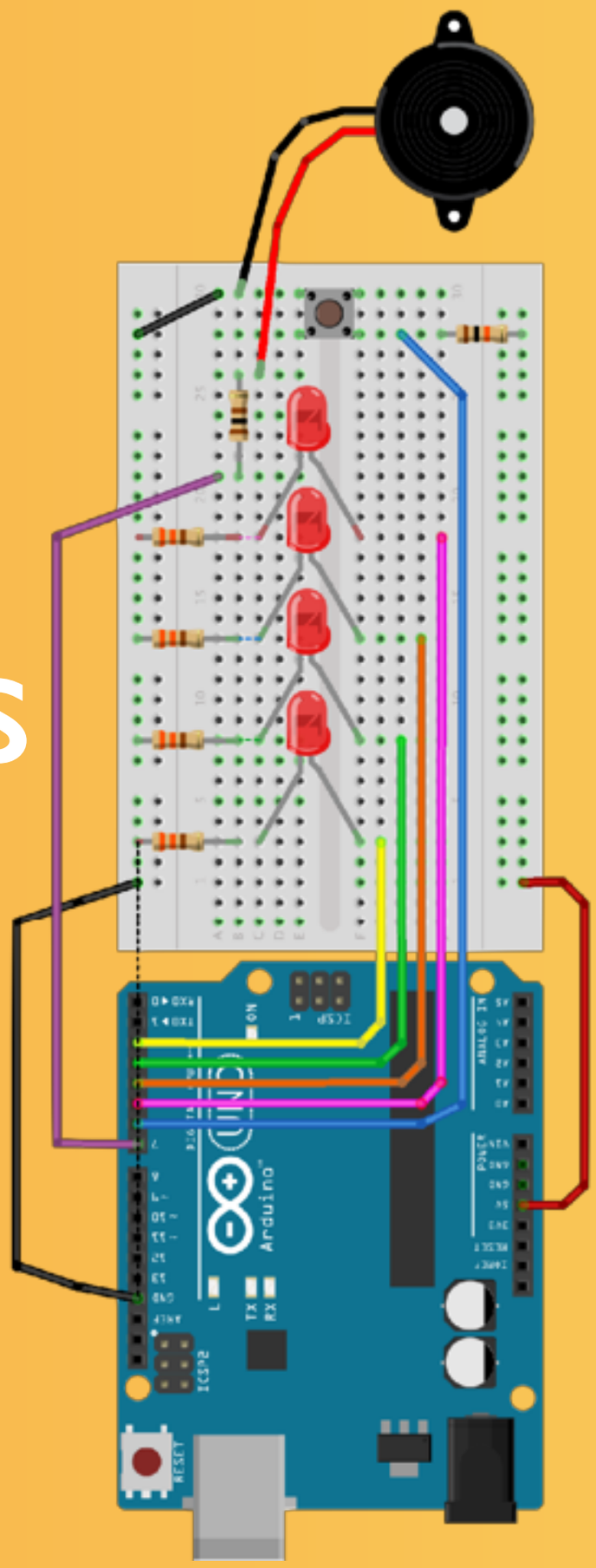
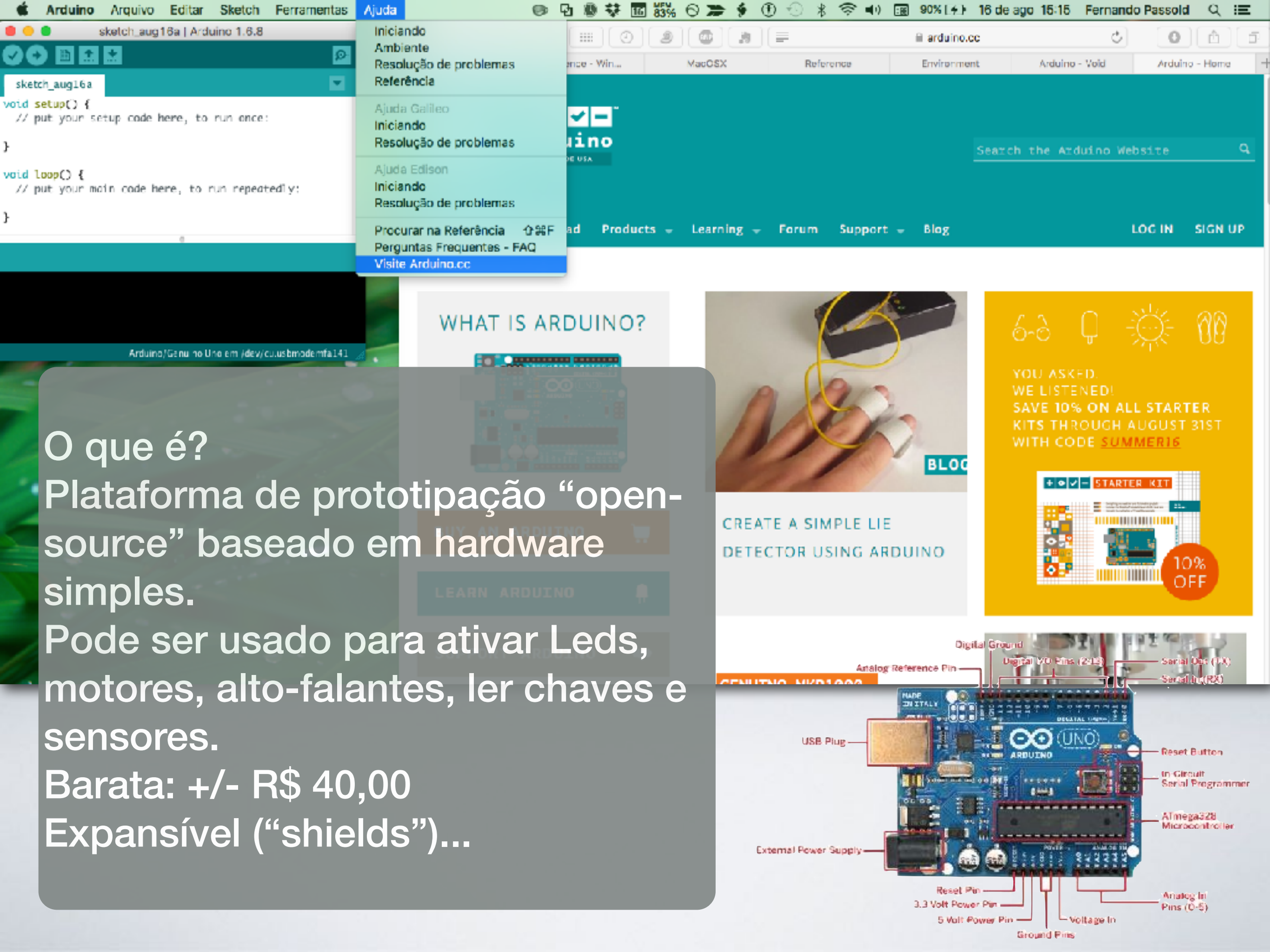
Intro. Sist. uMicroprocessados

Intro. Eng. Elétrica

Prof. Dr.Eng. **Fernando Passold**





O que é?

Plataforma de prototipação “open-source” baseado em hardware simples.

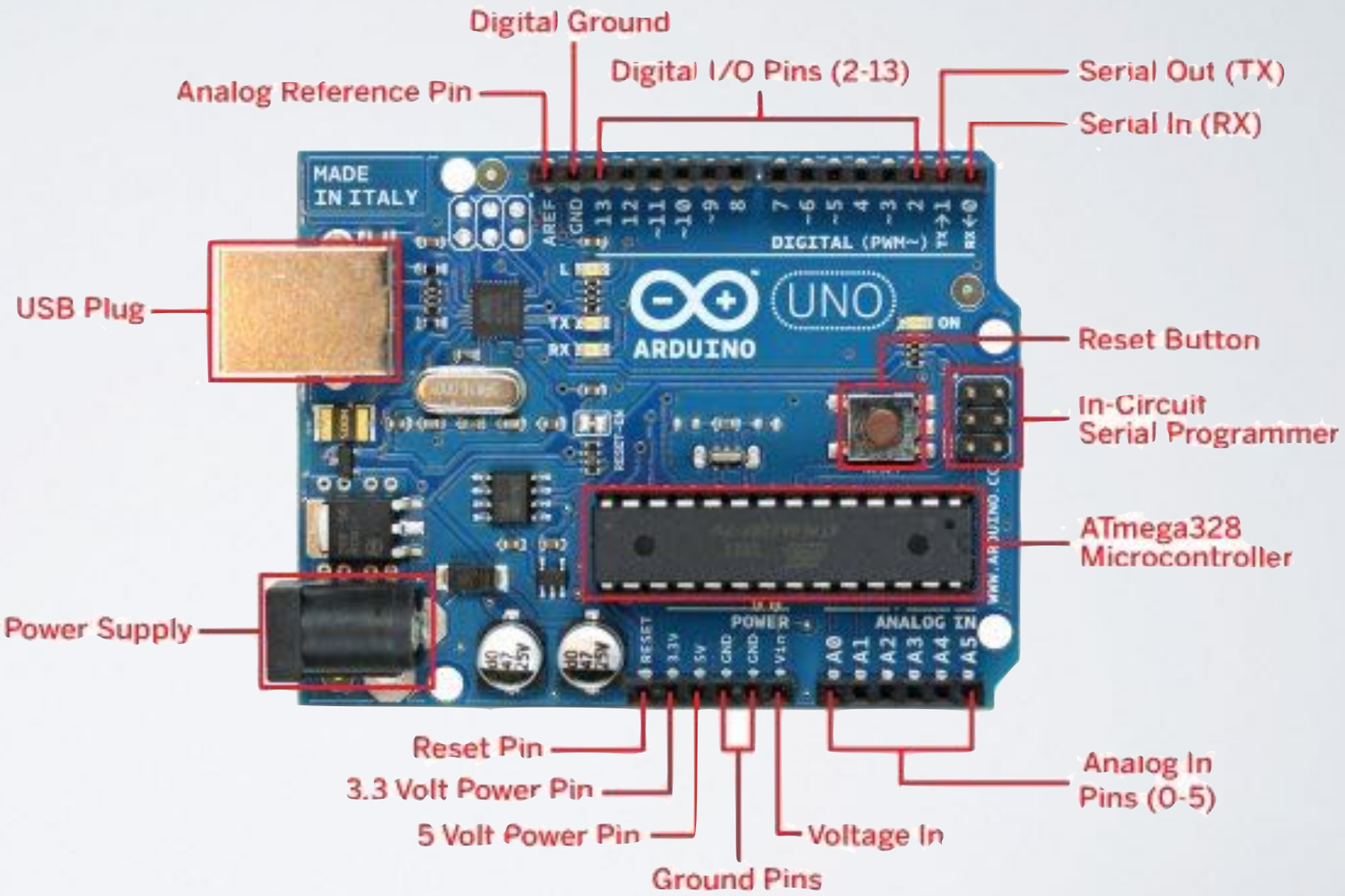
Pode ser usado para ativar Leds, motores, alto-falantes, ler chaves e sensores.

Barata: +/- R$ 40,00

Expansível (“shields”)...

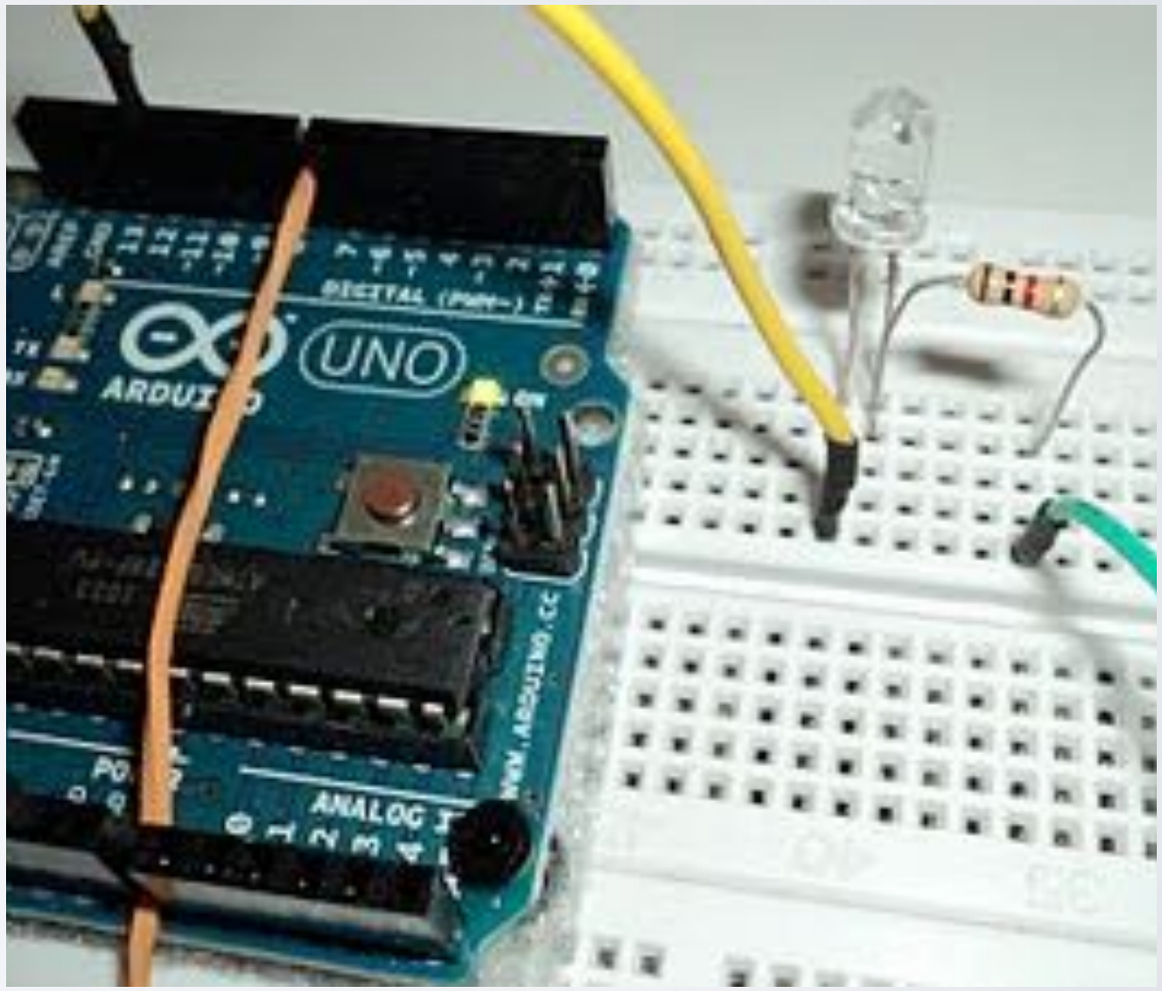
Características Técnicas Arduíno Uno$_{REV3}$ (Hardware)

- Microcontrolador: ATmega328P;
- Frequência de operação: 16 MHz; (ou 1 instrunção à cada 1/ (4*16.000.000) seg = à cada 15,625 ns. Em 1 seg ≈ 64.000.000 instruções
- Alimentação interna: 5 Volts;
- Alimentação externa: 7 ~ 12 Volts;
- Pinos de I/O: 14 (6 c/PWM~);
- Pinos com A/D: 6;
- Capacidade drenar 20 mA/pino;
- Memória FLASH (programa): 32 KBytes (0,5 KB = bootloader);
- SRAM (dados): 2 KBytes;



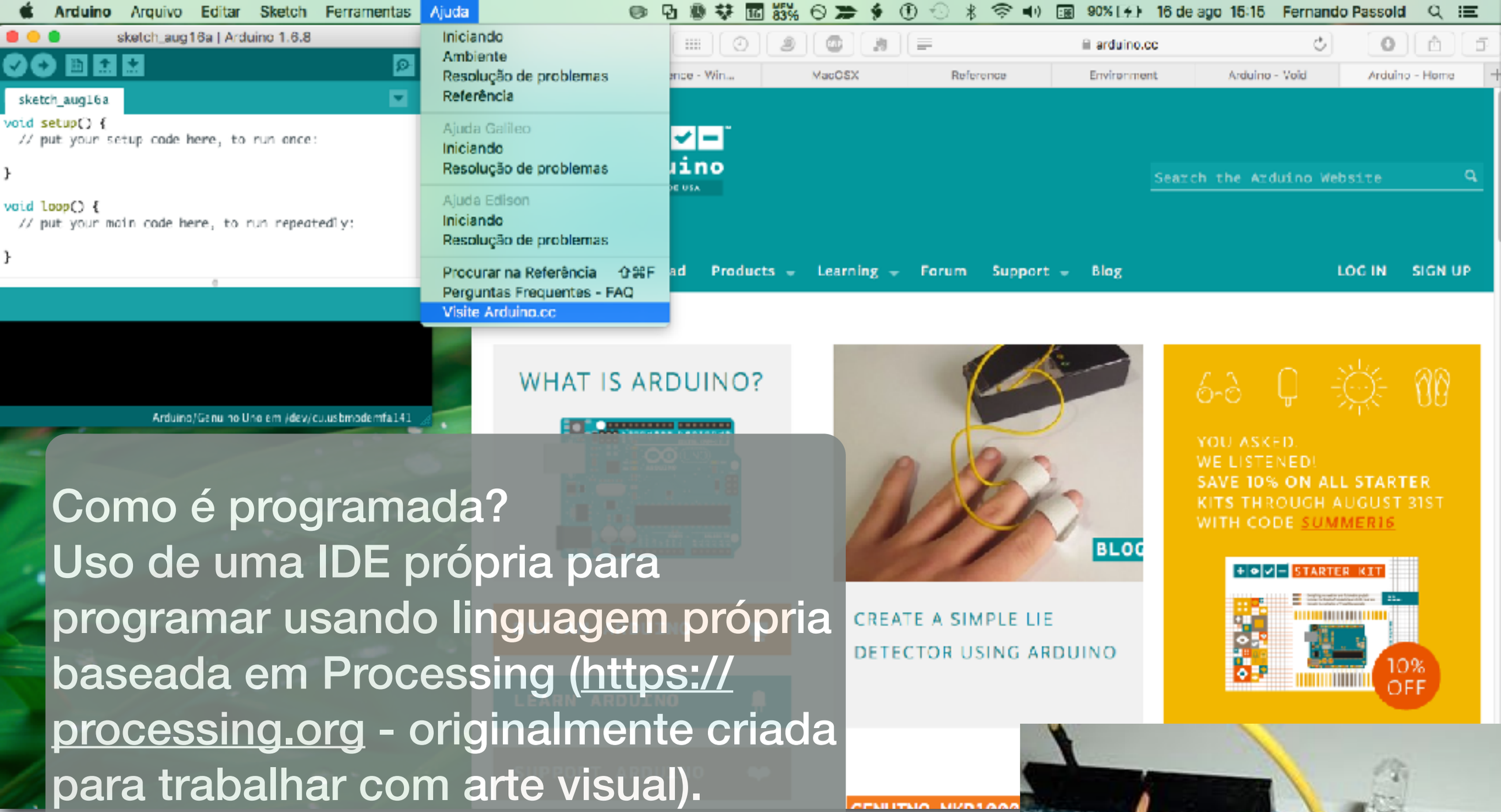
Como se trabalha com uma placa microcontrolada?

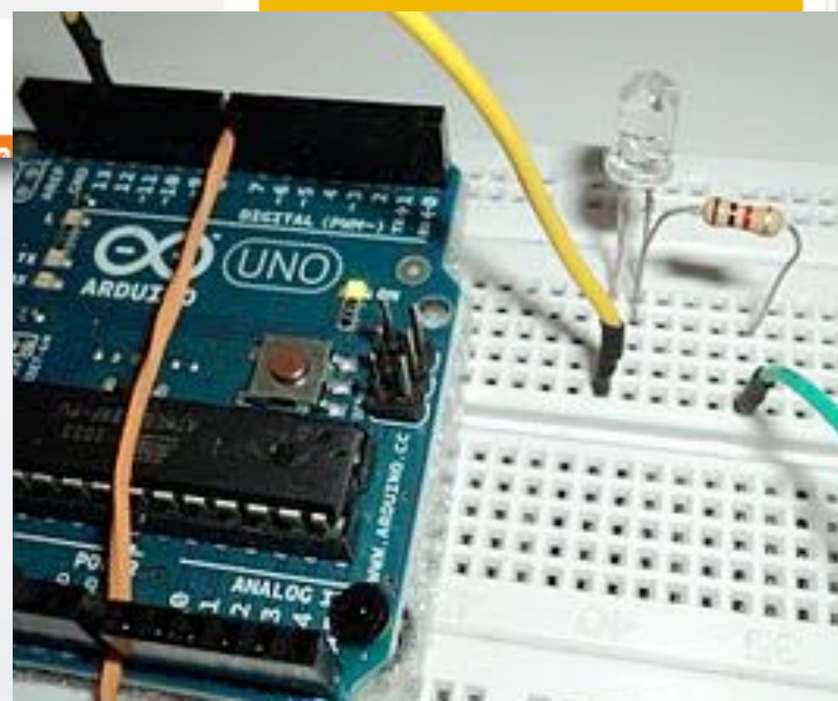
- 50% **Hardware**!
- 50% **Software**!



➥ O que define o resultado obtido é a **inteligência** usada tanto no hardware quanto no software

Como é programada?
Uso de uma IDE própria para programar usando linguagem própria baseada em Processing (https://processing.org - originalmente criada para trabalhar com arte visual).
O código se asemelha à linguagem C.
Multiplataforma: Windows, Mac, Linux.

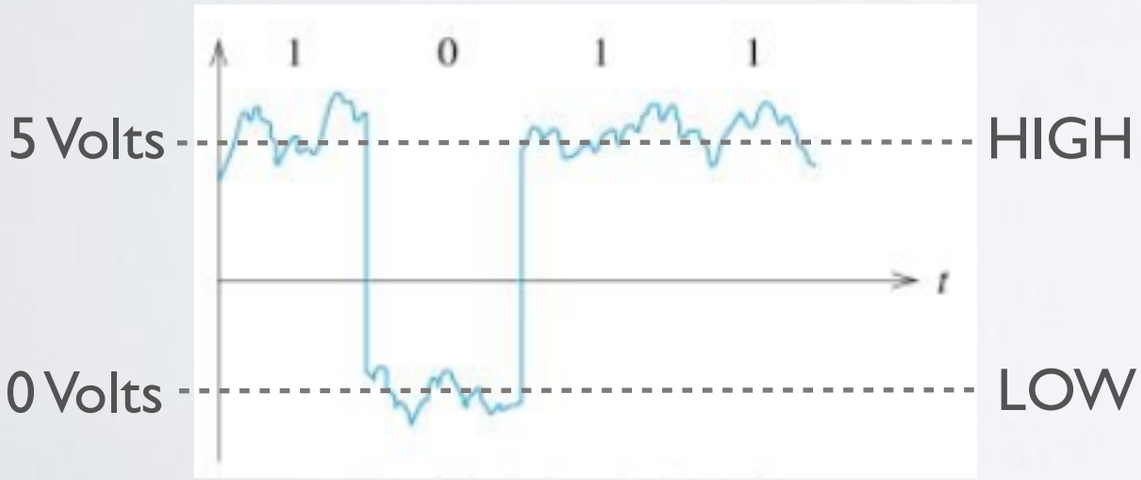


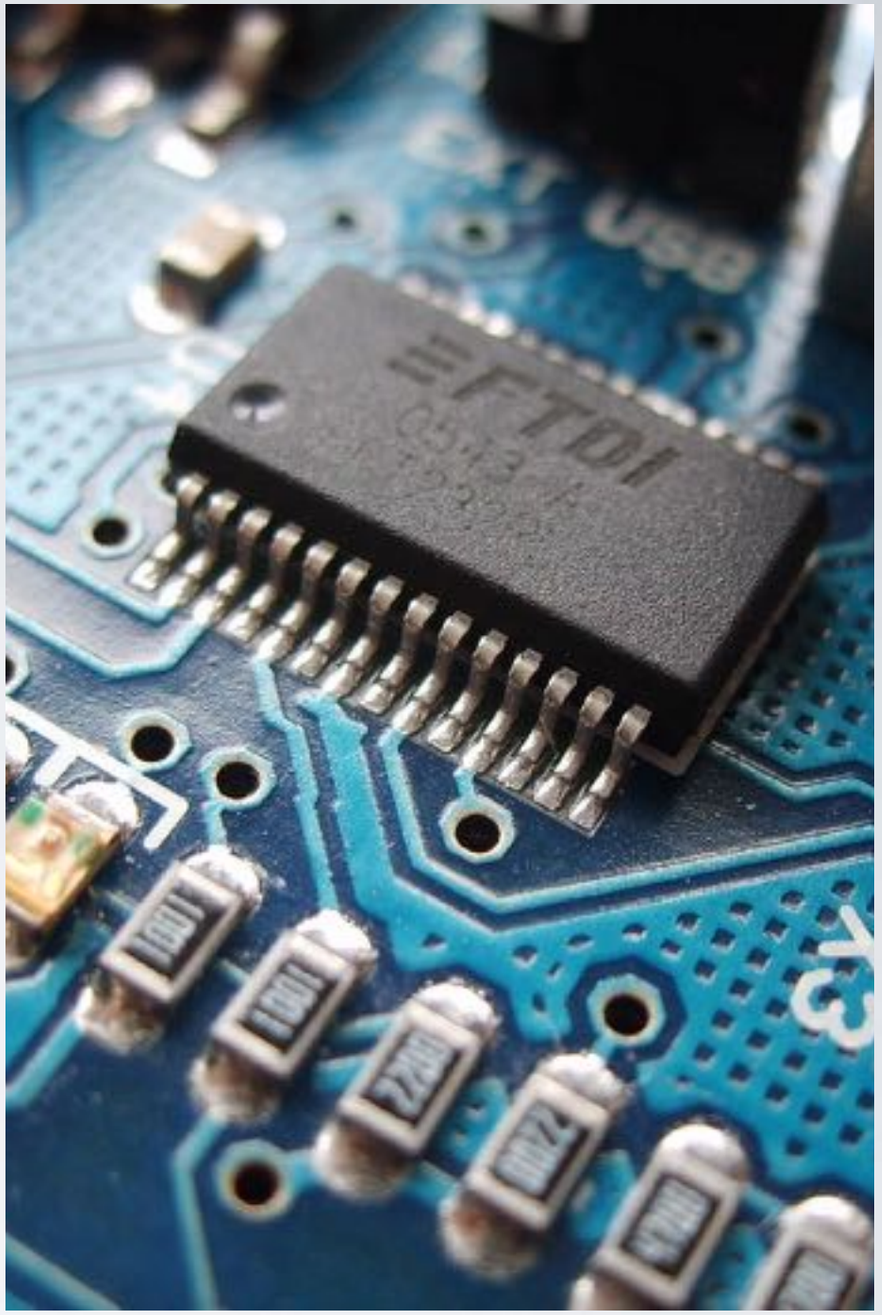


Conceitos Básicos: Eletrônica Digital

Níveis lógicos:

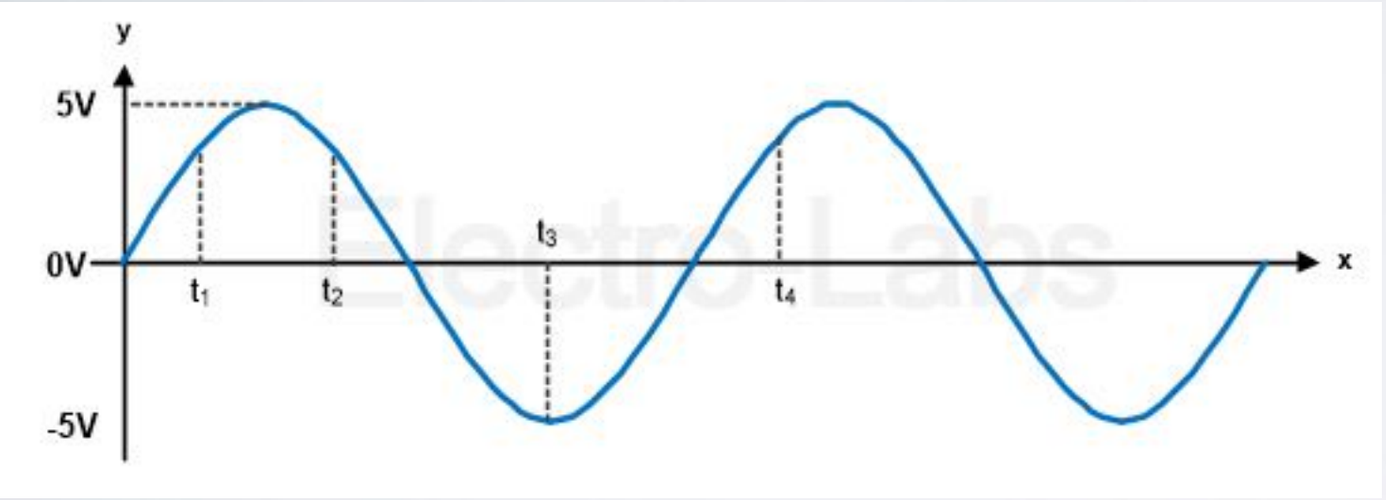
Bit:	Tensão	Lógica básica	Equivalência
“1”	Alto (HIGH)	VERDADEIRO	“Ligado”
“0”	Baixo (LOW)	FALSO	“Desligado”

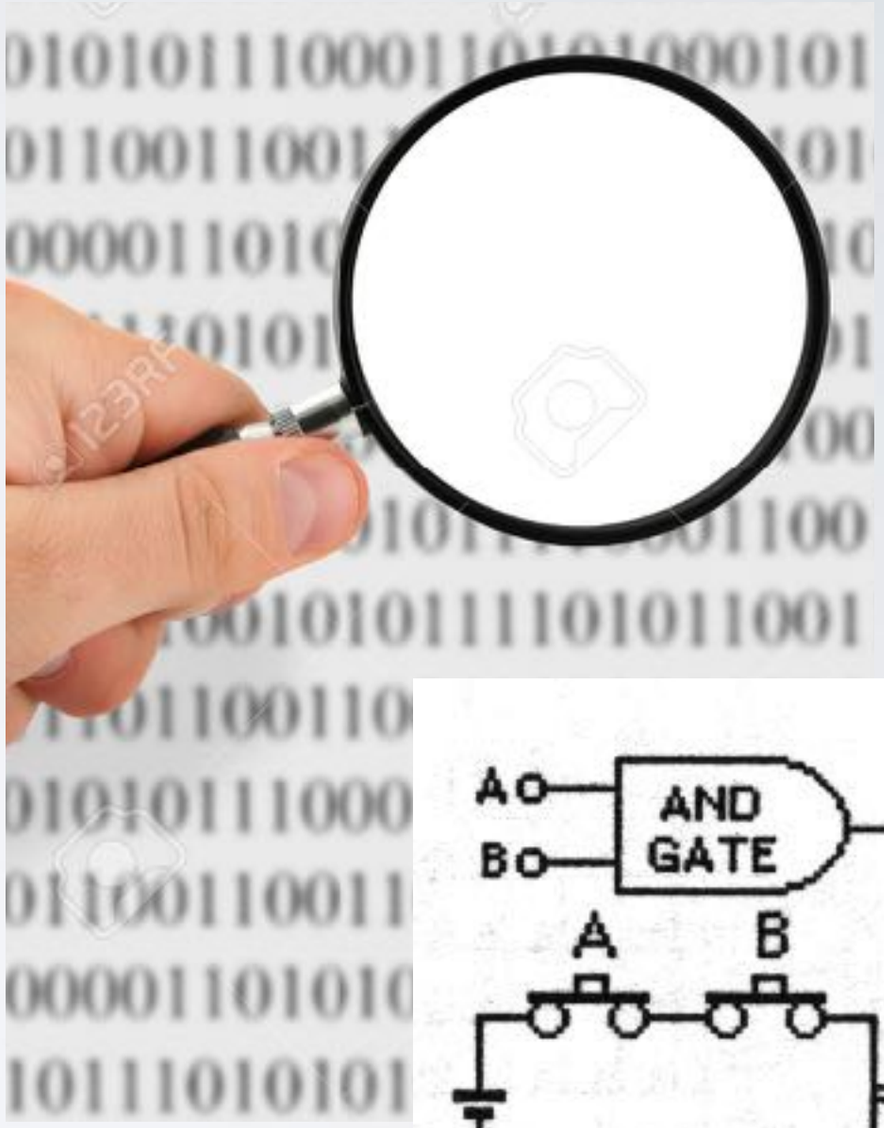




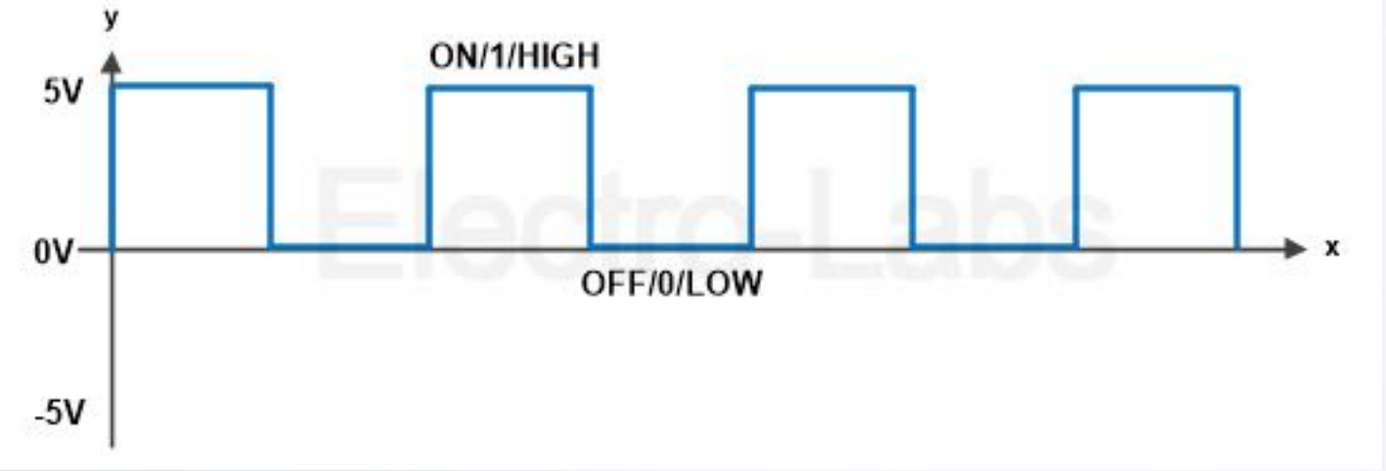
Conceitos Básicos
Sinal Analógico x Digital

Analógico:



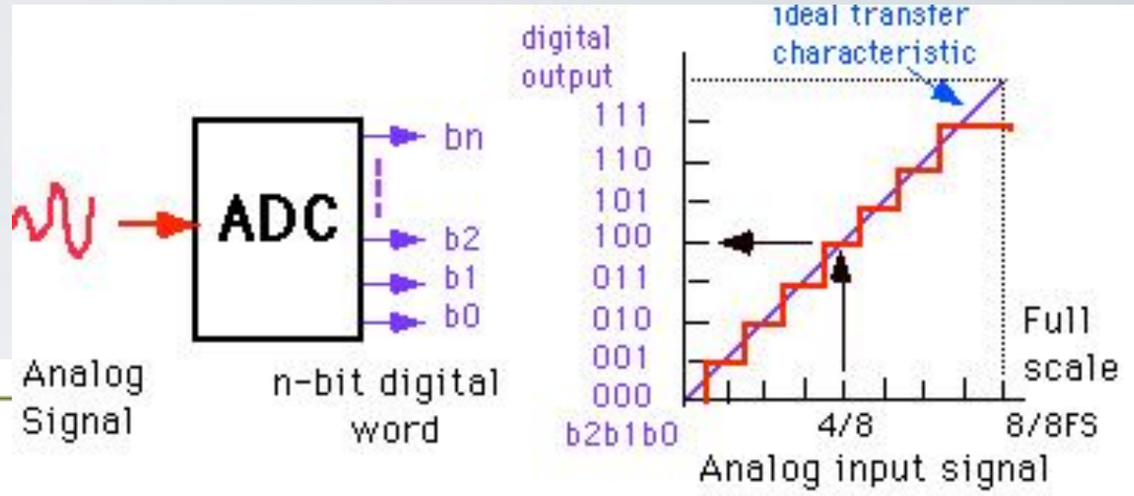


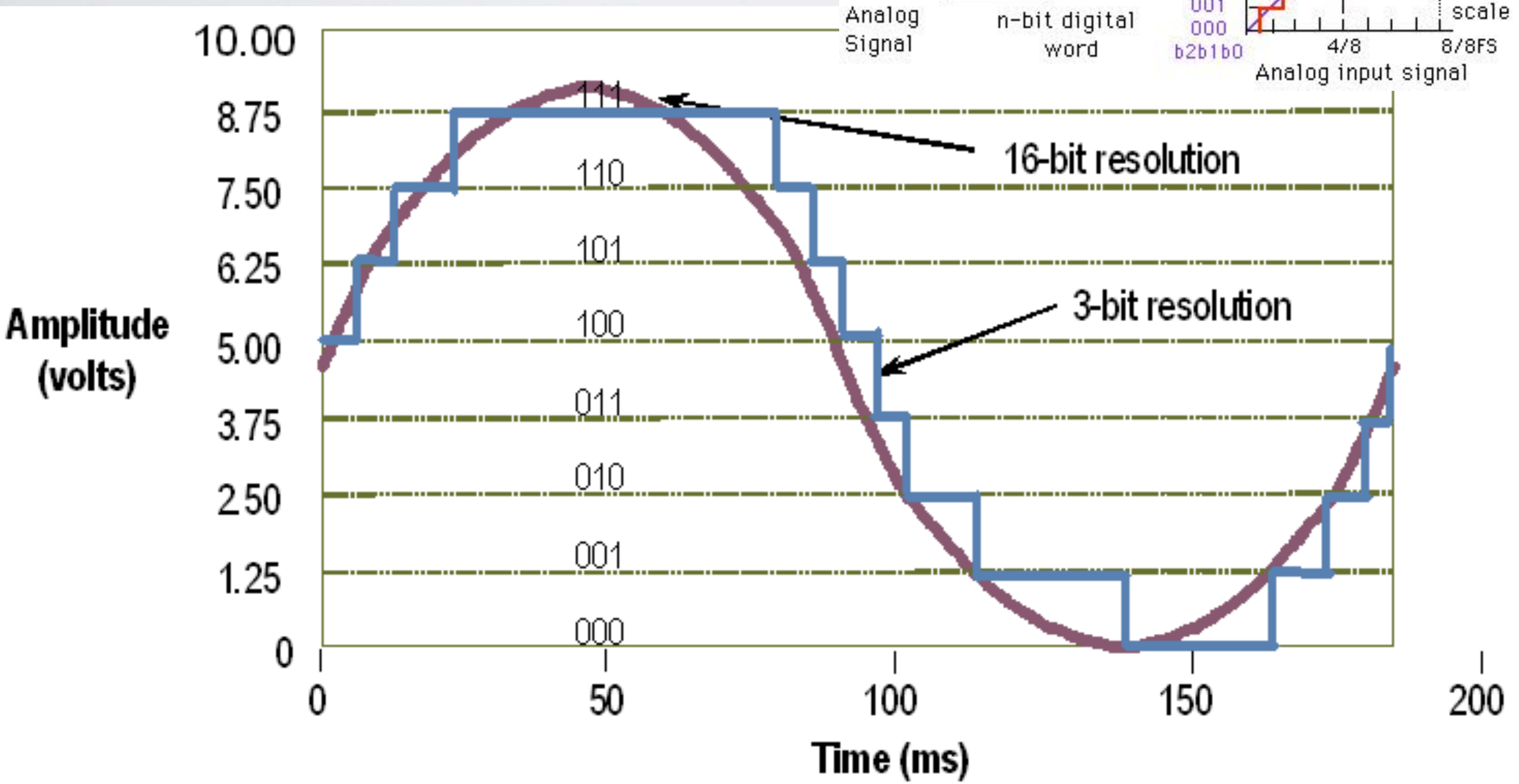
Digital:



Conversão Analógica

Ou A/D

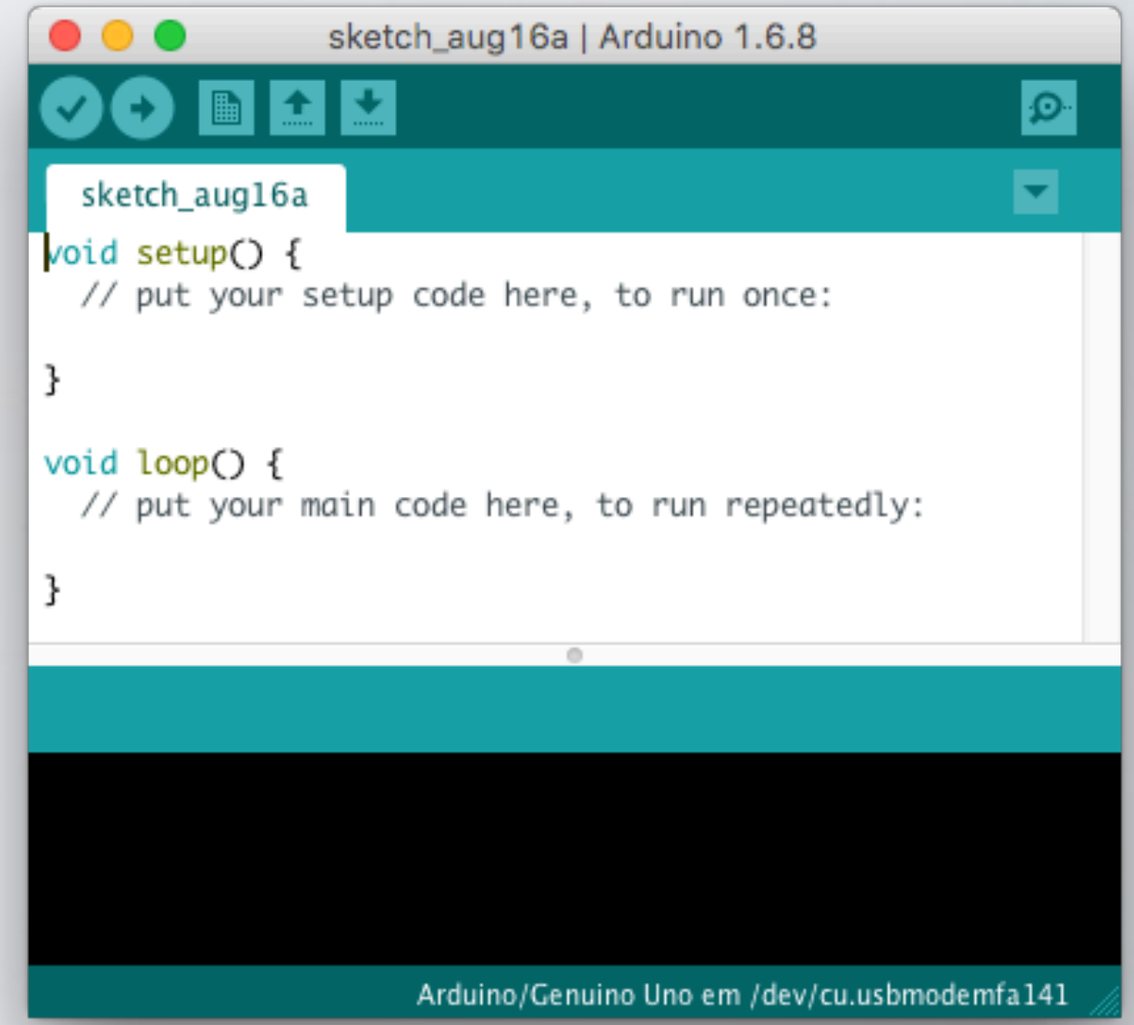




Como Programar?

