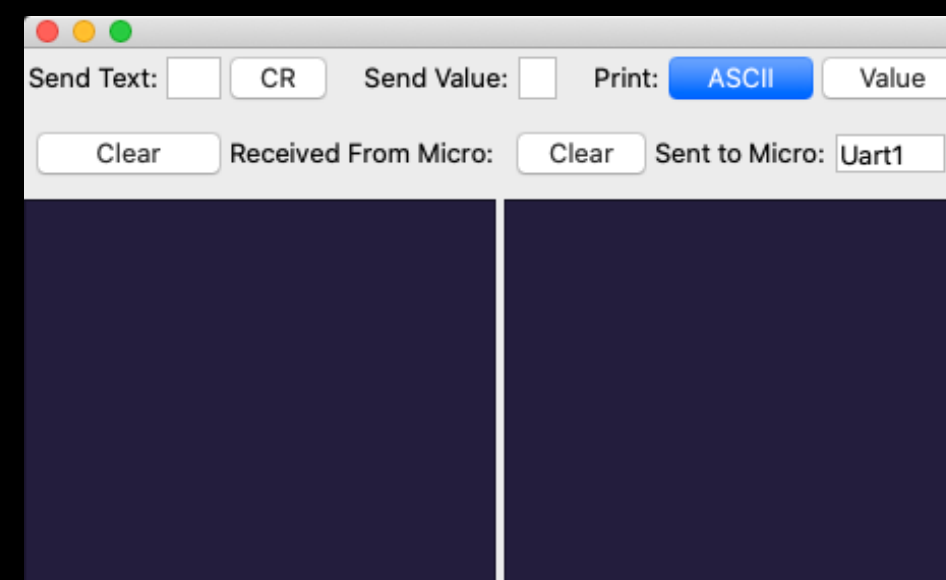
Projeto06a.simu



Iniciar Simulação



Parar Simulação



Tela do Serial



Clique no Botão

Opções da Placa do Arduino

Etapas para Rodar o Código no SimulIDE

```
#include <TimerOne.h>
```

```
...
```

```
Timer1.initialize(1000000);  
Timer1.attachInterrupt(funcao);
```



```
#define USE_TIMER_1 true
```

```
#include <TimerInterrupt.h>
```

```
...
```

```
ITimer1.init();
```

```
ITimer1.attachInterruptInterval(1000, funcao);
```

em milissegundos!



Substituição da Biblioteca TimerOne pela TimerInterrupt





## Testes Iniciais

Mantenha o **LED 1 aceso** constantemente. Exporte o binário e teste no Simulide, como mostram os slides anteriores.  
↳ DICA: use as funções `pinMode` e `digitalWrite` na `setup`.

**Exiba o número -4.12 no display** por 2 segundos ao iniciar o programa.

↳ DICA: use as funções `display.set` e `display.show`.

**Ao apertar o Botão 2, mude o estado do LED 2** (entre aceso/apagado).

↳ DICA: use a `GButton` e uma variável global tipo `bool` (`true/false`) para representar o estado do LED 2.

