

Introdução à Plataforma Arduino



Estrutura do Curso

- Pontos Abordados
 - Plataforma Arduino
 - Prototipagem em protoboard
 - Conceitos básicos de eletrônica e programação em C
 - Montagem de projetos

“Introdução à Plataforma Arduino”, de Saulo Machado Jacques pode ser usado, compartilhado e modificado livremente, desde que citada a fonte, segundo a Licença **Creative Commons - Atribuição 4.0 Internacional**



Material Disponível

Materiais do Curso com Acesso Livre

<https://github.com/smjacques/Horta-Automatizada-ESDI>

The screenshot shows the GitHub repository page for 'smjacques / Horta-Automatizada-ESDI'. The repository has 3 commits, 1 branch, 0 releases, and 1 contributor. The latest commit was 13 minutes ago. The repository contains files for Slides, esquemas, imagens, sketches, and README.md.

This repository

smjacques / Horta-Automatizada-ESDI

3 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

File	Commit	Time
Slides	first commit	13 minutes ago
esquemas	first commit	13 minutes ago
imagens	first commit	13 minutes ago
sketches	first commit	13 minutes ago
README.md	Update README.md	21 minutes ago

A Placa Arduino

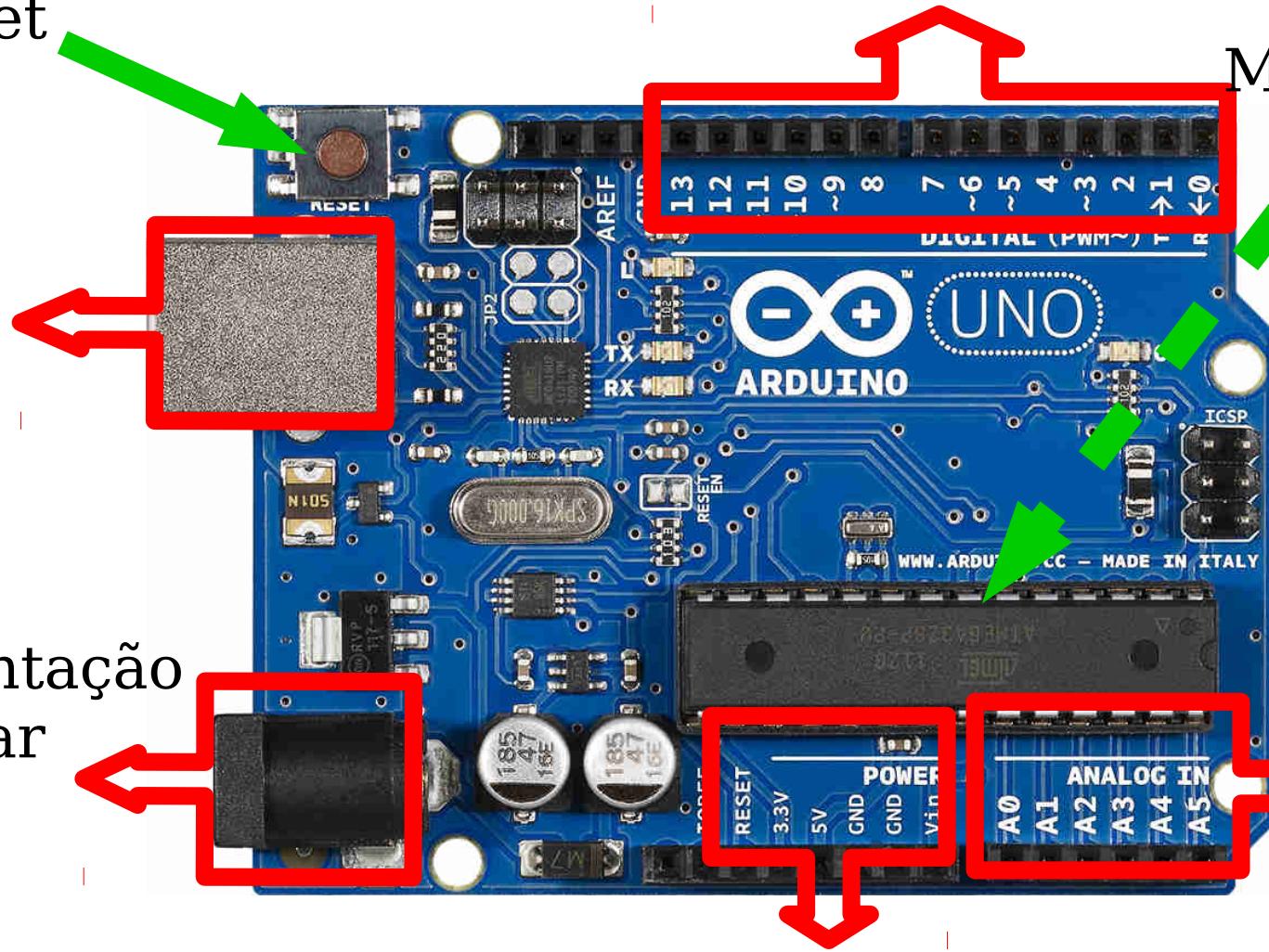
13 pinos digitais (0 V **ou** 5 V)
(INPUT/OUTPUT configurável)