Arduino

- Linguagem própria baseada em Processing (similar à Linguagem C)



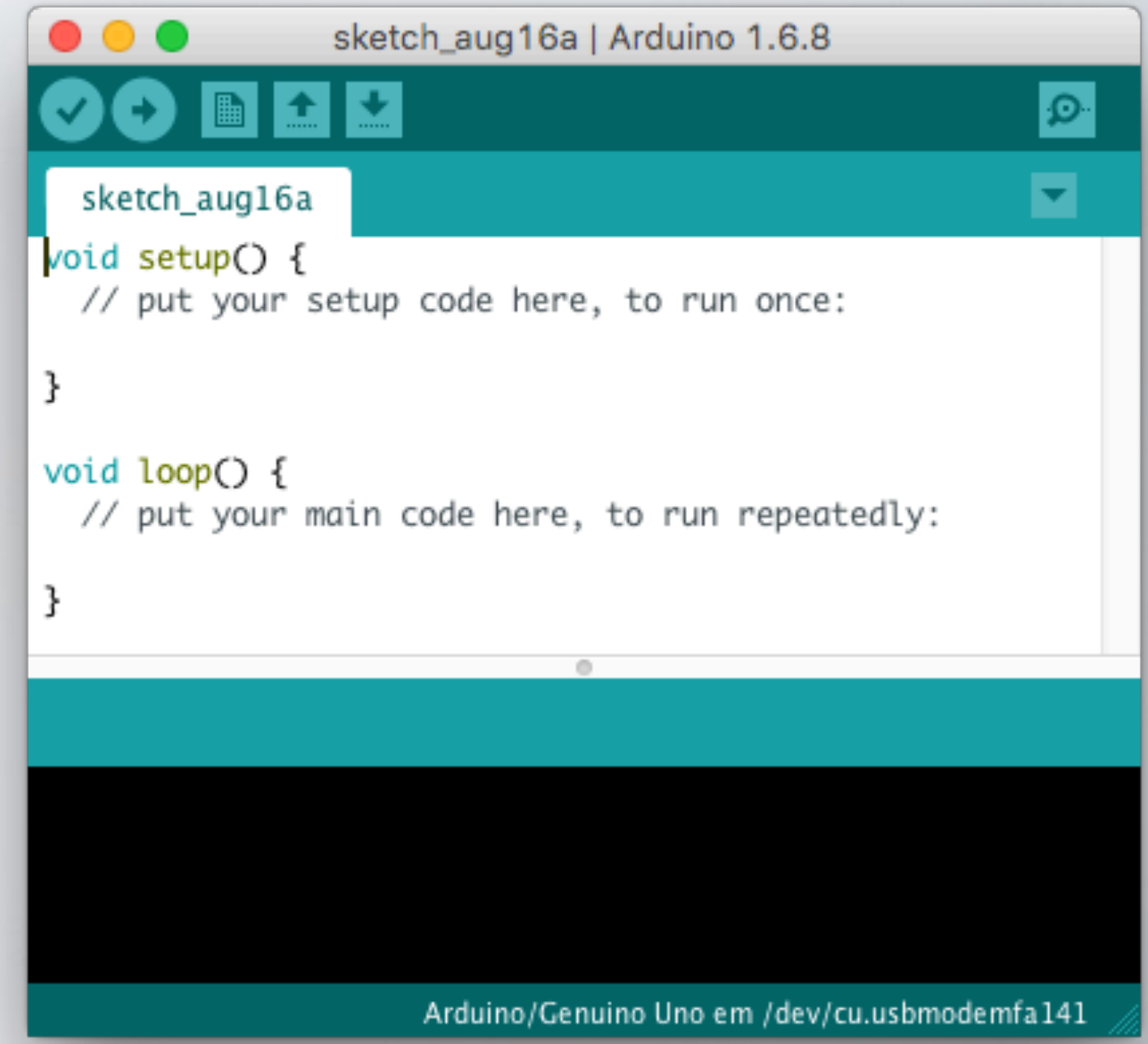
Como Programar?

Arduino

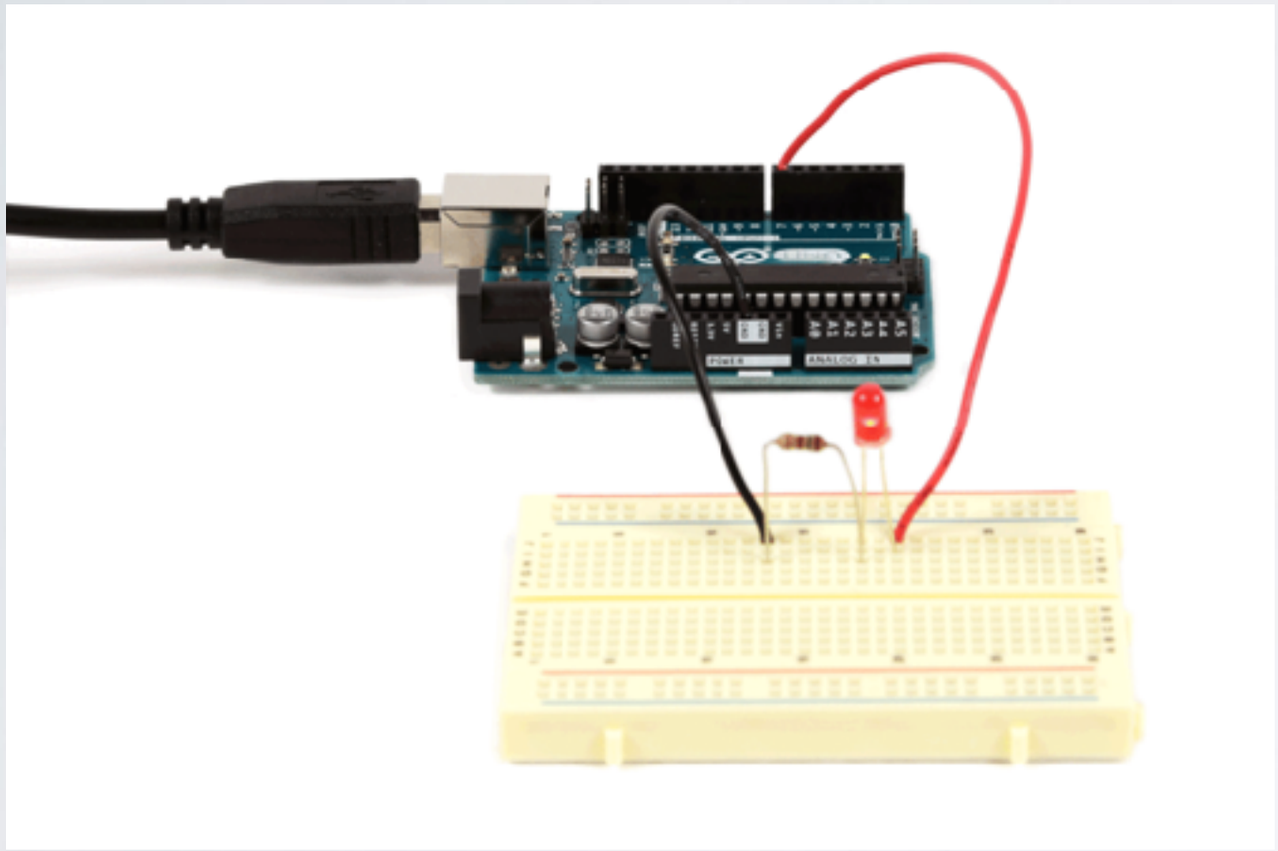
- Detalhes...

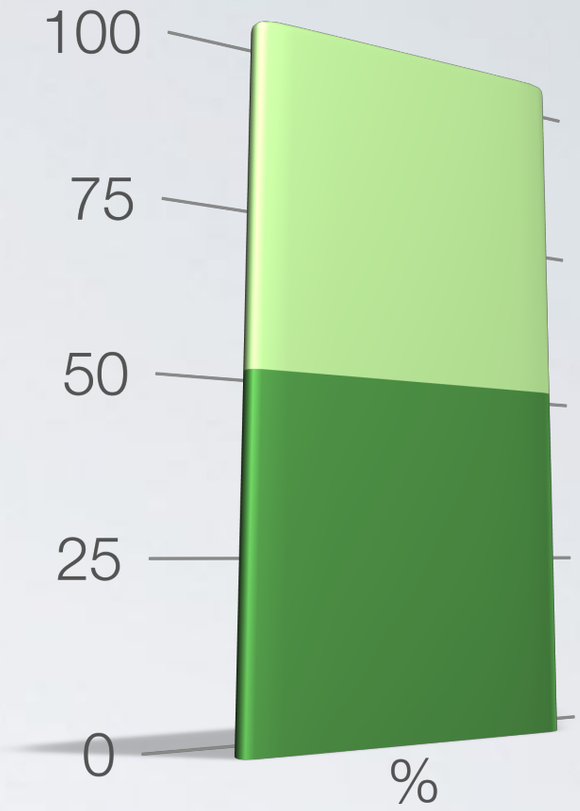
Parte do código que só é realizado **uma única vez.** Para inicializar entradas/saídas e variáveis.

Parte do **programa principal** (“main”). O código dentro deste espaço é **repetido indefinidamente**.



Primeiro Programa: Piscar um Led





- Software

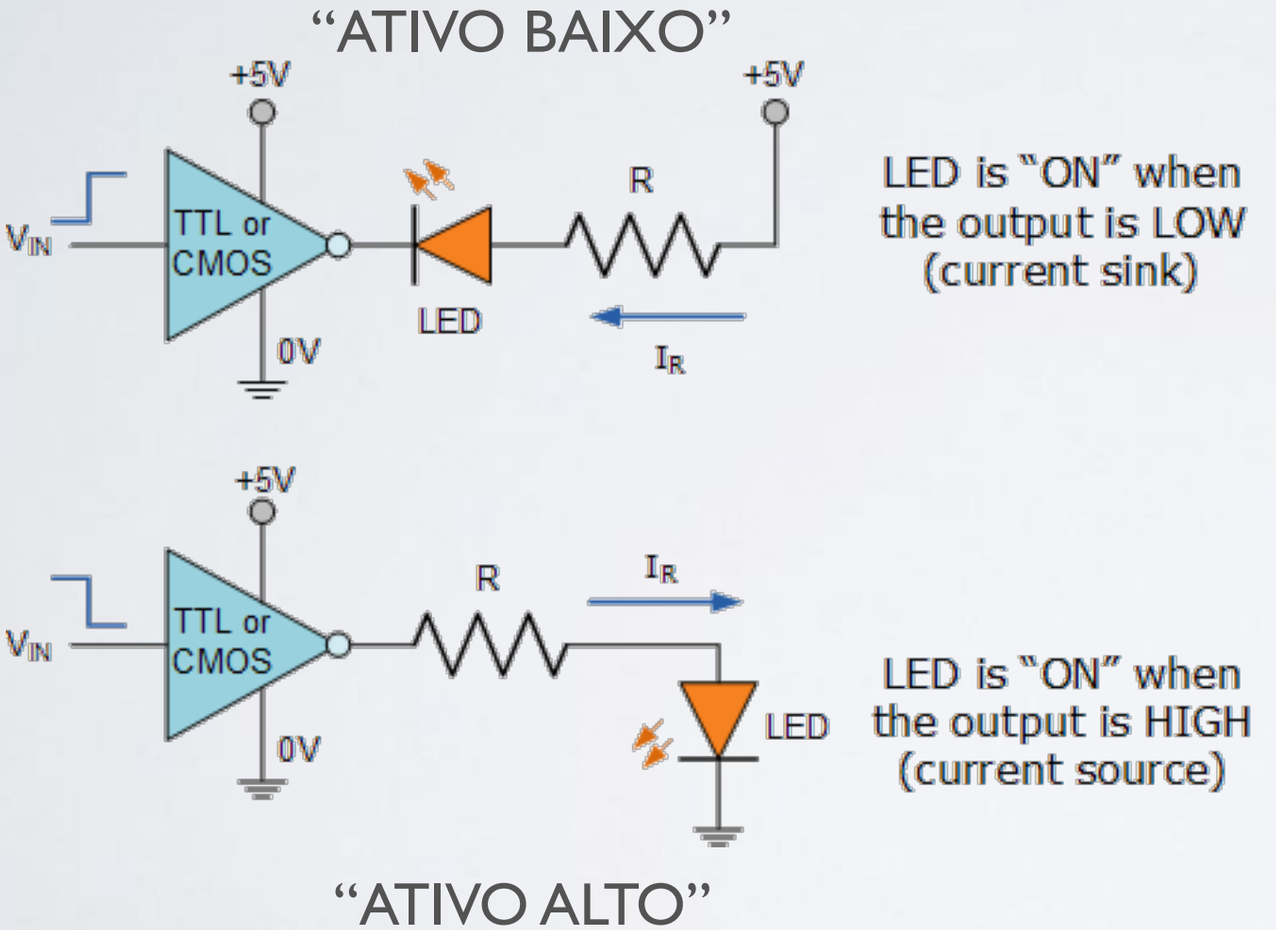
+

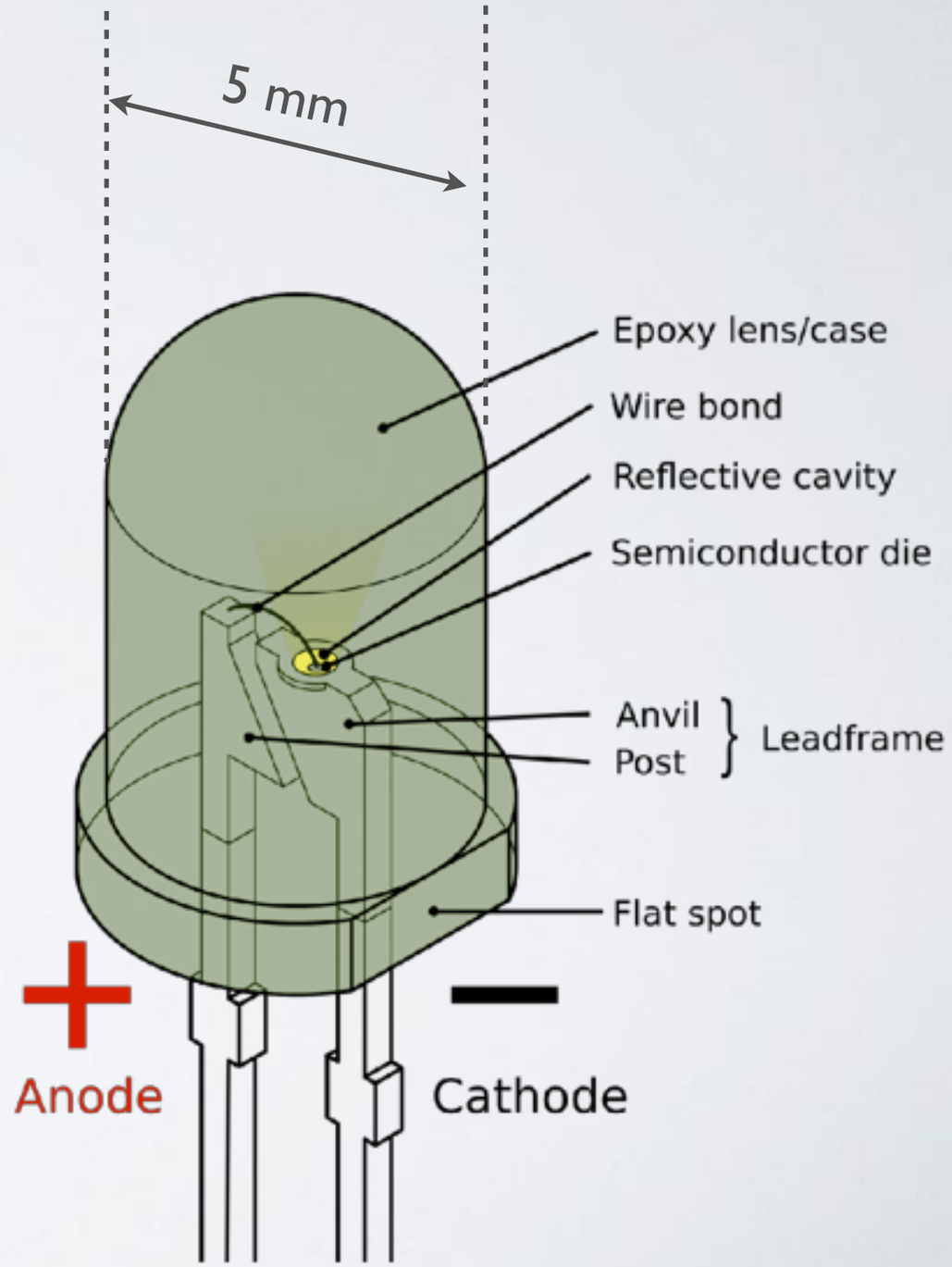
- Hardware

Exp.1) Como ativar um Led

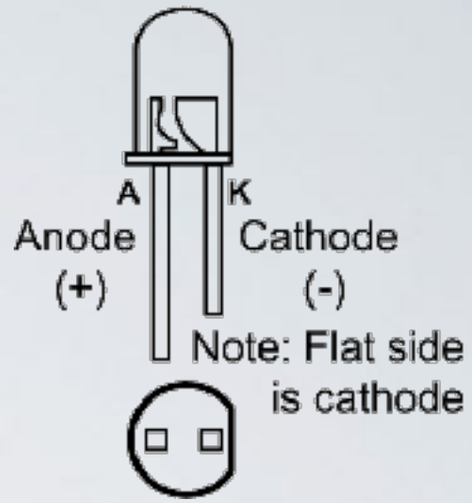
- Circuito elétrico: **Note** ☞ possui polaridade!

Configurações possíveis:



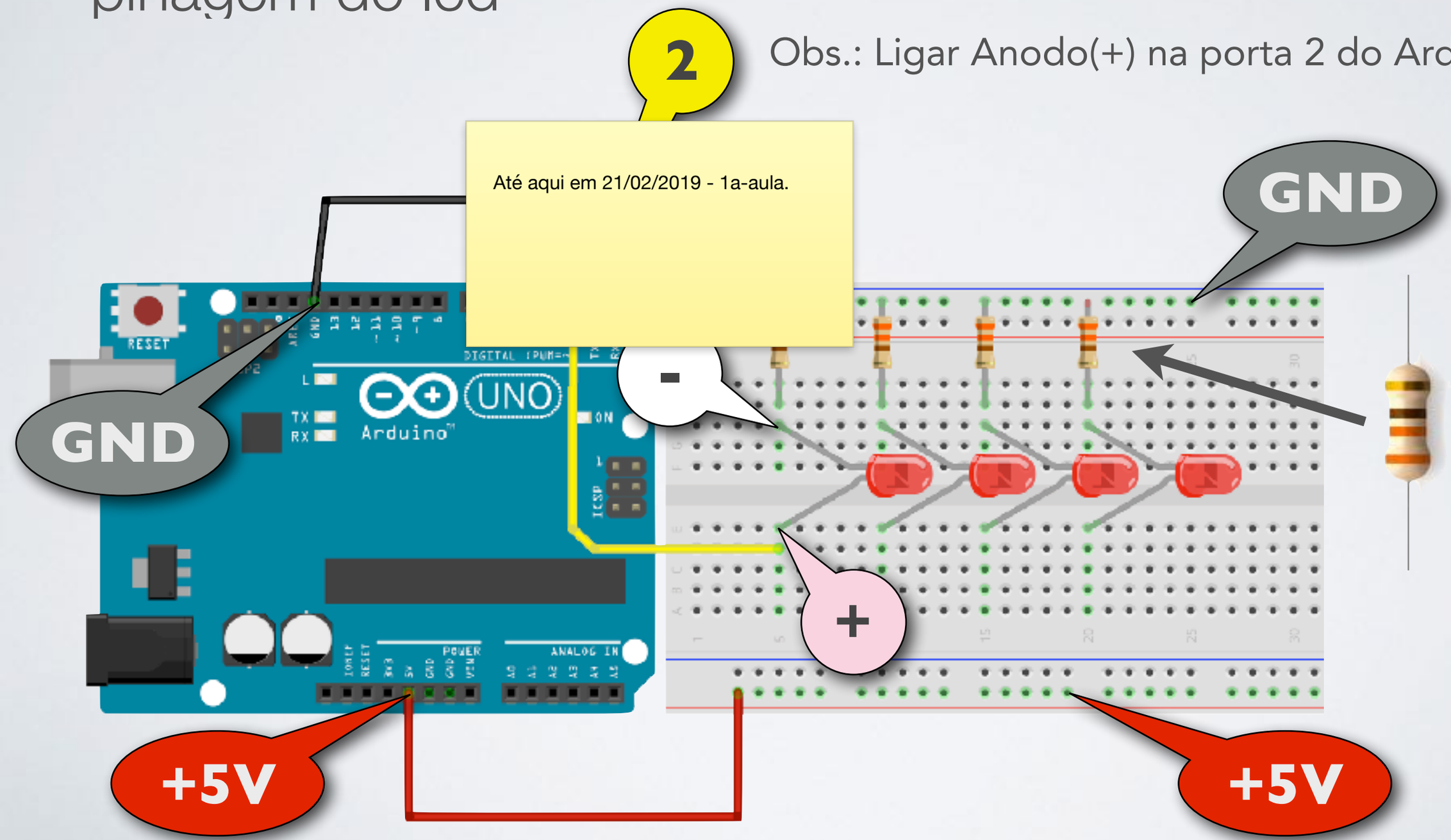


Exp_1) Piscar um Led!



- Diagrama elétrico + pinagem do led

Obs.: Ligar Anodo(+) na porta 2 do Arduino.

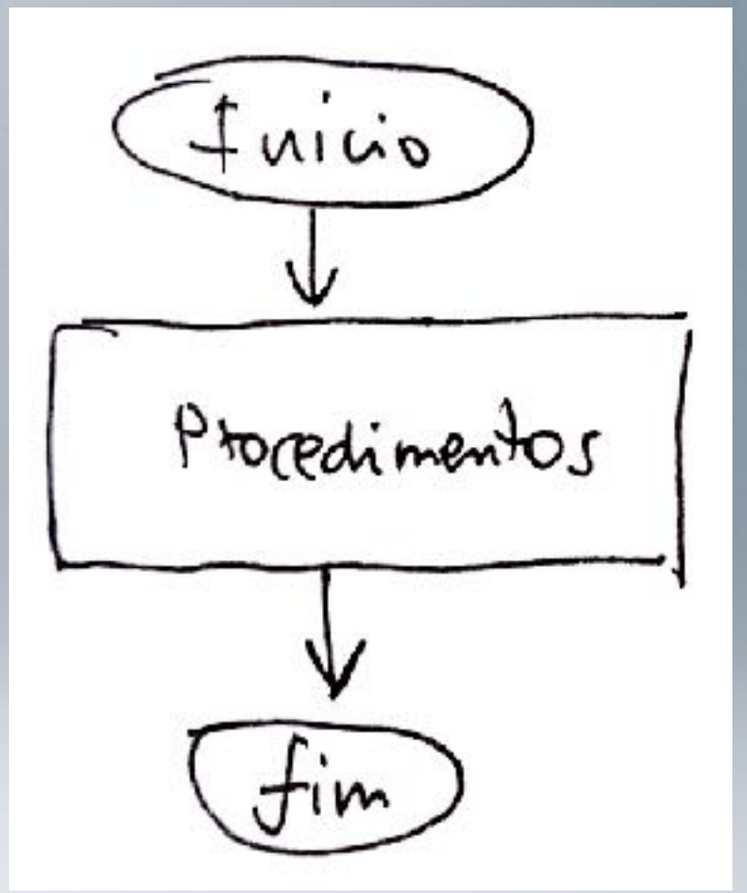


Resistores 330Ω:
laranja;
laranja;
marrom.

O que é Programar?

Todos os programas usam instruções básicas como blocos de construção. Aqui estão alguns dos mais comuns, em português:

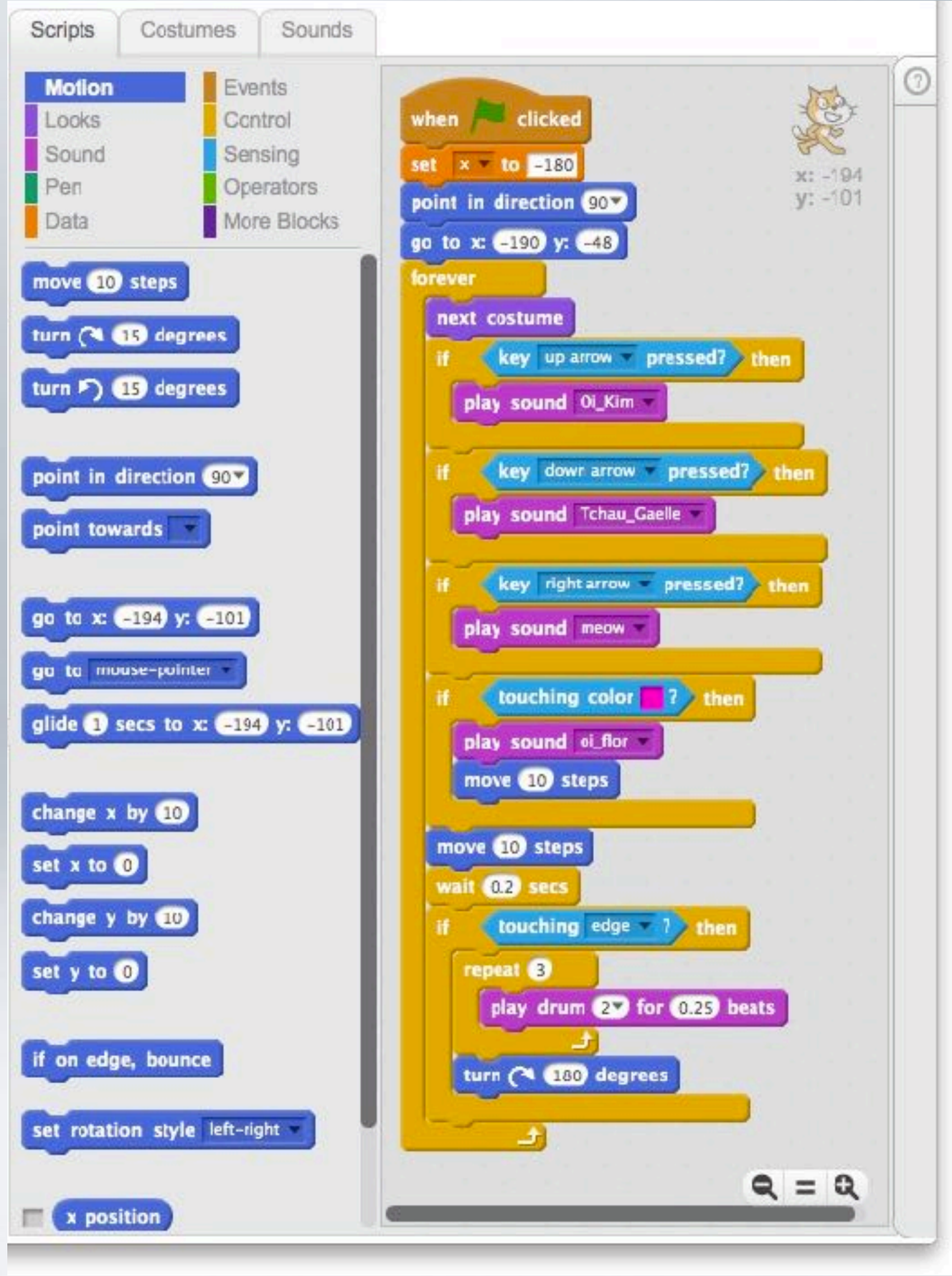
- "Faça isso; então faça isso. ”
- “Se esta condição for verdadeira, execute esta ação; caso contrário, faça essa ação.”
- “Repita essa ação esse número de vezes. ”
- "Continue fazendo isso até que essa condição seja verdadeira.”



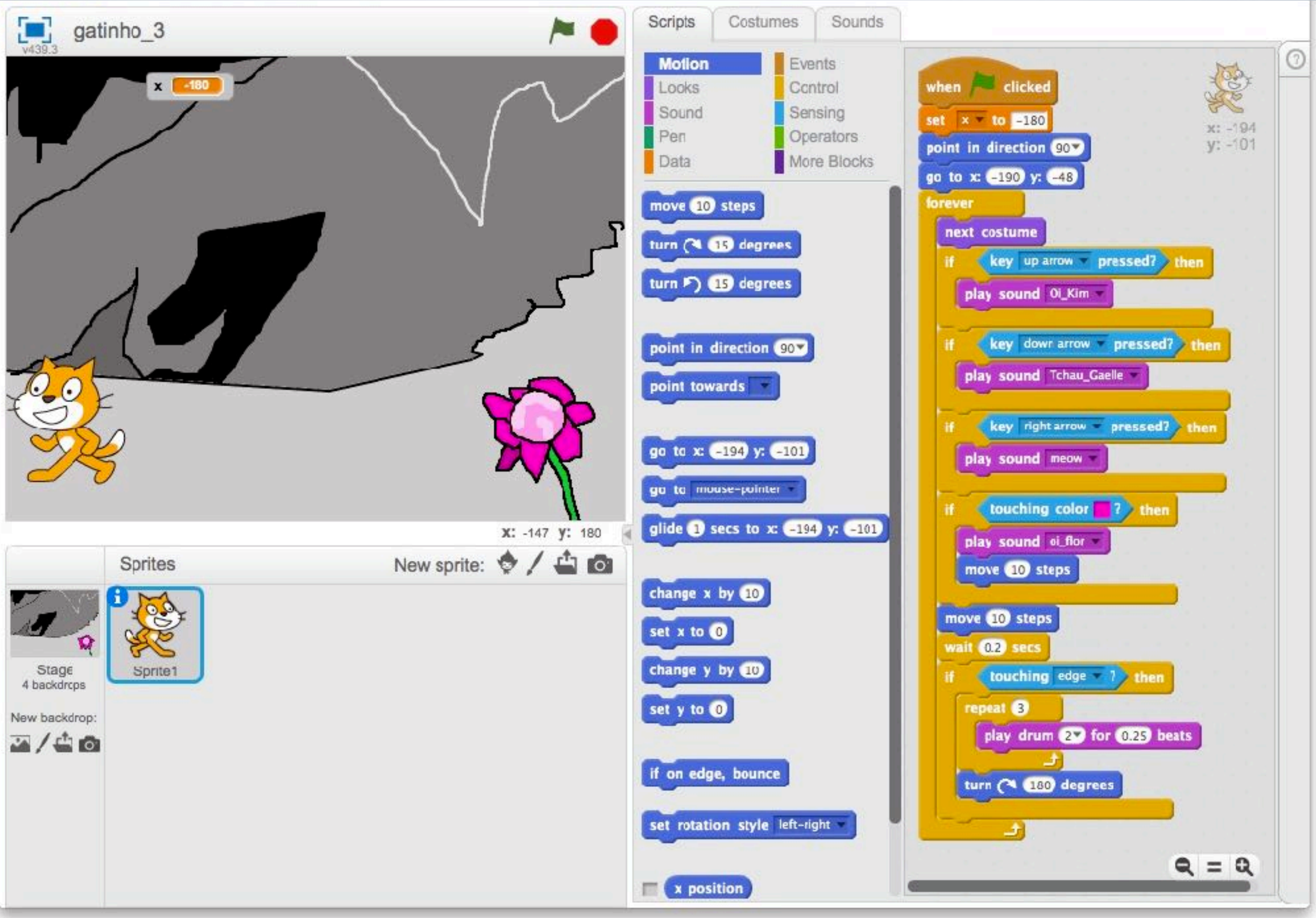
Você pode combinar esses blocos de construção para implementar decisões mais complexas também.

FORMAS DE Programação

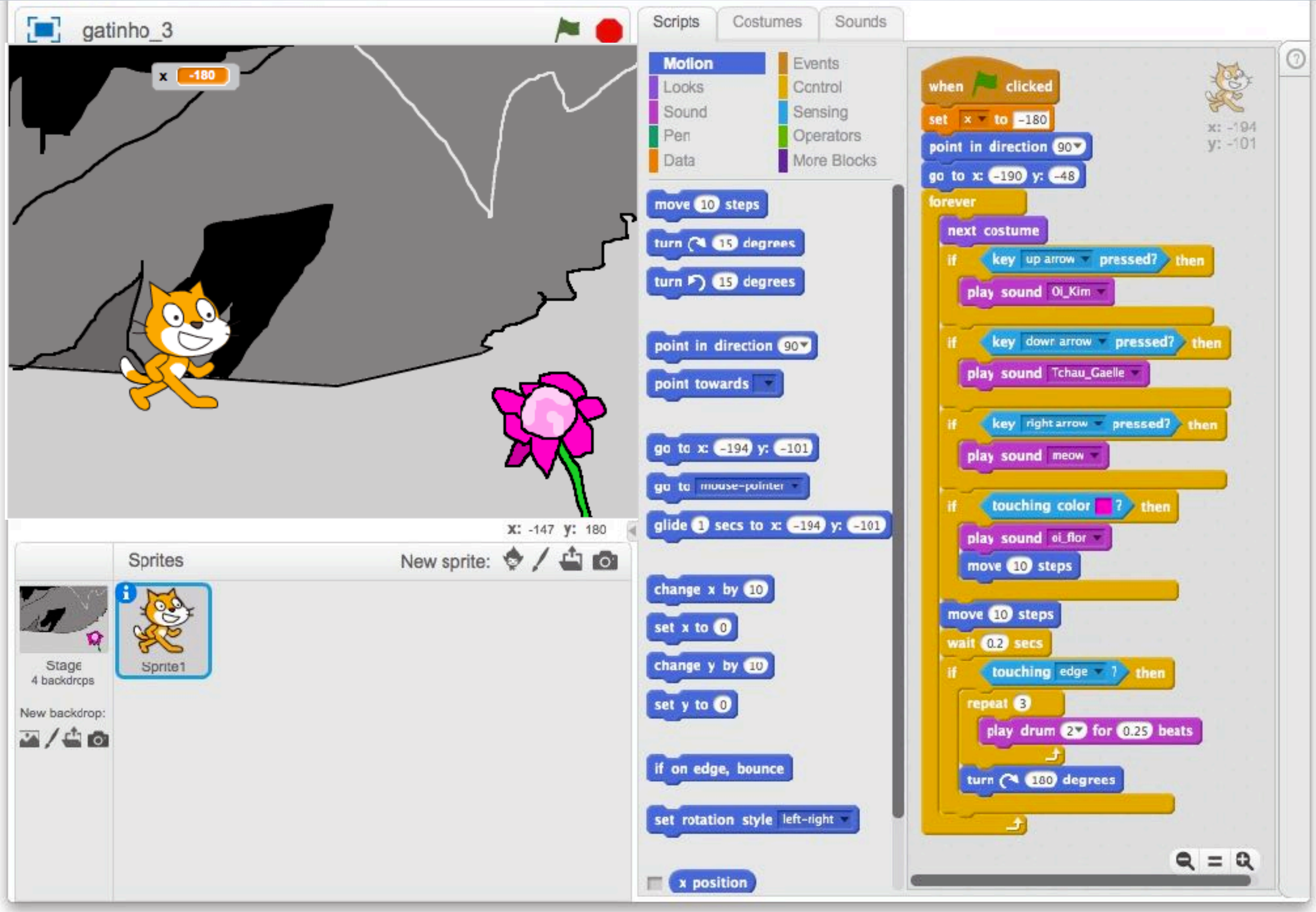
- Gráfica (icônica)
- Textual



Exemplo: Scratch Project (https://scratch.mit.edu)



Exemplo: Scratch Project (https://scratch.mit.edu)



Exemplo: Scratch Project (https://scratch.mit.edu)

FORMAS DE Programação

- Gráfica (icônica)
- Textual

movimento_leds_vai_volta | Arduino 1.8.5

movimento leds vai volta

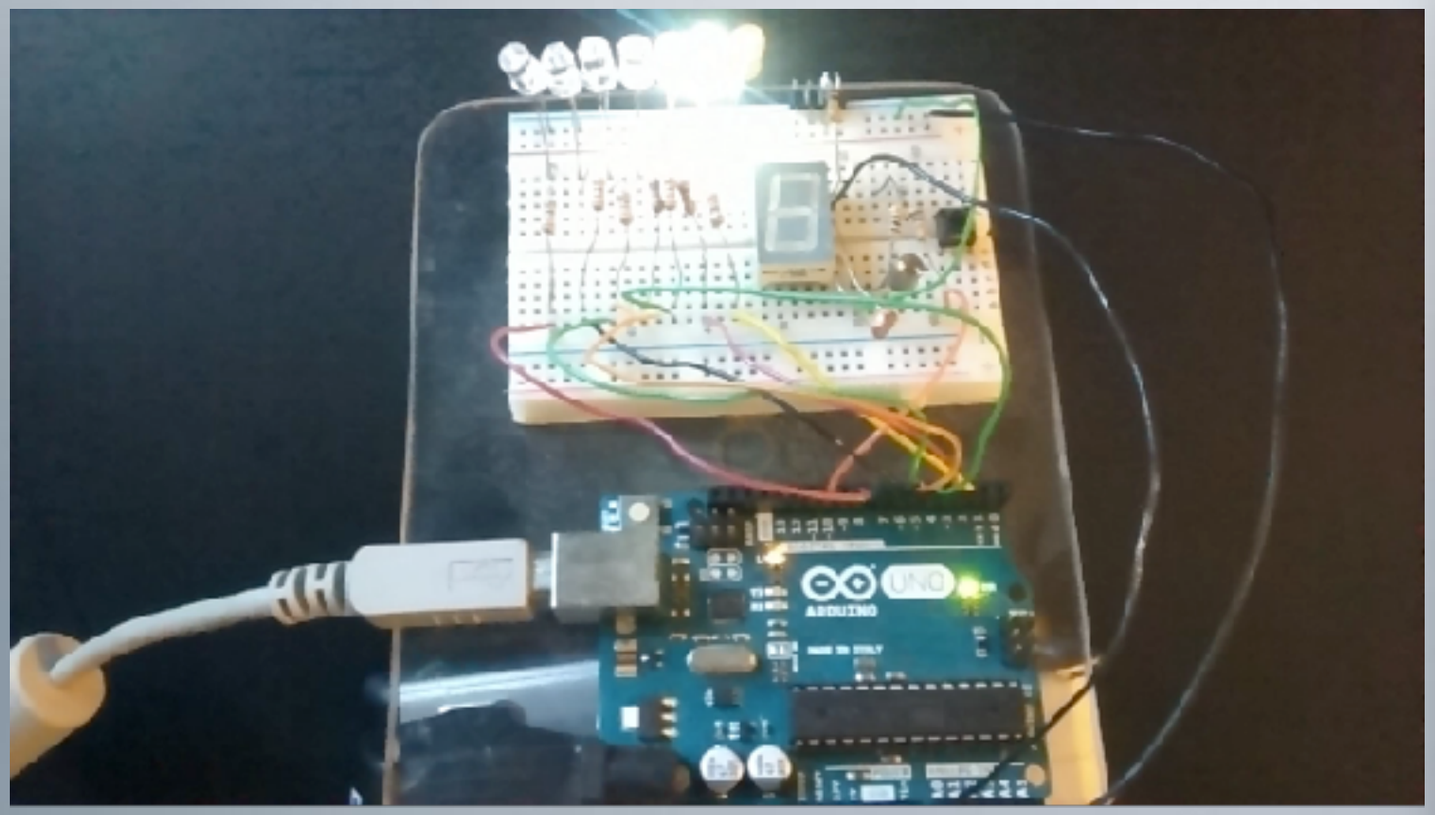
```
void setup() {
  // put your setup code here, to run once:
  int i; // declara a variável i como inteiro (sem casas decimais)
  for (i=2; i <=8; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, HIGH); // Ativa Led pino i
    delay(50);
  }
  delay(50);
  for (i=7; i>=5; i--){
    digitalWrite(i, LOW); // Apaga Led pino i
    delay(100);
  }
  delay(750);
  digitalWrite(2, HIGH); // Liga Led do pino 2
}

void loop() {
  // put your main code here, to run repeatedly:
  int i;
  int tempo=50;

  // Movimento de "ida"
  // digitalWrite(2, HIGH); // Liga Led do pino 2
  delay(tempo);
  for (i=2; i<=7; i++){
    digitalWrite(i, LOW); // Apaga Led atual (pino i)
    digitalWrite(i+1, HIGH); // Ativa próximo Led (pino i+1)
    delay(tempo);
  } // Notar que 2 <= i <= 4
  // Laço de repetição anterior termina com Led 5 ativado!

  // Movimento de "volta"
  for (i=8; i>=3; i--){
    digitalWrite(i, LOW); // Apaga Led atual (pino i)
    digitalWrite(i-1, HIGH); // Ativa Led anterior (pino i-1)
    delay(tempo);
  }
  // Notar que termina com Led 2 ativado!
}
```

Salvo.

O sketch usa 1100 bytes (3%) de espaço de armazenamento para programa
Variáveis globais usam 9 bytes (0%) de memória dinâmica, deixando 203

19 Arduino/Genuino Uno em /dev/cu.usbmodemFD131

Exemplo: Linguagem “C” Arduino IDE (https://www.arduino.cc/en/Main/Software)



- Gráfica (icônica)
- Textual

movimento_leds_vai_volta | Arduino 1.8.5

movimento_leds_vai_volta

```
void setup() {
  // put your setup code here, to run once:
  int i; // declara a variável i como inteiro (sem casas decimais)
  for (i=2; i <=8; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, HIGH); // Ativa Led pino i
    delay(50);
  }
  delay(50);
  for (i=7; i>=3; i--){
    digitalWrite(i, LOW); // Apaga Led pino i
    delay(100);
  }
  delay(750);
  digitalWrite(2, HIGH); // Liga Led do pino 2
}

void loop() {
  // put your main code here, to run repeatedly:
  int i;
  int tempo=50;

  // Movimento de "ida"
  // digitalWrite(2, HIGH); // Liga Led do pino 2
  delay(tempo);
  for (i=2; i<=7; i++){
    digitalWrite(i, LOW); // Apaga Led atual (pino i)
    digitalWrite(i+1, HIGH); // Ativa próximo Led (pino i+1)
    delay(tempo);
  } // Notar que 2 <= i <= 4
  // Laço de repetição anterior termina com Led 5 ativado!

  // Movimento de "volta"
  for (i=8; i>=3; i--){
    digitalWrite(i, LOW); // Apaga Led atual (pino i)
    digitalWrite(i-1, HIGH); // Ativa Led anterior (pino i-1)
    delay(tempo);
  }
  // Notar que termina com Led 2 ativado!
}
```

Salvo.

O sketch usa 1100 bytes (3%) de espaço de armazenamento para programas
Variáveis globais usam 9 bytes (0%) de memória dinâmica, deixando 203

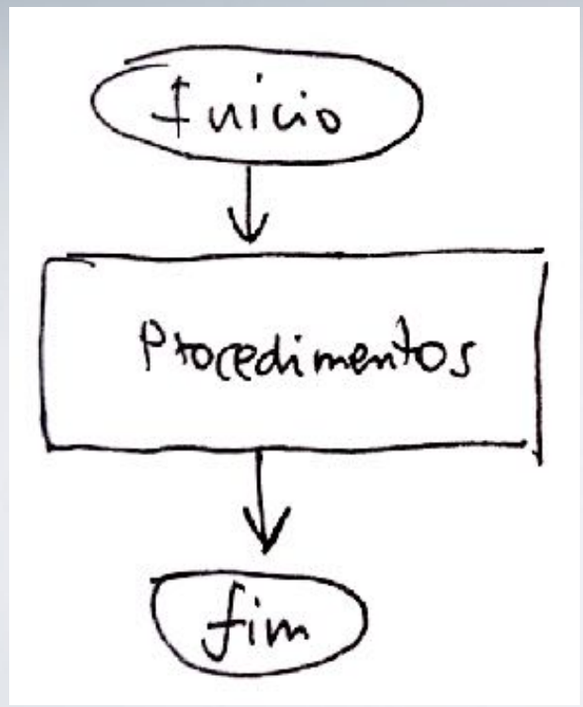
19 Arduino/Genuino Uno em /dev/cu.usbmodemFD131

Exemplo: Linguagem “C” Arduino IDE
(https://www.arduino.cc/en/Main/Software)

O que é Programar?

Por exemplo, aqui estão as instruções de programação, chamadas de código-fonte, para um programa simples escrito na linguagem de programação C do Arduino.

Começando no topo, o sistema executa cada linha de código (algumas linhas são executadas apenas se uma determinada condição for verdadeira ou então o sistema executa alguma outra linha) até atingir a parte inferior.