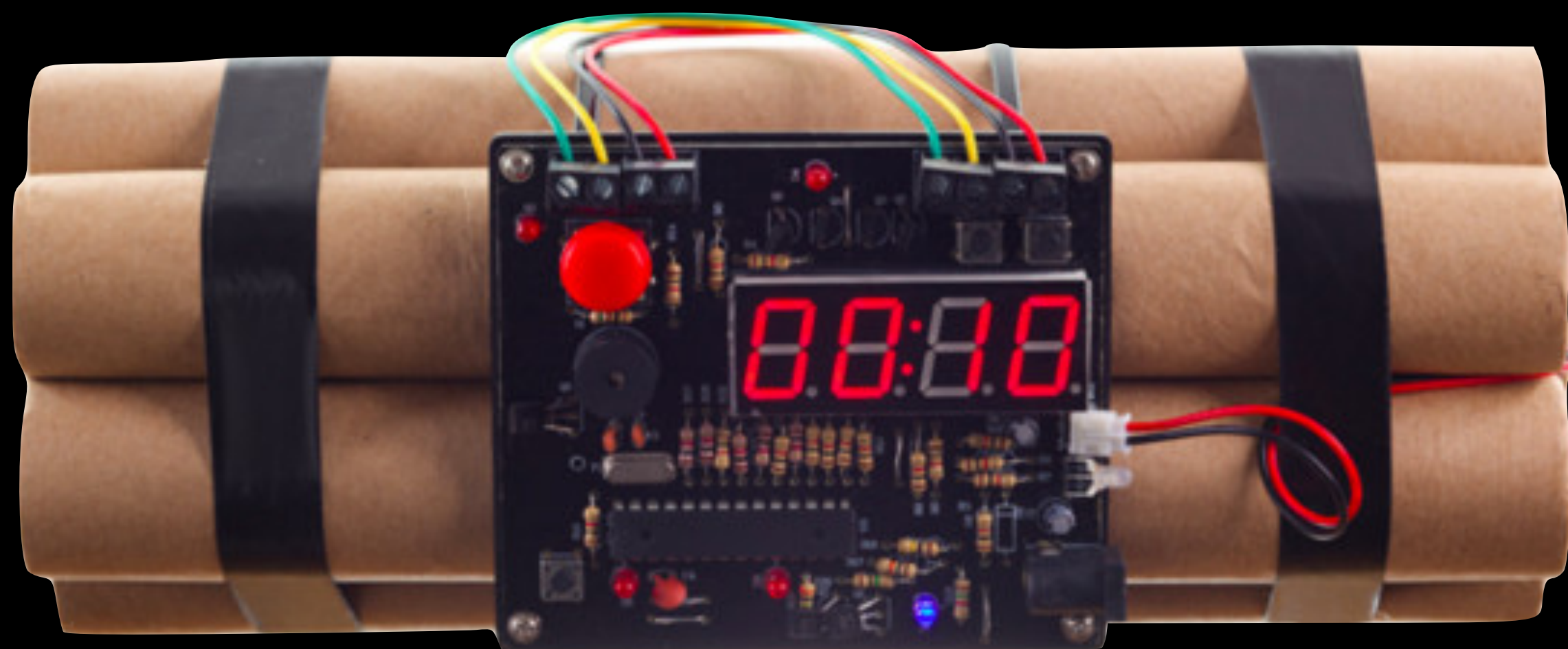
Durante o resto do programa, **exiba no display quantas vezes o Botão 3 foi pressionado**.

↳ DICA: use a `display.set` e `display.update` na `loop` e uma variável global de contagem.

**Imprima a contagem via serial a cada 2 segundos**, sem interferir no display de 7 segmentos.

↳ DICA: use a nova biblioteca `TimerInterrupt`.