reset

USB

Alimentação
Auxiliar

Alimentação (3.3V e 5V) e
terra (GND = 0V)



Microcontrolador
Atmega328

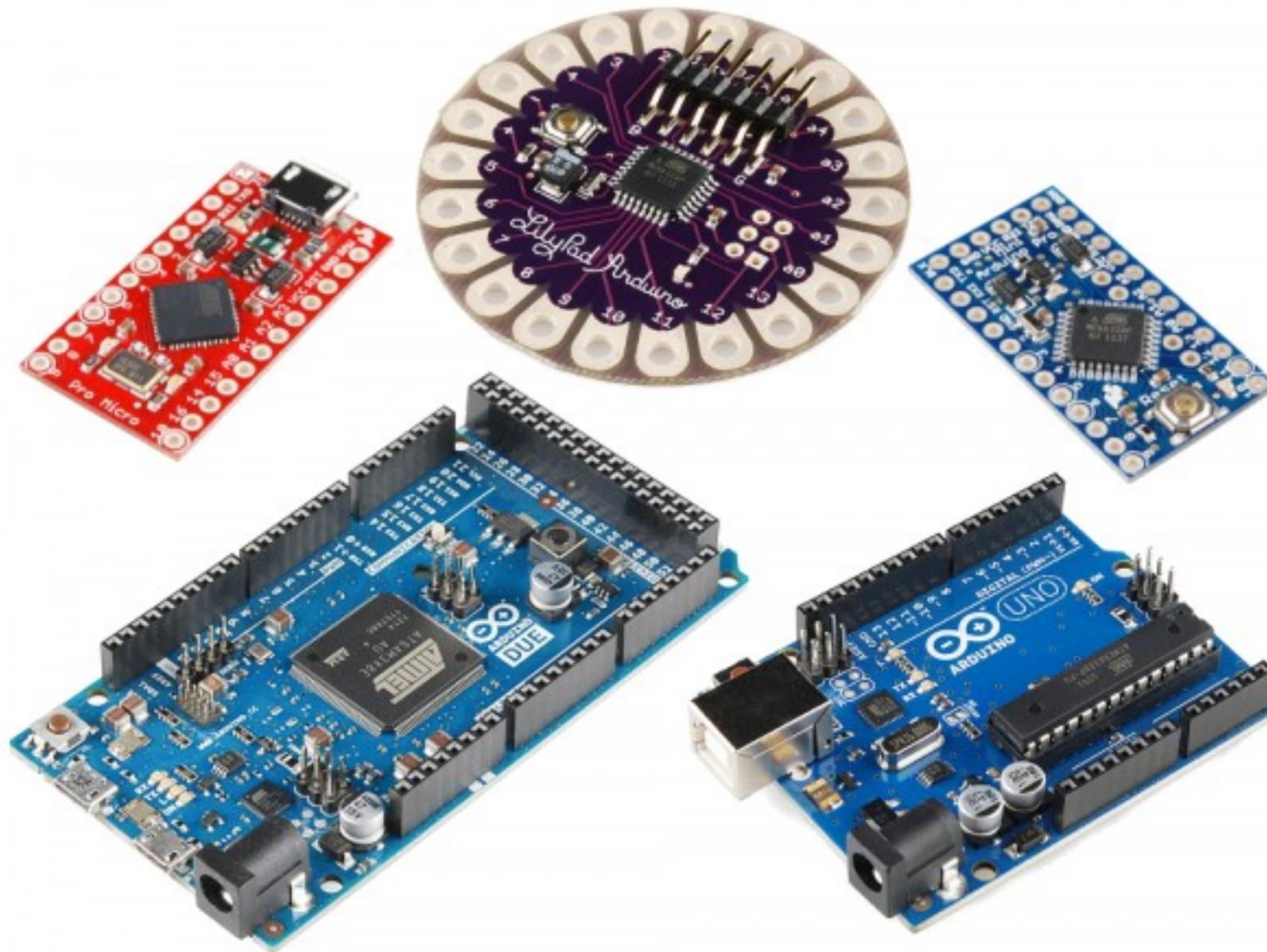
- ✓ Clock: 16 MHz
- ✓ EEPROM: 1 KB (bootloader)
- ✓ SRAM: 2 KB
- ✓ Flash: 32 KB ("pendrive")
- ✓ 5 V - 50 mA