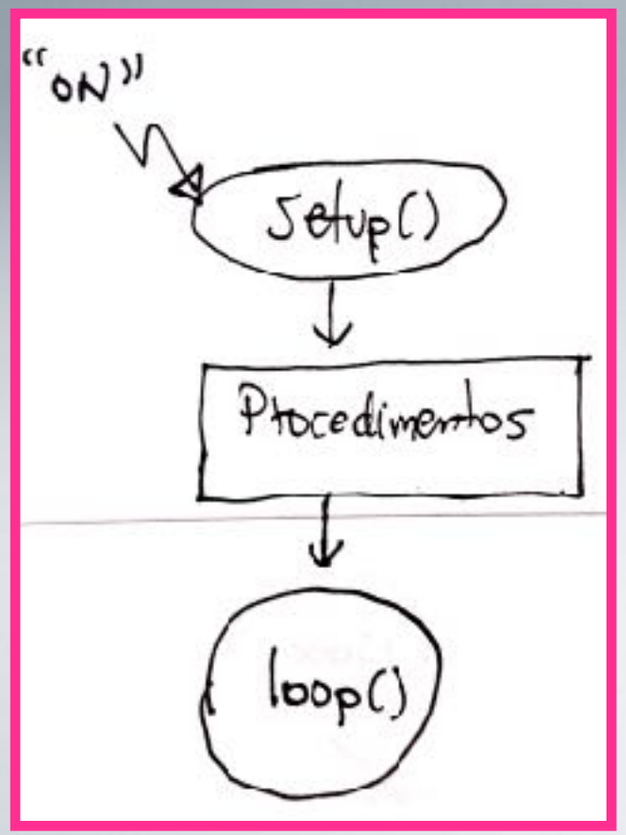


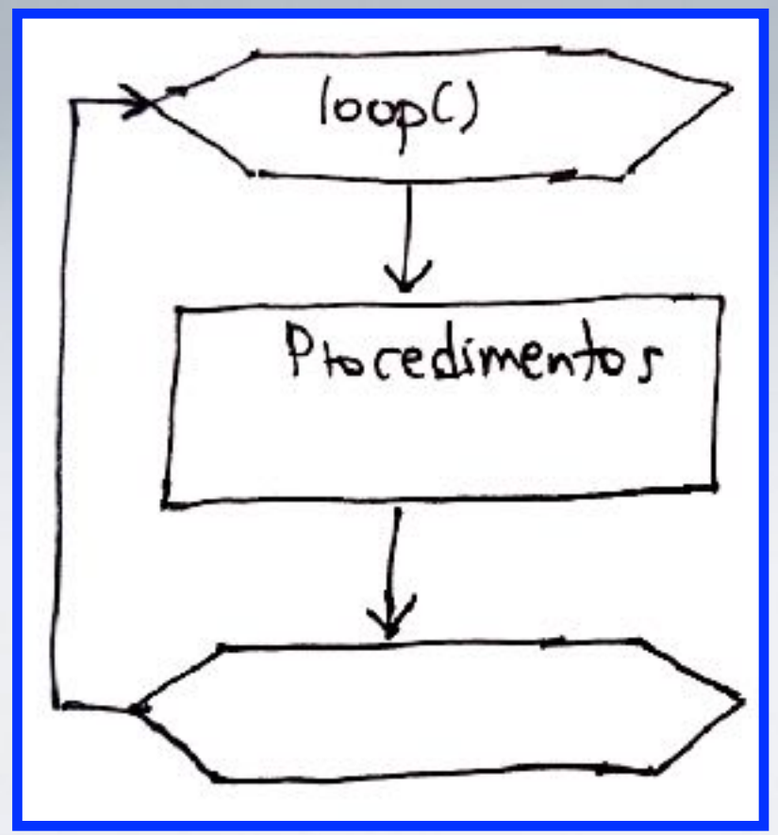

```
void setup() {
  // put your setup code here, to run once:
  int i; // declara a variável i como inteiro (sem casas
decimais)
  for (i=2; i <=8; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, HIGH); // Ativa Led pino i
    delay(50);
  }
  delay(50);
  for (i=2; i<=8; i++){
    digitalWrite(i, LOW); // Apaga Led pino i
    delay(100);
  }
  delay(250);
  digitalWrite(2, HIGH); // Liga Led do pino 2
}

void loop() {
  // put your main code here, to run repeatedly:
  int i;
  int tempo=50;

  // Movimento de "ida"
  // digitalWrite(2, HIGH); // Liga Led do pino 2
  delay(tempo);
  for (i=2; i<=7; i++){
    digitalWrite(i, LOW); // Apaga Led atual (pino i)
    digitalWrite(i+1, HIGH); // Ativa próximo Led (pino
i+1)
    delay(tempo);
  } // Notar que 2 <= i <= 4
  // Laço de repetição anterior termina com Led 5
ativado!

  // Movimento de "volta"
  for (i=8; i>=3; i--){
    digitalWrite(i, LOW); // Apaga Led atual (pino i)
    digitalWrite(i-1, HIGH); // Ativa Led anterior (pino i-1)
    delay(tempo);
  }
  // Notar que termina com led 2 ativado!
}
```

O que é Programar?

No caso **específico** do Arduino existem 2 blocos principais:

- **setup()** ☞ Executado uma **única** vez (logo no início);
- **loop()** ☛ Executado indefinidamente (**repetido indefinidamente**)

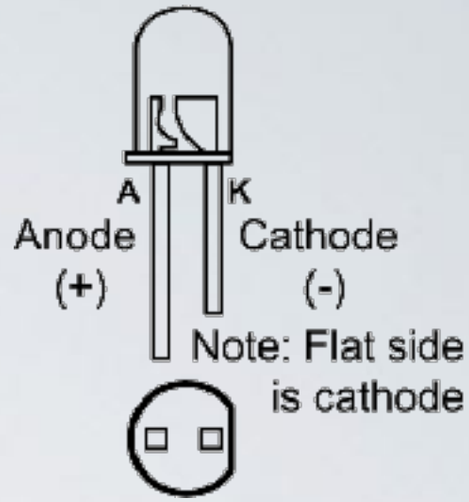


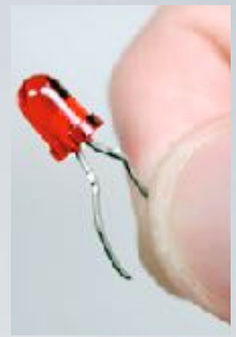



```
void setup() {
  // put your setup code here, to run once:
  int i; // declara a variável i como inteiro (sem casas decimais)
  for (i=2; i <=8; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, HIGH); // Ativa Led pino i
    delay(50);
  }
  delay(50);
  for (i=2; i<=8; i++){
    digitalWrite(i, LOW); // Apaga Led pino i
    delay(100);
  }
  delay(250);
  digitalWrite(2, HIGH); // Liga Led do pino 2
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  int i;
  int tempo=50;

  // Movimento de "ida"
  // digitalWrite(2, HIGH); // Liga Led do pino 2
  delay(tempo);
  for (i=2; i<=7; i++){
    digitalWrite(i, LOW); // Apaga Led atual (pino i)
    digitalWrite(i+1, HIGH); // Ativa próximo Led (pino i+1)
    delay(tempo);
  } // Notar que 2 <= i <= 4
  // Laço de repetição anterior termina com Led 5 ativado!

  // Movimento de "volta"
  for (i=8; i>=3; i--){
    digitalWrite(i, LOW); // Apaga Led atual (pino i)
    digitalWrite(i-1, HIGH); // Ativa Led anterior (pino i-1)
    delay(tempo);
  }
  // Notar que termina com led 2 ativado!
}
```

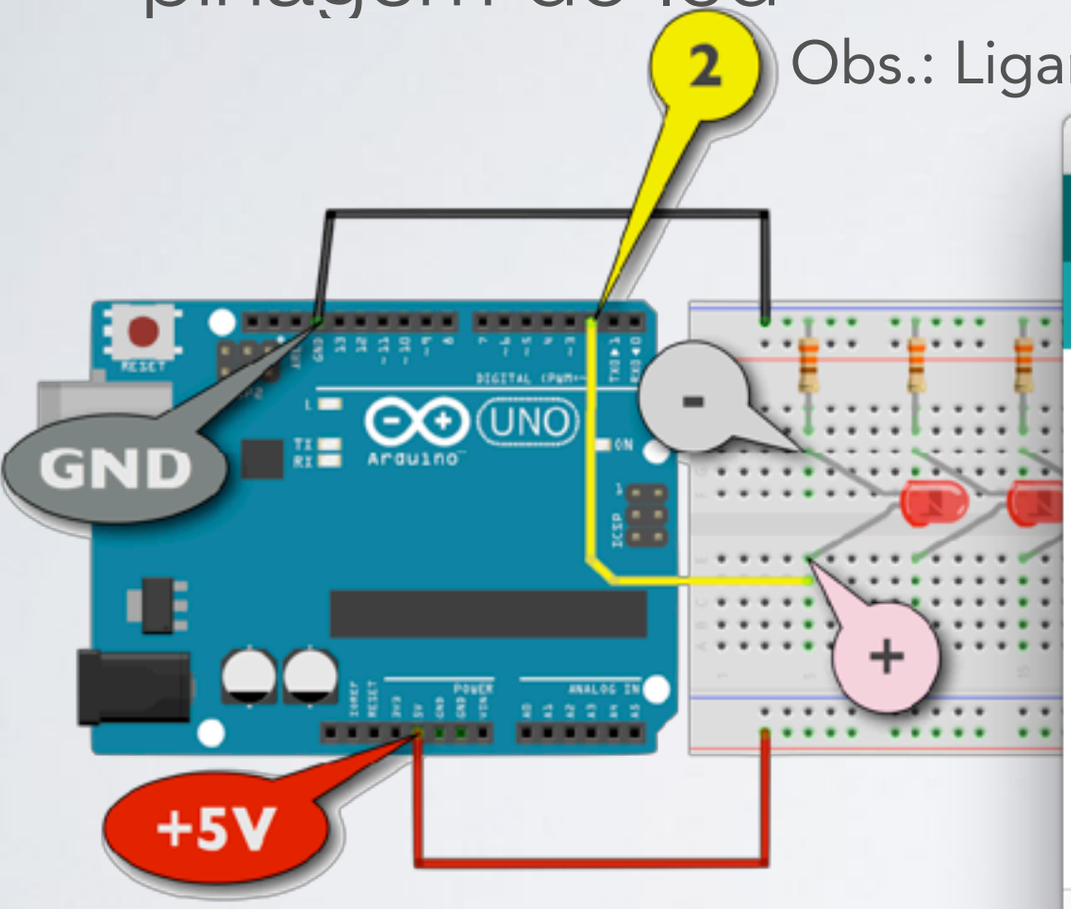
Exp_1) Piscar um Led!

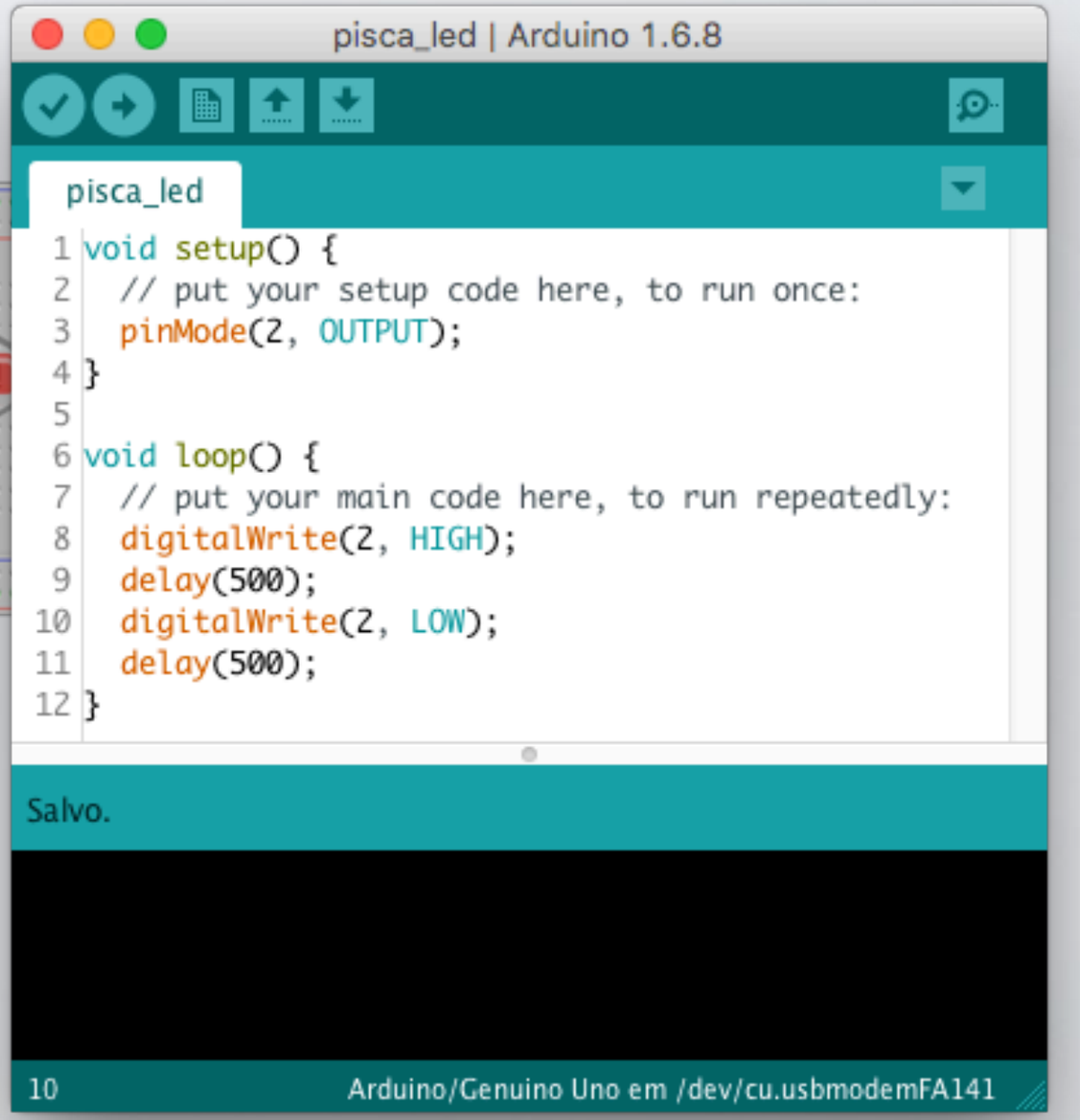




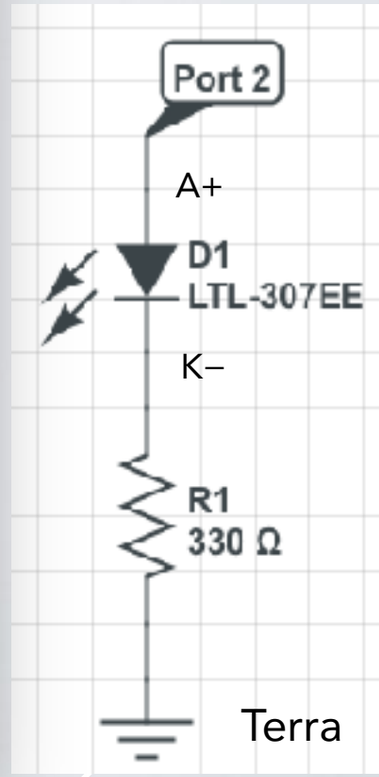
- Diagrama elétrico + pinagem do led

Obs.: Ligar Anodo(+) na porta 2 do Arduino.

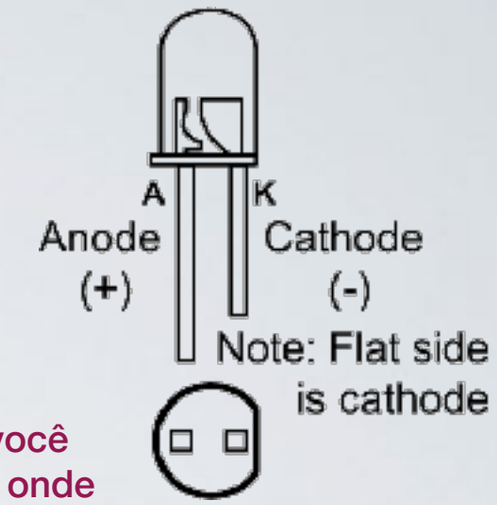




Resistor 330Ohms:
laranja;
laranja;
marrom.



Exp_1) Piscar um Led!



- **Software: Melhorando o código...**

  Obs.: Ligar Anodo(+) na porta 2 do Arduino.

Facilita se você mudar pino onde conectou led

pisca_led | Arduino 1.6.8

pisca_led

```
1  void setup() {
2    // put your setup code here, to run once:
3    pinMode(2, OUTPUT);
4  }
5
6  void loop() {
7    // put your main code here, to run repeat
8    digitalWrite(2, HIGH);
9    delay(500);
10   digitalWrite(2, LOW);
11   delay(500);
12 }
```

Salvo.

10 Arduino/Genuino Uno em /dev/cu.usbmo

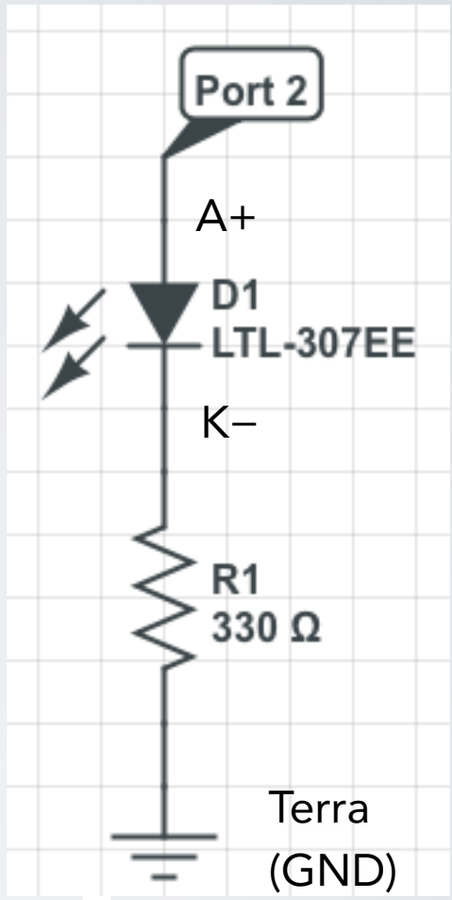
pisca_led2 | Arduino 1.6.8

pisca_led2

```
1  const int ledPin =  2;  // the number of the LED pin
2
3  void setup() {
4    // put your setup code here, to run once:
5    pinMode(ledPin, OUTPUT);
6  }
7
8  void loop() {
9    // put your main code here, to run repeatedly:
10   digitalWrite(ledPin, HIGH);
11   delay(500);
12   digitalWrite(ledPin, LOW);
13   delay(500);
14 }
```

Salvo.

5 Arduino/Genuino Uno em /dev/cu.usbmodemFA141

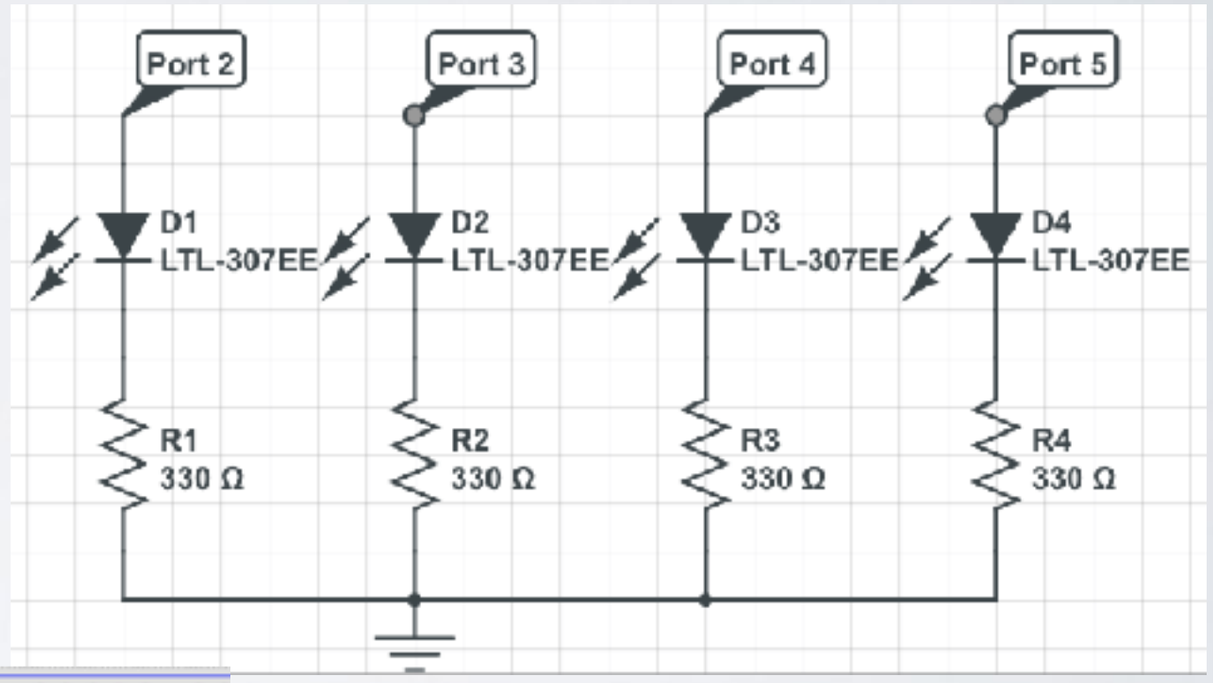


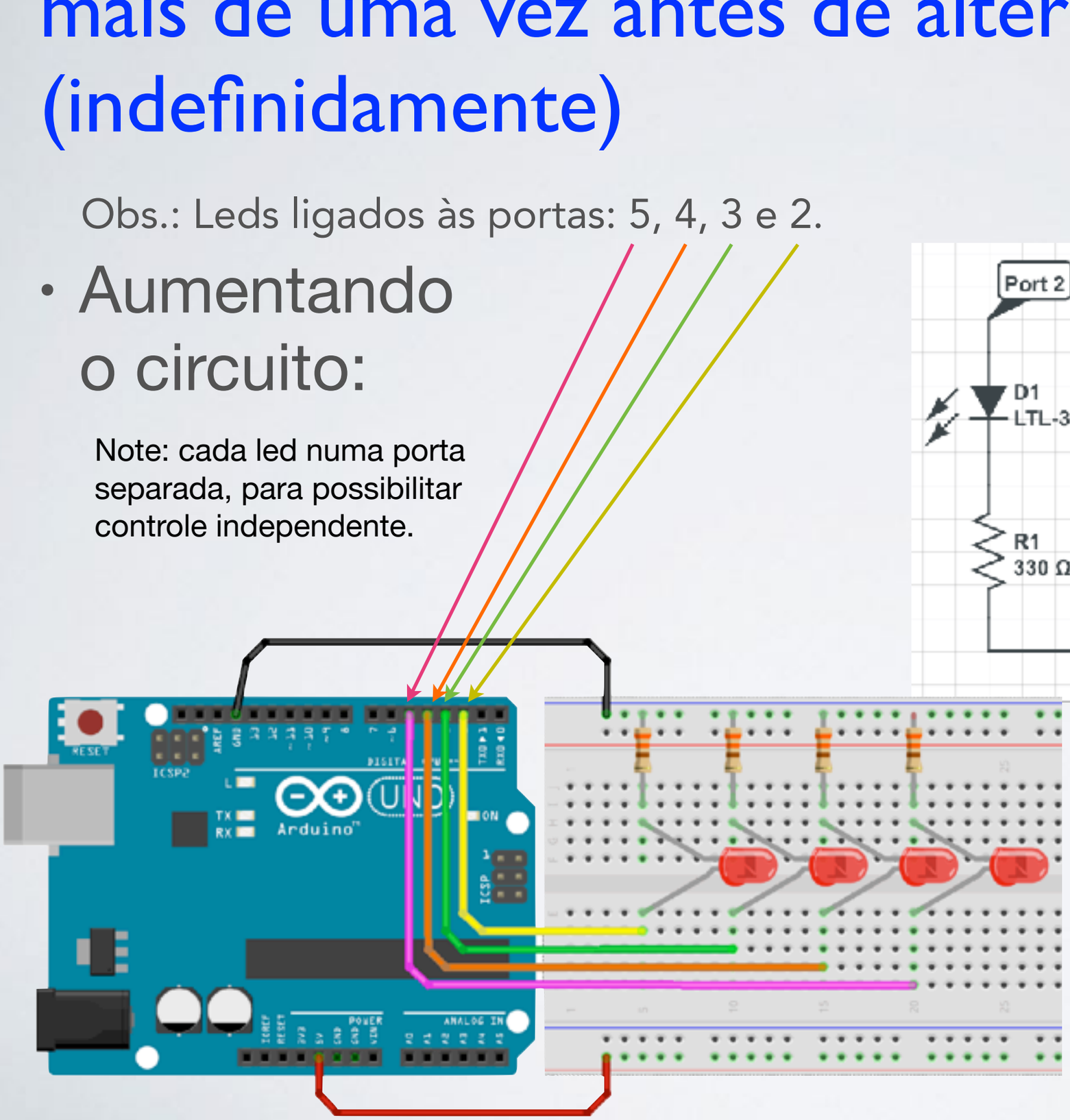
Exp_2) Piscar leds aos pares…
mais de uma vez antes de alternar os pares (indefinidamente)

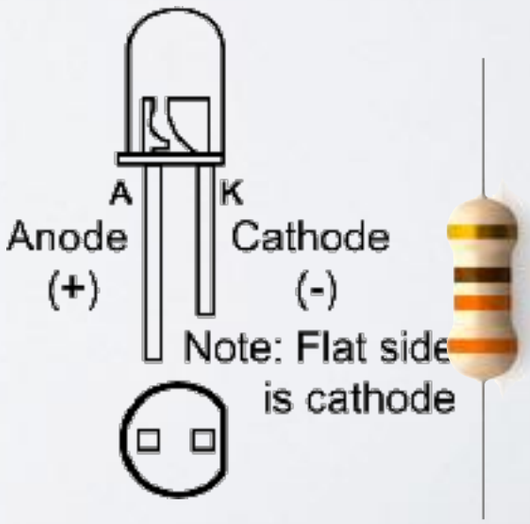
Obs.: Leds ligados às portas: 5, 4, 3 e 2.

- Aumentando o circuito:

Note: cada led numa porta separada, para possibilitar controle independente.

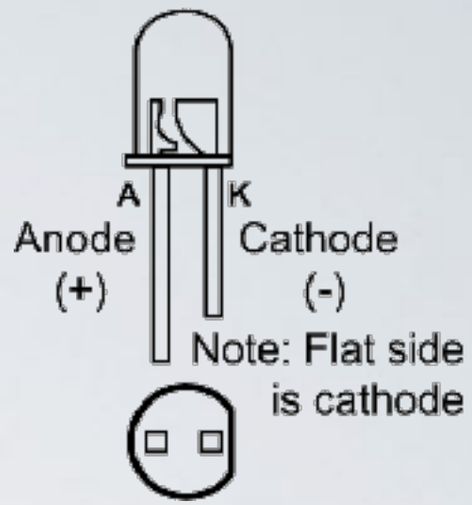






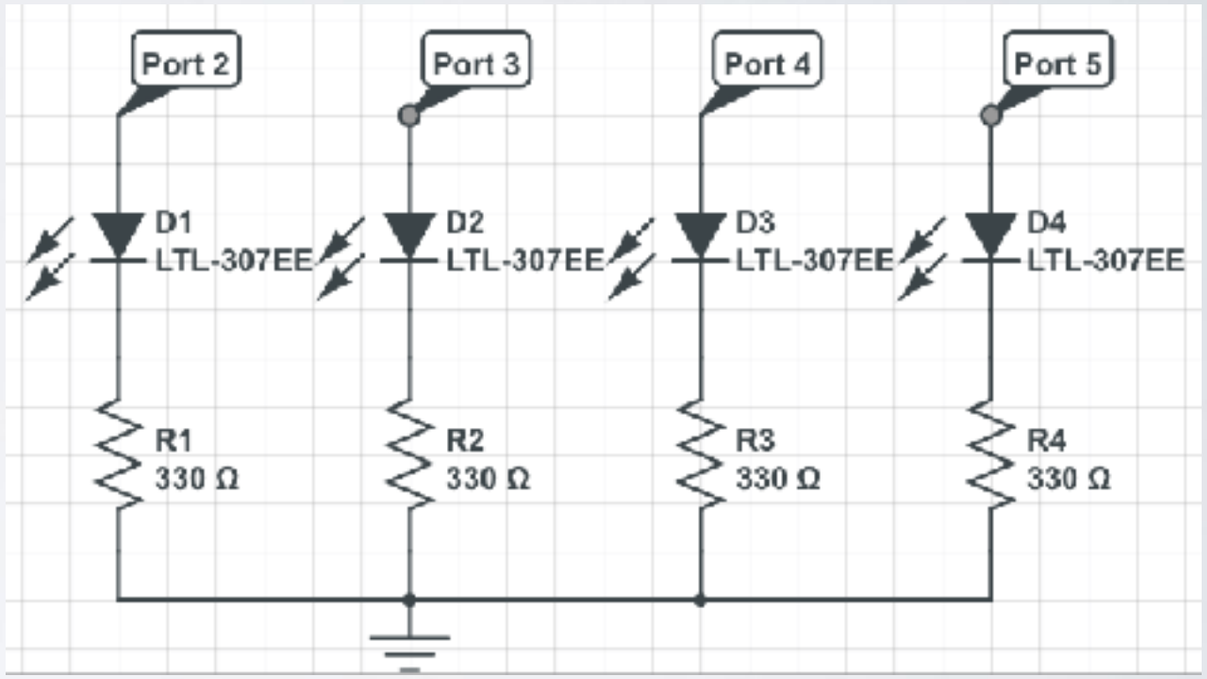
Resistor 330Ohms: laranja; laranja; marrom.

Exp_2) Alerta Visual!



- Circuito

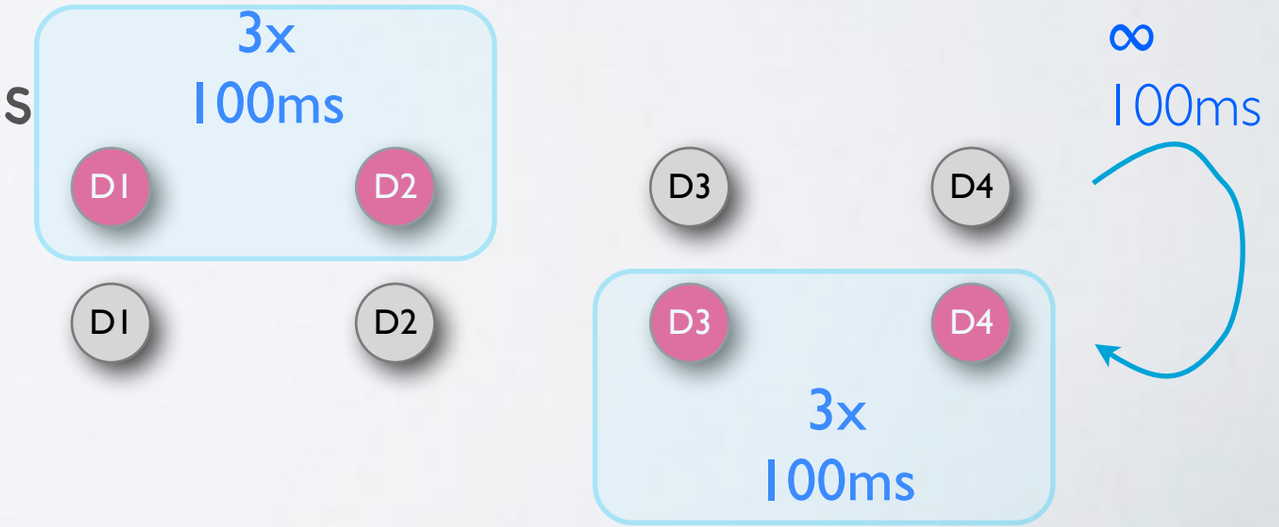
  Obs.: Leds ligados às portas: 5, 4, 3 e 2.



- Problemas:

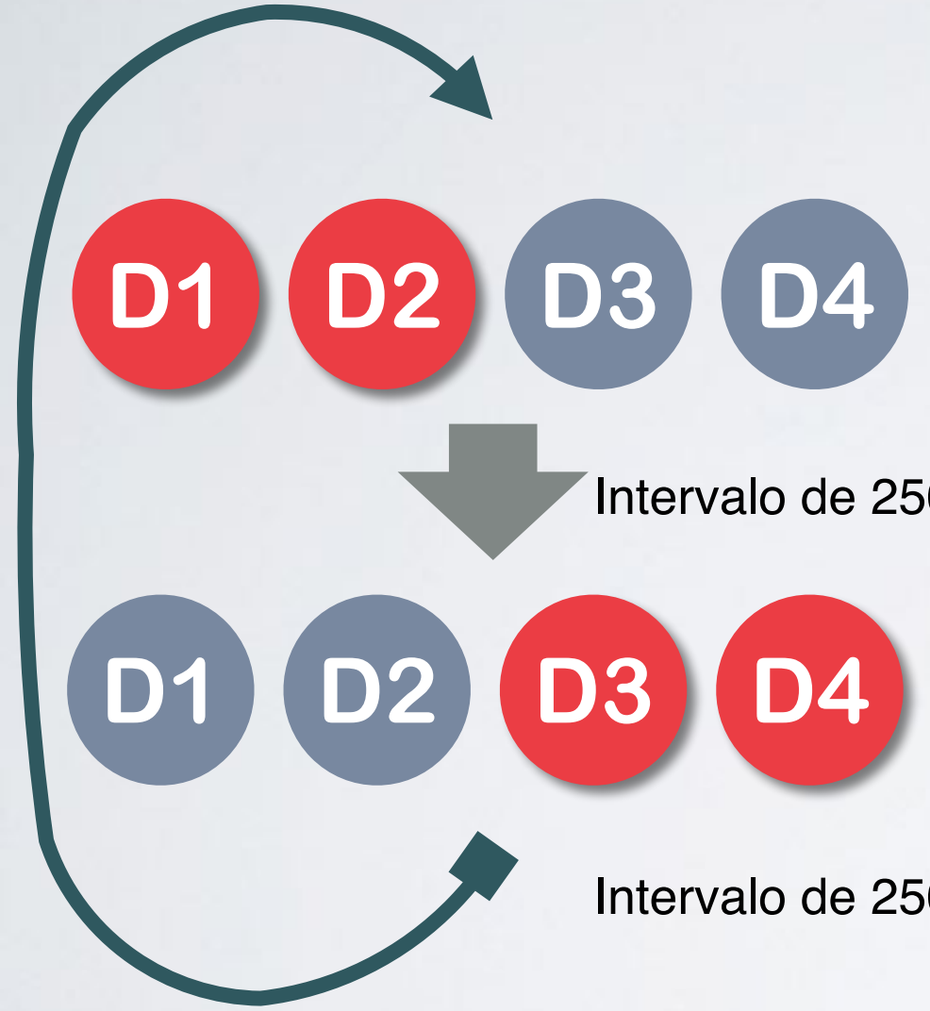
Até aqui em 10/05/2018

2. Piscar 2 leds de c
   vez que acionar 1 par, pisca-lo 3
   vezes a cada 100ms e somente
   depois repetir o mesmo
   procedimetno para os outros 2 pares
   de leds.
   (imita viatura/ambulância)



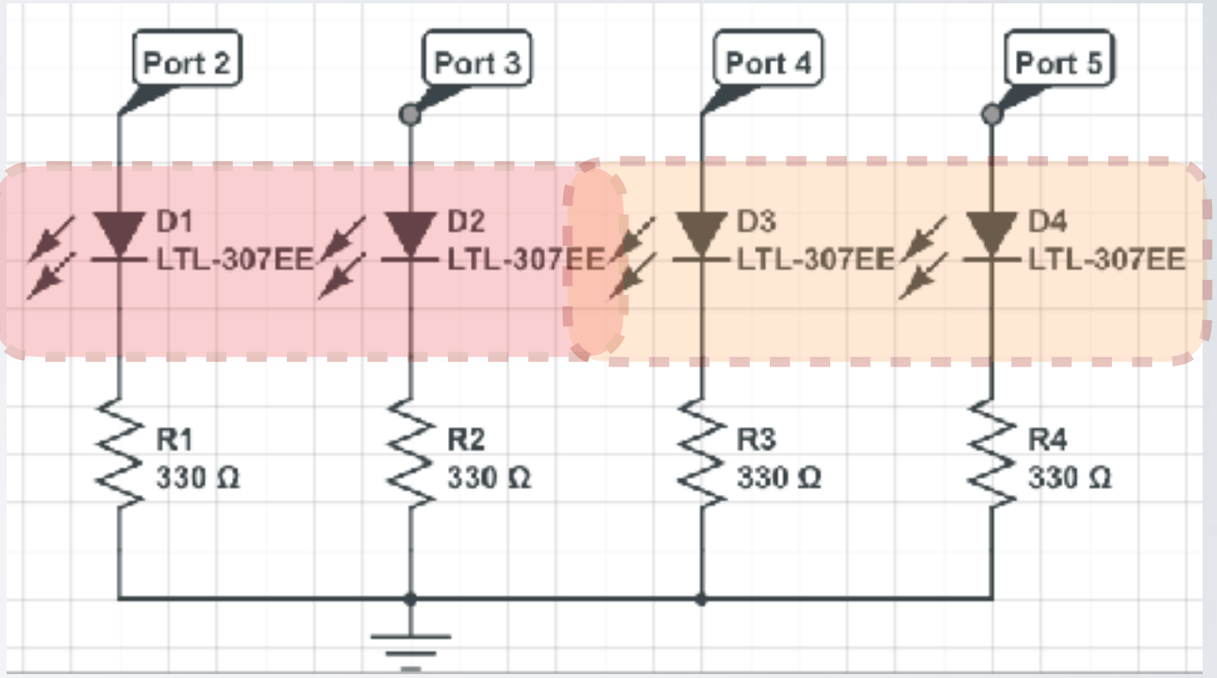
Exp_2) Piscar leds aos pares…

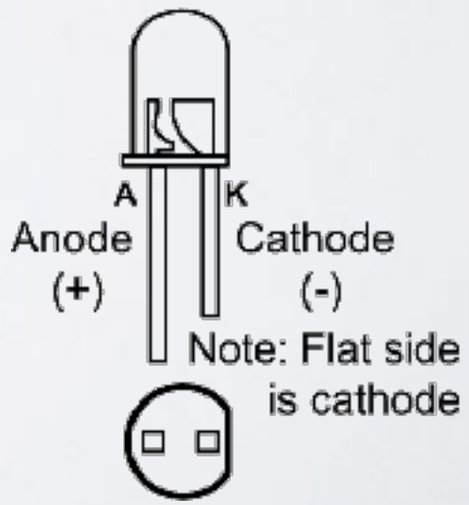
mais de uma vez antes de alternar os pares (indefinidamente)



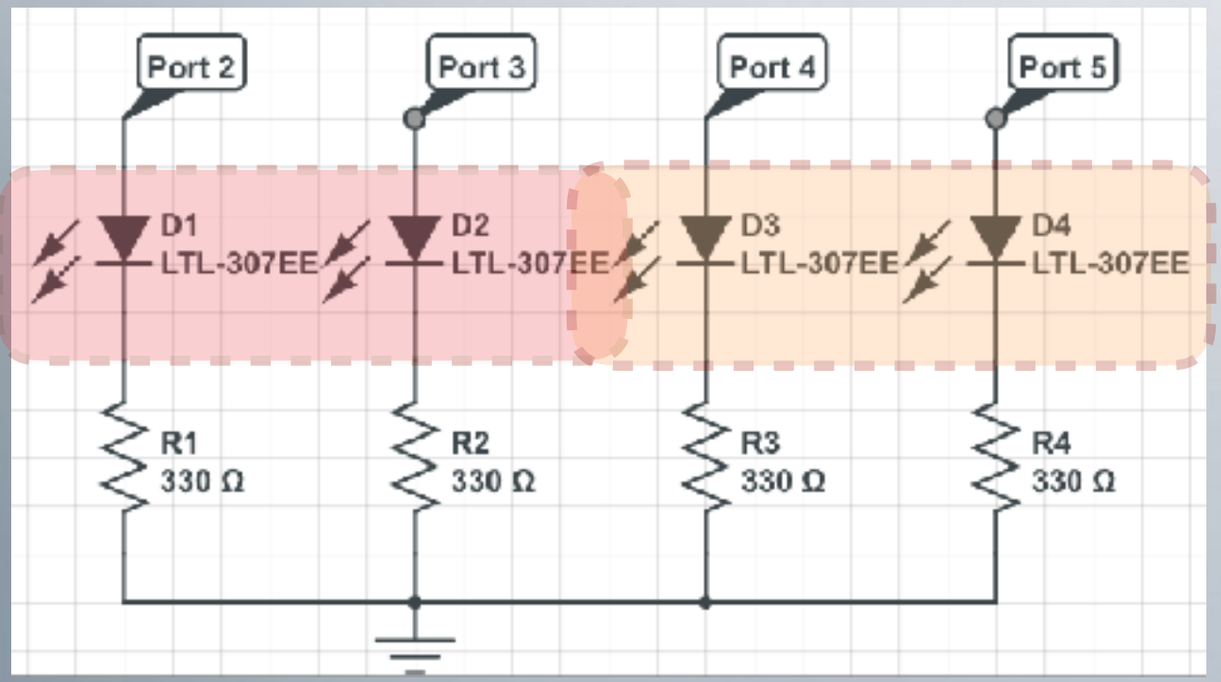
Pisquem 3x (intervalo de 250 ms)

Pisquem 3x (intervalo de 250 ms)





Resistor 330Ohms: laranja; laranja; marrom.



Exp_2) Piscar leds aos pares…

mais de uma vez antes de alternar os pares

Solução inicial (mais simples):

```
void setup(){

    pinMode(2, OUTPUT);
    digitalWrite(2, LOW);

    pinMode(3, OUTPUT);
    digitalWrite(3, LOW);

    pinMode(4, OUTPUT);
    digitalWrite(4, LOW);

    pinMode(5, OUTPUT);
    digitalWrite(5, LOW);

}
```

```
void loop(){

    //
    digitalWrite(2, HIGH);
    digitalWrite(3, HIGH);
    delay(tempo);
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    delay(tempo);
    //

    //
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
    delay(tempo);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    delay(tempo);
    //

}
```

Repetir 3 x !
(3 x 6 = 18 linhas)

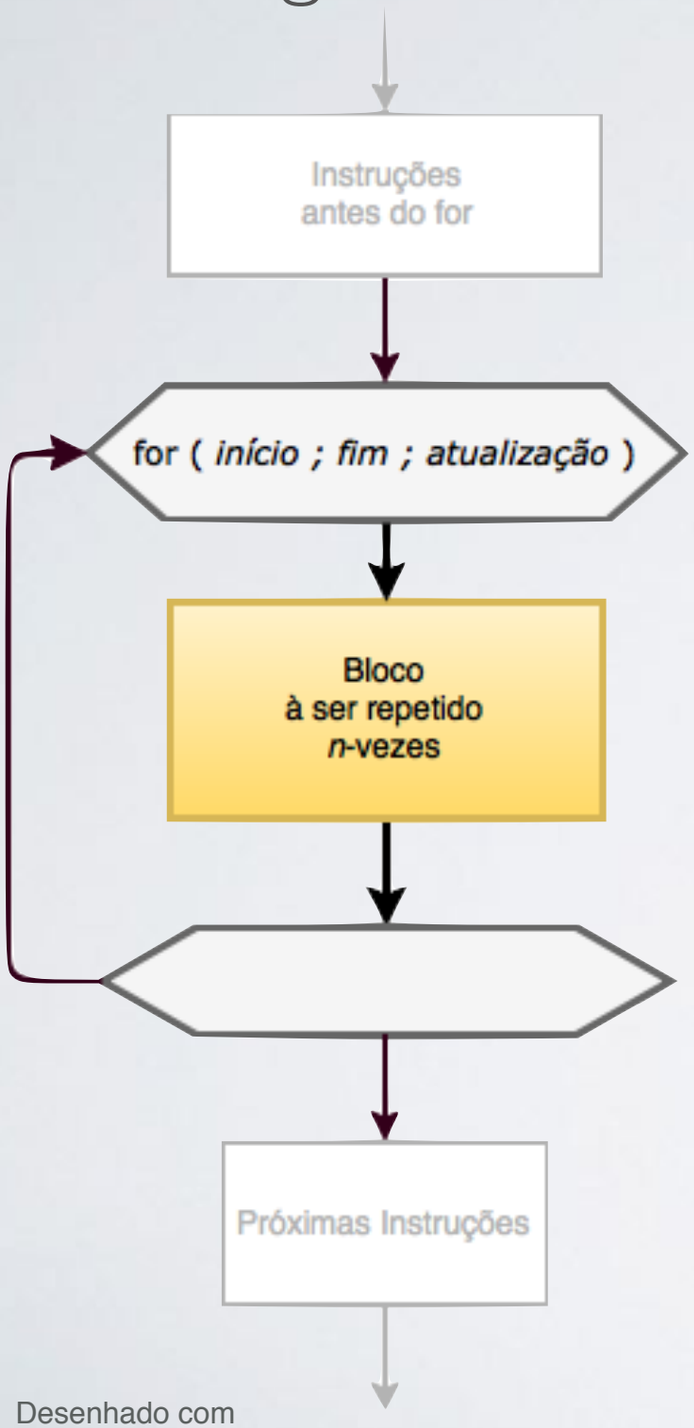
Repetir 3 x !
(3 x 6 = 18 linhas)

Total = 48 linhas de código!

- Dá para fazer mais “simples" ?