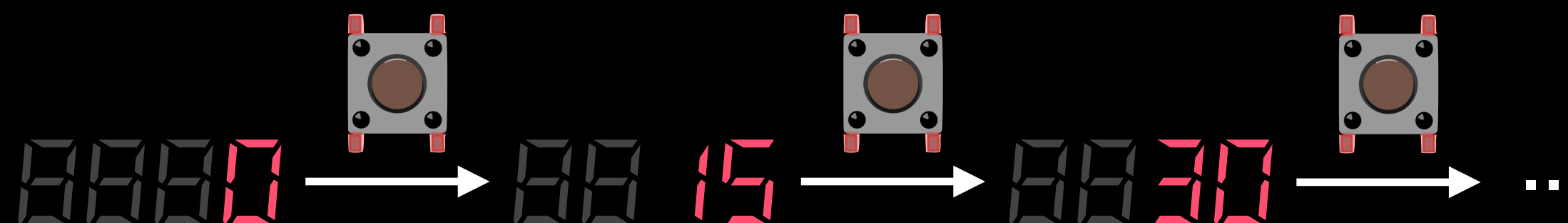
# Implementação



Contagem Regressiva de 7 Segmentos

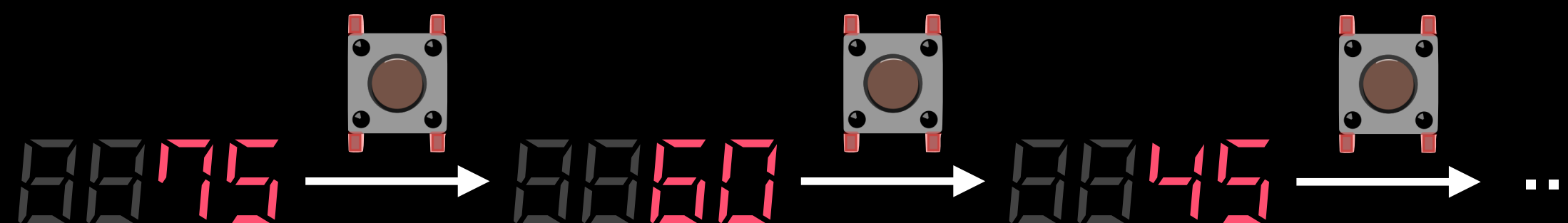
## Botão 1

aumenta tempo (fase de ajuste inicial)



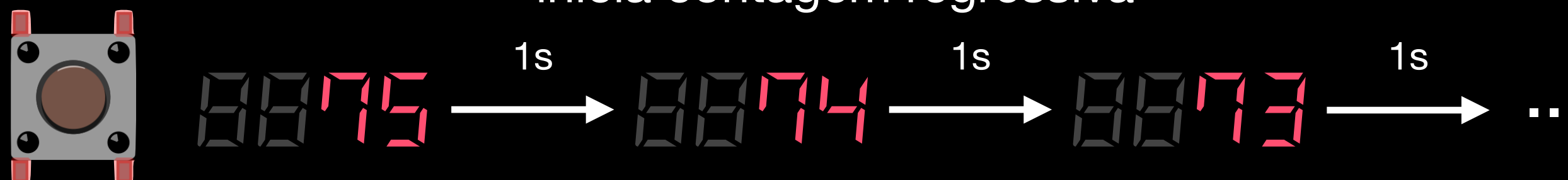
## Botão 2

diminui tempo (fase de ajuste inicial)



## Botão 3

inicia contagem regressiva



Controle da Contagem Regressiva pelos Botões

88'75 → 01.15

75 segundos



1 minuto e  
15 segundos



OU

envia número 1.15  
para o display

envia número 115  
para o display  
e acende o ponto

Ajuste Final da Exibição do Tempo





## Implementação

Crie uma variável global para o tempo da contagem (em segundos). **Exiba continuamente o valor dessa variável** no display.

↳ DICA: use a `display.set` e `display.update` na loop.

**Aumente o valor da variável de tempo em 15s** ao apertar o Botão 1 **e diminua-o em 15s** ao apertar o Botão 2. Não permita que o valor fique negativo!

↳ DICA: use a `GButton`.

Ao apertar o Botão 3, **inicie a contagem, diminuindo o valor a cada segundo** se ele for maior que zero.

↳ DICA: Use uma variável global que indique se a contagem está em andamento. Crie na setup um Timer que esteja sempre rodando, e diminua o tempo somente se estiver em andamento.

**Se o valor chegar a zero** e a contagem estiver em andamento, pare a contagem e toque a campainha brevemente.

↳ DICA: lembre-se que o `delay` não funciona dentro do timer.

Mude o **formato de exibição** de segundos para minutos e segundos (ex: **01.30** em vez de 90).

↳ DICA: use divisão e resto da divisão (%).