6 pinos
de entrada
análogica
(lê tensões
entre 0 V e 5 V)

Fonte da imagem:

http://en.wikipedia.org/wiki/Arduino#mediaviewer/File:Arduino_Duemilanove_2009b.jpg

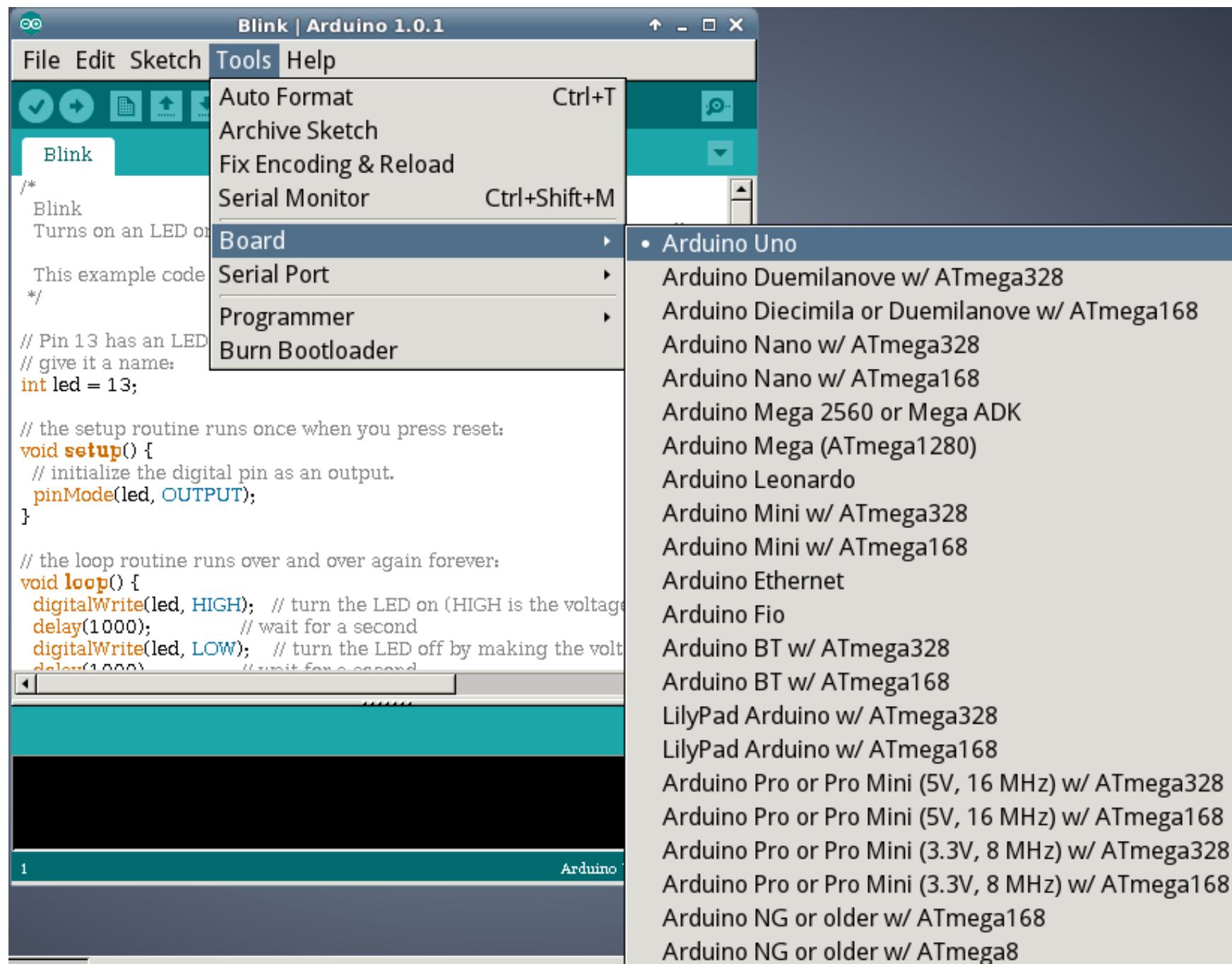
Variantes: Uno, Mega, Mini, Micro, etc.