Laço de Repetição: FOR

- Fluxograma:



Desenhado com
www.draw.io

- Estrutura básica:

```
for ( inicio; teste; atualização )
    // Bloco à ser repetido
}
```

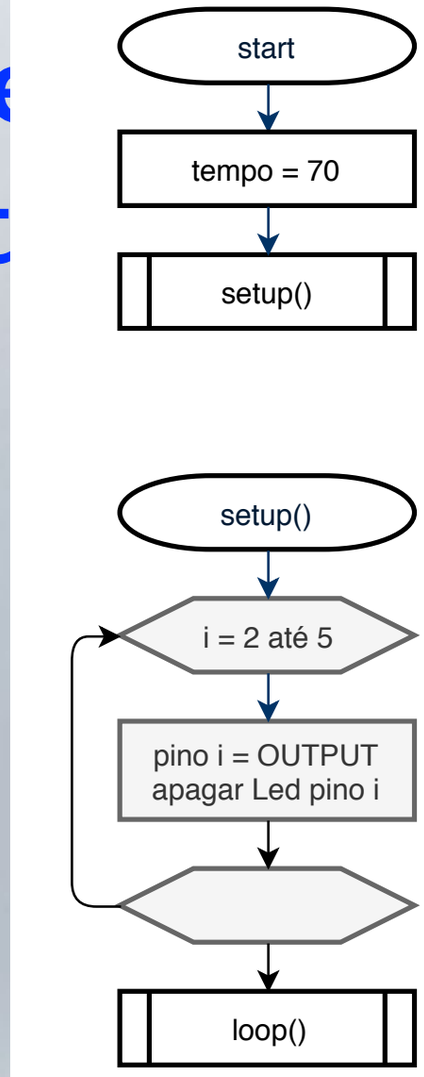
- Exemplo:

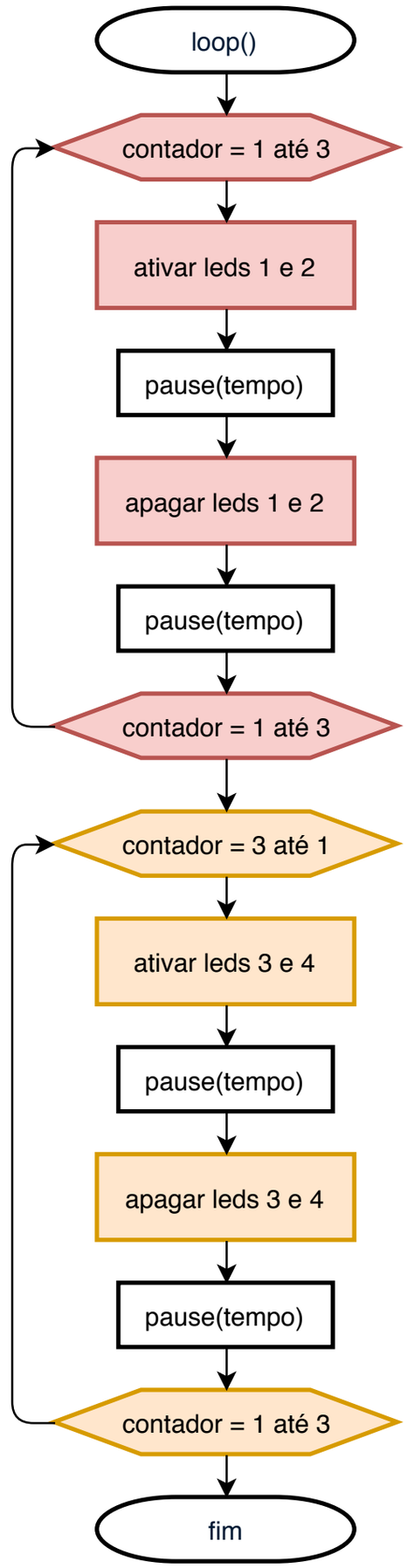
```
int i; // declarar variável
// for crescente:
for ( i=2; i<=5; i++ ) {
    digitalWrite(i, HIGH); // ativa led
    delay(100);
    digitalWrite(i, LOW); // apaga led-i
}

// for decrescente:
for ( i=5; i>=2; i-- ) {
    digitalWrite(i, HIGH); // ativa led
    delay(100);
    digitalWrite(i, LOW); // apaga led-i
}
```

Exp_2) Piscar leds aos pare
mais de uma vez antes de alt
(indefinidamente)

Solução melhorada (usando laço**s** de repetição):

```
int tempo = 70;
int i;
void setup(){
  for (i=2; i=<5; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, LOW);
  }
}

void loop(){
  int contador;
  for (contador=1; contador <= 3; contador++ ){
     digitalWrite(2, HIGH);
     digitalWrite(3, HIGH);
     delay(tempo);
     digitalWrite(2, LOW);
     digitalWrite(3, LOW);
     delay(tempo);
  }
  for (contador = 3 ; contador>=1 ; contador–){
     digitalWrite( 4, HIGH);
     digitalWrite(5, HIGH);
     delay(tempo);
     digitalWrite(4, LOW);
     digitalWrite(5, LOW);
     delay(  tempo   );
  }
}
```





Obs:

Variáveis globais x locais

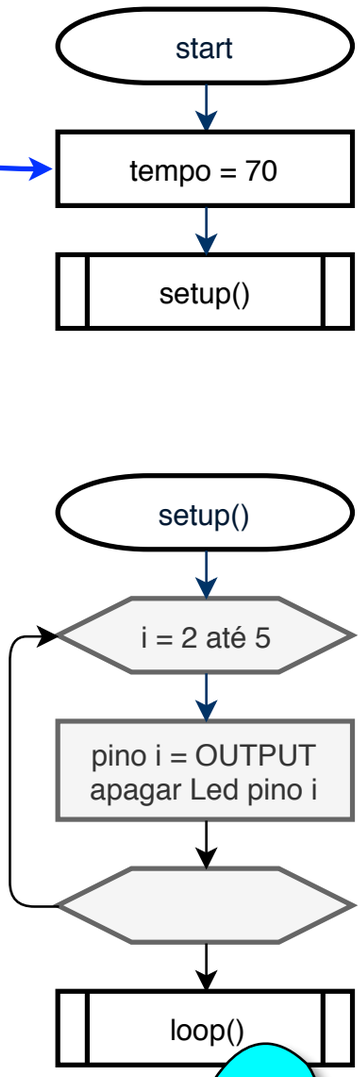
Global

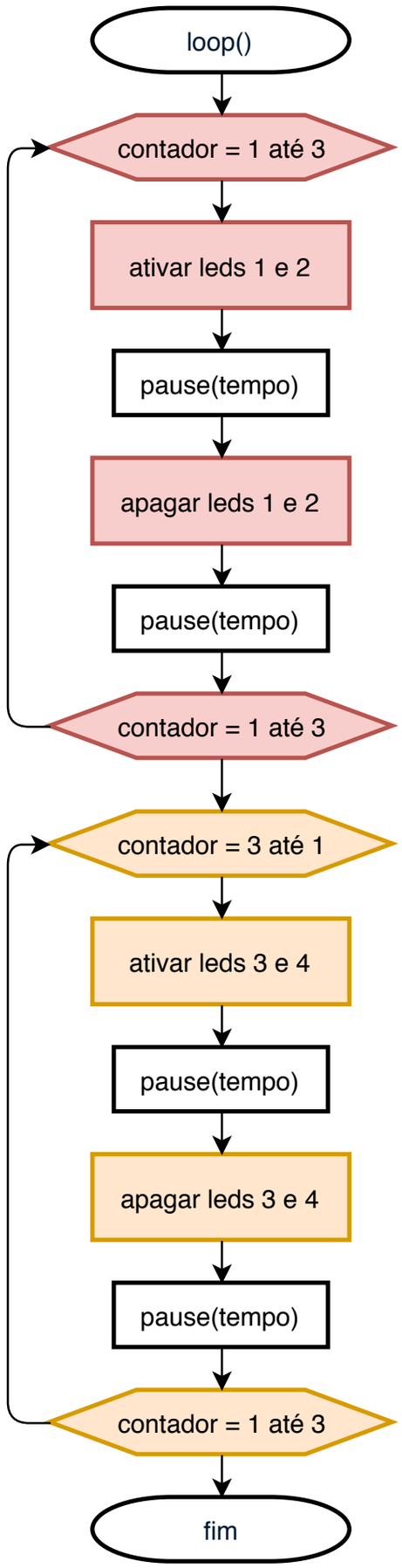
Obs.: deve ser declarada fora de qualquer bloco do código!
Desvantagem: qualquer bloco pode alterar seu valor (para o bem ou para o mau).

```
int tempo = 70;
int i;
void setup(){
  for (i=2; i=<5; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, LOW);
  }
}

void loop(){
  int contador;
  for (contador=1; contador <= 3; contador++ ){
     digitalWrite(2, HIGH);
     digitalWrite(3, HIGH);
     delay(tempo);
     digitalWrite(2, LOW);
     digitalWrite(3, LOW);
     delay(tempo);
  }
  for (contador = 3 ; contador>=1 ; contador–){
     digitalWrite( 4, HIGH);
     digitalWrite(5, HIGH);
     delay(tempo);
     digitalWrite(4, LOW);
     digitalWrite(5, LOW);
     delay(  tempo   );
  }
}
```

Globais

Globais:
Valem em qualquer parte do código!





Obs:

Variáveis globais x locais

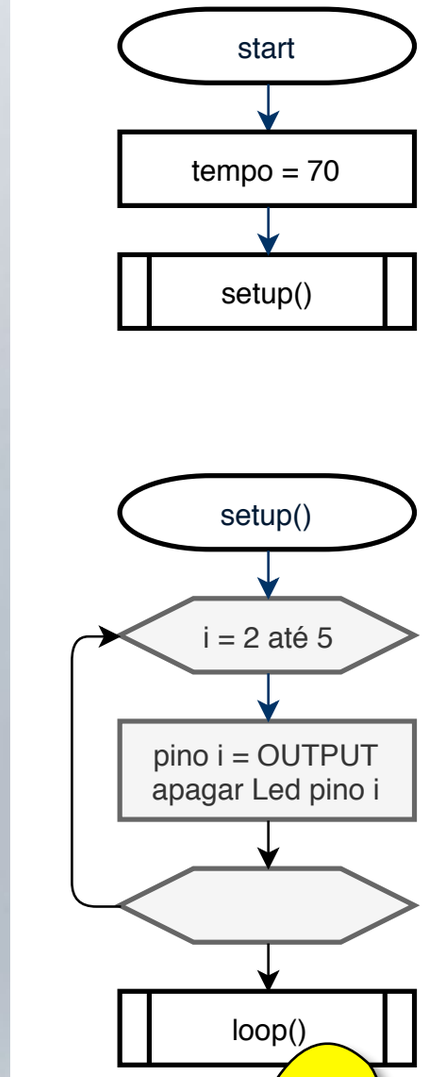
Obs.:deve ser declarada dentro de um bloco do código!
Desvantagem: só pode ser acessada (alterada) pelo bloco onde a mesma foi declarada.

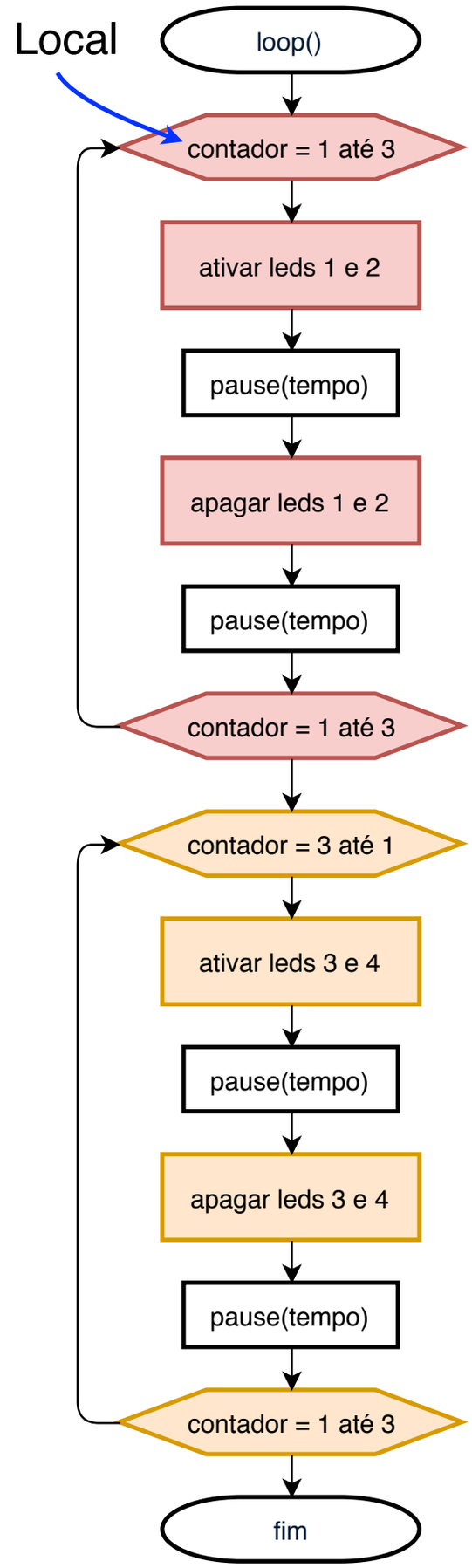
```
int tempo = 70;
int i;
void setup(){
  for (i=2; i=<5; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, LOW);
  }
}

void loop(){
  int contador;
  for (contador=1; contador <= 3; contador++ ){
     digitalWrite(2, HIGH);
     digitalWrite(3, HIGH);
     delay(tempo);
     digitalWrite(2, LOW);
     digitalWrite(3, LOW);
     delay(tempo);
  }
  for (contador = 3 ; contador>=1 ; contador–){
     digitalWrite( 4, HIGH);
     digitalWrite(5, HIGH);
     delay(tempo);
     digitalWrite(4, LOW);
     digitalWrite(5, LOW);
     delay(  tempo   );
  }
}
```

Globais

Local

Locais:
Valem somente dentro do bloco onde foram declaradas!





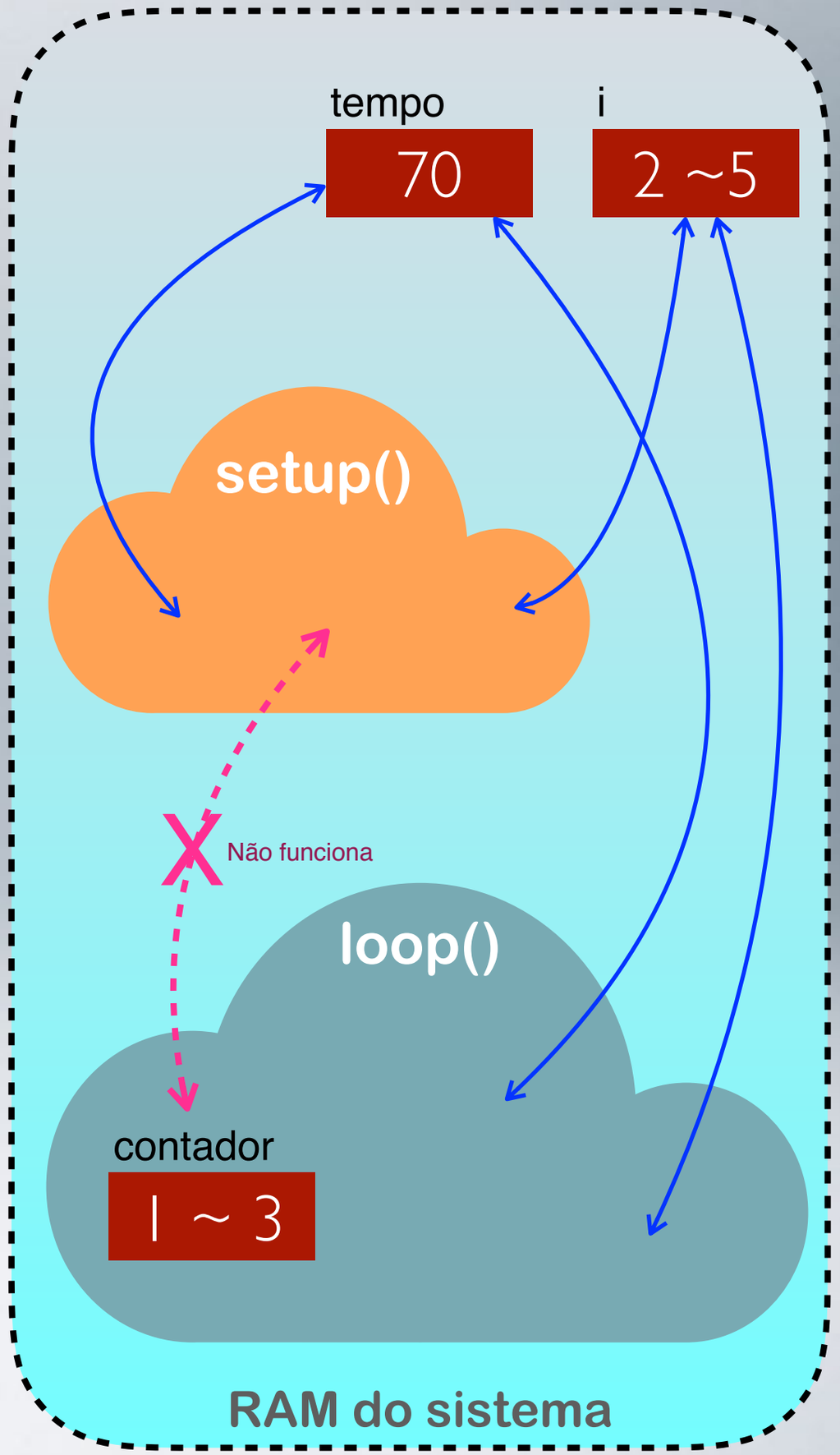
Obs:
Variáveis globais x locais

Obs.: A variável ‘contador' (local) só pode ser acessada e alterada dentro do bloco loop().

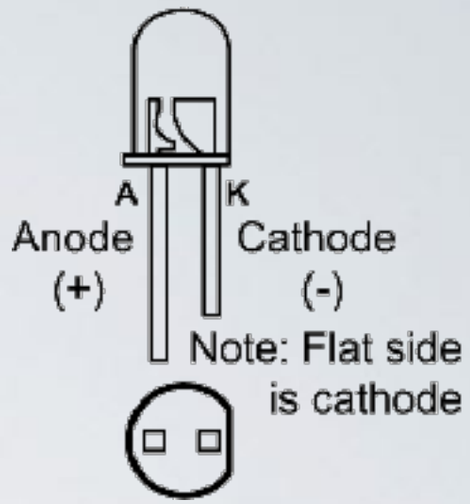
```
int tempo = 70;
int i;
void setup(){
  for (i=2; i=<5; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, LOW);
  }
}

void loop(){
  int contador;
  for (contador=1; contador <= 3; contador++ ){
     digitalWrite(2, HIGH);
     digitalWrite(3, HIGH);
     delay(tempo);
     digitalWrite(2, LOW);
     digitalWrite(3, LOW);
     delay(tempo);
  }
  for (contador = 3 ; contador>=1 ; contador–){
     digitalWrite( 4, HIGH);
     digitalWrite(5, HIGH);
     delay(tempo);
     digitalWrite(4, LOW);
     digitalWrite(5, LOW);
     delay(  tempo   );
  }
}
```

Globais

Local

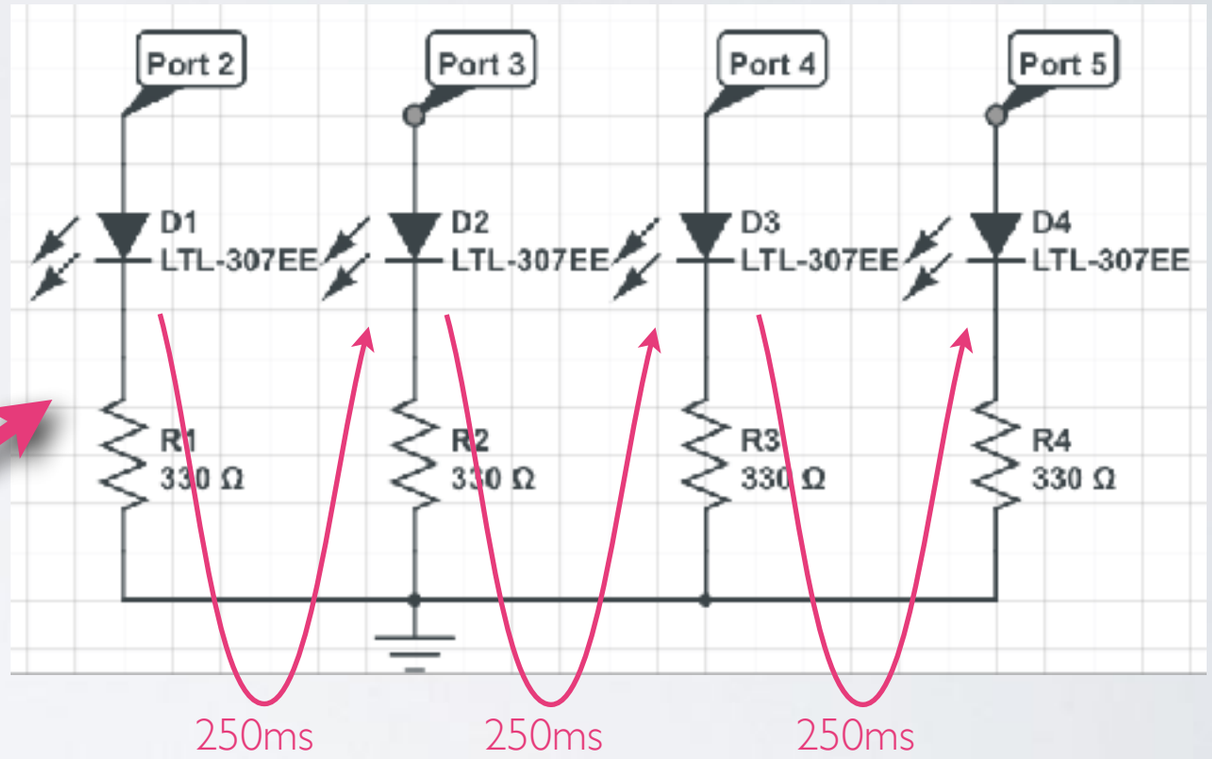


Exp_3) Acionar sequencialmente 4 Leds!

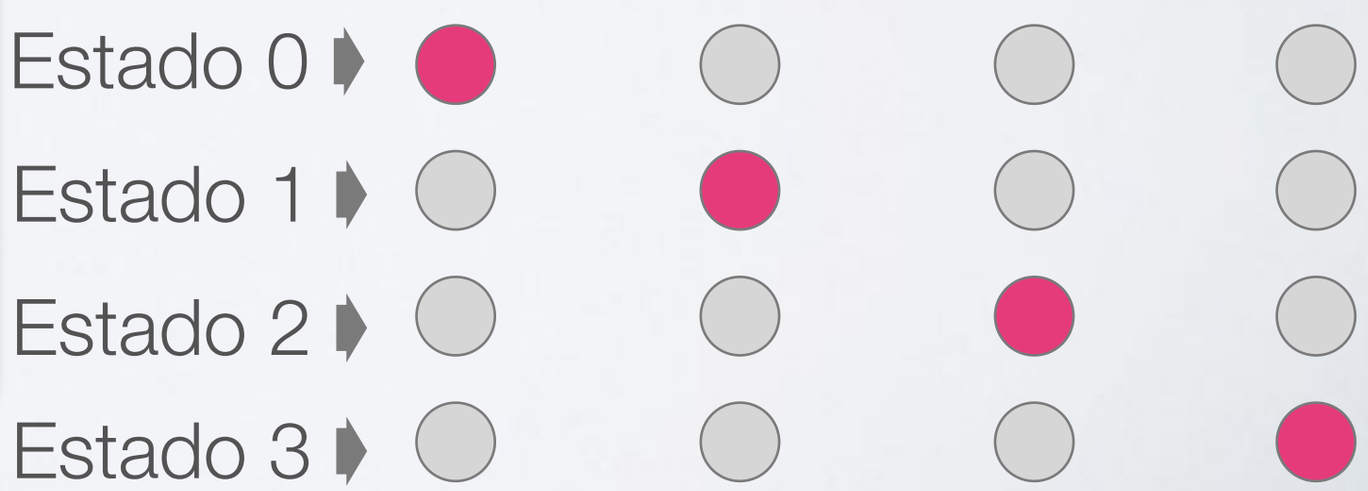


- Circuito
  Obs.: Leds ligados às portas: 5, 4, 3 e 2.
- Problemas:

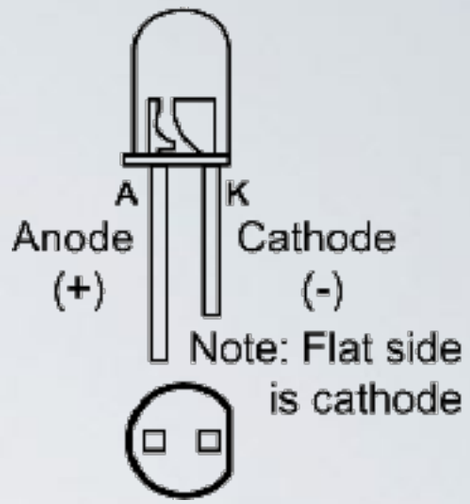
1. Piscar na sequência: 1 led depois do outro, somente 1 led acesso de cada vez.




```
sketch_sep12a §
1  void setup() {
2    // put your setup code here, to run once:
3    pinMode(2, OUTPUT);
4    pinMode(3, OUTPUT);
5    pinMode(4, OUTPUT);
6    pinMode(5, OUTPUT);
7  }
8
9  void loop() {
10   // put your main code here, to run repeatedly:
11
12 }
```



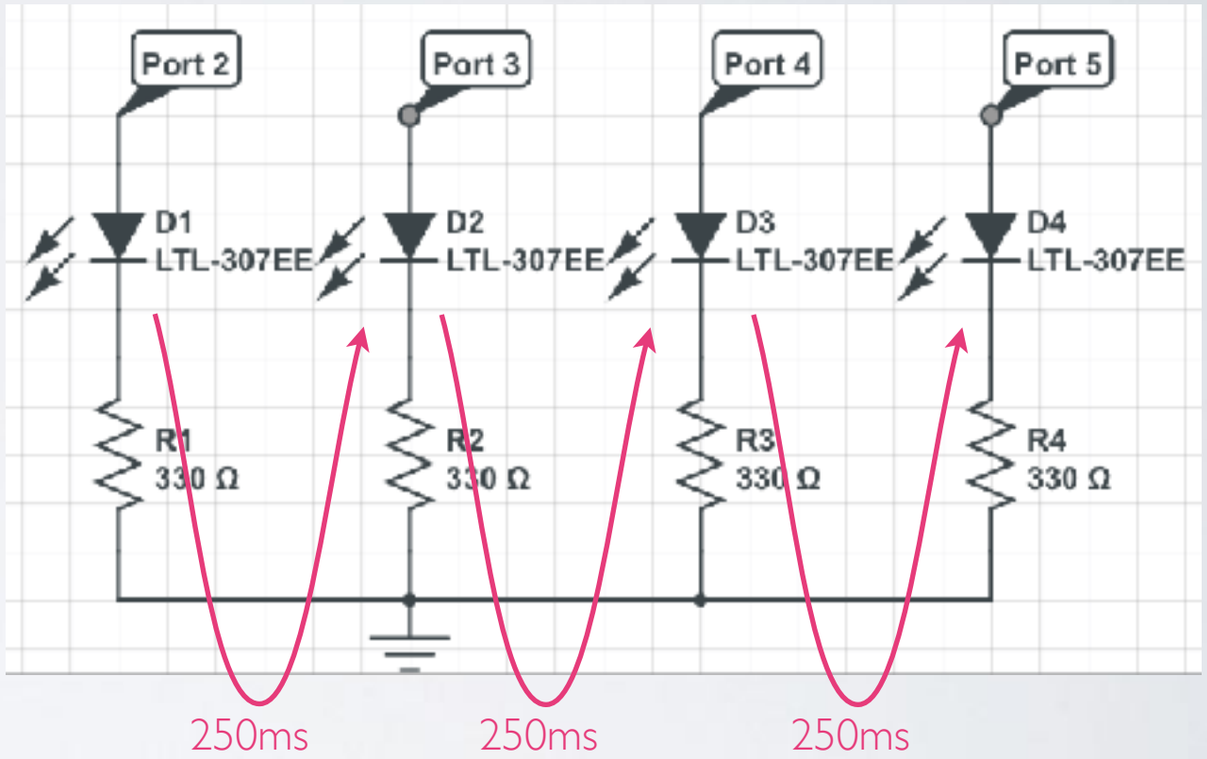
Exp_3) Acionar sequencialmente 4 Leds!



- Circuito

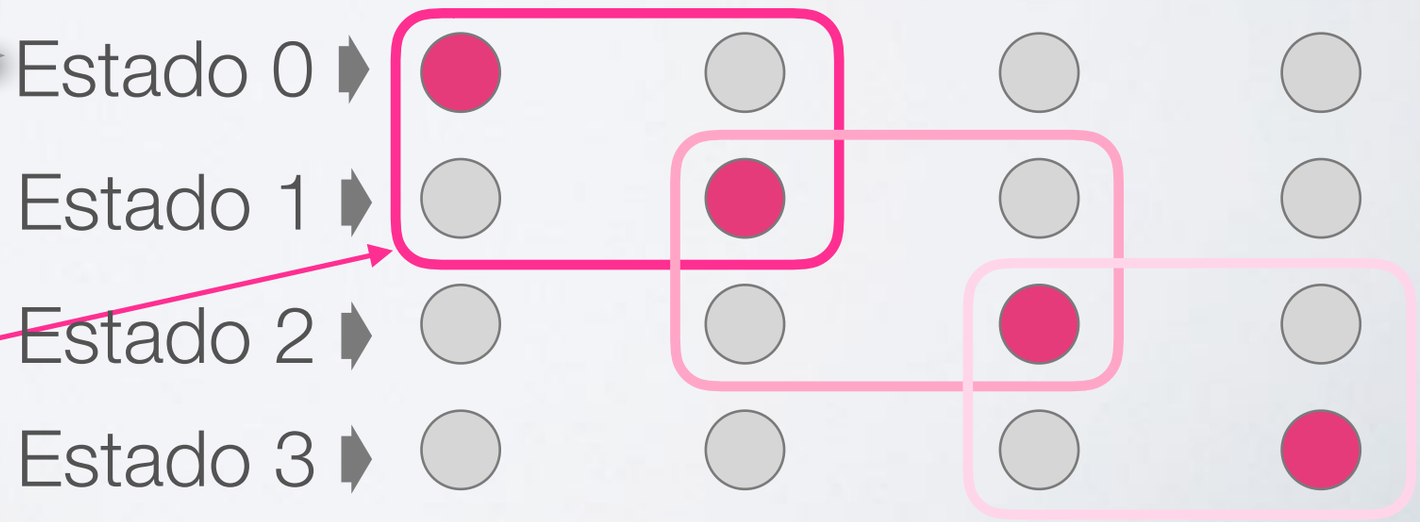
  Obs.: Leds ligados às portas: 5, 4, 3 e 2.
- Problemas:

1. Piscar na sequência: 1 led depois do outro, somente 1 led acesso de cada vez.



Significa que cada estado, algo como:

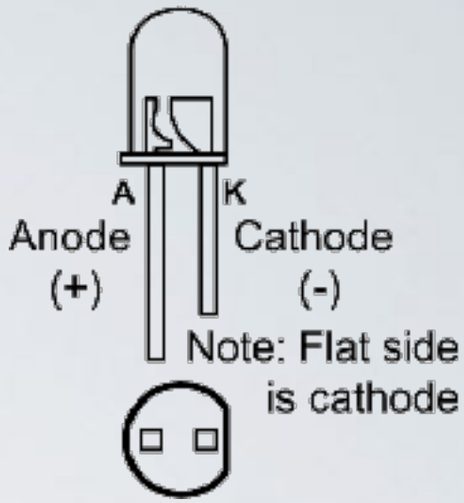
```
:
void loop(){
   // Estado 0:
   digitalWrite(2, HIGH);
   digitalWrite(3, LOW);
   digitalWrite(4, LOW);
   digitalWrite(5, LOW);
   :
```



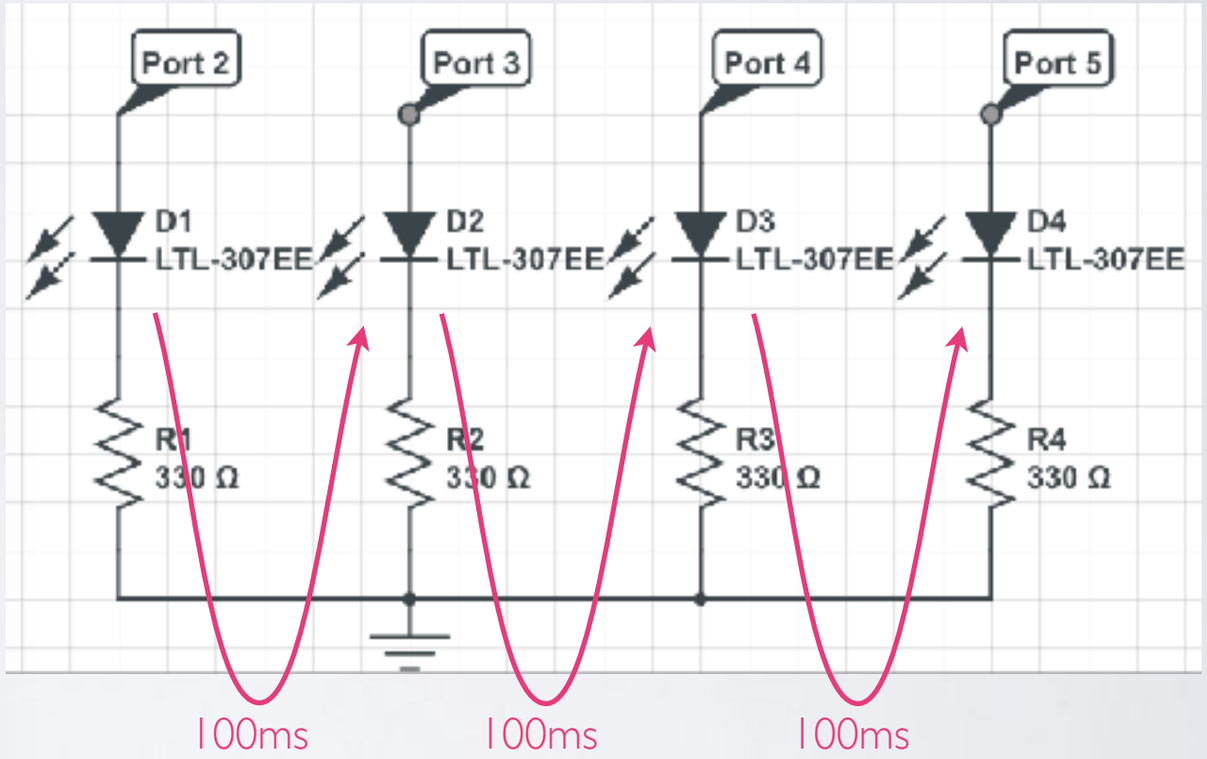
Note porém que só 2 leds mudam de nível lógico entre um estado e outro!

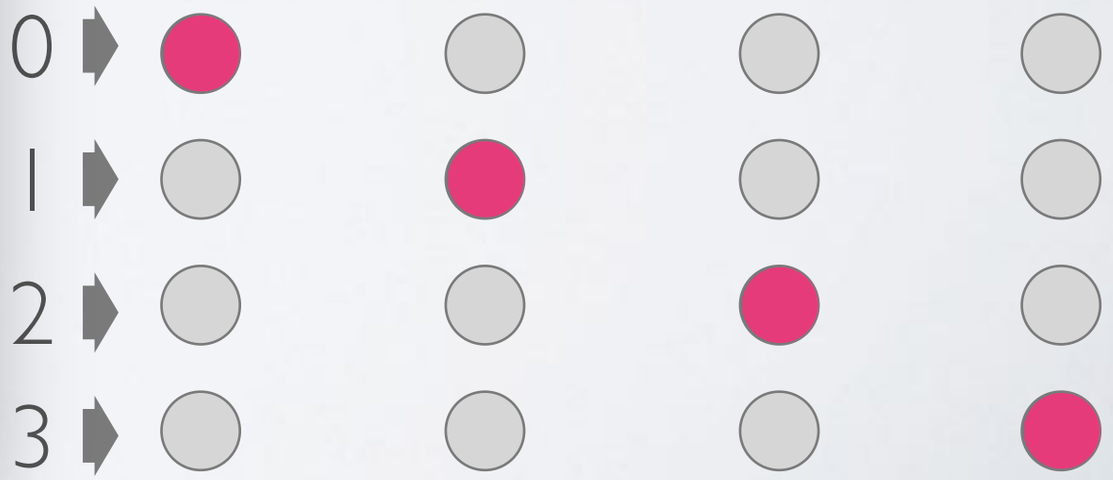
Exp_3) Acionar sequencialmente 4 Leds!

Outra forma + prática ➠ uso de "**for**",
(laço de repetição)



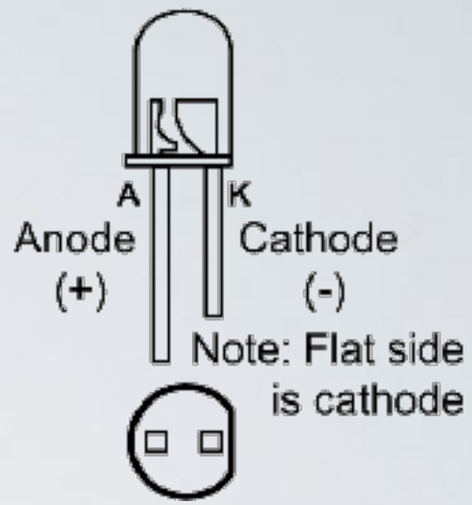

```
sketch_sep12a | Arduino 1.6.8

sketch_sep12a §

void setup() {
  // put your setup code here, to run once:
  int i; // declara a variável i como inteiro (sem casas decimais)
  for ( i=2; i <=5; i++ ) {
    pinMode( i, OUTPUT);
  }
}


void loop() {
  // put your main code here, to run repeatedly:

}

7                    Arduino/Genuino Uno em /dev/cu.usbmodemFA141
```





Exp_3) Acionar sequencialmente 4 Leds!