Aperfeiçoamento



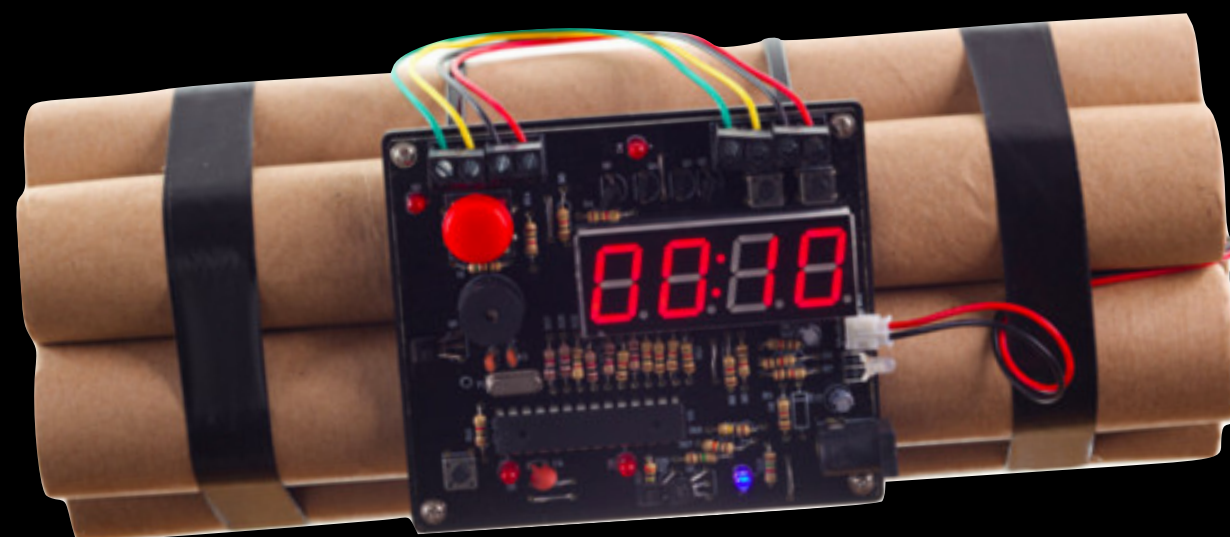
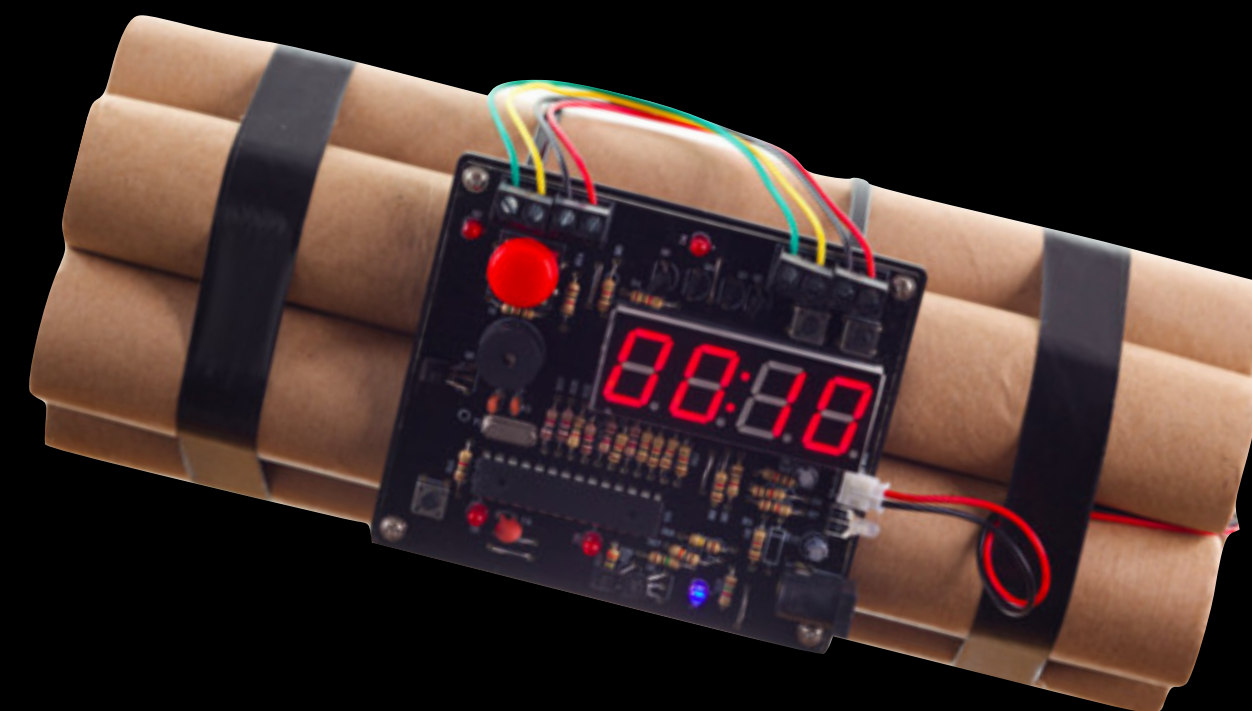