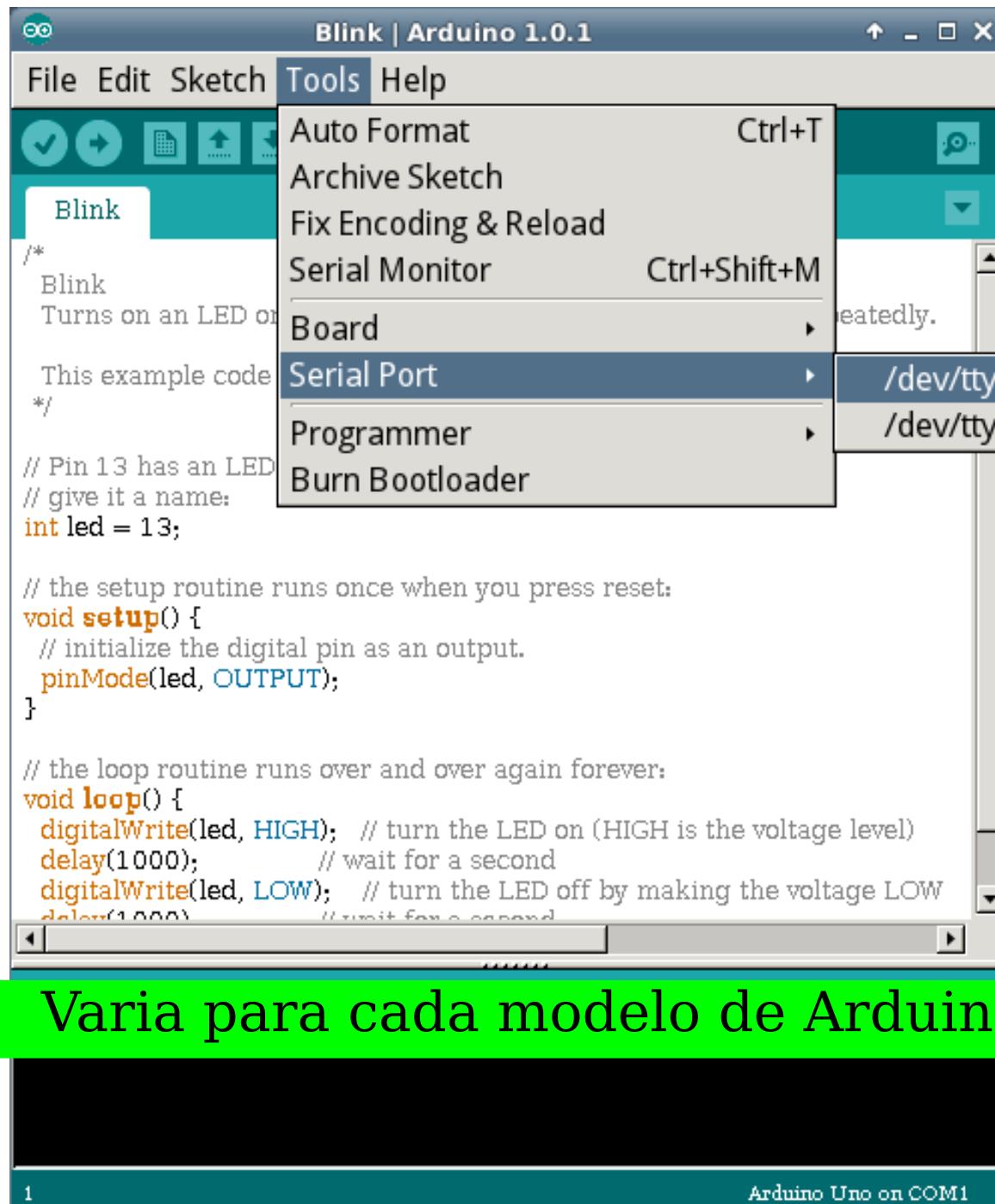
Primeiros Passos

Testes

Selecionando Modelo da Placa



Configurando a Porta Serial



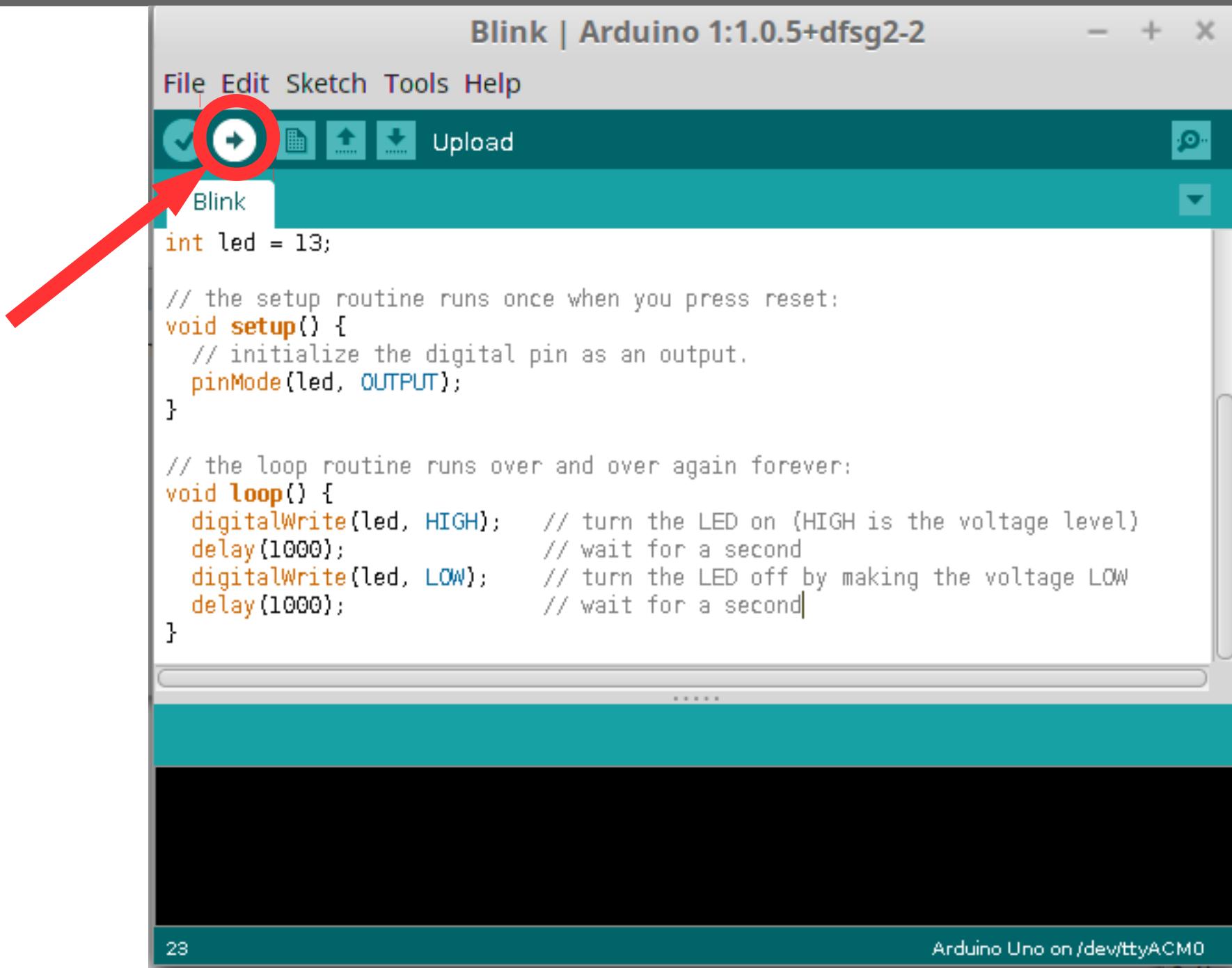
- Varia para cada modelo de Arduino

Testando: Carregue o exemplo Blink



File --> Examples --> Basics --> **Blink**

Fazendo upload para placa



Primeiros Passos: O Código

Estrutura principal

```
void setup()
{
    // executa uma vez ao ligar a placa
}

void loop()
{
    // repete execução enquanto estiver ligada
}
```

- Isso é um sketch Arduino vazio;
- Já está sintaticamente correto: pode compilar;
- Mas a placa nada fará...

Configurando uma saída digital

```
// Pin 13 has an LED connected on most Arduino boards.  
// give it a name:  