Outra forma + prática ➠ uso de "**for**",
(laço de repetição)

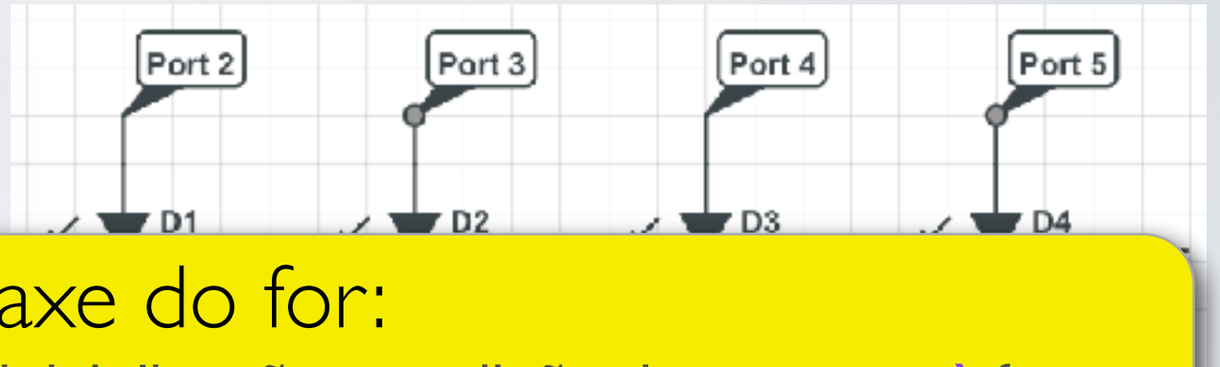


sketch_sep12a | Arduino 1.6.8

sketch_sep12a §

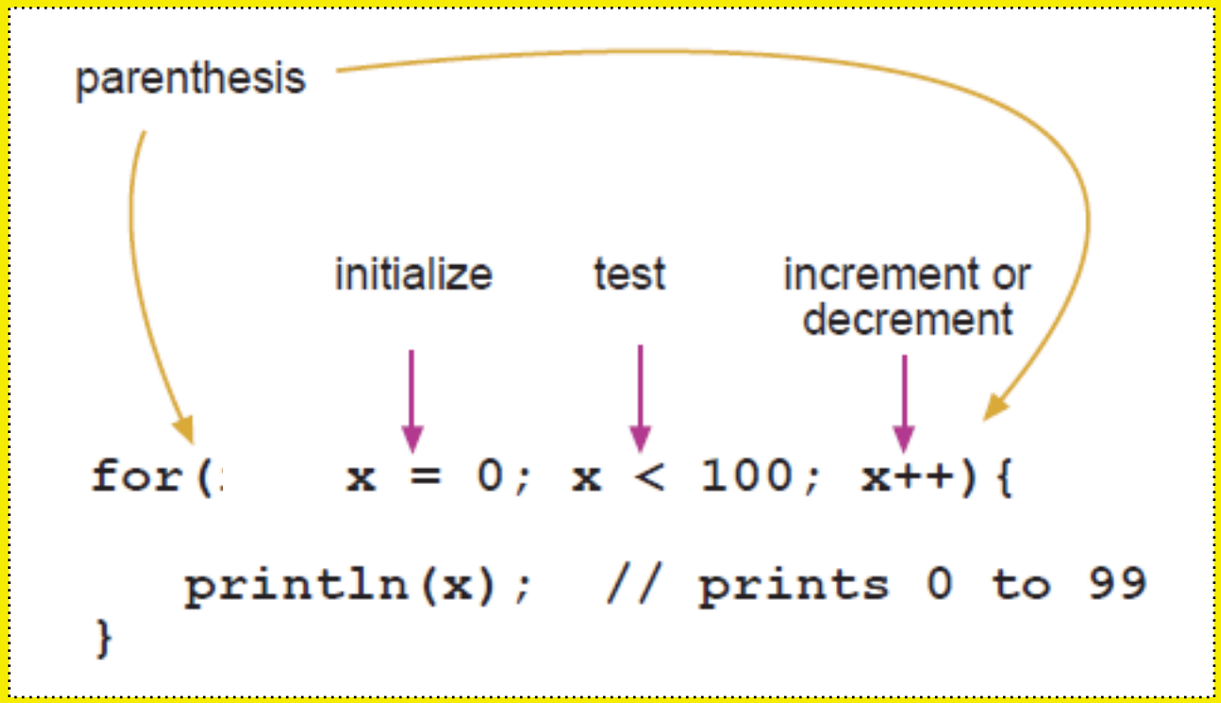
```
void setup() {
  // put your setup code here, to run once:
  int i; // declara a variável i como inteiro (sem casa
decimais)
  for ( i=2; i <=5; i++ ) {
    pinMode( i, OUTPUT);
  }
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

7 Arduino/Genuino Uno em /dev/cu.usbmodemFA1



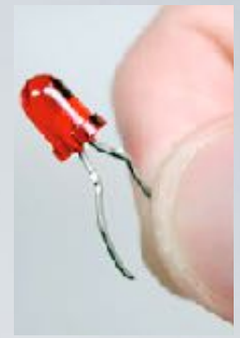
Sintaxe do for:

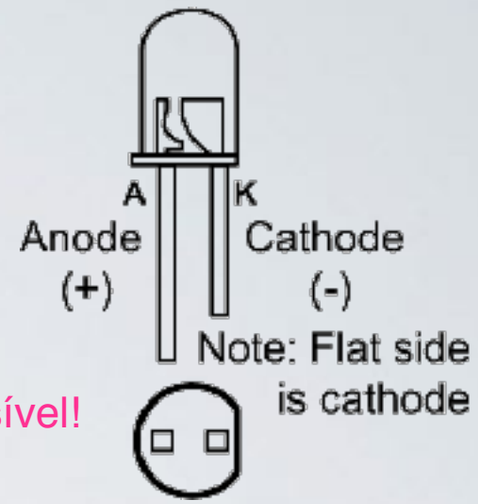
```
for ( inicialização; condição; incremento ) {
   // declaração(ões);
}
```



Exp_3) Acionar sequencialmente 4 Leds!

Outra forma + prática ➠ uso de "**for**",
(laço de repetição)





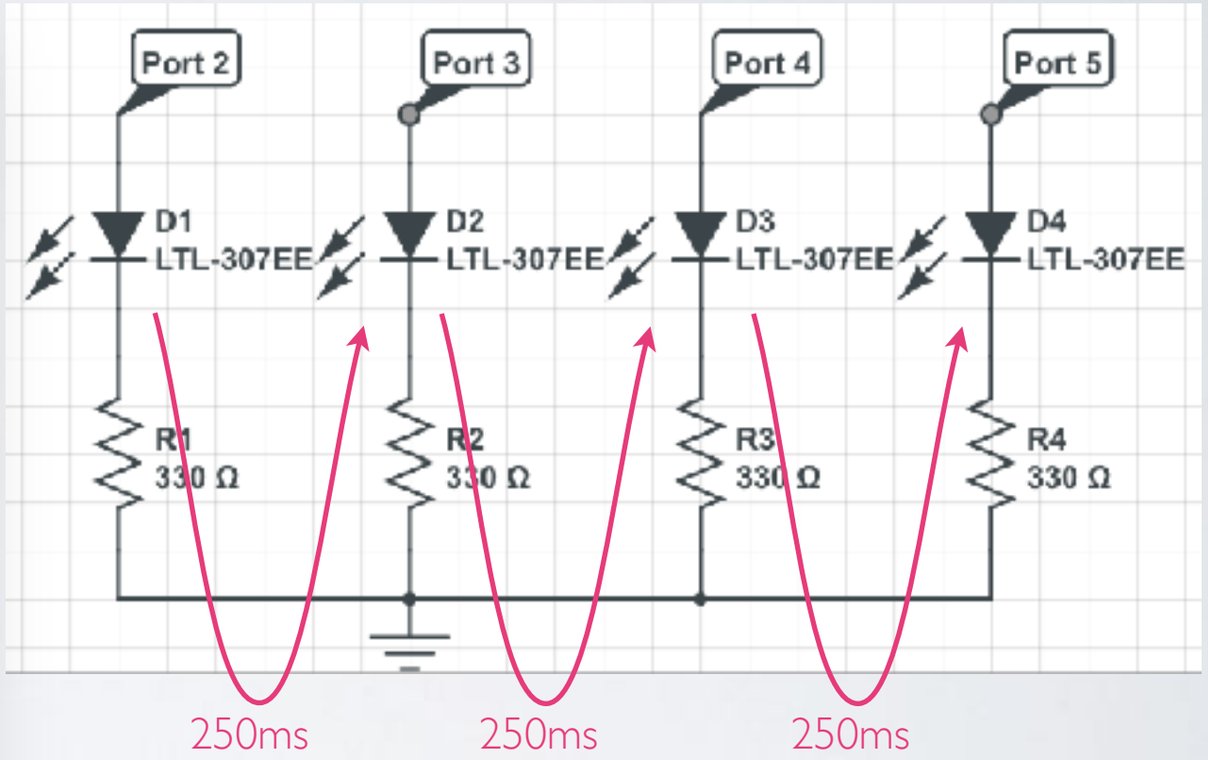
Uma solução possível!

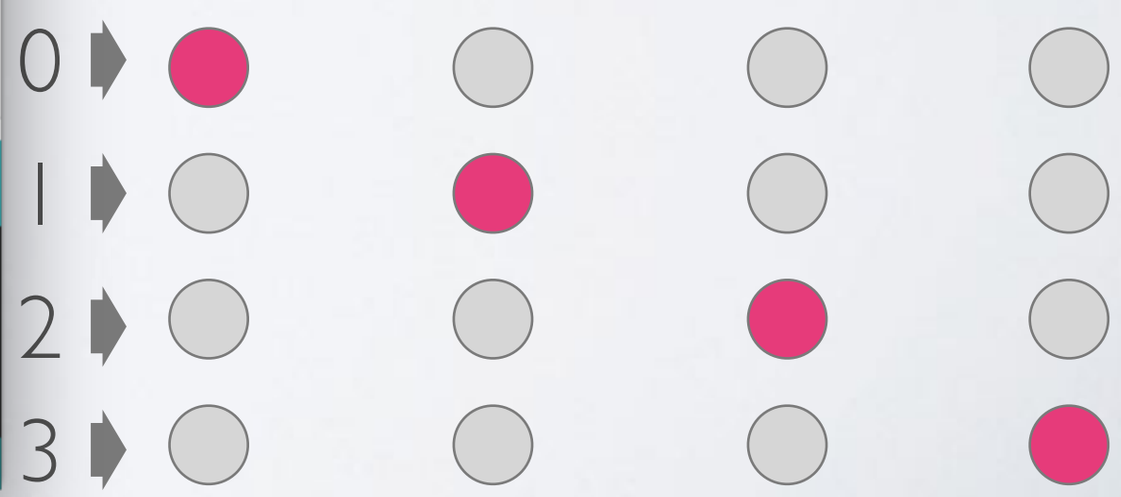
exemplo_for_4_leds | Arduino 1.6.8

exemplo_for_4_leds

```
void setup() {
  // put your setup code here, to run once:
  int i; // declara a variável i como inteiro (sem casas
decimais)
  for ( i=2; i <=5; i++ ) {
    pinMode( i, OUTPUT);
  }
}

void loop() {
  // put your main code here, to run repeatedly
  int i;
  for (i=2; i<=5; i++){
     digitalWrite(i, HIGH); // ativa led
     delay(100);
     digitalWrite(i, LOW); // apaga led-i
  }
}
```

13 Arduino/Genuino Uno em /dev/cu.usbmodemFA141





Exp_3) Acionar sequencialmente 4 Leds!

Outra forma + prática ➠ uso de "**for**",
(laço de repetição)

exemplo_for_4_leds | Arduino 1.6.8

exemplo_for_4_leds

```
void setup() {
  // put your setup code here, to run once:
  int i; // declara a variável i como inteiro (sem casas decimais)
  for ( i=2; i <=5; i++ ) {
    pinMode( i, OUTPUT);
  }
}

void loop() {
  // put your main code here, to run repeatedly
  int i;
  for (i=2; i<=5; i++){
     digitalWrite(i, HIGH); // ativa led
     delay(100);
     digitalWrite(i, LOW); // apaga led-i
  }
}
```

13 Arduino/Genuino Uno em /dev/cu.usbmodemFA141

Com for

0 1 2 3

sketch_sep12a | Arduino 1.6.8

sketch_sep12a §

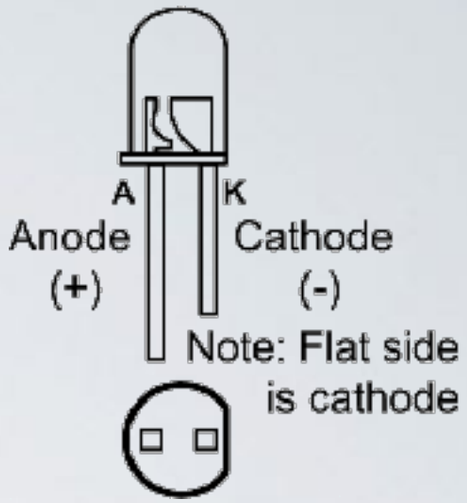
```
1  void setup() {
2    // put your setup code here, to run once:
3    pinMode(2, OUTPUT); digitalWrite(2, LOW);
4    pinMode(2, OUTPUT); digitalWrite(3, LOW);
5    pinMode(4, OUTPUT); digitalWrite(4, LOW);
6    pinMode(5, OUTPUT); digitalWrite(5, LOW);
7  }
8
9  void loop() {
10   // put your main code here, to run repeatedly:
11   // Estado 0:
12   digitalWrite(2, HIGH);
13   delay(250);
14   digitalWrite(2, LOW);
15   // Estado 1:
16   digitalWrite(3, HIGH);
17   delay(250);
18   digitalWrite(3, LOW);
19   // Estado 2:
20   digitalWrite(4, HIGH);
21   delay(250);
22   digitalWrite(4, LOW);
23   // Estado 3:
24   digitalWrite(4, HIGH);
25   delay(250);
26   digitalWrite(4, LOW);
27 }
```

Code formatted for the Arduino forum has been copied to the clipbo

24 Arduino/Genuino Uno em /dev/cu.usbmodemFA141

SEM for

Exp_4) Efeito de “Vai-e-Volta”

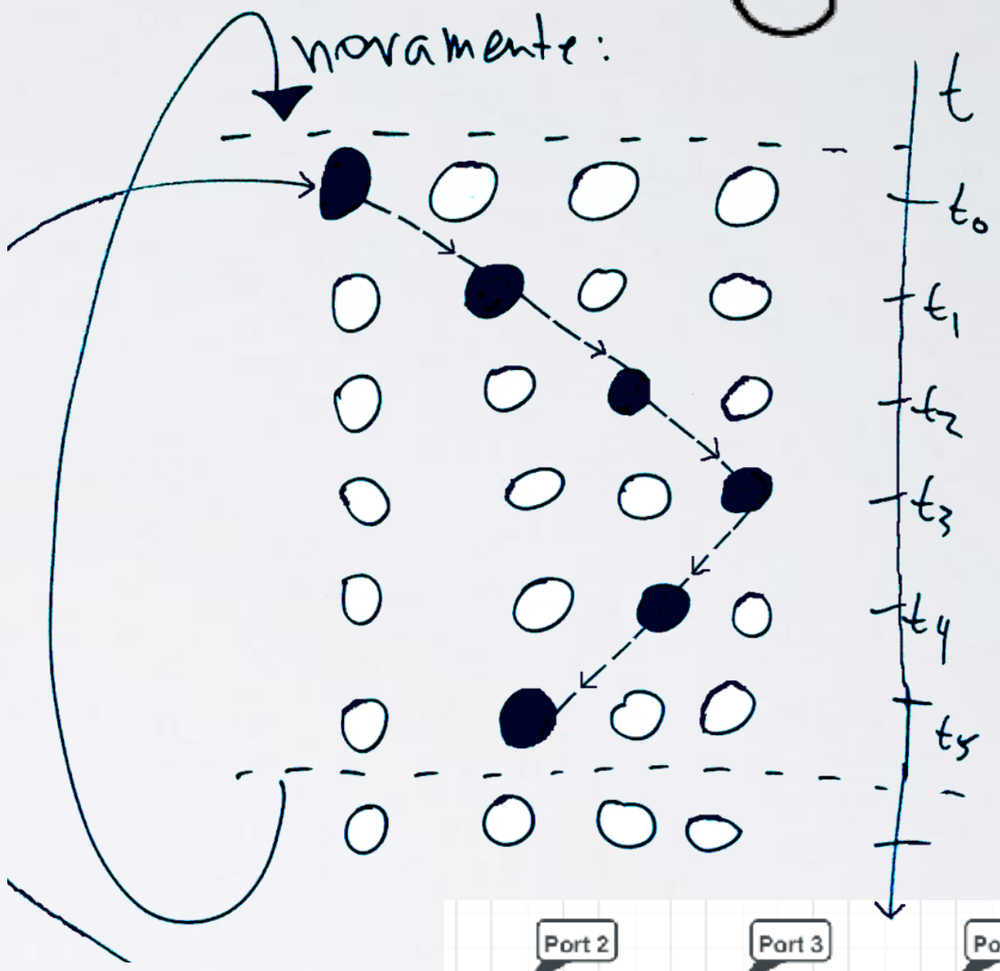


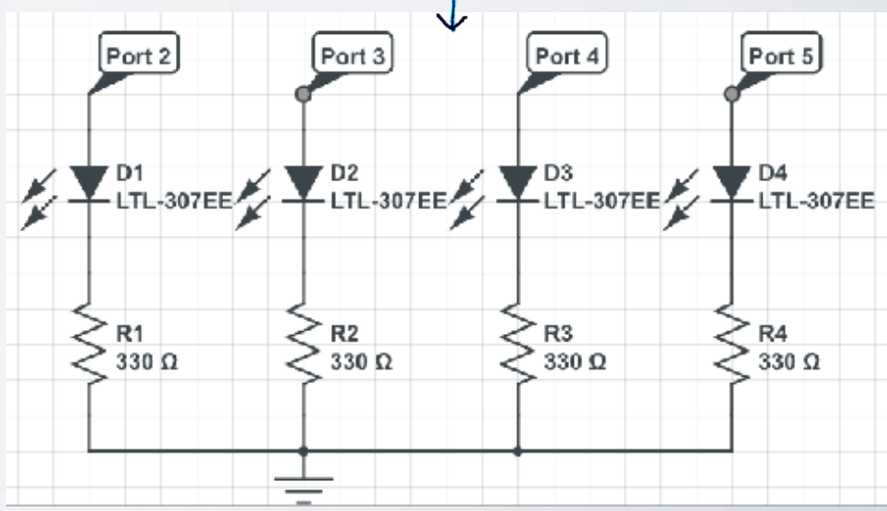
- Circuito

  Obs.: Leds ligados às portas: 5, 4, 3 e 2.

- Problemas:

3. Criar efeito visual de vai-e-vêm: Obs.: ativar 1 led após o outro, mas somente 1 led acesso de cada vez, realizando “movimento de vai-e-vêm”. Note: mais estados são necessários.

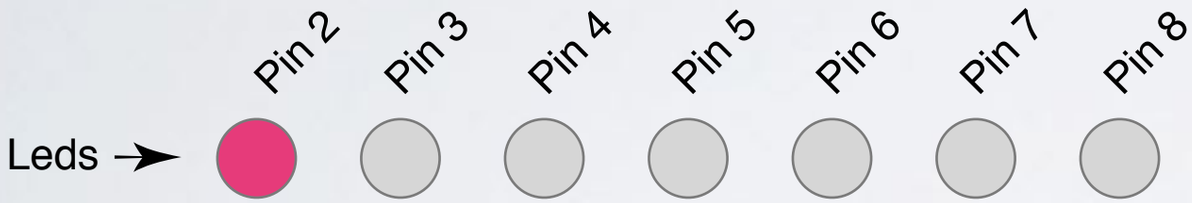




Exemplo de outro efeito visual.

Código

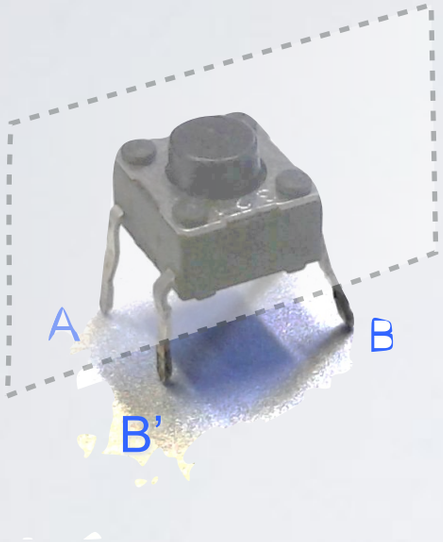
- Qual é o Resultado obtido aqui?

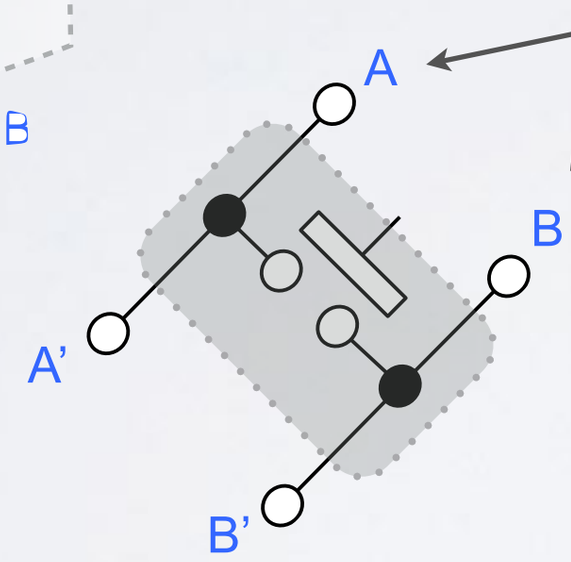



```
void setup() {
  // put your setup code here, to run once:
  int i; // declara a variável i como inteiro (sem casas decimais)
  for (i=2; i <=8; i++){
    pinMode(i, OUTPUT);
    digitalWrite(i, HIGH); // Ativa Led pino i
    delay(50);
  }
  delay(50);
  for (i=2; i<=8; i++){
    digitalWrite(i, LOW); // Apaga Led pino i
    delay(100);
  }
  delay(250);
  digitalWrite(2, HIGH); // Liga Led do pino 2
}

void loop() {
  // put your main code here, to run repeatedly:
  int i;
  int tempo=50;

  // Movimento de "ida"
  // digitalWrite(2, HIGH); // Liga Led do pino 2
  delay(tempo);
  for (i=2; i<=7; i++){
    digitalWrite(i, LOW); // Apaga Led atual (pino i)
    digitalWrite(i+1, HIGH); // Ativa próximo Led (pino i+1)
    delay(tempo);
  } // Notar que 2 <= i <= 4
  // Laço de repetição anterior termina com Led 5 ativado!

  // Movimento de "volta"
  for (i=8; i>=3; i--){
    digitalWrite(i, LOW); // Apaga Led atual (pino i)
    digitalWrite(i-1, HIGH); // Ativa Led anterior (pino i-1)
    delay(tempo);
  }
  // Notar que termina com led 2 ativado!
}
```

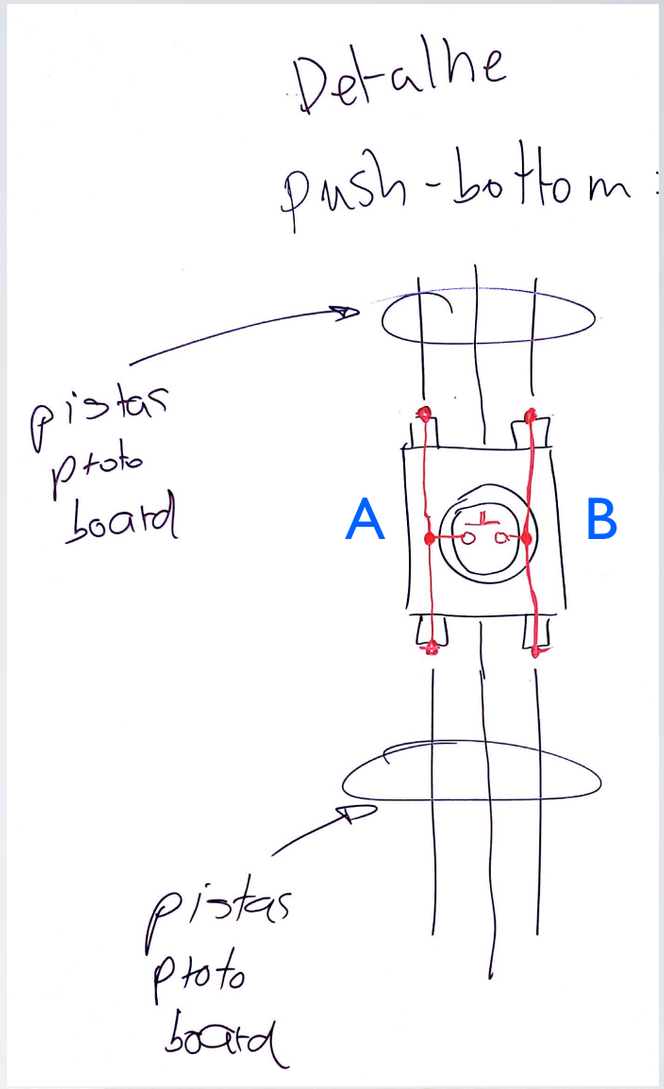
Exp: Acrescentando Chave

- Detalhe (físico) da chave:

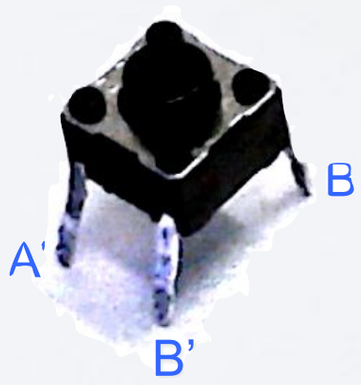


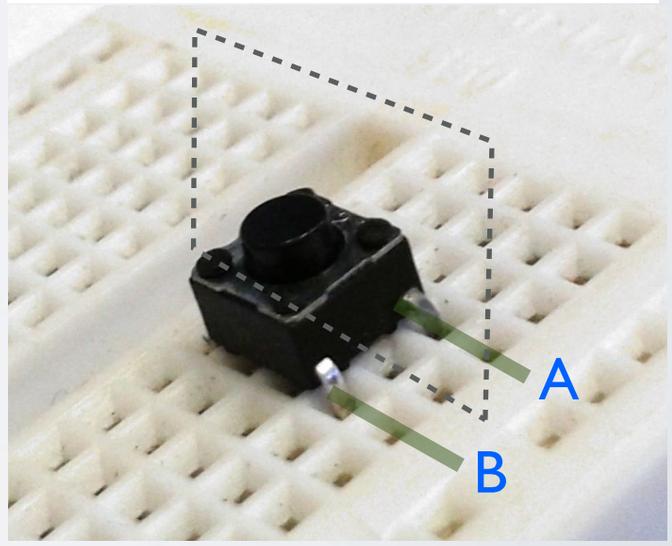


A ligação elétrica que se realiza quando se pressiona a chave ocorre entre os lados A e B!



Obs.:
Repare que já existe uma ligação elétrica interna entre os pontos A e A' e entre B e B'





Exp: Acrescentando Chave

- Ligação (elétrica) da chave:

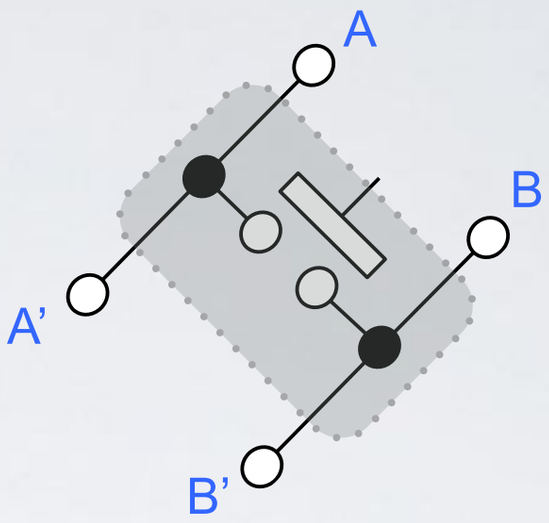
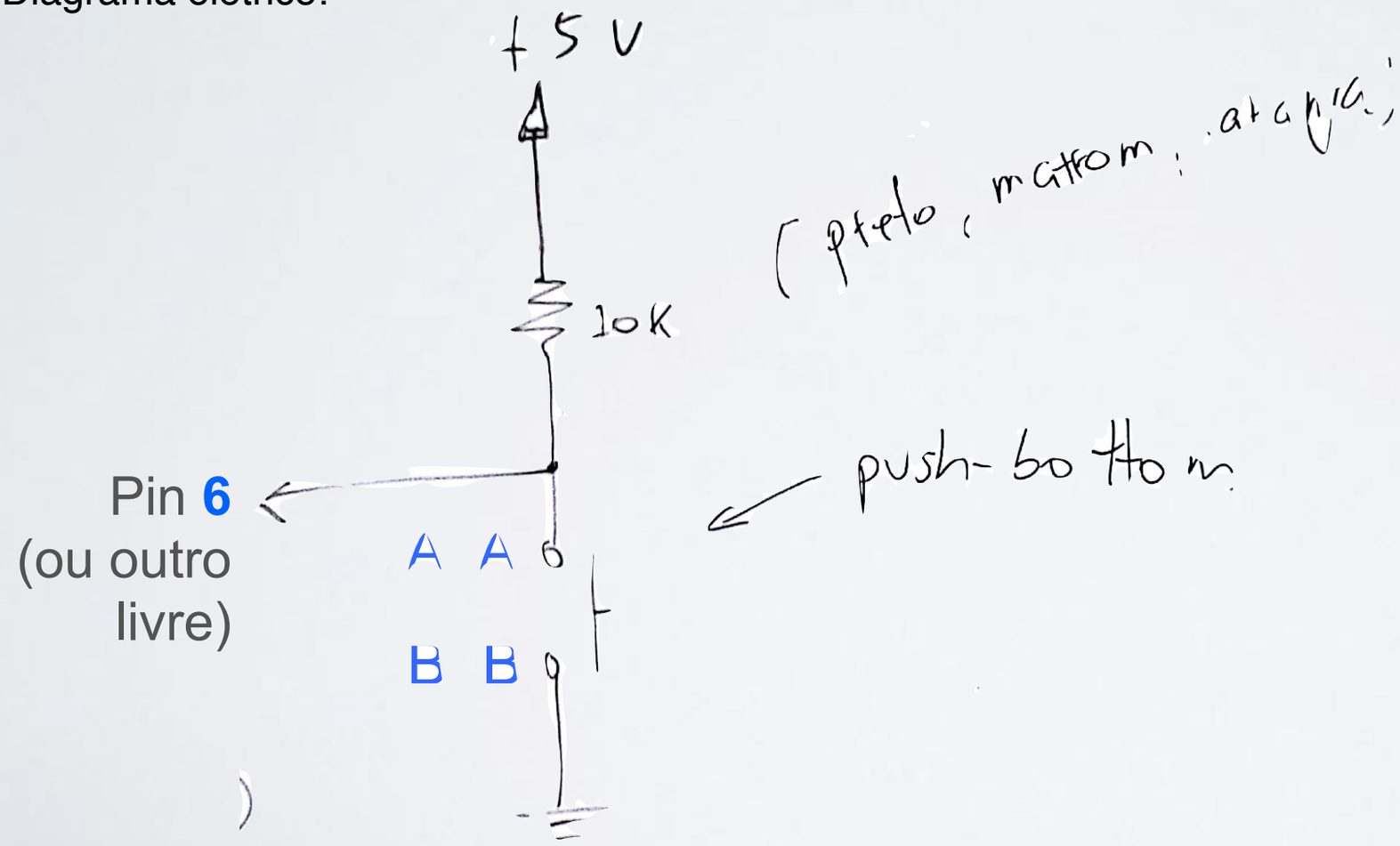
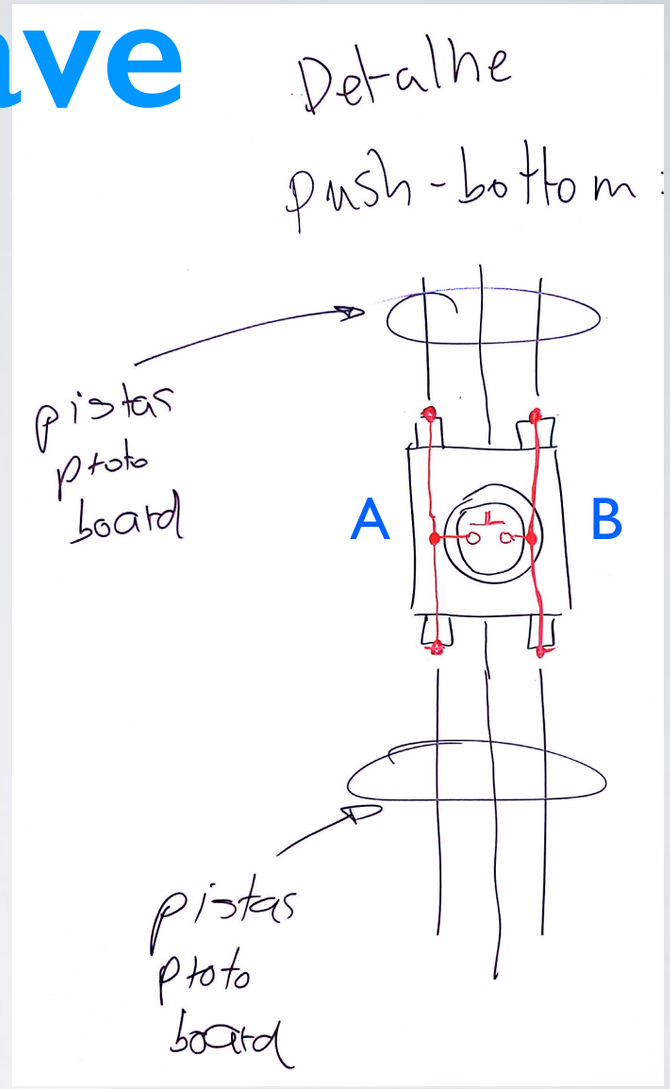
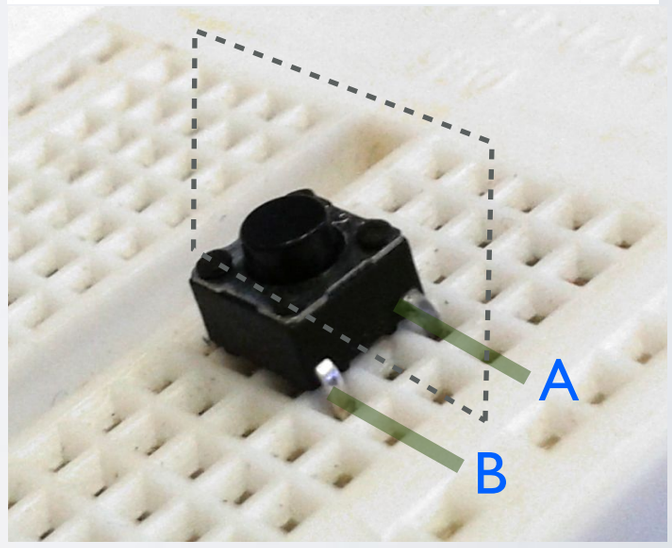


Diagrama elétrico:

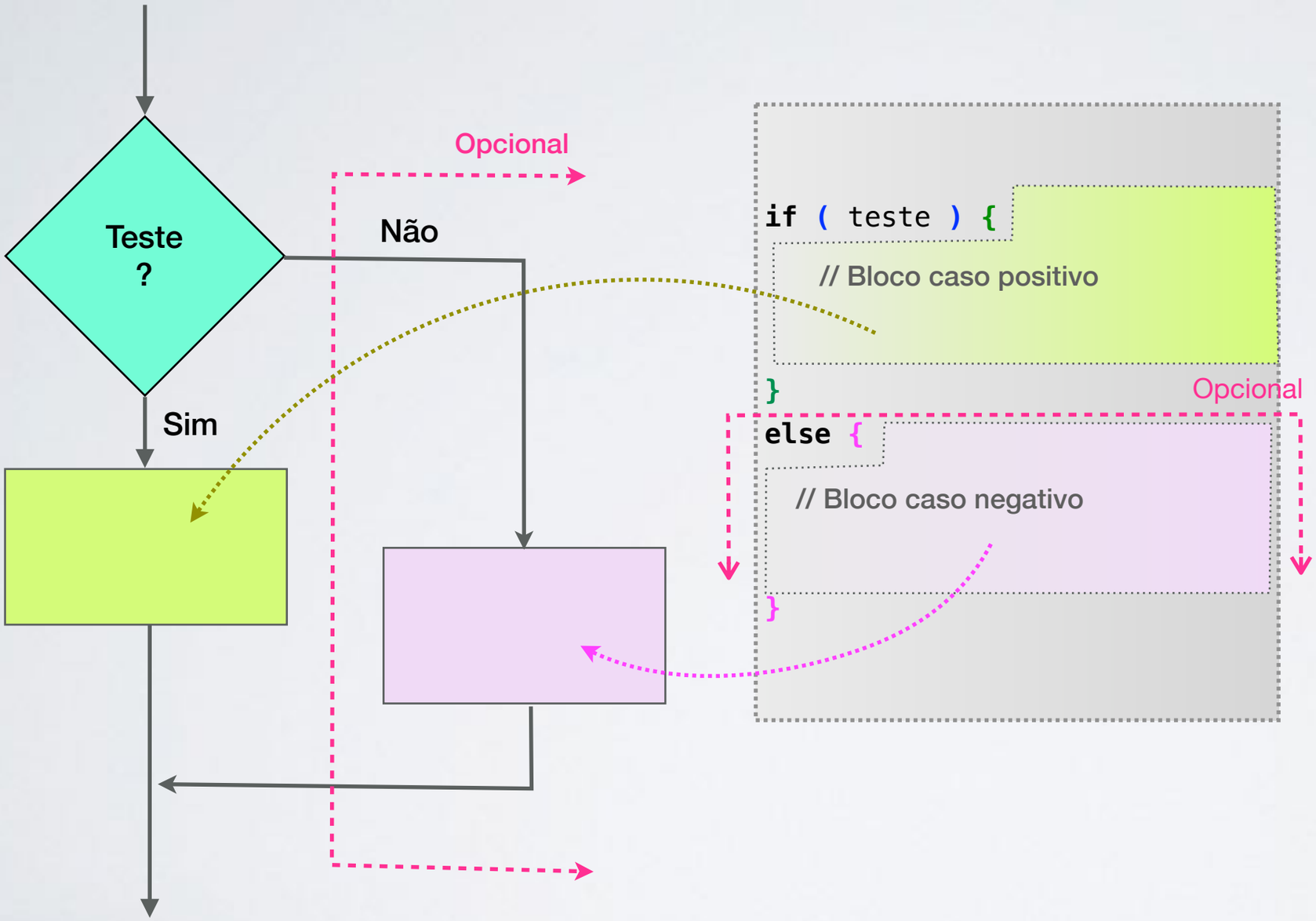






Estrutura Para “Desvios Condicionais: IF’s

- Fluxograma:
- Estrutura básica:
- Exemplo:




```
if ( teste ) {
    // Bloco caso positivo
}
Opcional
else {
    // Bloco caso negativo
}
```

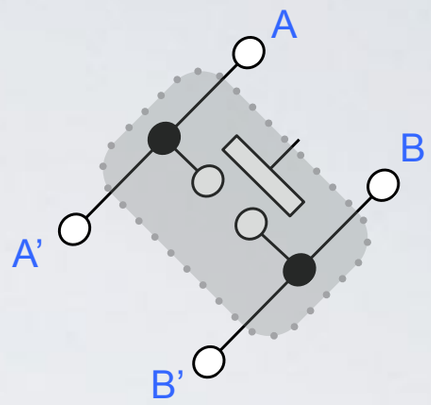
```
// lê estado chave:
buttonState = digitalRead(buttonPin);
if (buttonState == LOW) {
    // chave pressionada, faça algo


}
else {
    // chave não pressionada


}
```

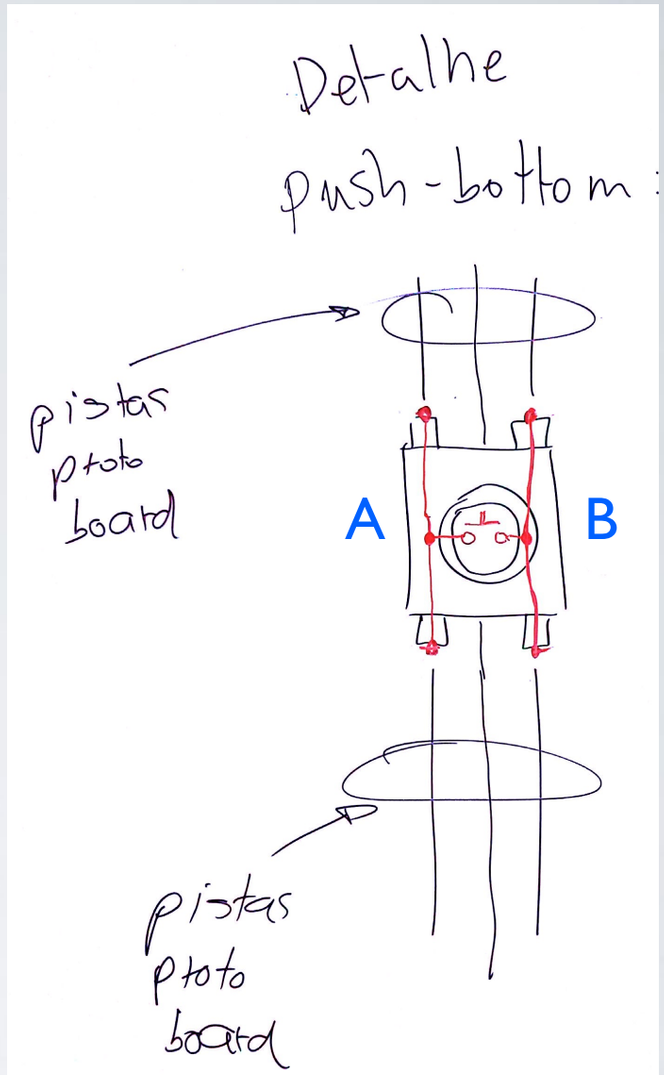
Operadores Relacionais:

==	Igual que
!=	Diferente de
<	Menor que
>	Maior que
<=	Menor ou igual que
>=	Maior ou igual que
(. .) && (. .)	(..) e (..)
(. .) || (. .)	(..) ou (..)

Exp: Usando Chave

- Programação (software) usando a chave:

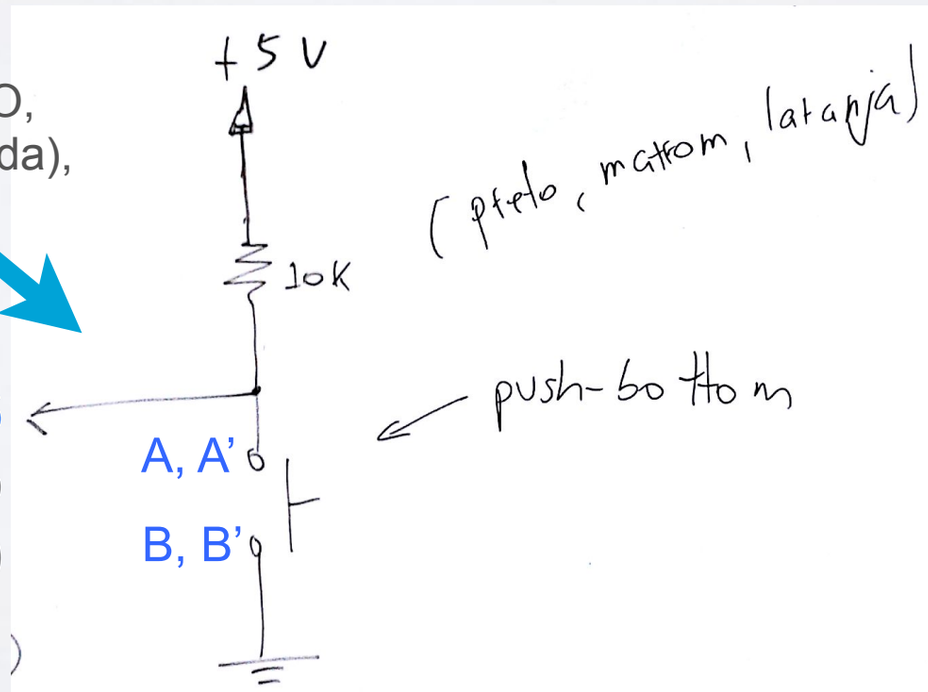


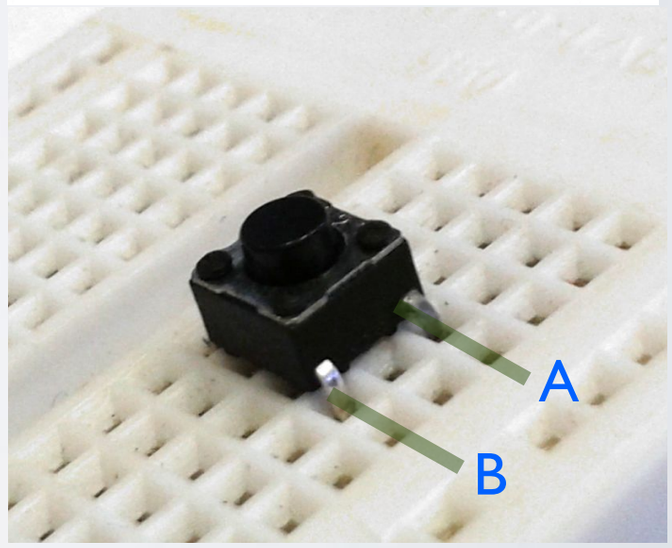



```
const int buttonPin = 6;  // pino 6 com pushbutton
int buttonState = 0;  // variável para estado do pushbutton
...
buttonState = digitalRead(buttonPin); // lê estado chave
if (buttonState == LOW) {
   // chave pressionada, faça algo
   ...
}
```

Note:
Somente enquanto a chave está pressionada no Pin 6
vamos “enxergar”, nível lógico BAIXO,
caso contrário (chave não pressionada),
Pin 6 = ‘1’.

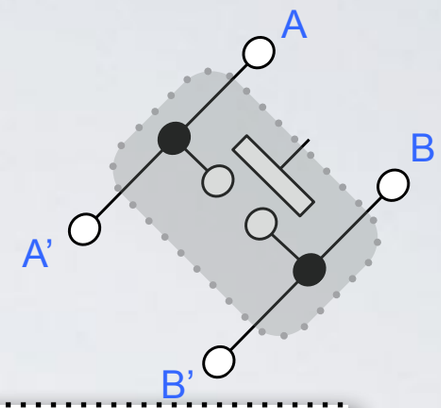
Pin **6**
(ou outro livre)

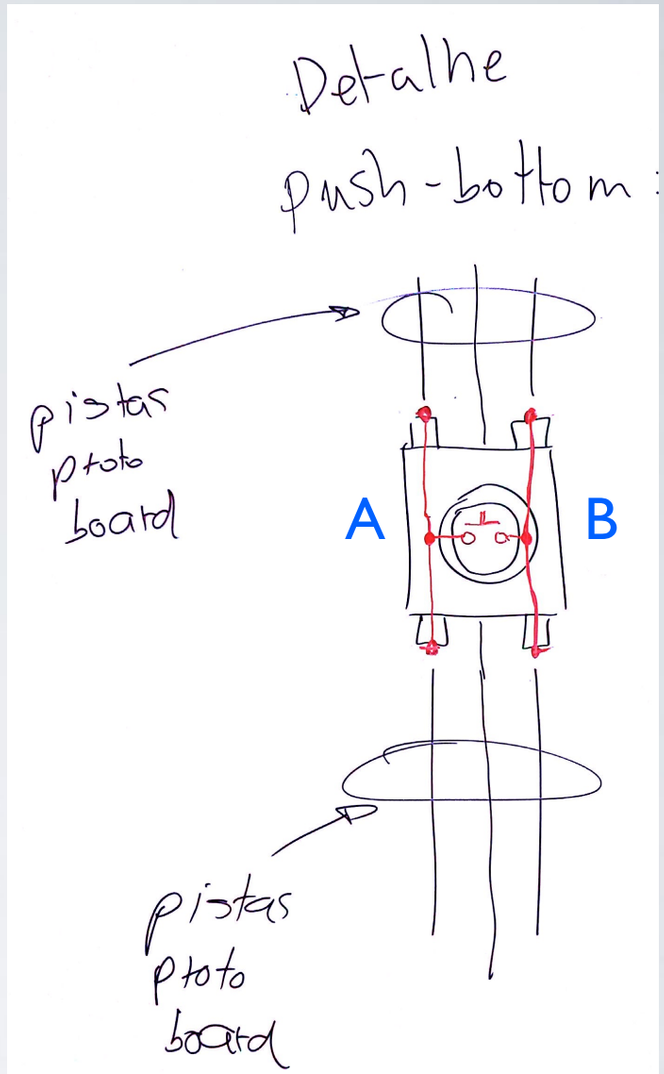


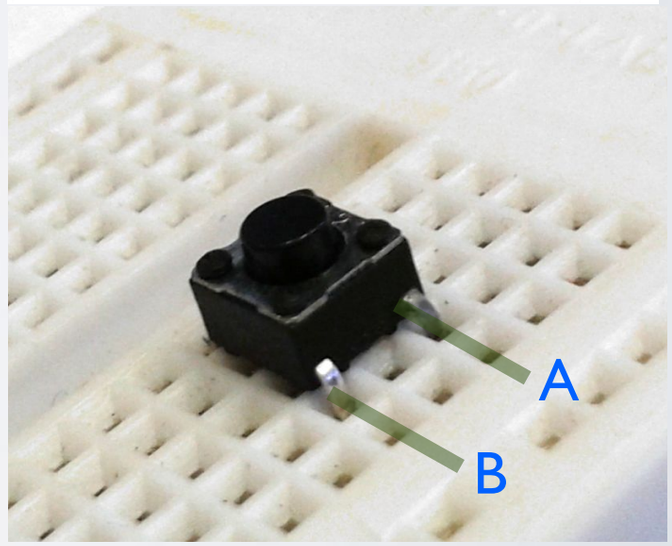


Exp: Usando Chave

- Sugestão de 1º programa (software) usando a chave:







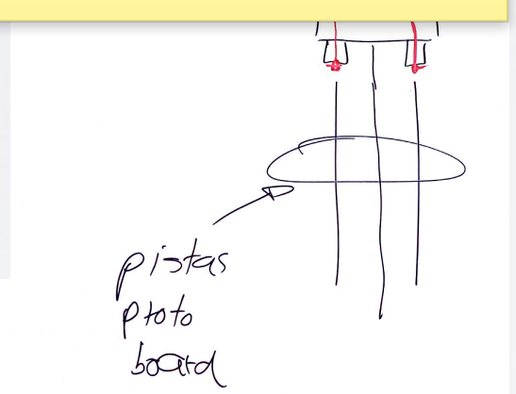