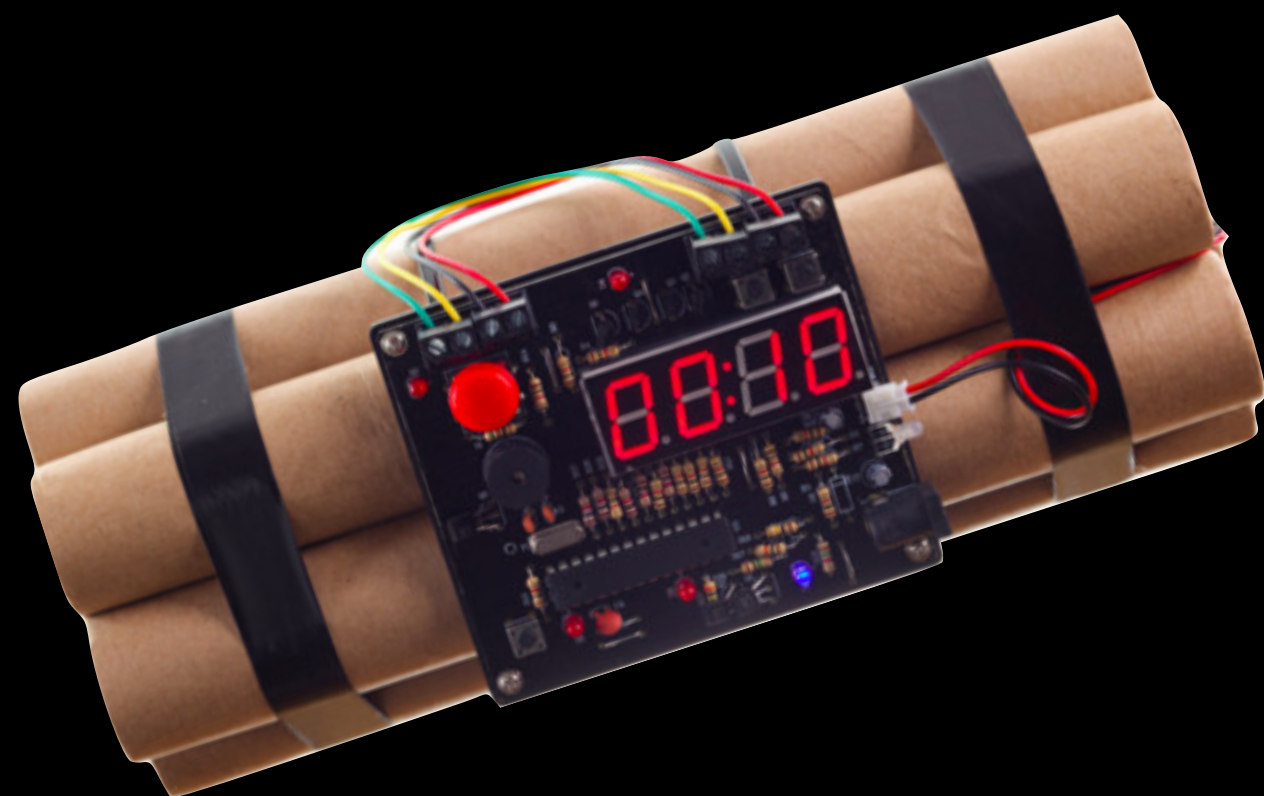
p06b\_implementacao.ino