int led = 13;  
  
// the setup routine runs once when you press reset:  
void setup(){  
    // initialize the digital pin as an output.  
    pinMode(led, OUTPUT);  
}  
  
void loop()  
{  
    // repete execução quando estiver ligada  
}
```

- (sintaxe) Declarando variáveis:

```
<tipo> <nome>;  
<tipo> <nome> = <valor_inicial>;
```

- (sintaxe) Chamando/executando funções disponíveis:

```
nome_da_função(arg1, arg2, ..., argN);
```

“Ligue, espere, desligue, espere...”

```
// Pin 13 has an LED connected on most Arduino boards.  
// give it a name:  
int led = 13;  
  
// the setup routine runs once when you press reset:  
void setup(){  
    // initialize the digital pin as an output.  
    pinMode(led, OUTPUT);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
    digitalWrite(led, HIGH);    // turn the LED on (HIGH is the voltage level)  
    delay(1000);                // wait for a second  
    digitalWrite(led, LOW);     // turn the LED off by making the voltage LOW  
    delay(1000);                // wait for a second  
}
```

- Funções básicas da biblioteca do Arduino:

pinMode(num_do_pino, INPUT | OUTPUT)

digitalWrite(num_do_pino, LOW | HIGH)

delay(milliseconds)

+info: consulte a documentação!

www.arduino.cc/en/Reference

The screenshot shows the Arduino Reference homepage with a navigation bar at the top. The main content is organized into three columns: Structure, Variables, and Functions.