```
// set pin numbers:
const int buttonPin = 6;  // pino 6 com pushbutton
const int ledPin =  3;    // pino 3 com LED

int buttonState = 0;  // variável para estado do pushbutton

void setup() {
   // inicializado pino do LED como saída
   pinMode(ledPin, OUTPUT);
   digitalWrite(ledPin, LOW);  // inicia com led apagado
  // inicializa pino do pushbutton como entrada:
  pinMode(buttonPin, INPUT);
   // outras inicializações (se necesssário)
}

void loop() {
  digitalWrite(ledPin, LOW);  // apaga led
  buttonState = digitalRead(buttonPin); // lê estado chave
  if (buttonState == LOW) {
    digitalWrite(ledPin, HIGH); // acende LED por 50ms
    delay(50);
  }
}
```

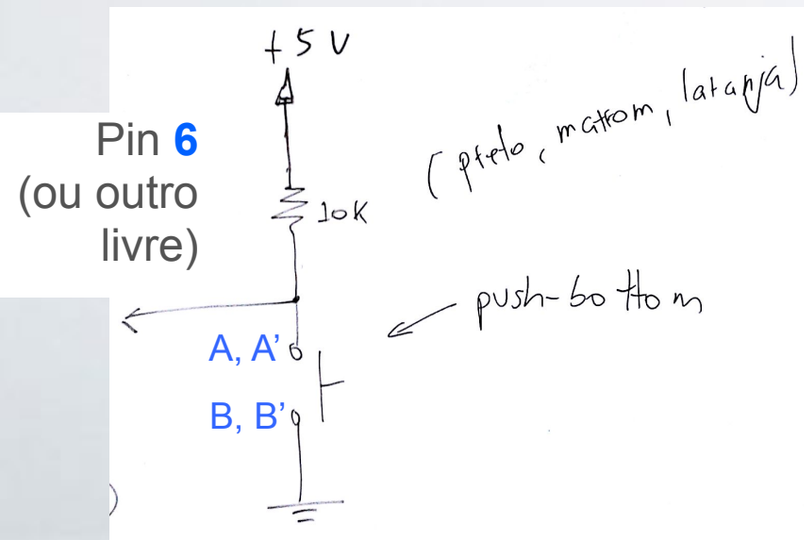
Problema_1:

Dicas/Sugestões:

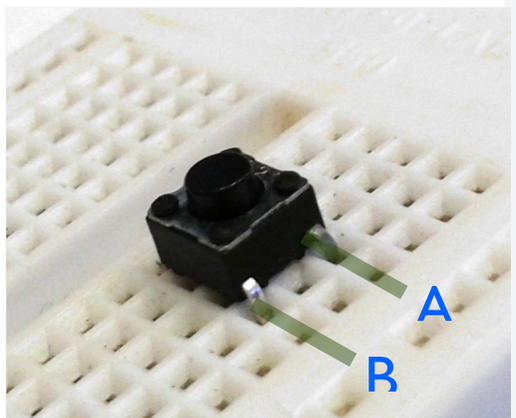
```
const int buttonPin = 6;  // pino 6 com
pushbutton
int buttonState = 0;  // variável para
estado do pushbutton
...
buttonState = digitalRead(buttonPin); // lê
estado chave
if (buttonState == LOW) {
   // chave pressionada, faça algo
   ...
}
```

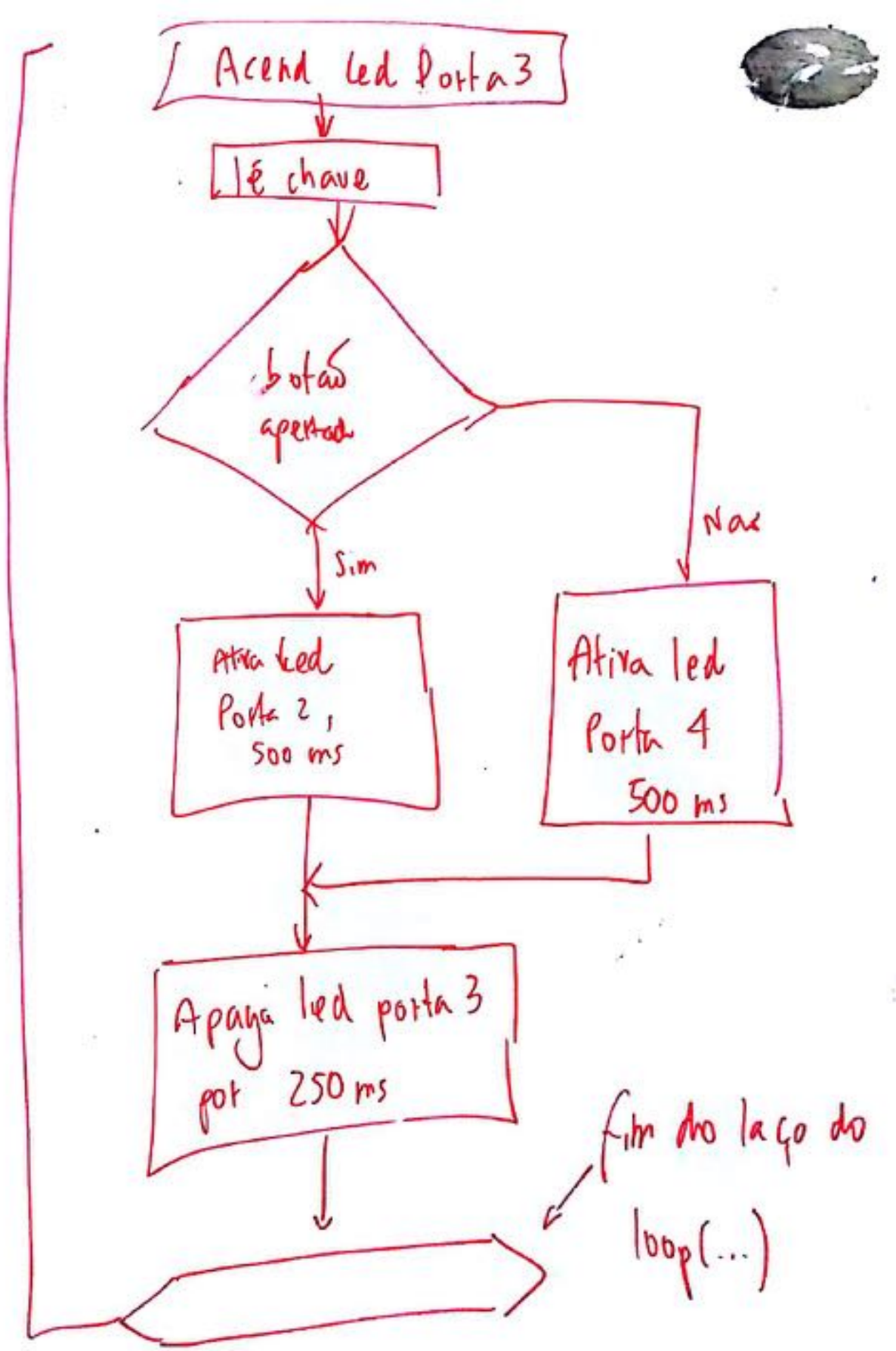
Até aqui em 17/05/2018





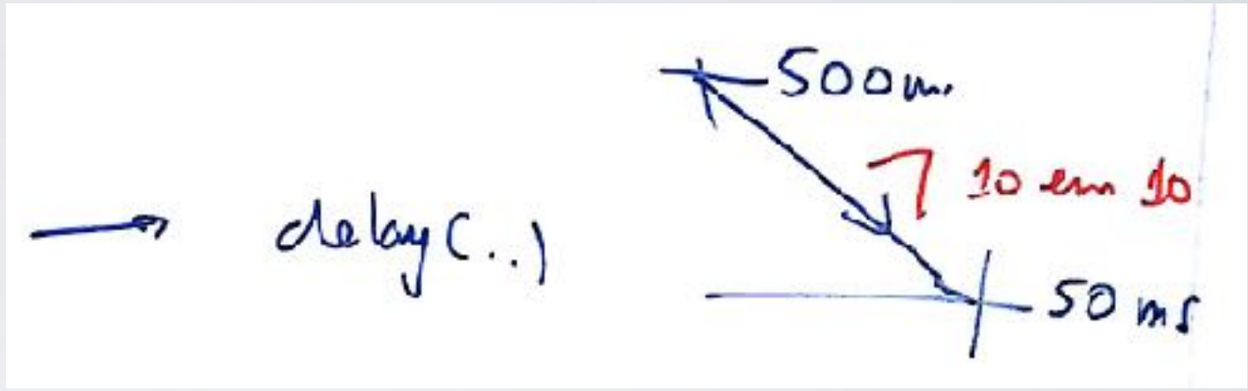
Pin **6** (ou outro livre)





Problema_2:

Enquanto NÃO se aperta a chave,
led pisca cada vez mais rápido.
Quando a chave é pressionada,
o led volta a piscar lento.



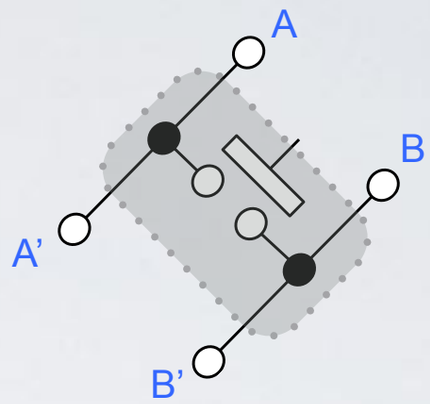

```pseudo
Ligar Led
delay(tempo)
tempo=tempo-10
if tempo < 10
   tempo = 50
end
Apaga Led
Lê chave
if chave pressionada
   tempo = 500
end
delay(tempo)
```

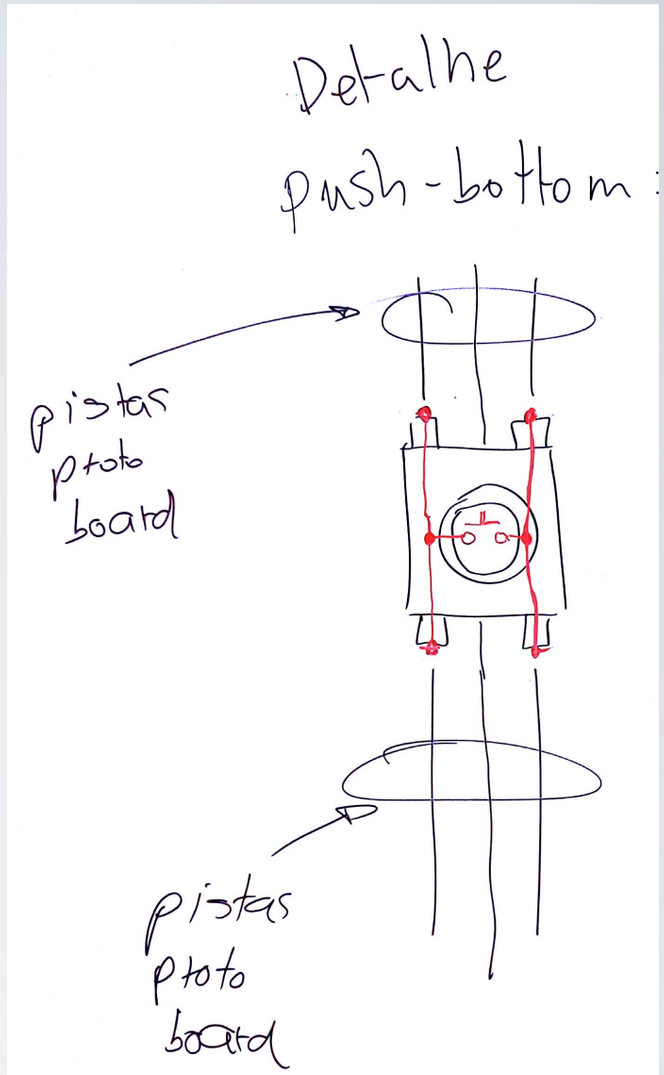
Markdoc pseudoooode

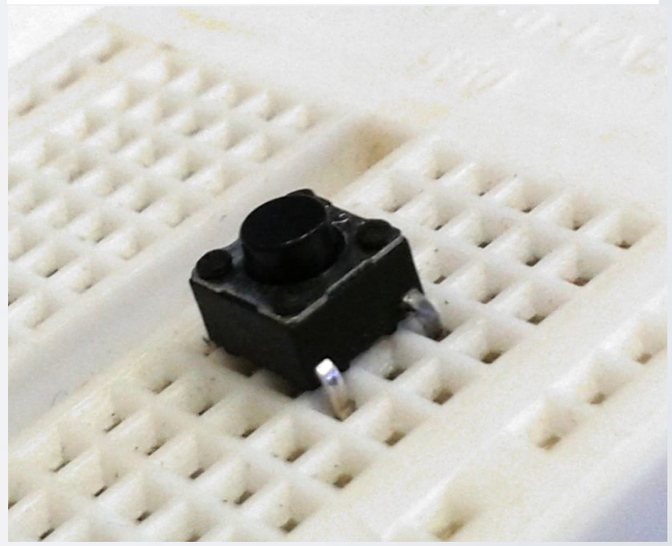
```arduino
const int buttonPin = 6;  // pino 6 com pushbutton
int buttonState = 0;  // variável para estado do
pushbutton
...
buttonState = digitalRead(buttonPin); // lê estado
chave
if (buttonState == LOW) {
   // chave pressionada, faça algo
   ...
}
```

Outro Exemplo Usando Chave

```
const int buttonPin = 6;  // pino 6 com pushbutton
int buttonState = 0;  // variável para estado do pushbutton

void setup() {
  // put your setup code here, to run once:
  pinMode(buttonPin, INPUT);
  // falta declarar pinos dos leds como OUTPUTs
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(2, LOW); digitalWrite(3, LOW); // apaga os leds
  digitalWrite(4, HIGH);
  buttonState = digitalRead(buttonPin); // lê estado chave
  if(buttonState==LOW){
    // caso verdadeiro
    digitalWrite(2, HIGH); // acende led do pino 2
    delay(100);
    digitalWrite(2, LOW); // apaga o led
    }
  else{
    // caso falso
    digitalWrite(3, HIGH); // acende led do pino 3
    delay(100);
    digitalWrite(3, LOW); // apaga o led
  }
  // código continua...
  digitalWrite(4, LOW);
  delay(50);
}
```

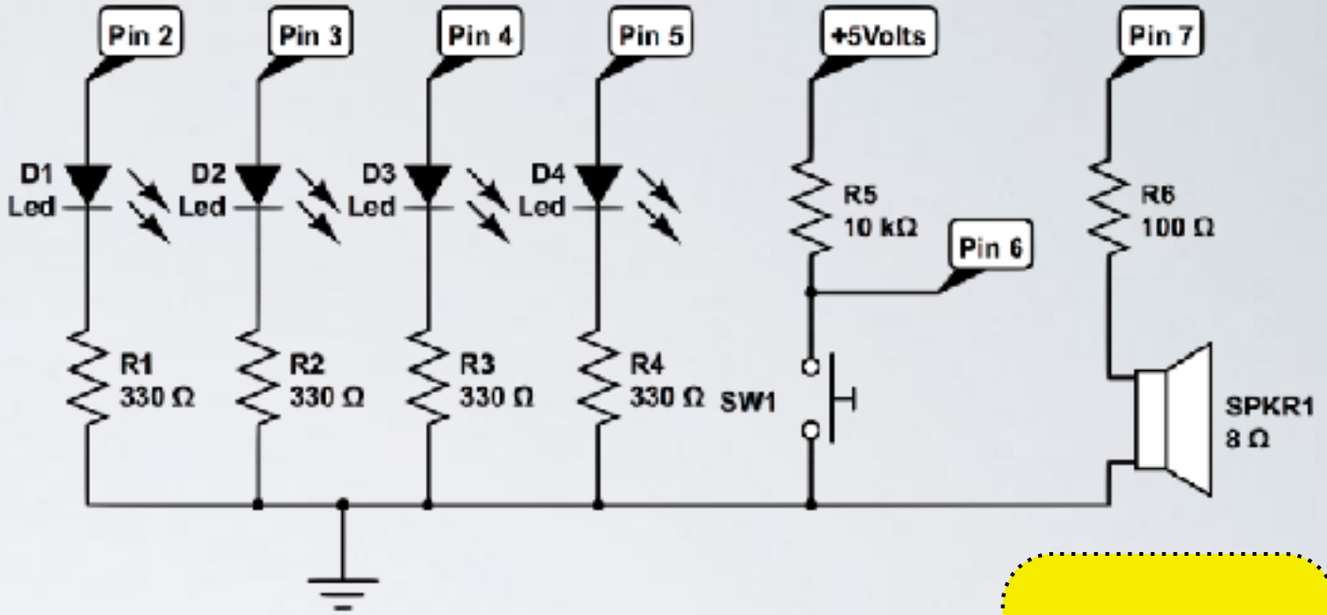


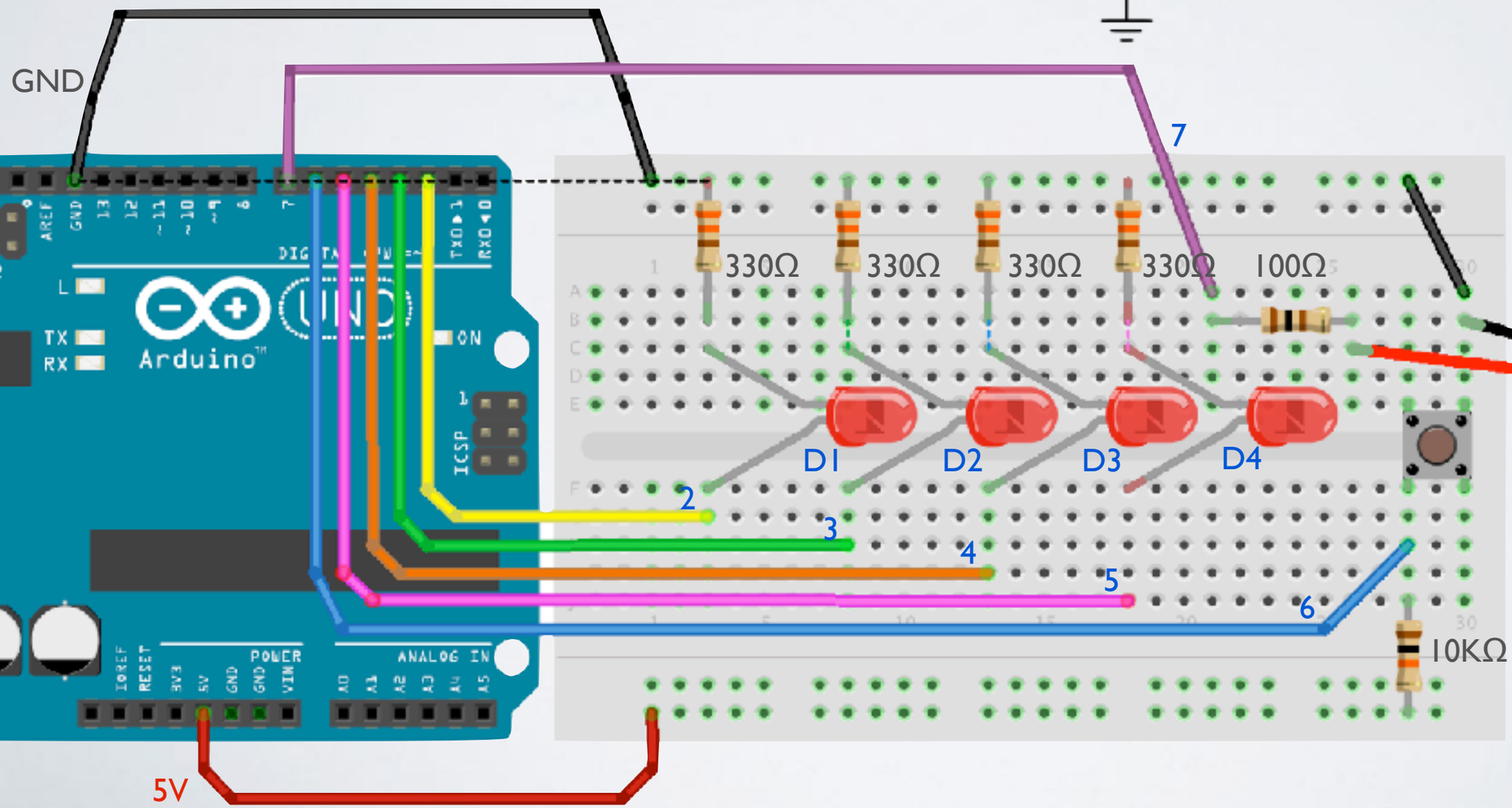




Exp: Produzindo Sons

- Uso de alto-falante ou buzzer + chave + melodias
- Circuito:





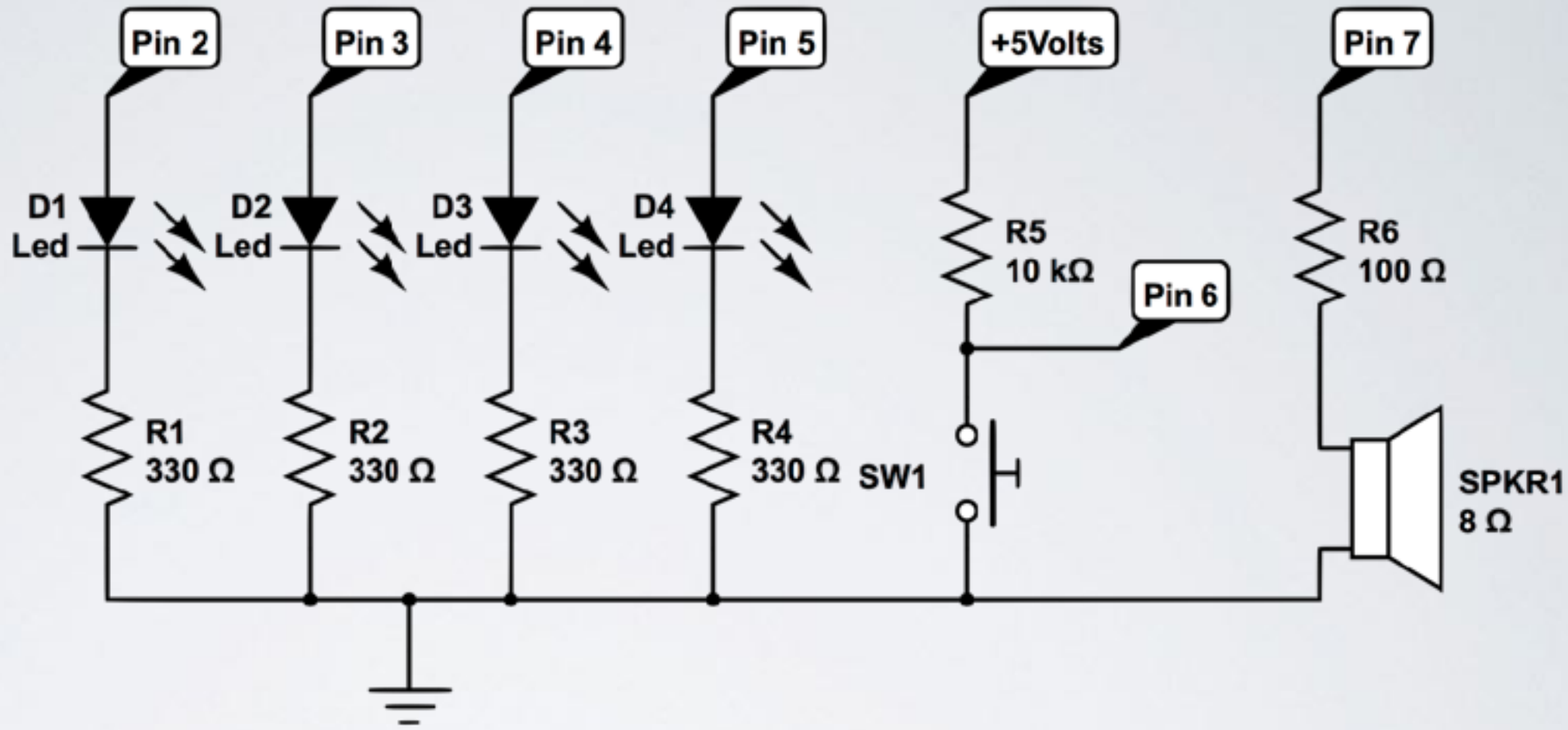
Pin 7: Buzzer
Pin 6: Chave
Pin 5: Led D4
Pin 4: Led D3
Pin 3: Led D2
Pin 2: Led D1

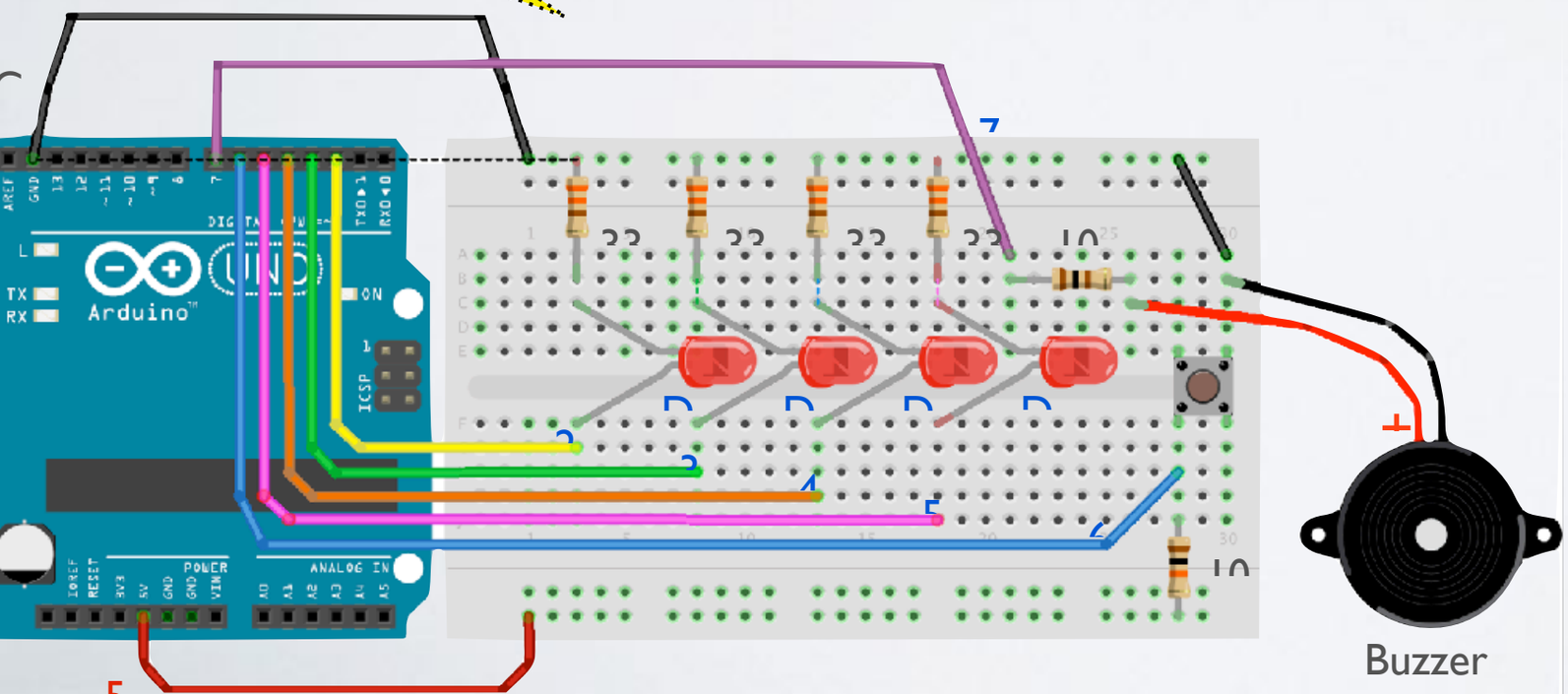
Exp: Produzindo Sons

- Uso de alto-falante ou buzzer + chave + melodias

- Circuito:

Pin 7: Buzzer
Pin 6: Chave
Pin 5: Led D4
Pin 4: Led D3
Pin 3: Led D2
Pin 2: Led D1



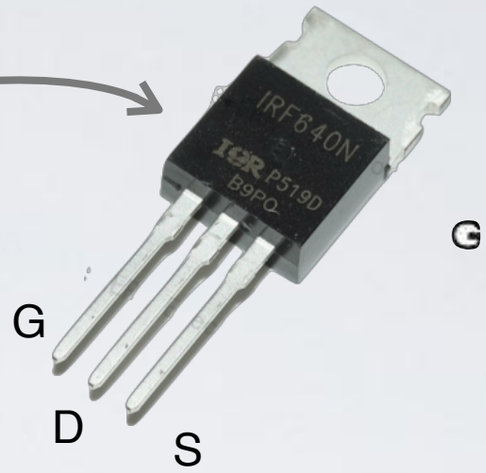


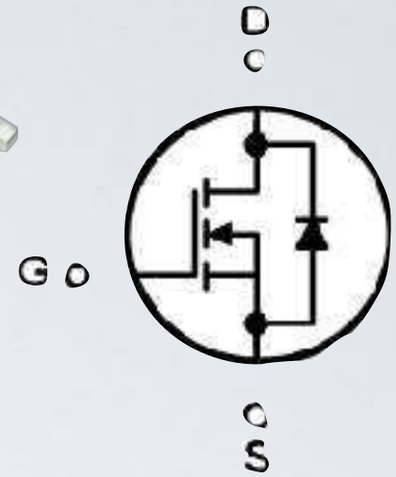
Conectando o Alto-FalantE

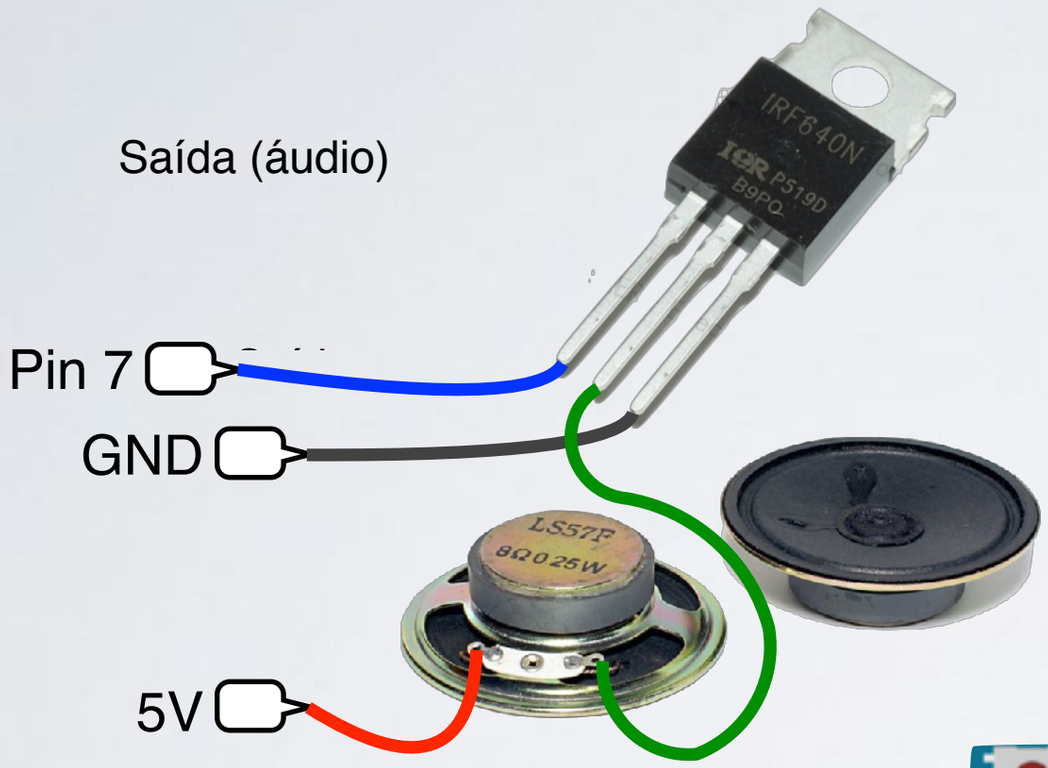
- Uso do FET IRF640:

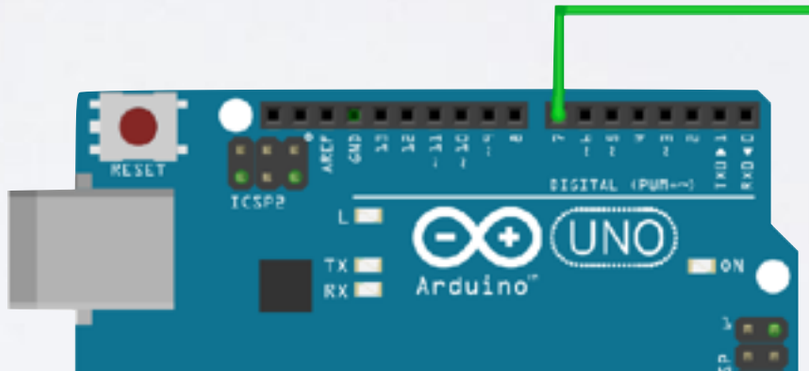
Conectando o Alto-FalantE

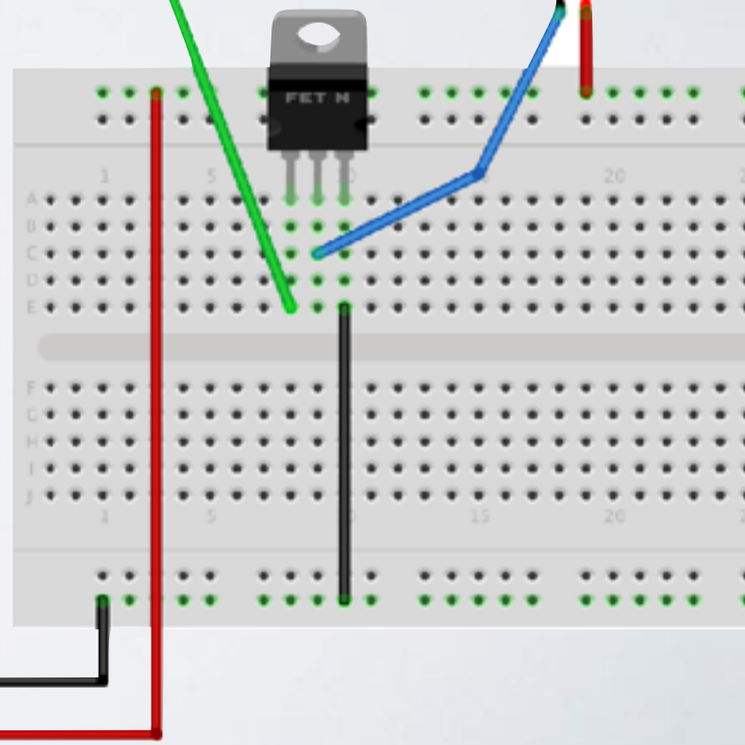
- Uso do FET IRF640:











Exp: Produzindo Sons

- Uso de alto-falante ou buzzer + chave + melodias

Sintaxe:

- **tone**(pino, frequência):
- **tone**(pino, frequência, duração);

➧ gera onda quadrada, ciclo de 50%

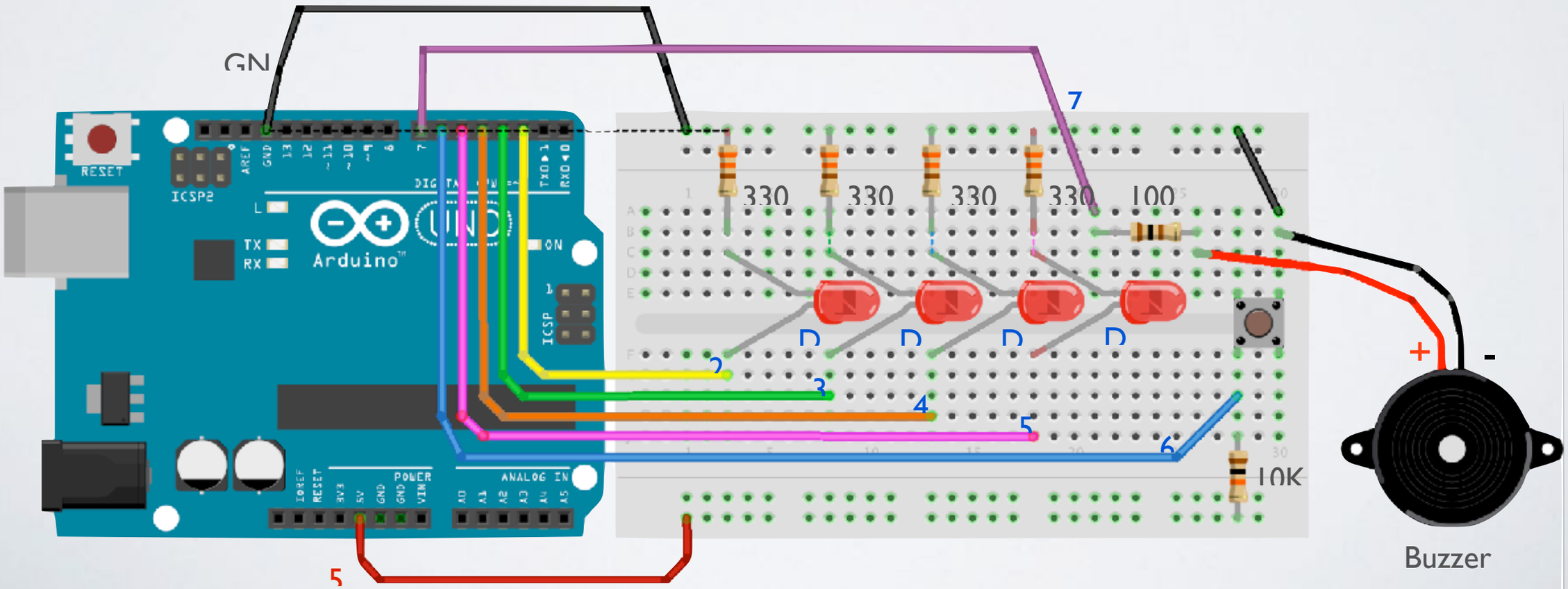
31 < freq. < 6.535 Hz

- **Software** ➪ alto-falante:
  - **tone**(speakerPin, notes[thisNote], noteDurationDefault);
  - **noTone**(speakerPin);

Ex.:
494Hz ('B4')
20ms

Não esquecer de inicializar:

- **pinMode**(speakerPin, OUTPUT);



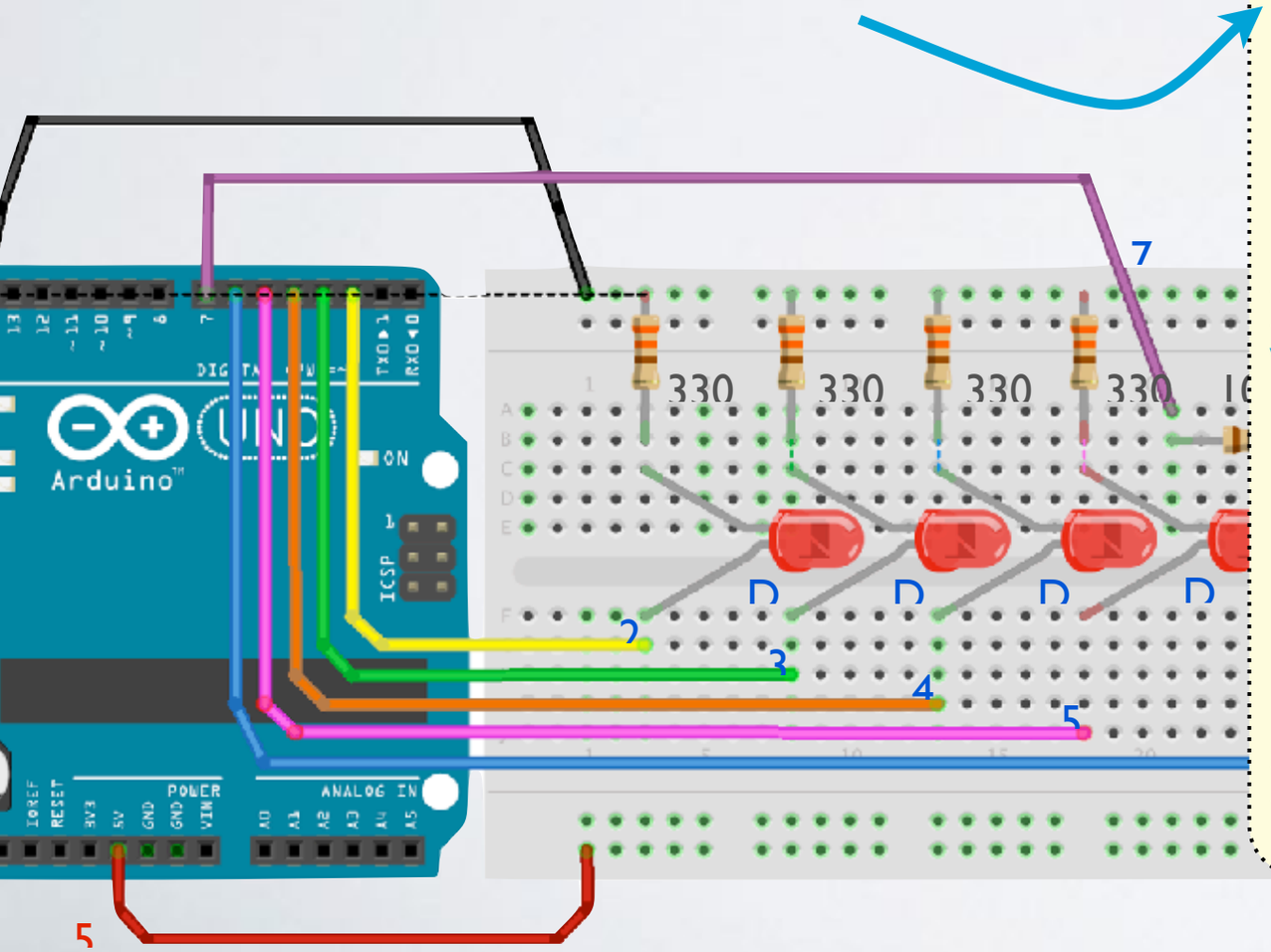
Exp: Produzindo

- Uso de alto-falante ou buzzer + chave + melodias

- Software ➪ buzzer:

tone(speakerPin, notes[thisNote], noteDurationDefault);
noTone(speakerPin);

teste inicial




```
// Sugestão de software para teste da chave
//  + buzzer + led

// setting pin numbers:
const int buttonPin = 6; // the number of the
pushbutton pin
const int ledPin =  3; // the number of the LED pin
const int speakerPin = 7; // pin number for the
speaker
const int noteDurationDefault = 20; // play notes
for 20 ms
int buttonState = 0; // variable for reading the
pushbutton status
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  // initialize pin 0 as an output pin
  pinMode(speakerPin, OUTPUT);
  digitalWrite(ledPin, HIGH);
  tone(speakerPin, 494, 150);
  noTone(speakerPin);
  digitalWrite(ledPin, LOW);
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == LOW) {
    digitalWrite(ledPin, HIGH);
    tone(speakerPin, 494, 150);
    tone(speakerPin, 196, 150);
    noTone(speakerPin);
    digitalWrite(ledPin, LOW);
  }
}
```

Buzzer

Outro Exemplo explorando Som:

Bloco de Inicialização variáveis e constantes (globais)

```
// Sugestão 2 de software para teste da chave
//  + buzzer + led

// setting pin numbers:
const int buttonPin = 6; // the number of the pushbutton pin
const int ledPin =  3; // the number of the LED pin
const int speakerPin = 7; // pin number for the speaker
const int noteDurationDefault = 20; // play notes for 20 ms
int buttonState = 0; // variable for reading the pushbutton status
```

Opcional

Bloco do Setup

```
// Parte do código que só é executada uma única vez
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  // initialize pin 0 as an output pin
  pinMode(speakerPin, OUTPUT);
  digitalWrite(ledPin, HIGH);
  tone(speakerPin, 494, 150);
  noTone(speakerPin);
  digitalWrite(ledPin, LOW);
}
```

Bloco do Loop