cópia  
-----▶



p06c\_aperfeicoamento.ino

Cópia do Código da Implementação para o Aperfeiçoamento



Múltiplas Contagens



Tempo da contagem atual

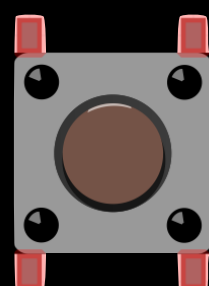
05.33

Indicação da contagem atual



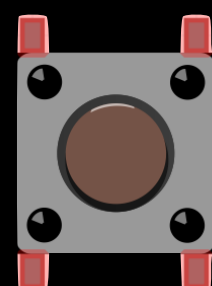
Botão 1

aumenta tempo



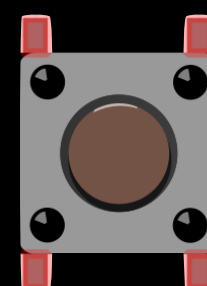
Botão 2

diminui tempo



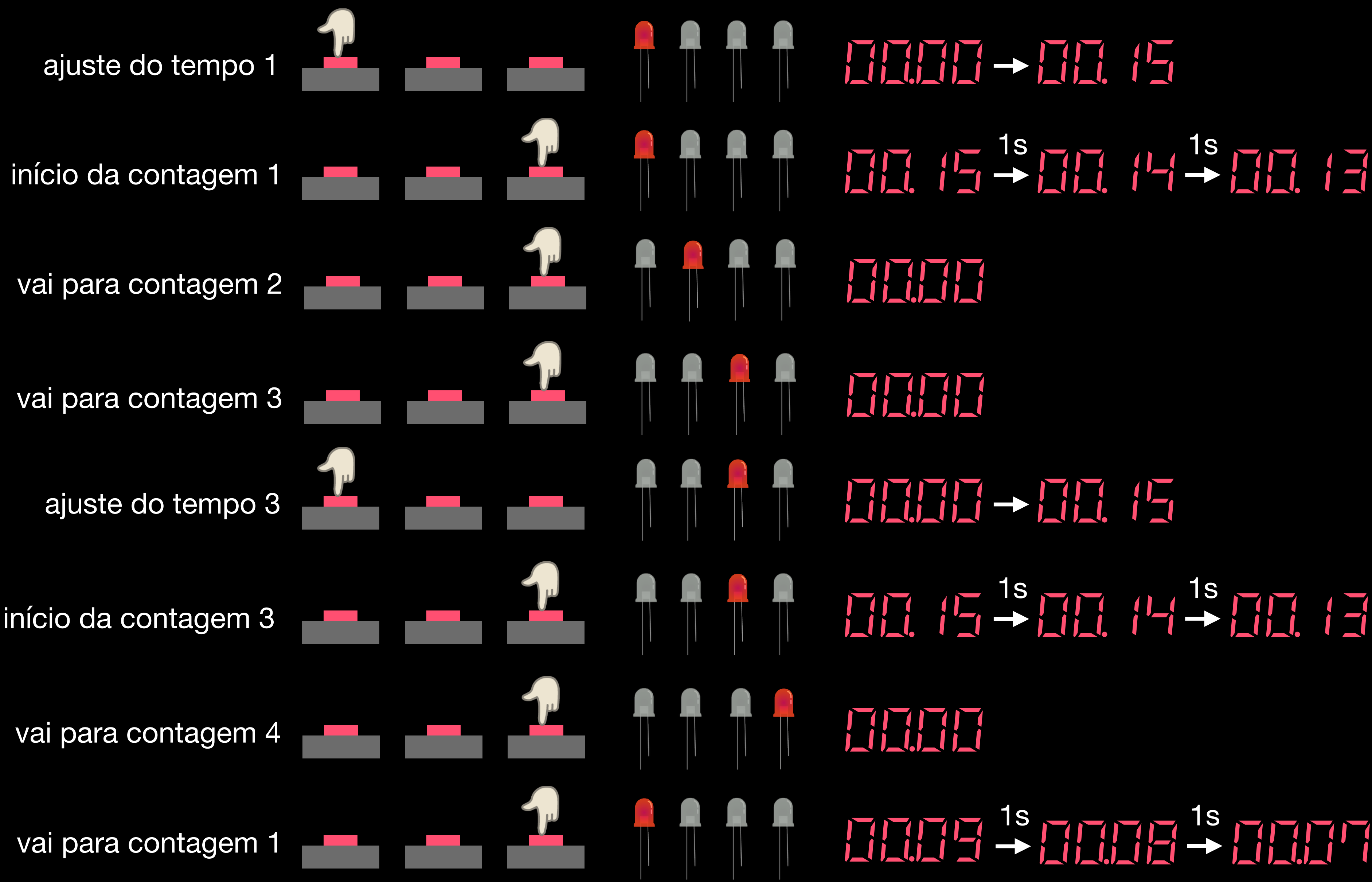
Botão 3

inicia contagem / próxima contagem



Controle de 4 Timers





Exemplo de Controle de 4 Timers

```
int tempo = 0;
```

```
bool emAndamento = false;
```



```
int tempo[] = {0, 0, 0, 0};
```

```
bool emAndamento[] = {false, false, false, false};
```

```
int indiceDaContagemAtual = 0;
```

Conversão de Variáveis para Listas