```
// Parte do código que é repetida n-vezes
void loop() {
  // put your main code here, to run repeatedly:
  int f; // f = frequência que varia
  buttonState = digitalRead(buttonPin);
  if (buttonState == LOW) {
    digitalWrite(ledPin, HIGH);
    for (f=32; i<=15000; f = f + 10){
      tone(speakerPin, f, 20);
      delay(20);
    }
    noTone(speakerPin);
    digitalWrite(ledPin, LOW);
  }
}
```

Obs.: Note que todos os programas no Arduíno seguem esta “estrutura"!

Bloco do Setup

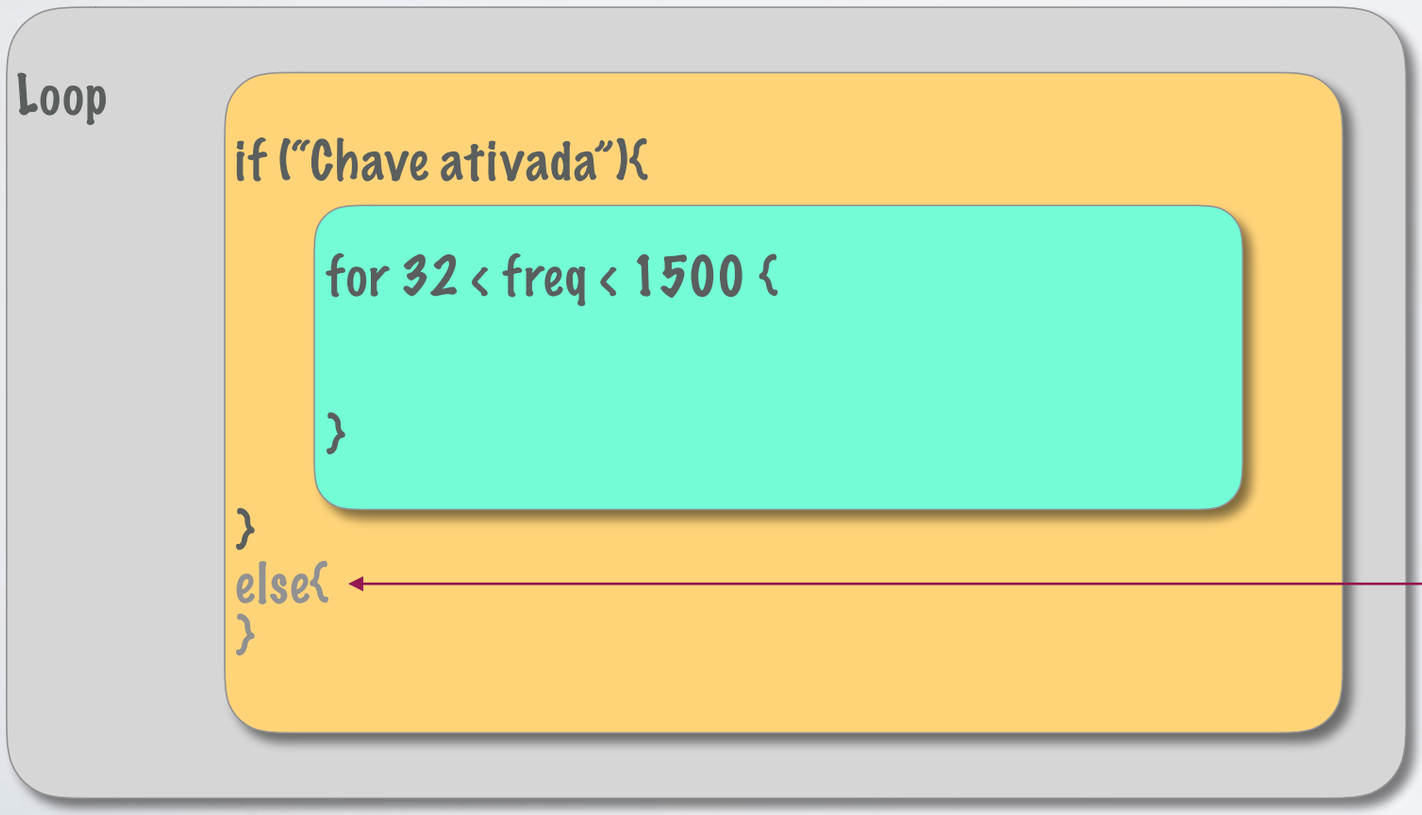
```
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  // initialize pin 0 as an output pin
  pinMode(speakerPin, OUTPUT);
  digitalWrite(ledPin, HIGH);
  tone(speakerPin, 494, 150);
  noTone(speakerPin);
  digitalWrite(ledPin, LOW);
}
```

Bloco do Loop

```
// Parte do código que é repetida n-vezes
void loop() {
  // put your main code here, to run repeatedly:
  int f; // f = frequência que varia
  buttonState = digitalRead(buttonPin);
  if (buttonState == LOW) {
    digitalWrite(ledPin, HIGH);
    for (f=32; i<=15000; f = f + 10){
      tone(speakerPin, f, 20);
      delay(20);
    }
    noTone(speakerPin);
    digitalWrite(ledPin, LOW);
  }
}
```

Note:

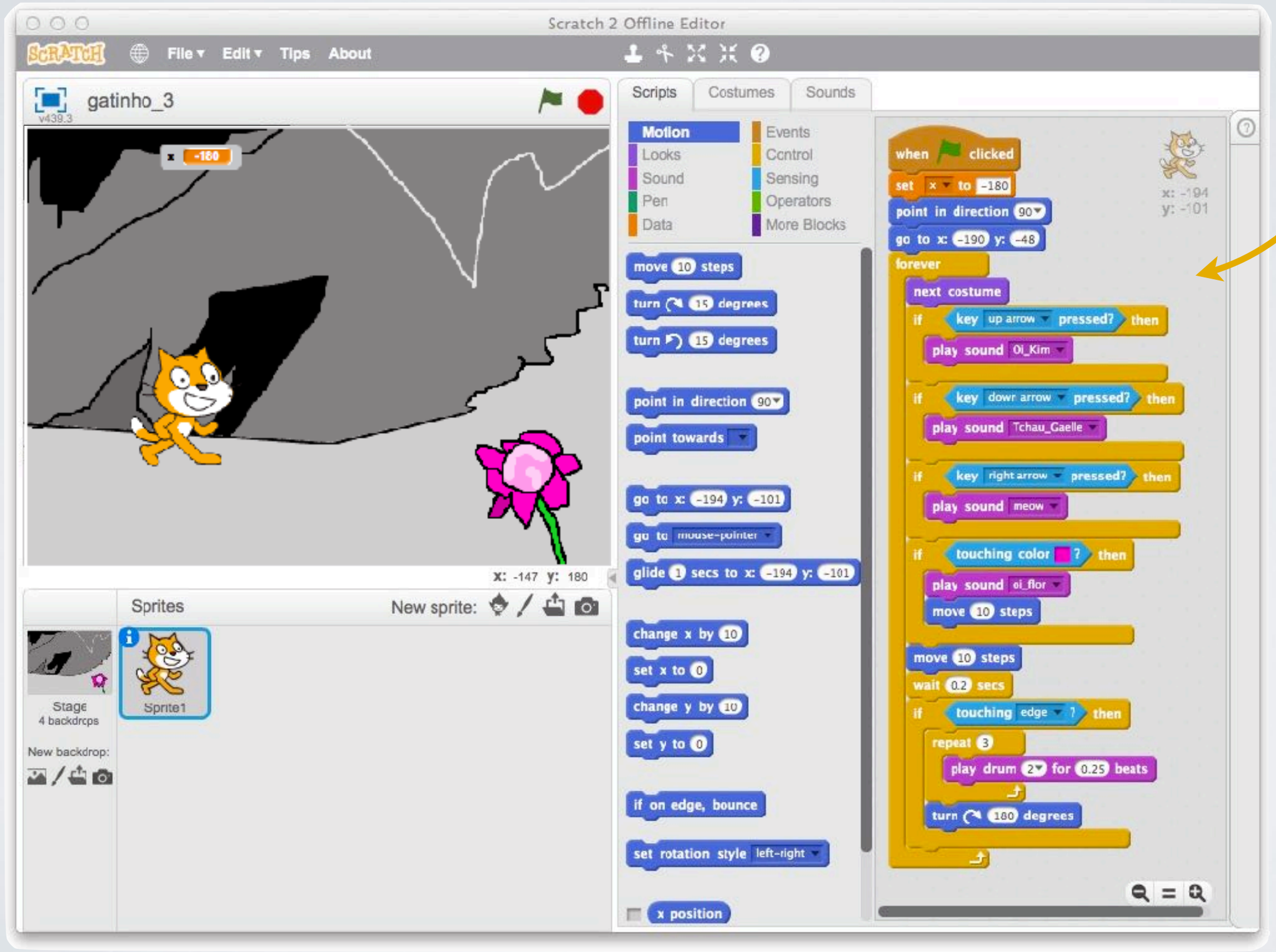
Dentro do Bloco do Loop, existem 2 blocos um dentro do outro:



Note:
Posso "cascatear" blocos;
Posso colocar um bloco dentro de outro!

Imagine seu programa organizado em blocos!

Tal qual se raciocina no “**Scratch**”, apenas com a diferença neste caso, que, ao invés da linguagem ser gráfica (caso do Scratch), a linguagem do Arduíno exige texto (código) escrito !



Uso de biblioteca com tons musicais...

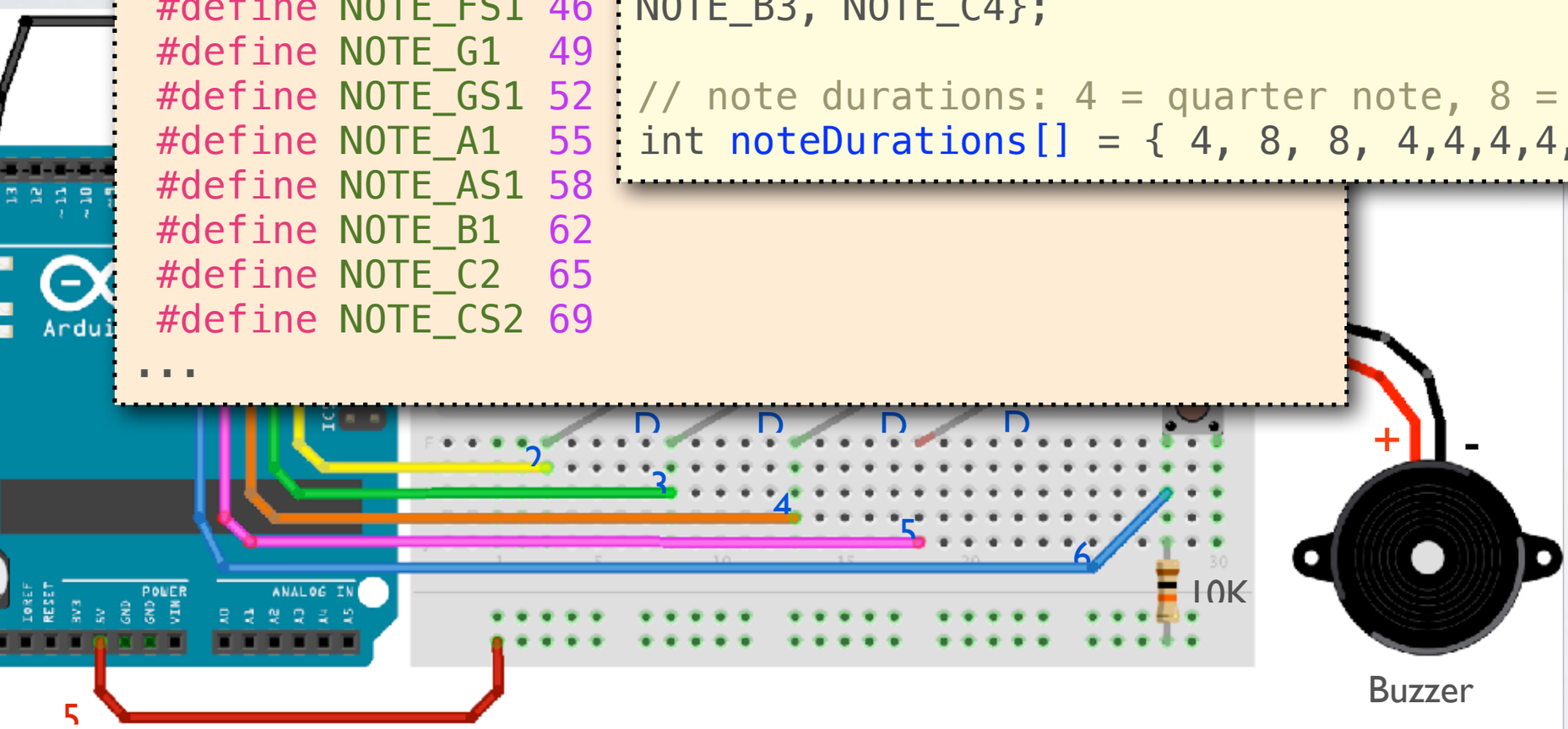
- Facilitar programação...

uso de vetores ou array's[]...

pitches.h:

```
/*************************
* Public Constants
*************************/

 #define NOTE_B0  31
 #define NOTE_C1  33
 #define NOTE_CS1 35
 #define NOTE_D1  37
 #define NOTE_DS1 39
 #define NOTE_E1  41
 #define NOTE_F1  44
 #define NOTE_FS1 46
 #define NOTE_G1  49
 #define NOTE_GS1 52
 #define NOTE_A1  55
 #define NOTE_AS1 58
 #define NOTE_B1  62
 #define NOTE_C2  65
 #define NOTE_CS2 69
...
```

```
#include "pitches.h"

// set pin numbers:
const int buttonPin = 10;   // the number of the pushbutton pin
const int ledPin =  3;      // the number of the LED pin
const int speakerPin = 8;   // pin number for the speaker
const int noteDurationDefault = 20;   // play notes for 20 ms

// notes in the melody:
int notes[] = { NOTE_A4, NOTE_B4, NOTE_C3 };

int melody[] = { NOTE_C4, NOTE_G3, NOTE_G3, NOTE_GS3, NOTE_G3,0,
NOTE_B3, NOTE_C4};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = { 4, 8, 8, 4,4,4,4,4 };
```



melody2.ino

melody2 | Arduino 1.6.8

melody2 pitches.h

```
/*
 *  Melody2.cc
 *  Fernando Passold, em 29 ago 2016
 *  Ref.: http://itp.nyu.edu/physcomp/labs/labs-arduino-
 *
 */
#include "pitches.h"

// set pin numbers:
const int buttonPin = 10;     // the number of the pushb
const int ledPin =  3;        // the number of the LED p
const int speakerPin = 8;              // pin number for
const int noteDurationDefault = 20;    // play notes for

// notes in the melody:
int notes[] = {
  NOTE_A4, NOTE_B4, NOTE_C3 };

int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_GS3, NOTE_G3,0, NOTE_B3, NOTE_C4};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4,4,4,4,4 };

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

```

melody2 | Arduino 1.6.8

melody2 pitches.h

```
41
42 void setup() {
43   // initialize the LED pin as an output:
44   pinMode(ledPin, OUTPUT);
45   // initialize the pushbutton pin as an input:
46   pinMode(buttonPin, INPUT);
47
48   pinMode(speakerPin, OUTPUT); // initialize pin 0 as an output pin
49
50   digitalWrite(ledPin, HIGH);
51   for (int thisNote = 0; thisNote < 4; thisNote++) {
52     tone(speakerPin, notes[thisNote], noteDurationDefault);
53     //pause for the note's duration plus 30 ms:
54     delay(noteDurationDefault);
55   }
56   noTone(speakerPin);
57   digitalWrite(ledPin, LOW);
58 }
59
```

Salvo.

melody2.ino

melody2 | Arduino 1.6.8

melody2 | Arduino 1.6.8

melody2 | Arduino 1.6.8

melody2 pitches.h

```
1  /*
2   *  Melody2.cc
3   *  Fernando Passold, em 29 ago 2016
4   *  Ref.: http://itp.nyu.edu/physcomp/labs/labs-arduino-digital-and-
5   *
6   */
7  #include "pitches.h"
8
41
42 void setup() {
43   // initialize the LED pin as an output:
44   pinMode(ledPin, OUTPUT);
45   // initialize the pushbutton pin as an input:
46   pinMode(buttonPin, INPUT);
47
48   pinMode(speakerPin, OUTPUT); // initialize pin 0 as an output pin
49
50   digitalWrite(ledPin, HIGH);
51   for (int thisNote = 0; thisNote < 4; thisNote++) {
52     tone(speakerPin, notes[thisNote], noteDurationDefault);
53     //pause for the note's duration plus 30 ms:
54     delay(noteDurationDefault);
55   }
56   noTone(speakerPin);
57   digitalWrite(ledPin, LOW);
58 }
59
```

Salvo.

melody2 pitches.h

```
59
60 void loop() {
61   // put your main code here, to run repeatedly:
62   // read the state of the pushbutton value:
63   buttonState = digitalRead(buttonPin);
64
65   // check if the pushbutton is pressed.
66   // if it is, the buttonState is HIGH:
67   if (buttonState == HIGH) {
68     // turn LED on:
69     digitalWrite(ledPin, HIGH);
70     for (int thisNote = 0; thisNote < 8; thisNote++) {
71       // to calculate the note duration, take one second
72       // divided by the note type.
73       //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
74       int noteDuration = 1000/noteDurations[thisNote];
75       tone(speakerPin, melody[thisNote], noteDuration);
76       //pause for the note's duration plus 30 ms:
77        delay(noteDuration +30);
78   }
79   noTone(speakerPin);
80   digitalWrite(ledPin, LOW);
81   } else {
82     // turn LED off:
83     digitalWrite(ledPin, HIGH);
84     delay(150);
85     digitalWrite(ledPin, LOW);
86     delay(150);
87   }
88 }
```

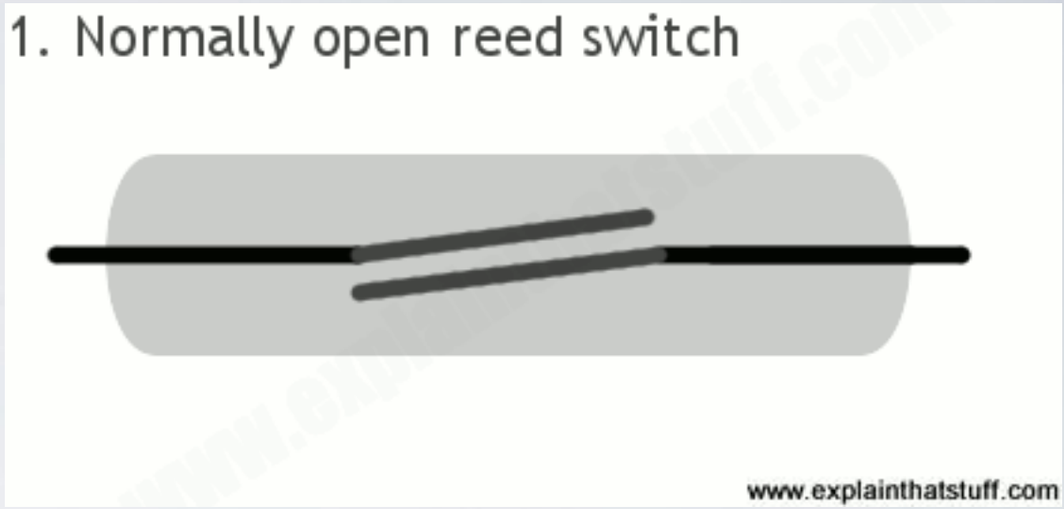
Salvo.

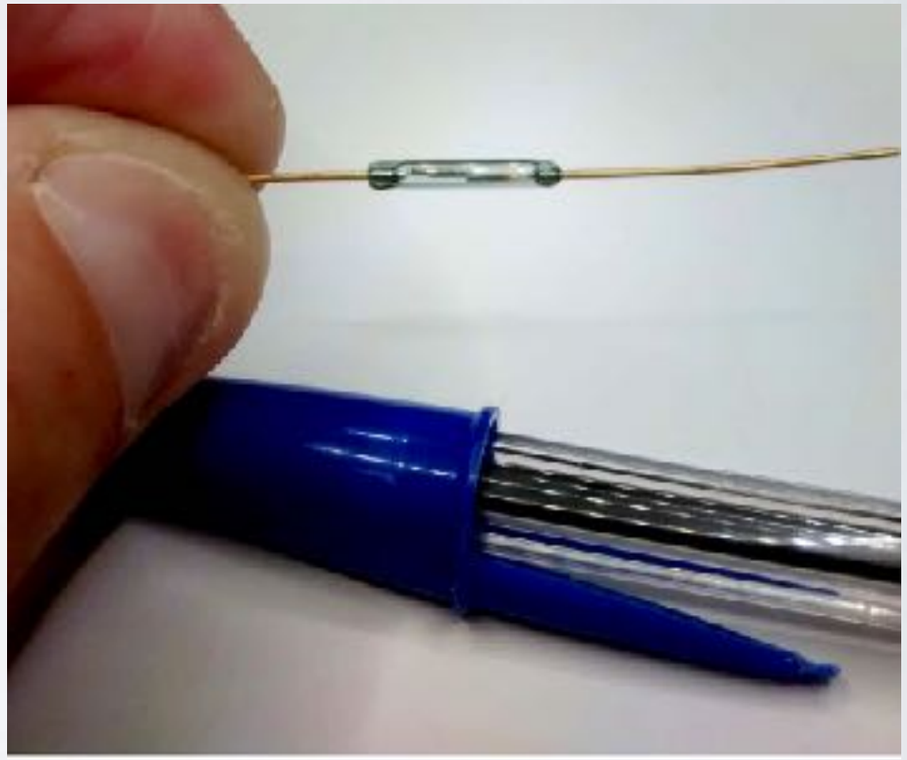
41 Arduino/Genuino Uno em /dev/cu.usbr

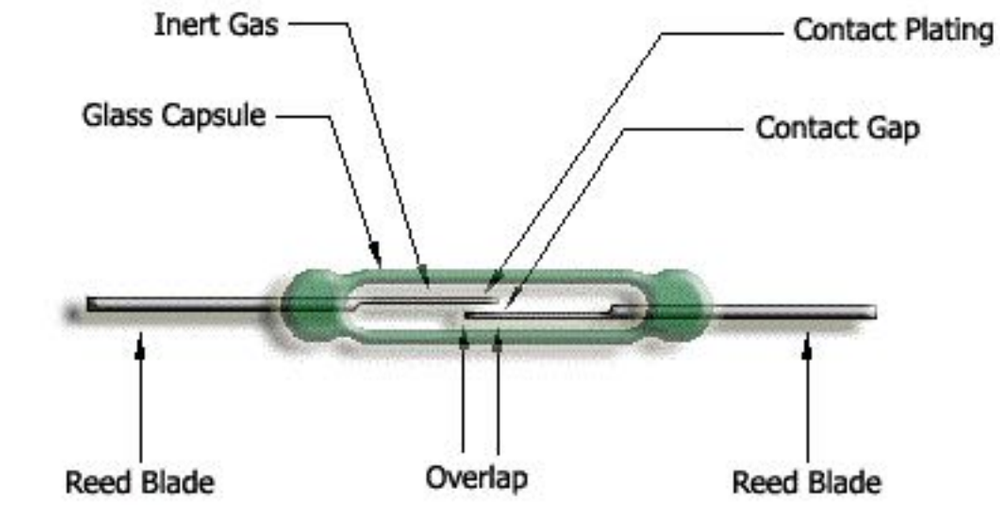
Uso de REED-SWITCHES

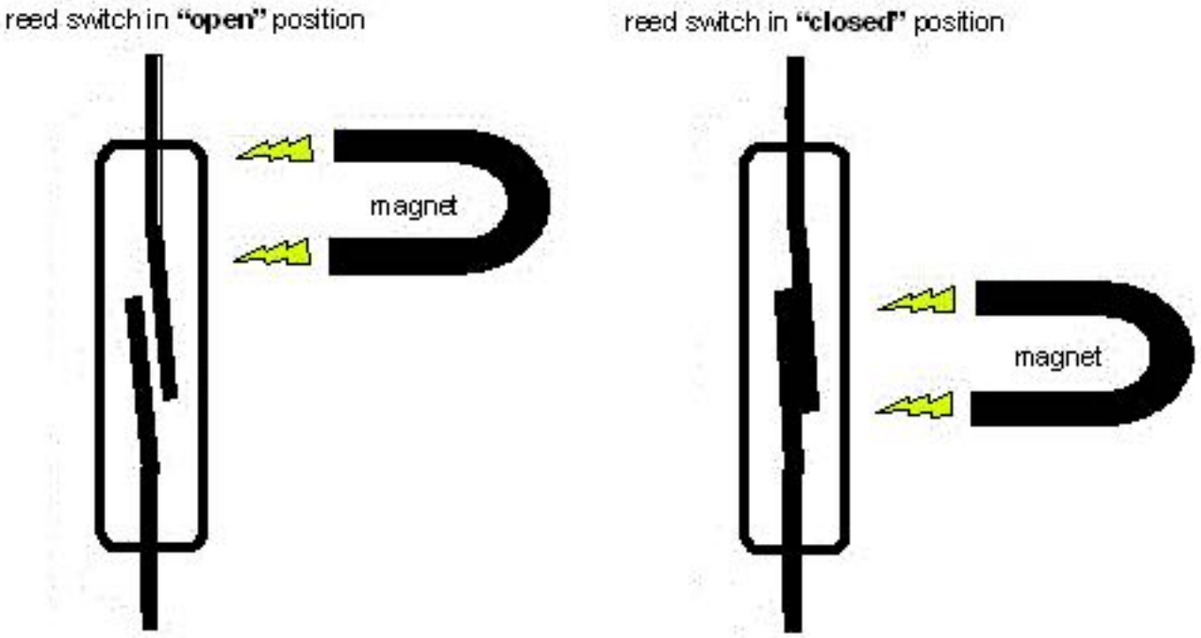
Outro Sensor (Entrada de info

- Exemplos:





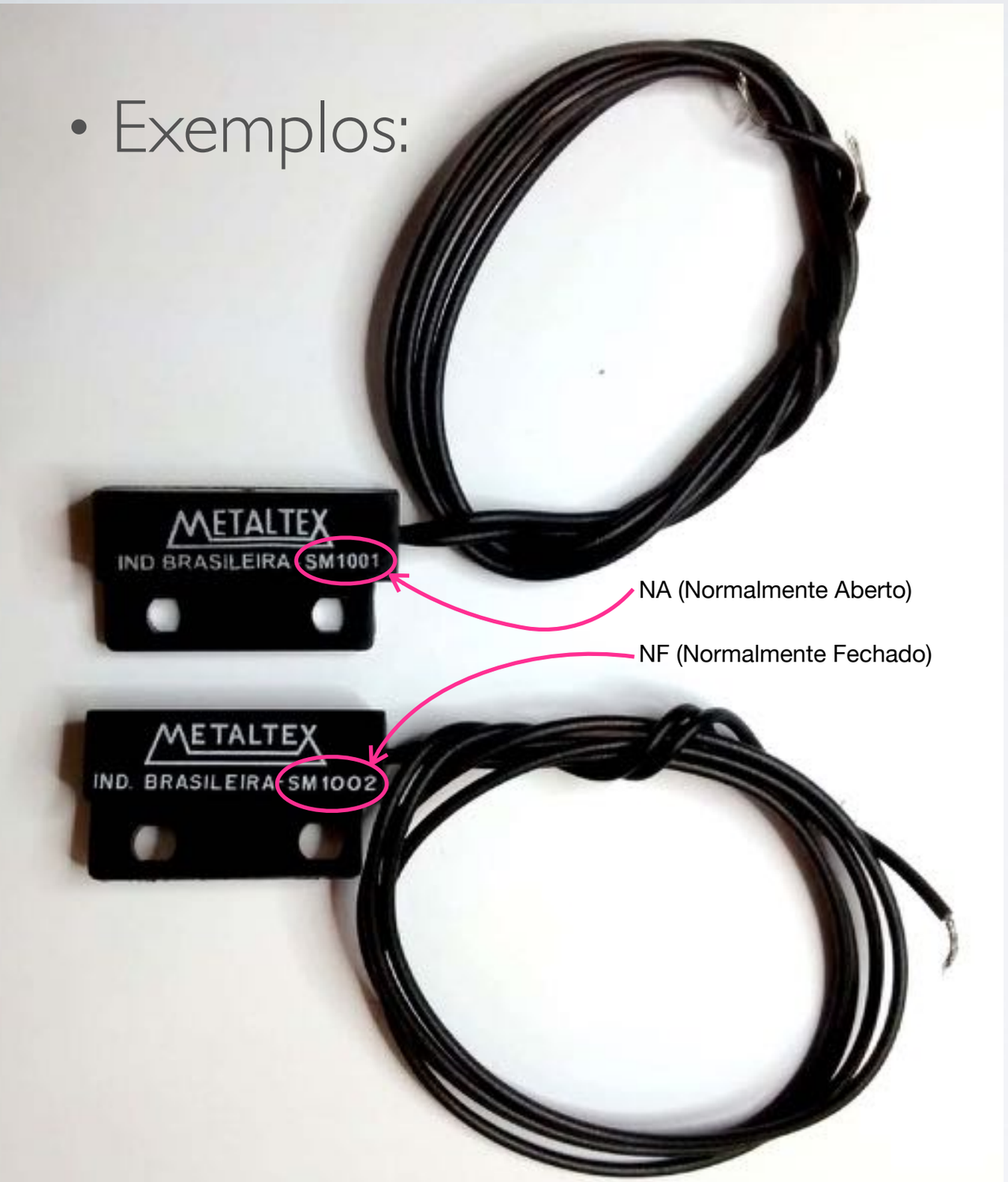


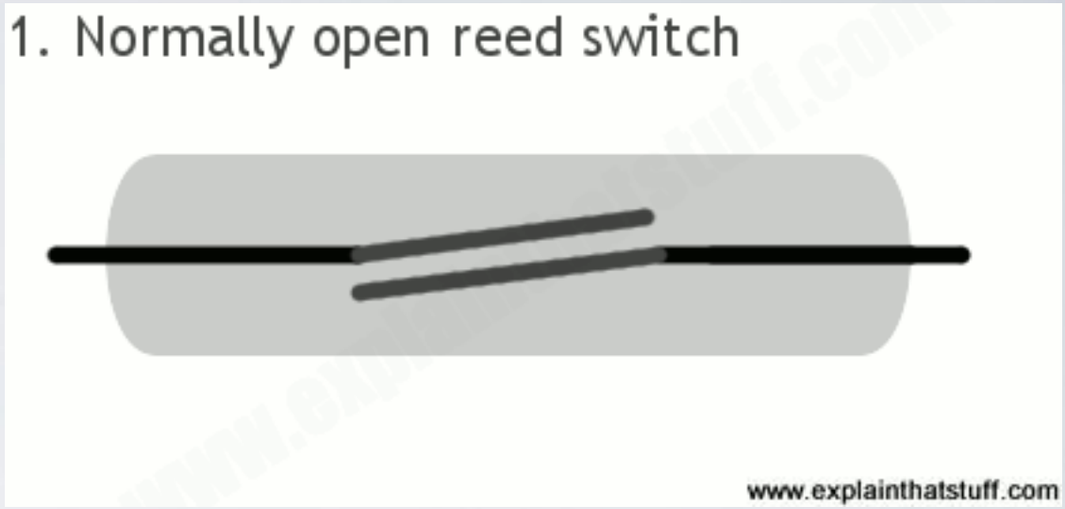


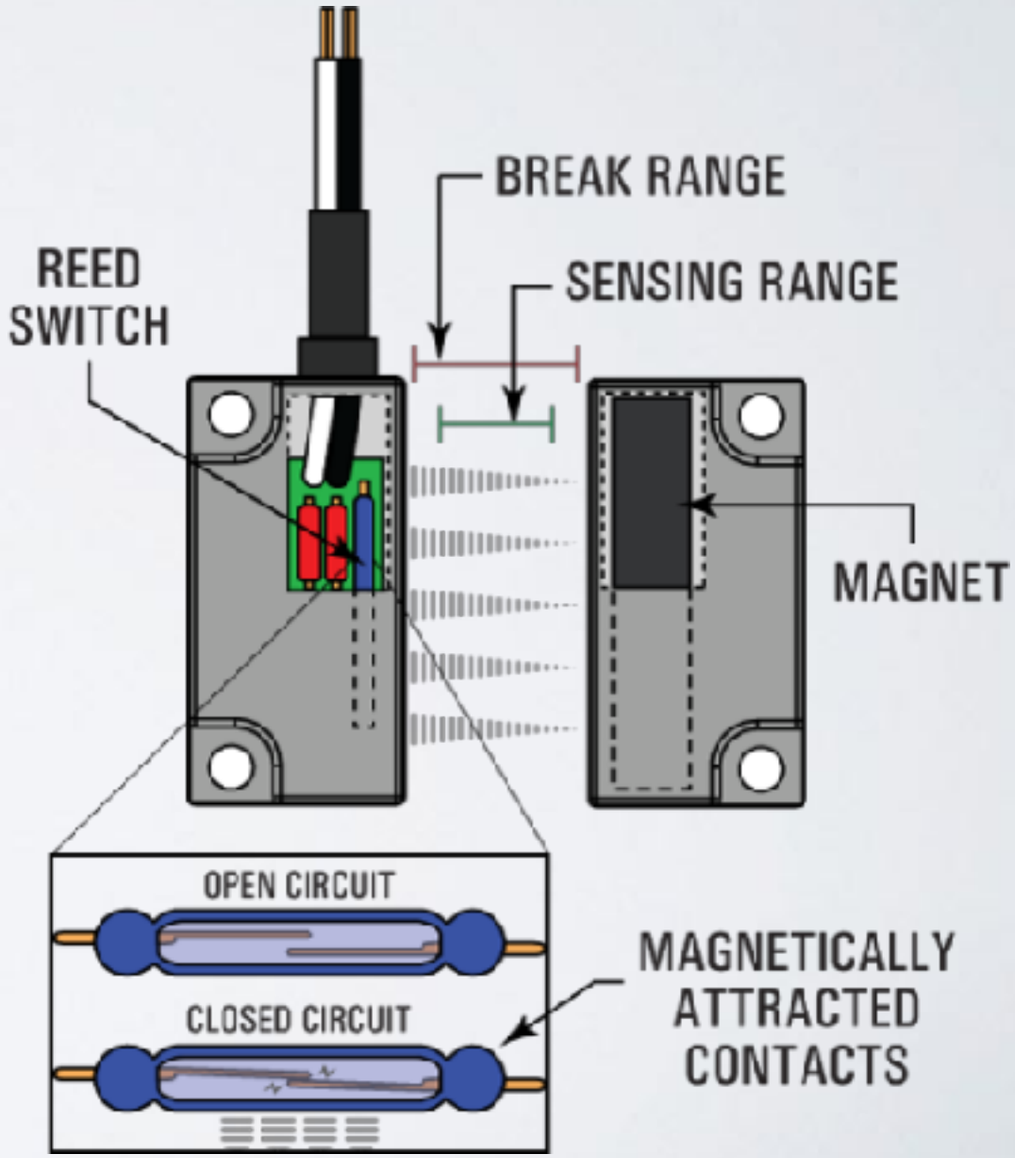
Uso de REED-SWITCHES

Outro Sensor
(Entrada de info.)

- Exemplos:





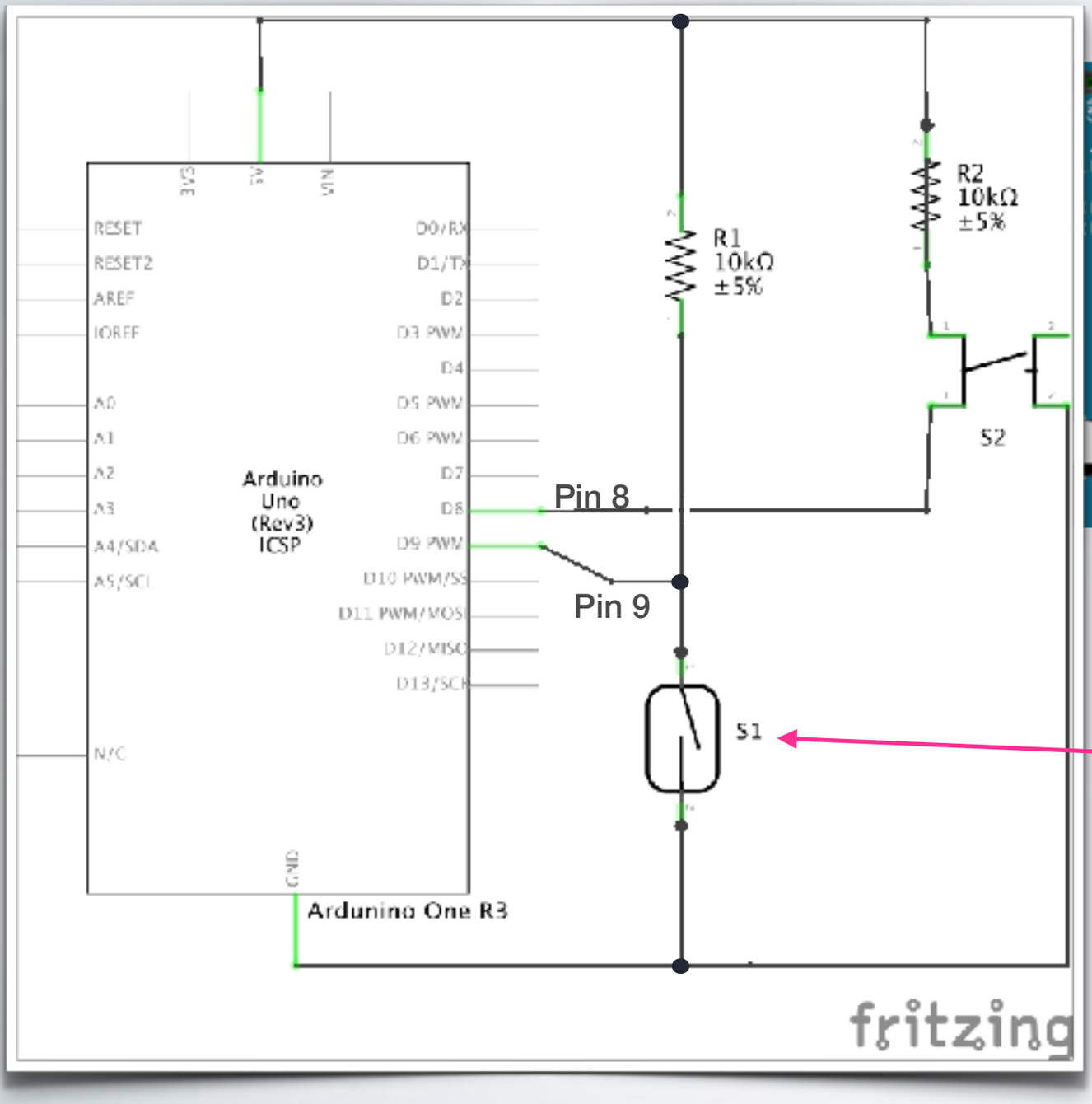


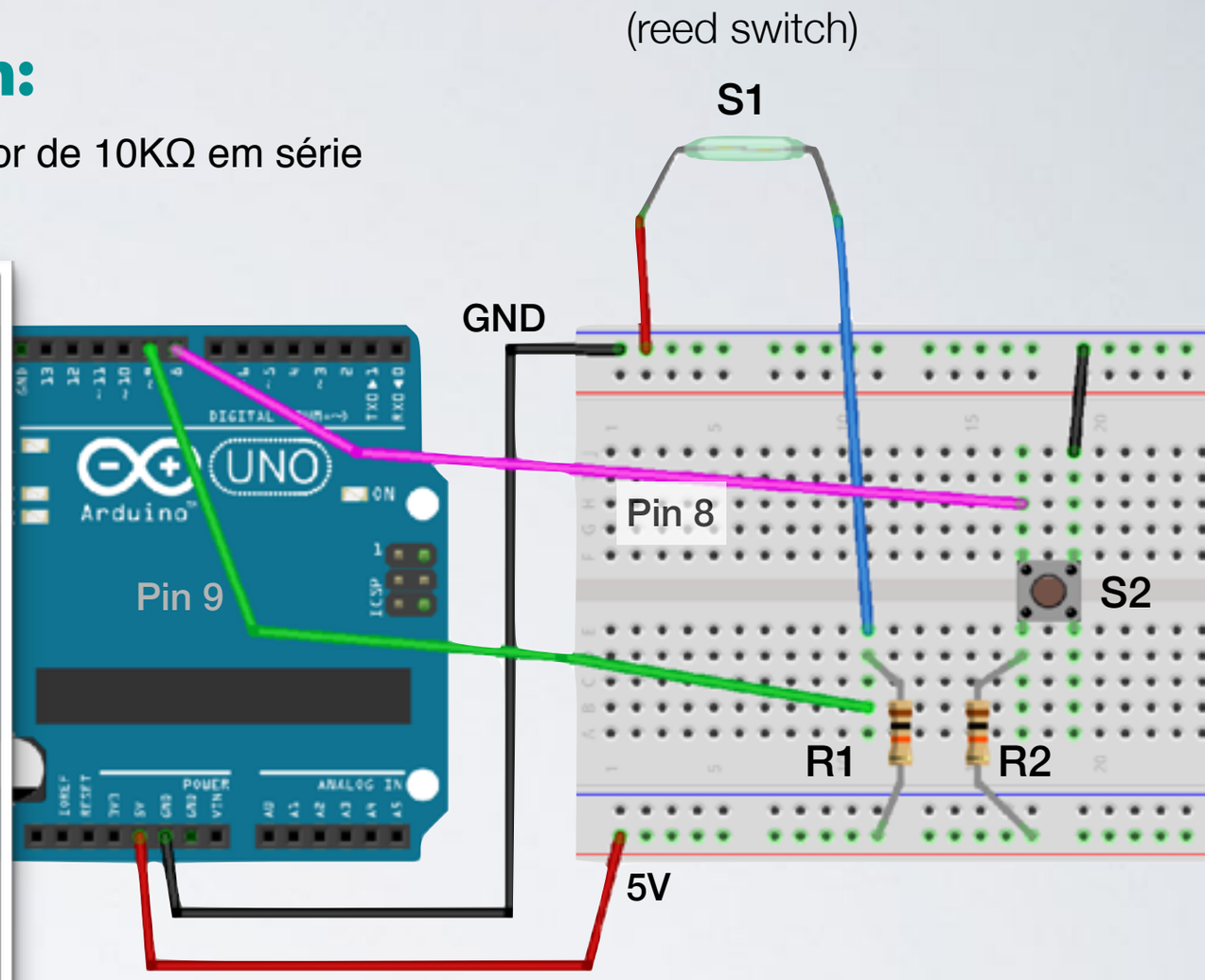
Uso de REED-SWITCHES

Outro Sensor

Usa-se tal qual como o push buton:

Isto é, não esquecer o resistor de 10KΩ em série no circuito da chave!





Obs.:
O Reed-switch (S1) é Normalmente Aberto, isto é:

1) quando não há um imã emparelhado com o mesmo: S1 se comporta como um circuito aberto; no caso da figura, o Pin 9 (do Arduino) = HIGH (ativar alarme).
2) quando o imã está encostado em S1, Pin 9 = LOW (alarme não dispara).

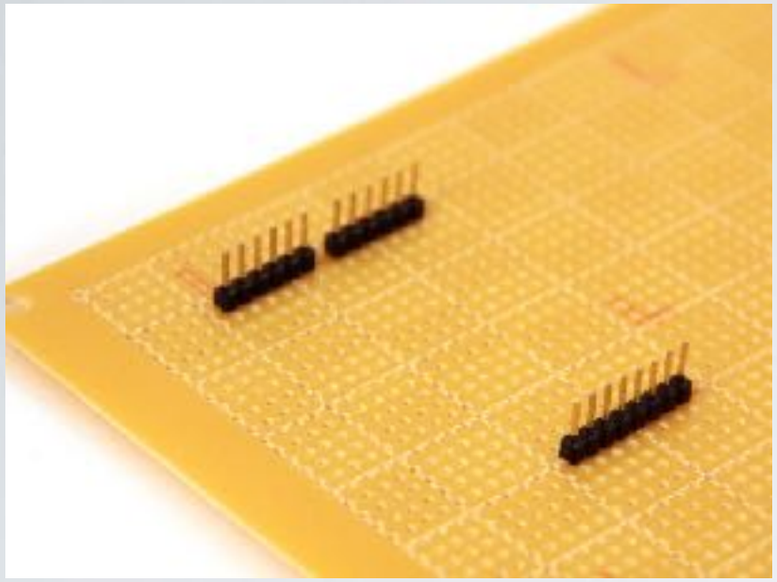
DESAFIO FinaL

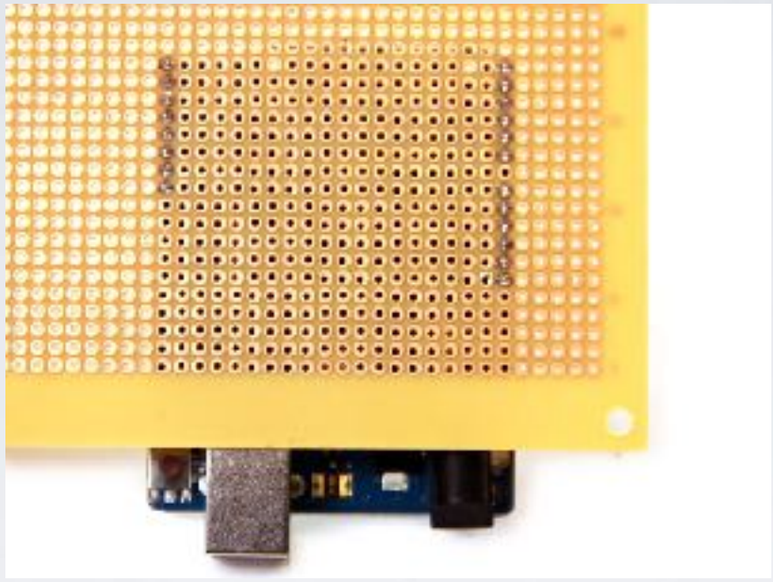
Circuito de alarme contendo

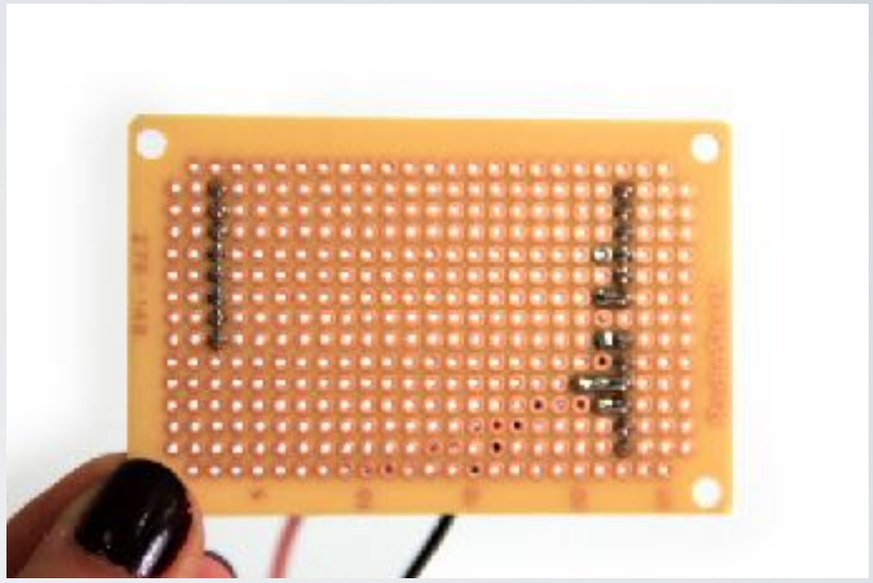
- “teclado”: 4 teclas (push-buttons) para entrada da senha (sequencia de 4 números:
  Detalhe: teclado deve “travar" temporariamente depois cada tentativa errada; subsequentes tentativas erradas incrementam o período de tempo no qual o teclado fica bloqueado!
- Led’s para informar “status" do alarme:
  (o) Armado (o) Desligado (o) (Teclado) Bloqueado
- 1 sensor do tipo reed-switch;
- 1 x Alto-falante:
  feedback sonoro ☞ “dispara alarme”, toca melodia de “welcome" se senha correta.

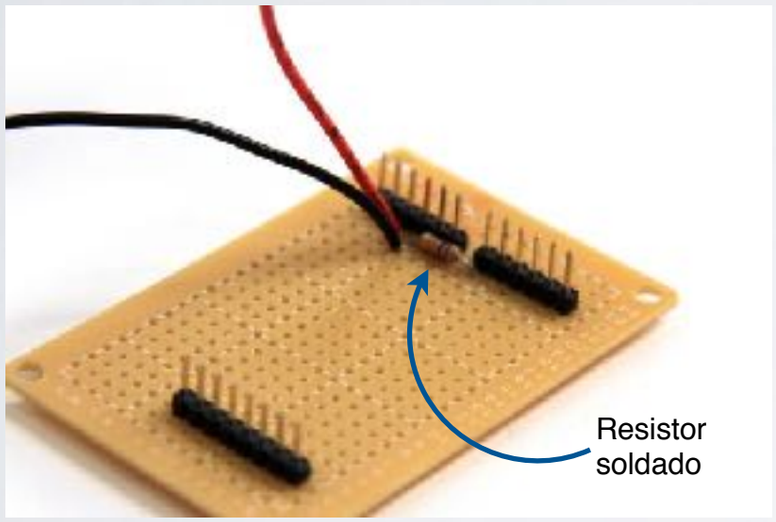
- Detalhe: o circuito deverá ser “montado" numa placa de circuito impresso universal (como um “shield" para o Arduino One) e apresentado no último dia de aula: 28/06/2016.

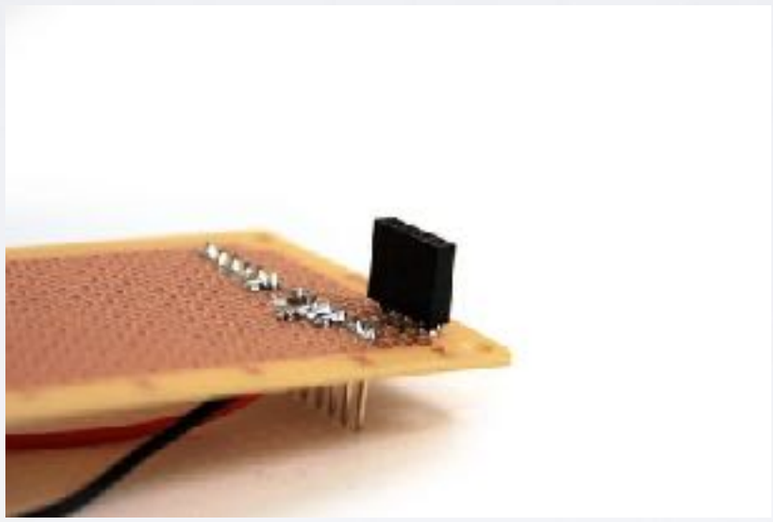
Placas Universais => Shield

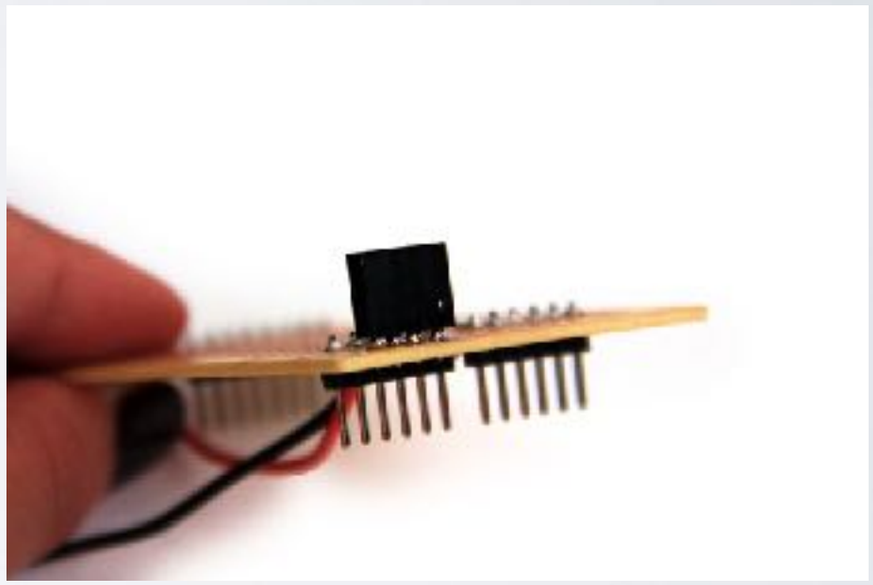


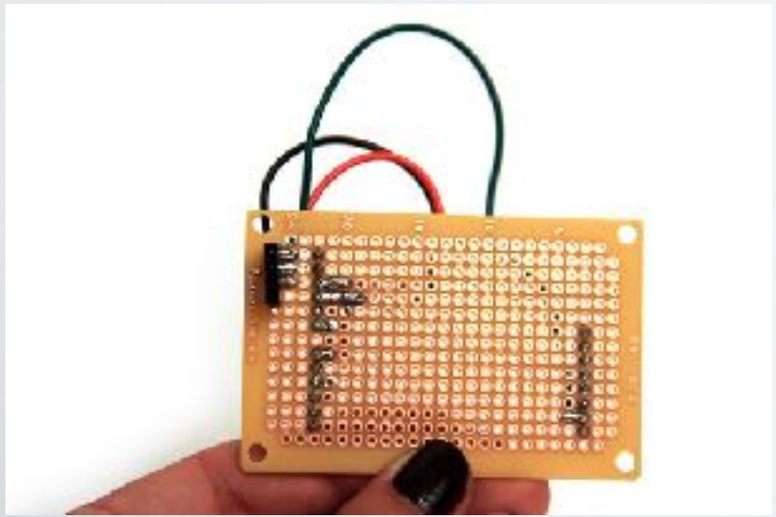










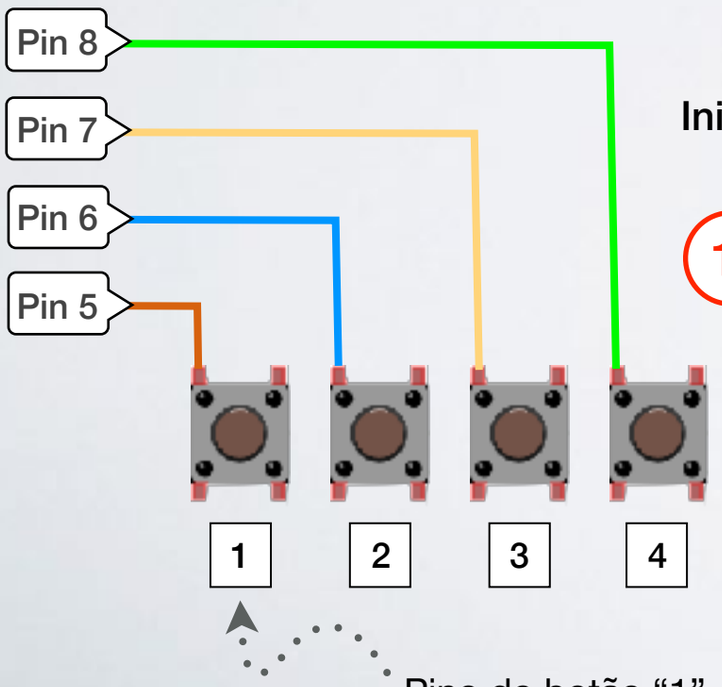


Dicas:
Como Capturar Senha

1. Guardar senha num vetor

2. Ler tecla à tecla (incrementado um contador/índice para “posição" do caracter da senha), comparando caracter à caracter. Em caso de erro, bloqueia-se teclado e se re-inicia contador/índice para “posição” do caracter da senha.

3. Exemplo:



Pino do botão “1” → posição = 0 → pino = buttonPin[0] = 5;

Parte do Código (exemplo) ➨

2.