Converta as variáveis de tempo e emAndamento para listas e crie uma variável para o índice atual (começando em 0). Corrija essas variáveis ao longo do código, fazendo o acesso nas listas. Teste o programa e veja se ainda funciona com apenas o primeiro contador.



## Aperfeiçoamento

Ao apertar o Botão 3: se a contagem atual estiver em andamento ou zerada, troque para a contagem seguinte; caso contrário, inicie a contagem atual (que nem na implementação).

Indique qual a contagem que está sendo exibida, acendendo o LED respectivo.

Para cada uma dos 4 contagens: se estiver em andamento, diminua o valor do tempo a cada segundo. A campainha deve tocar mesmo que a contagem que terminou não esteja sendo exibida.

↪ DICA: modifique a função associada ao Timer1 para lidar com as 4 contagens.

Desafio Extra



p06c\_aperfeicoamento.ino

cópia  
-----▶



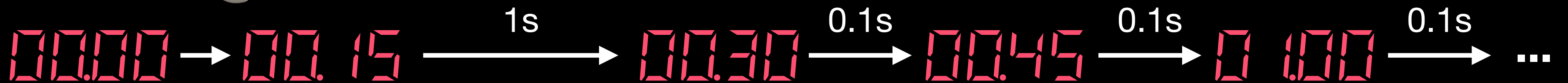
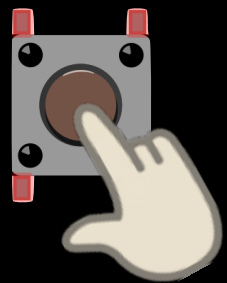
p06d\_desafio.ino