```
void loop(){
  // index = 0; // -> inicia leitura senha
  int i;

  // lê “teclado”
   buttonPressed = 0; // nada apertado
   for (i=0; i<=3; i++){  // verifica as 4 teclas
      buttonState = digitalRead(buttonPin[i]);
      if (buttonState == LOW) {
         buttonPressed = i + 1; // No. botão apertado - 1
         if (buttonPressed == senha[index]) {
            // Só segue adiante no momento que usuário "soltar" tecla
            // necessário para evitar "auto-repetição" indevida
            do {
               buttonState[i] = digitalRead(buttonPin[i]);
            } while (buttonState == LOW);
            // usuário acertou caracter, prepara para próximo
            // caracter da senha
            index = index + 1;
         }
         else{
            // usuário errou caracter da senha
            index = 0;
            // Falta “bloquear" teclado
            ...
         } // fim if testando senha
      } // fim if tecla pressionada
   } // fim do for leitura teclado
   if (index >= 4){
      // leu toda senha, desbloquear alarme
      ...
   }
   ... // continua restante do programa…
} // fim do loop()
```

Inicialização:

1.

```
int buttonPin[ ] = { 5, 6, 7, 8 };  // vetor pinos das chaves
int buttonState[ ] = { 1, 1, 1, 1}; // vetor teste chaves
int senha[ ] = { 4, 3, 2, 1 };      // vetor com a senha
int index = 0;

void setup() {
  // put your setup code here, to run once:
   int i;
   for (i=0; i<=3; i++){
      pinMode( buttonPin[i], INPUT);
   }
   ...
}
```

senha[1] = 3

Exemplo:
Captura de Senha

```
#include "pitches.h"
//                  0  1  2  3
int buttonPin[ ] = { 7, 8, 9, 10 };  // vetor pinos onde estão chaves
int senha[ ]  = { 4, 3, 2, 1 };       // vetor com a senha
int ledPin[ ] = { 6, 5, 4, 3, 2 };    // vetor pinos onde estão leds
const int LedOrange = 4;          // pino do led Laranja
const int speakerPin = 11;        // pino do alto-faltante
const int LIGADO = 1;
const int DESLIGADO = 0;
int alarme = LIGADO;

int cont = 0;                     // controla “velocidade" com que pisca
const int MAX = 10000;      // led laranja (alarme ativado)
int StatusLedOrange = 0;

const int noteDurationDefault = 20;   // play notes for 20 ms
int notes[ ] = { NOTE_A4, NOTE_B4, NOTE_C3 };
int warning[ ] = { NOTE_A4, NOTE_D4, NOTE_G4, NOTE_D4, NOTE_A4,
  NOTE_D4};

// notes in the melody
// Ref.: https://www.arduino.cc/en/Tutorial/toneMelody
int melody[ ] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3,
  NOTE_C4
};
// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[ ] = {  4, 8, 8, 4, 4, 4, 4, 4 };

int buttonState = 1;
int index = 0;              // controla posição do caracter da senha
```

```
void setup() {
  // put your setup code here, to run once:
  int i;
  // configurando Leds
  for ( i=0; i <= 4; i++ ) {
    pinMode ( ledPin[ i ], OUTPUT );
    digitalWrite ( ledPin[ i ], HIGH ); // também liga os leds
  }
  // configurando chaves
  for ( i=0; i <= 3; i++ ) {
    pinMode ( buttonPin[ i ], INPUT);
  }
  pinMode(speakerPin, OUTPUT);
  for ( i=0; i < 3; i++ ) {
    tone(speakerPin, notes[ i ], noteDurationDefault);
    delay(noteDurationDefault);
  }
  // apagando Leds
  for (i = 4; i >= 0; i--){
    digitalWrite (ledPin[ i ], LOW);
  }
  noTone(speakerPin);
  alarme = LIGADO;
  digitalWrite ( ledPin[LedOrange], HIGH);
  cont = 0; // contador para controlar como led laranja pisca
  // indicativo de alarme ativado
}
```

Continua com bloco loop() próximo slide ➡

Este programa ocupa: 4504 bytes (max = 32.256 bytes).
Suas variáveis globais ocupam 112 bytes de 2028 bytes.

Exemplo:

```
void loop() {
  // put your main code here, to run repeatedly:
  int i, j, t, buttonPressed;

  if ( alarme == LIGADO ) {

    if ( cont < MAX ) {
      cont++;
    }
    else{
      // togle logic level of led laranja
      StatusLedOrange = ! ( StatusLedOrange );
      cont = 0;
    }

   digitalWrite ( ledPin[ LedOrange ], StatusLedOrange);
   // falta testa sensor

   // inicia leitura da senha
   buttonPressed = 0;    // nada apertado
   for ( i=0; i <= 3; i++ ) {    // testa as 4 teclas
      buttonState = digitalRead(buttonPin[ i ]);
      if ( buttonState == LOW ) {
        // Só segue adiante no momento que usuário "soltar" tecla
        // necessário para evitar "auto-repetição" indevida
        do {
          buttonState = digitalRead(buttonPin[ i ]);
        } while ( buttonState == LOW );
        buttonPressed = i + 1; // descobre No. botão apertado
        if ( buttonPressed == senha[ index ] ) {
          // usuário acertou caracter da senha, prepara para próximo
          // caracter da senha
          digitalWrite (ledPin[ index ], HIGH); // acende led do caracter i+1 da senha
          index = index + 1;  // prepara para próximo caracter da senha
        }
        else{
```

```
        else {
          // usuário errou caracter da senha
          index = 0;  // prepara p/ ler senha do início novamente
          cont = 0;
          // Falta “bloquear" teclado
          // Dispara um alerta
          for ( j=0; j < 6; j++ ) {
            for ( t=0; t <= 3; t++ ){
              digitalWrite ( ledPin[ t ], HIGH );
            } // fim do for para acender os 4 leds
            tone( speakerPin, warning[ j ], noteDurationDefault );
            delay( noteDurationDefault );
            for ( t = 0; t <= 3; t++ ) {
              digitalWrite ( ledPin[ t ], LOW );
            } // fim for apagar os 4 leds
            delay( noteDurationDefault );
          } // fim do for para picar leds 5 vezes - senha incorreta !
          noTone( speakerPin );
        } // fim else teste caracter da senha
      } // fim if tecla pressionada
    } // fim do for varredura teclado
```

Continua no próximo slide ➡

Suas variáveis globais ocupam 112 bytes de 2028 bytes.

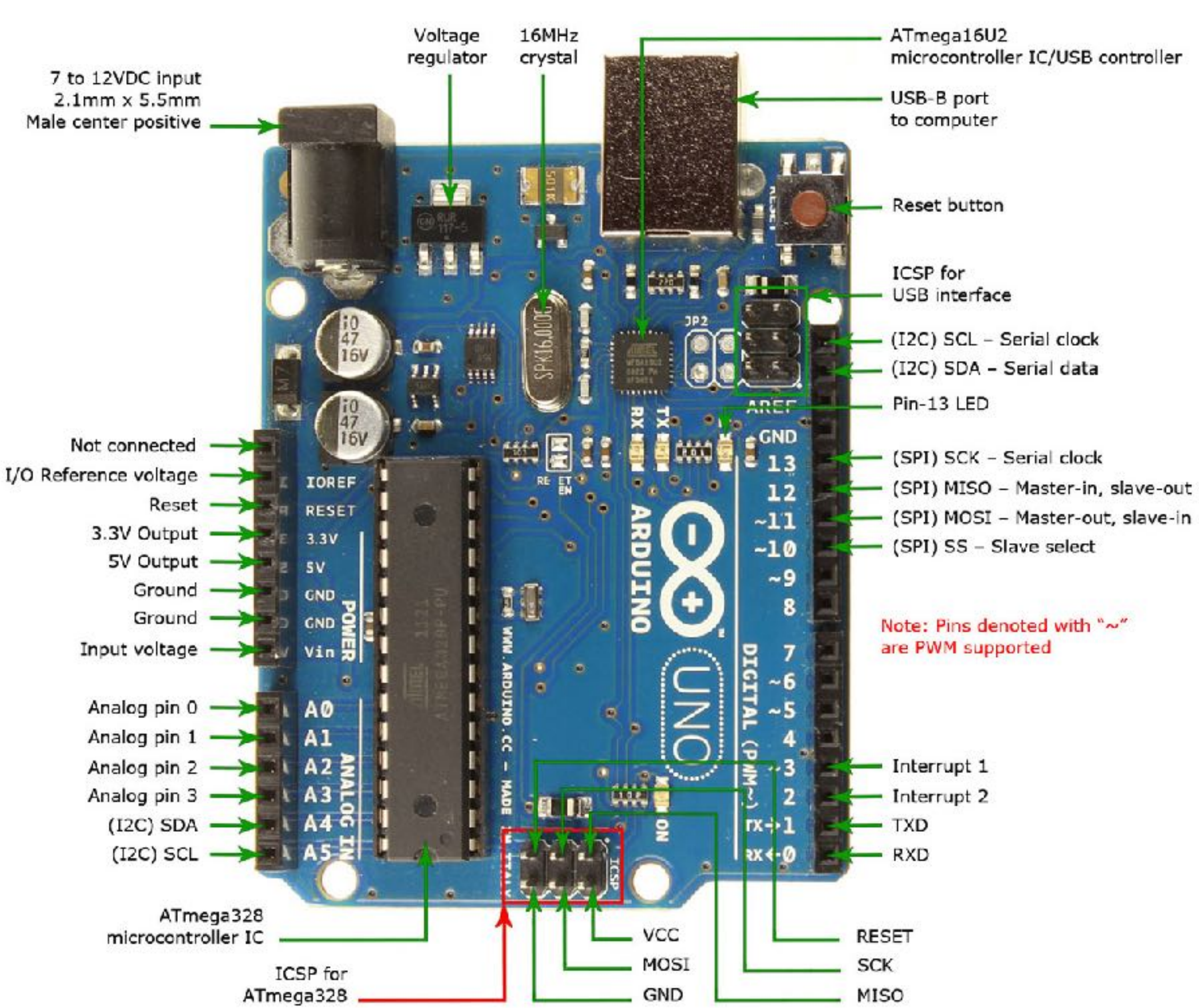
Exemplo:
Captura de Senha (Cont.)

```
// Se index alcançou 4 é porque usuário acertou a senha
if ( index >= 4 ) {
  // leu corretamente toda senha, desbloquear alarme
  alarme = DESLIGADO;
  digitalWrite ( ledPin[ LedOrange ], LOW); // apaga led Status alarme
  // toca melodia atestando desbloqueio alarme
  // Ref.: https://www.arduino.cc/en/Tutorial/toneMelody
  for ( int thisNote = 0; thisNote < 8; thisNote++ ) {
    // calculate the note duration, take one second divided by the note type
    // e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000 / noteDurations[ thisNote ];
    tone( speakerPin, melody[ thisNote ], noteDuration );

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay( pauseBetweenNotes );
    noTone( speakerPin );  // stop the tone playing
  } // fim do for que toca melodia acerto da senha
  // apaga os leds que acenderam conforme caracter da senha
  for ( t = 0; t <= 3; t++ ) {
    digitalWrite ( ledPin[ t ], LOW );
  }
  index = 0;  // já deixa pronto para ler senha do início novamente
  cont = 0;
  delay( 1000 );  // espera 1 segundo até continuar programa...
 } // fim do if da senha digitada corretamente
} // fim do if alarme LIGADO
```

```
else { // caso do alarme desligado
  // Segue o if para o caso de alarme desligado
  // Neste caso, espera usuário apertar qualquer tecla para
  // reativar alarme
  buttonPressed = 0;  // nenhuma tecla pressionada
  for ( i = 0; i <= 3; i++ ) {   // testa as 4 teclas
    buttonState = digitalRead( buttonPin[ i ] );
    if ( buttonState == LOW ) {
      buttonPressed = i + 1; // descobre No. botão apertado
      // Só segue adiante quando usuário "soltar" tecla
      // necessário para evitar "auto-repetição" indevida
      do {
        buttonState = digitalRead( buttonPin[ i ] );
      } while ( buttonState == LOW );
      tone( speakerPin, NOTE_C4, noteDurationDefault );
      noTone( speakerPin );
    } // fim do if verifica alguma tecla pressionada
  } // fim do for varredura teclado
  if ( buttonPressed > 0 ) {
    alarme = LIGADO;
    // começar a piscar led laranja para avisar de alarme
    // re-ativado, durante aprox. 10 segundos...
    for ( i = 0; i < 20; i++ ) {
      digitalWrite ( ledPin[ LedOrange ], HIGH );
      delay( 250 );
      tone( speakerPin, NOTE_C5, noteDurationDefault );
      digitalWrite ( ledPin[ LedOrange ], LOW);
      delay( 250 );
    } // fim do for avisando alarme re-ativado
    noTone( speakerPin );
  } // fim do if algum botão ativado
 } // fim do else alarme ligado
 // volta ao loop comum do Arduino
} // fim do loop()
```

Este programa ocupa: 4504 bytes (max = 32.256 bytes).
Suas variáveis globais ocupam 112 bytes de 2028 bytes.

fim

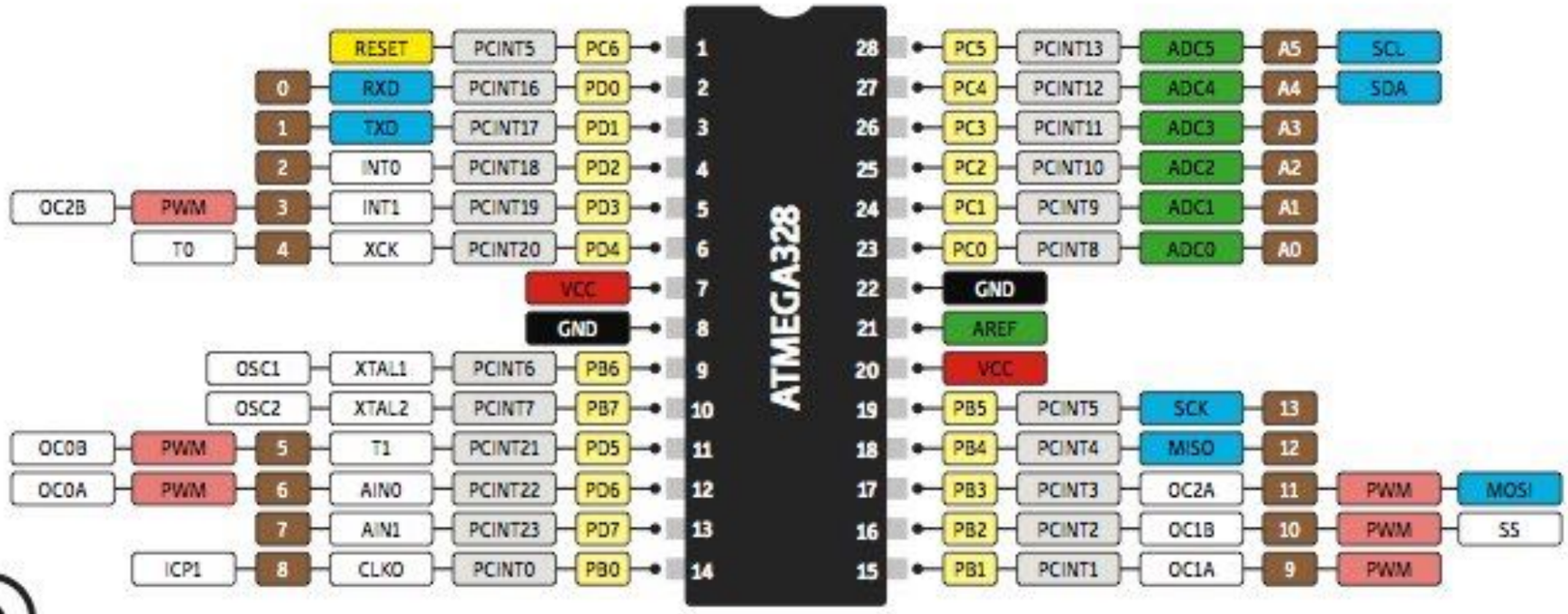

Voltage regulator
16MHz crystal
ATmega16U2 microcontroller IC/USB controller
7 to 12VDC input
2.1mm x 5.5mm
Male center positive
USB-B port to computer
Reset button
ICSP for USB interface
(I2C) SCL – Serial clock
(I2C) SDA – Serial data
Pin-13 LED
Not connected
I/O Reference voltage
Reset
3.3V Output
5V Output
Ground
Ground
Input voltage
(SPI) SCK – Serial clock
(SPI) MISO – Master-in, slave-out
(SPI) MOSI – Master-out, slave-in
(SPI) SS – Slave select
Note: Pins denoted with "~" are PWM supported
Analog pin 0
Analog pin 1
Analog pin 2
Analog pin 3
(I2C) SDA
(I2C) SCL
Interrupt 1
Interrupt 2
TXD
RXD
ATmega328 microcontroller IC
ICSP for ATmega328
VCC
MOSI
GND
RESET
SCK
MISO

THE UNOFFICIAL ARDUINO UNO PINOUT DIAGRAM

LEGEND

- GND
- POWER
- CONTROL
- PHYSICAL PIN
- PORT PIN
- ATMEGA328 PIN FUNC
- DIGITAL PIN
- ANALOG-RELATED PIN
- PWM PIN
- SERIAL PIN
- ARDUINO PIN

Source Total 150mA
Source Total 150mA
General Information
Pay Attention
No Really **PAY ATTENTION**
LED



GND
VIN 2.1 m
7-12v depending on current drawn

POWER BARREL JACK
USB JACK

Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package

Reserved

R3 Only
IOREF
RESET
3V3
5V
GND
GND
Vin

ATMEGA328

A0 A1 A2 A3 A4 A5

Sink Total 100mA
PCINT8 ADC0 PC0 23
PCINT9 ADC1 PC1 24
PCINT10 ADC2 PC2 25
PCINT11 ADC3 PC3 26
PCINT12 ADC4 PC4 27
PCINT13 ADC5 PC5 28
SDA
SCL

ICSP
RESET
SCK
MISO
5V
MOSI
GND

AREF
GND
13 12 11 10 9 8 7 6 5 4 3 2 TX 1 RX 0

SCL
SDA
AREF
R3 Only
GND

19 PB5 PCINT5 SCK MISO MOSI L
18 PB4 PCINT4
17 PB3 OC2A PCINT3 PWM
16 PB2 OC1B PCINT2 PWM SS
15 PB1 OC1A PCINT1 PWM
14 PB0 CLKO PCINT0 ICP1

Sink Total 100mA
13 PD7 AIN1 PCINT23
12 PD6 AIN0 PCINT22 PWM OC0A
11 PD5 T1 PCINT21 PWM
6 PD4 T0 PCINT20 XCK
5 PD3 INT1 PCINT19 PWM OC2B
4 PD2 INT0 PCINT18
3 PD1 TXD PCINT17 TX
2 PD0 RXD PCINT16 RX
Sink Total 100mA
Connected to the ATMega and used for USB program and communicating with it

www.pighixxx.com
29 JAN 2013

fritzing.org

arduino how us... | Interface a Sev... | 7 Segment Dis... | Seven Segmen... | How to Set up... | Fritzing Down...

fritzing

electronics made easy

Projects Parts **Download** Learning Services Contribute FORUM FAB

Fritzing is open source, free software. Be aware that the development of it depends on the active support of the community.
Select the download for your platform below.

Version 0.9.3b was released on **June 2, 2016.**

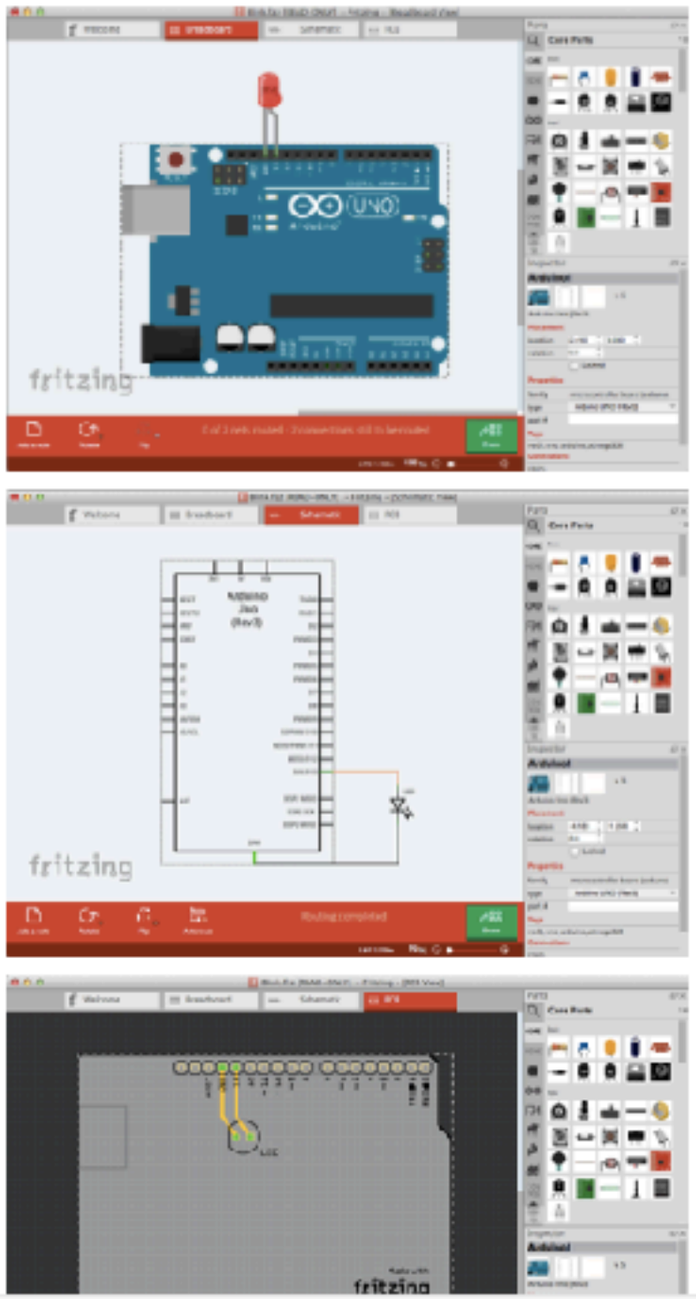
- **Windows** 32 bit
- **Windows** 64 bit
- **Mac OS X** 10.7 and up
- **Linux** 32 bit
- **Linux** 64 bit
- **Source** Github

Downloaded 3026224 times.

See what's new and the known issues.

Read the installation instructions below.

This version includes translations for:
Deutsch (German), **English**, **Español** (Spanish), **Français** (French), **Italiano** (Italian), **Nederlands** (Dutch), **Português (eu)** (Portuguese EU), **Português (br)** (Portuguese BR), **日本語** (Japanese), **中文 (简体)** (Chinese Simplified), **正體中文 (繁體)** (Chinese



Arduino Cheat Sheet V.02c

Mostly taken from the extended reference:
http://arduino.cc/en/Reference/Extended
Gavin Smith – Robots and Dinosaurs, The Sydney Hackspace

Structure
void **setup**() void **loop**()

Control Structures
```
if (x<5){ } else { }
switch (myvar) {
  case 1:
    break;
  case 2:
    break;
  default:
}
for (int i=0; i <= 255; i++){ }
while (x<5){ }
do { } while (x<5);
continue; //Go to next in do/for/while loop
return x; // Or 'return;' for voids.
goto      // considered harmful :-)
```

Further Syntax
// (single line comment)
/* (multi-line comment) */
#define DOZEN 12 //Not baker's!
#include <avr/pgmspace.h>

General Operators
= (assignment operator)
+ (addition) - (subtraction)
* (multiplication) / (division)
% (modulo)
== (equal to) != (not equal to)
< (less than) > (greater than)
<= (less than or equal to)
>= (greater than or equal to)
&& (and) || (or) ! (not)

Pointer Access
& reference operator
* dereference operator

Bitwise Operators
& (bitwise and) | (bitwise or)
^ (bitwise xor) ~ (bitwise not)
<< (bitshift left) >> (bitshift right)

Compound Operators
++ (increment) -- (decrement)
+= (compound addition)
-= (compound subtraction)
*= (compound multiplication)
/= (compound division)
&= (compound bitwise and)
|= (compound bitwise or)

Constants
HIGH | LOW
INPUT | OUTPUT
true | false
143 // **Decimal** number
0173 // **Octal** number
0b1101111 //**Binary**
0x7B // **Hex** number
7U // Force unsigned
10**L** // Force long
15**UL** // Force long unsigned
10.0 // Forces floating point
2.4e5 // 240000

Data Types
void
boolean (0, 1, false, true)
char (e.g. 'a' -128 to 127)
unsigned char (0 to 255)
byte (0 to 255)
int (-32,768 to 32,767)
unsigned int (0 to 65535)
word (0 to 65535)
long (-2,147,483,648 to 2,147,483,647)
unsigned long (0 to 4,294,967,295)
float (-3.4028235E+38 to 3.4028235E+38)
double (currently same as float)
sizeof(myint) // returns 2 bytes

Strings
```
char S1[15];
char S2[8]={'a','r','d','u','i','n','o'};
char S3[8]={'a','r','d','u','i','n','o','\0'};
  //Included \0 null termination
char S4[ ] = "arduino";
char S5[8] = "arduino";
char S6[15] = "arduino";
```

Arrays
```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
```

Conversion
char() **byte**()
int() **word**()
long() **float**()

Qualifiers
static // persists between calls
volatile // use RAM (nice for ISR)
const // make read-only
PROGMEM // use flash

Digital I/O
pinMode(pin, [INPUT,OUTPUT])
digitalWrite(pin, value)
int **digitalRead**(pin)
//Write High to inputs to use pull-up res

Analog I/O
analogReference([DEFAULT,INTERNAL,EXTERNAL])
int **analogRead**(pin) //Call twice if switching pins from high Z source
analogWrite(pin, value) // PWM

Advanced I/O
tone(pin, freqhz)
tone(pin, freqhz ,duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin, [MSBFIRST,LSBFIRST], value)
unsigned long **pulseIn**(pin, [HIGH,LOW])

Time
unsigned long **millis**() // 50 days overflow.
unsigned long **micros**() // 70 min overflow
delay(ms)
delayMicroseconds(us)

Math
min(x, y) **max**(x, y) **abs**(x)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
pow(base, exponent) **sqrt**(x)
sin(rad) **cos**(rad) **tan**(rad)

Random Numbers
randomSeed(seed) // Long or int
long **random**(max)
long **random**(min, max)

Bits and Bytes
lowByte() **highByte**()
bitRead(x,bitn) **bitWrite**(x,bitn,bit)
bitSet(x,bitn) **bitClear**(x,bitn)
bit(bitn) //bitn: 0-LSB 7-MSB

External Interrupts
attachInterrupt(interrupt, function, [LOW,CHANGE,RISING,FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()

Libraries:

Serial
begin([300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200])
end()
int **available**()
int **read**()
flush()
print()
println()
write()

EEPROM (#include <EEPROM.h>)
byte **read**(intAddr)
write(intAddr,myByte)

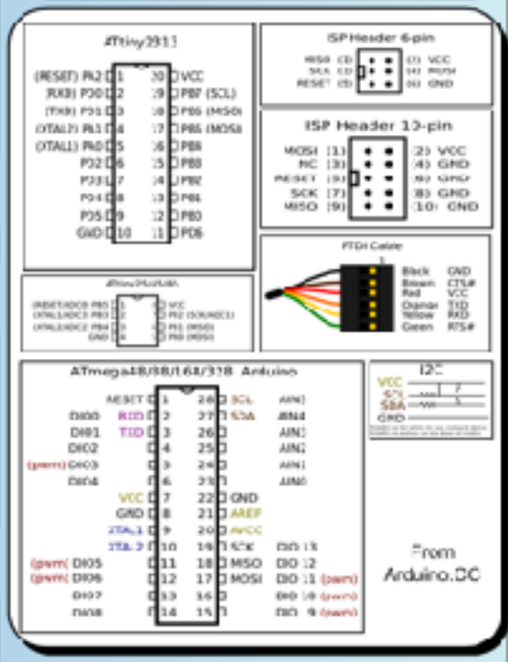
Servo (#include <Servo.h>)
attach(pin , [min_uS, max_uS])
write(angle) // 0-180
writeMicroseconds(uS) //1000-2000, 1500 is midpoint
read() // 0-180
attached() //Returns boolean
detach()

SoftwareSerial(RxPin,TxPin)
// #include<**SoftwareSerial**.h>
begin(longSpeed) // up to 9600
char **read**() // blocks till data
print(myData) or **println**(myData)

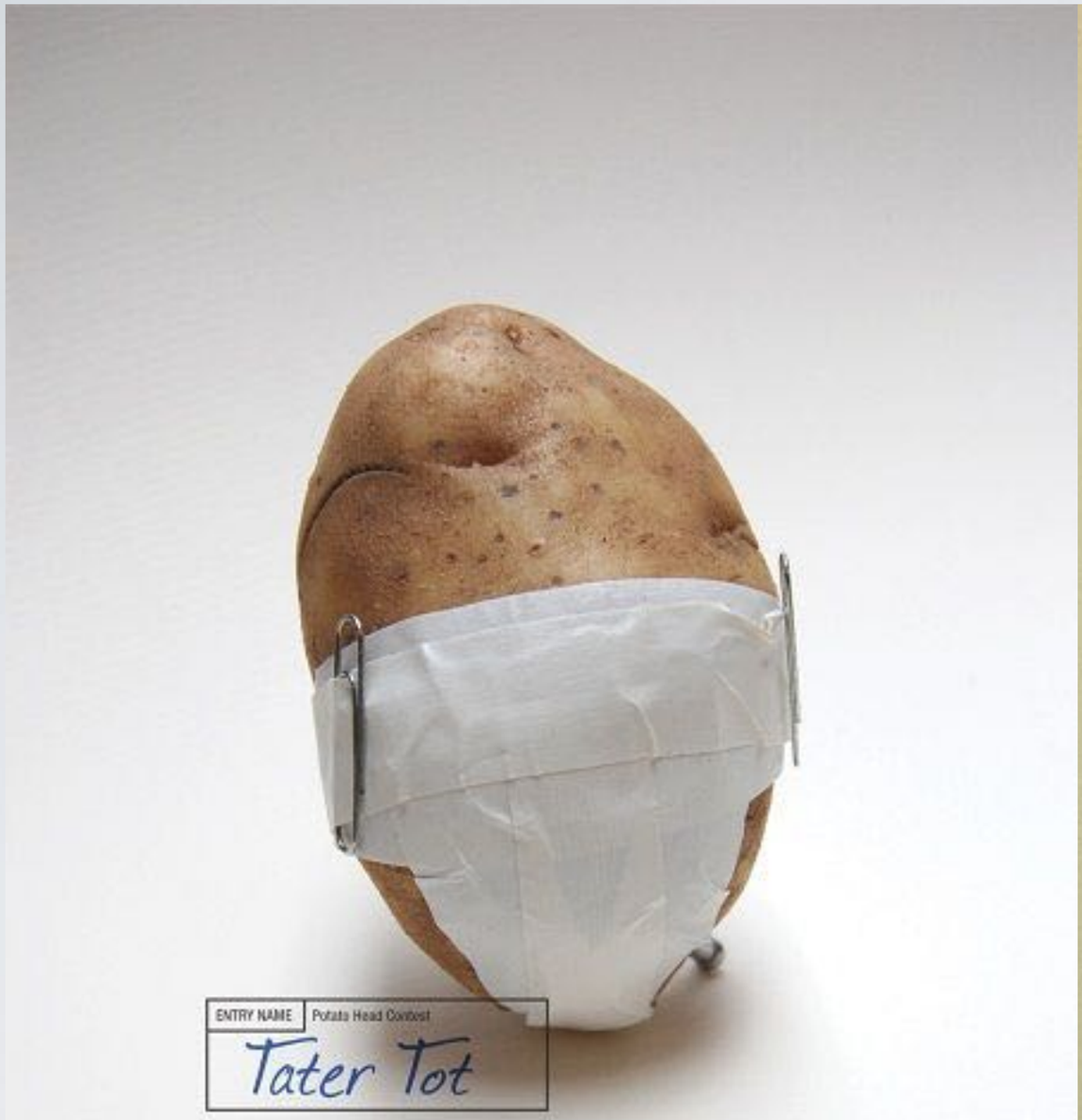
Wire (#include <Wire.h>) // For **I2C**
begin() // Join as master
begin(addr) // Join as slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(mybyte) // Step 2
send(char * mystring)
send(byte * data, size)
endTransmission() // Step 3
byte **available**() // Num of bytes
byte **receive**() //Return next byte
onReceive(handler)
onRequest(handler)

	ATMega168	ATMega328	ATmega1280
Flash (3k for bootloader)	16KB	32KB	128KB
SRAM	1KB	2KB	8KB
EEPROM	512B	1KB	4KB

	Duemilanove/ Nano/ Pro/ ProMini	Mega
# of IO	14 + 6 analog (Nano has 14+8)	54 + 16 analog
Serial Pins	0 - RX 1 - TX	0 - RX1 1 - TX1 19 - RX2 18 - TX2 17 - RX3 16 - TX3 15 - RX4 14 - TX4
Ext.Interrupts	2 - (Int 0) 3 - (Int 1)	2,3,21,20,19,18 (IRQ0- IRQ5)
PWM pins	5,6 - Timer 0 9,10 - Timer 1 3,11 - Timer 2	0-13
SPI	10 - SS 11 - MOSI 12 - MISO 13 - SCK	53 - SS 51 - MOSI 50 - MISO 52 - SCK
I2C	Analog4 - SDA Analog5 - SCK	20 - SDA 21 - SCL



Pics from Fritzing.Org under C.C. license



THINK YOU CAN DO BETTER?

Enter our first annual St. Patrick's Day Mr. Potato Head Contest. March 16 at 3PM. WYSE

HOW TO WORK BETTER.

1 DO ONE THING AT A TIME

2 KNOW THE PROBLEM

3 LEARN TO LISTEN

4 LEARN TO ASK QUESTIONS

5 DISTINGUISH SENSE FROM NONSENSE

6 ACCEPT CHANGE AS INEVITABLE

7 ADMIT MISTAKES

8 SAY IT SIMPLE

9 BE CALM

10 SMILE