Cópia do Código do Aperfeiçoamento para o Desafio



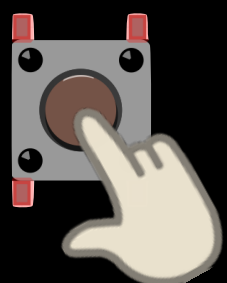
## Botão 1

ao apertar e segurar



## Botão 2

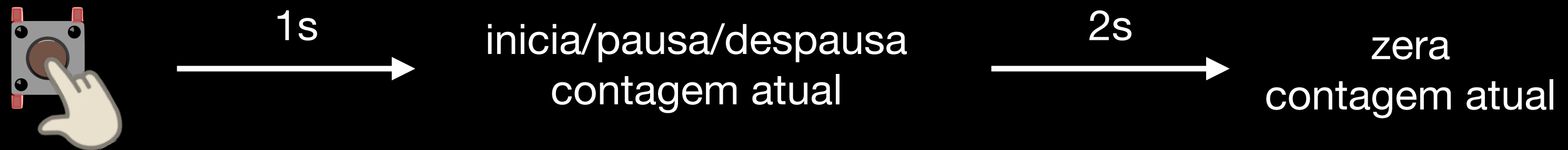
ao apertar e segurar



Melhoria nos Botões 1 e 2

## Botão 3

ao apertar e segurar



## Botão 3

ao soltar

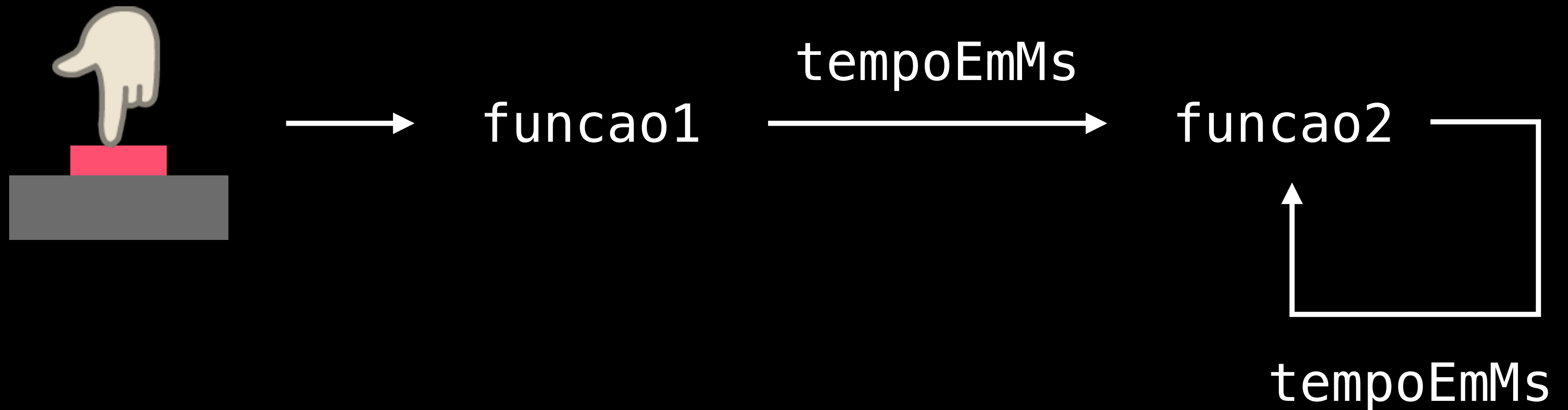


Melhoria no Botão 3

```
botao.setPressHandler(funcao1)
```

```
botao.setHoldHandler(funcao2)
```

```
botao.setHoldTime(tempoEmMs)
```



Funções para Evento de Botão Segurado

```
// definição função para quando o botão é segurado
```

```
botao.setHoldHandler(funcao)
```

```
// define tempo para considerar o botão como segurado
```

```
botao.setHoldTime(tempoEmMs)
```

ATENÇÃO: a função do evento "segurar" é chamada **ciclicamente**, cada vez que passa esse tempo.



Funções para Evento de Botão Segurado



## Desafio Extra

Ao segurar o Botão 1 ou o Botão 2 por mais de 1 segundo, **aumente/diminua 15s do tempo a cada 100 ms** enquanto o botão estiver sendo segurado.

↳ DICA: você pode usar as funções `setHoldHandler` e `setHoldTime` a qualquer momento.

Mude o comportamento anterior do Botão 3. Se ele estiver for segurado por 1 segundo, **inicie/pause/despause imediatamente a contagem atual**. Se ele se mantiver segurado por mais 2 segundos, **zere a contagem imediatamente, sem tocar a campainha**. Ao soltar o botão, **mude para a próxima contagem** caso o botão não tenha sido segurado por mais de 1 segundo.

↳ DICA: não use mais a função `setPressHandler` para o botão 3.





[janks.link/micro/projeto06.zip](https://janks.link/micro/projeto06.zip)

Material do Projeto 06