Structure:

- `setup()`
- `loop()`

Control Structures:

- `if`
- `if...else`
- `for`
- `switch case`
- `while`
- `do... while`
- `break`
- `continue`
- `return`
- `goto`

Further Syntax:

- `;` (semicolon)
- `{}` (curly braces)
- `//` (single line comment)
- `/* */` (multi-line comment)

Variables:

Constants:

- `HIGH | LOW`
- `INPUT | OUTPUT | INPUT_PULLUP`
- `LED_BUILTIN`
- `true | false`
- `integer constants`
- `floating point constants`

Data Types:

- `void`
- `boolean`
- `char`
- `unsigned char`
- `byte`
- `int`
- `unsigned int`
- `word`
- `long`
- `unsigned long`

Functions:

Digital I/O:

- `pinMode()`
- `digitalWrite()`
- `digitalRead()`

Analog I/O:

- `analogReference()`
- `analogRead()`
- `analogWrite() - PWM`

Due only:

- `analogReadResolution()`
- `analogWriteResolution()`

Advanced I/O:

- `tone()`
- `noTone()`
- `shiftOut()`
- `shiftIn()`

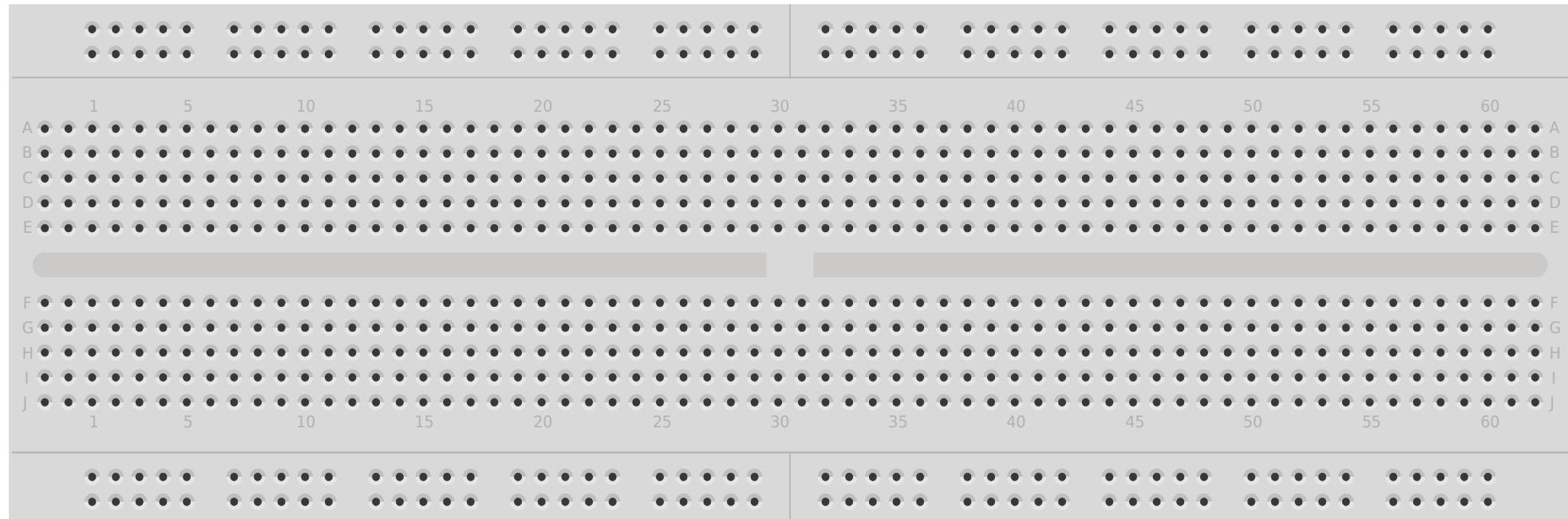
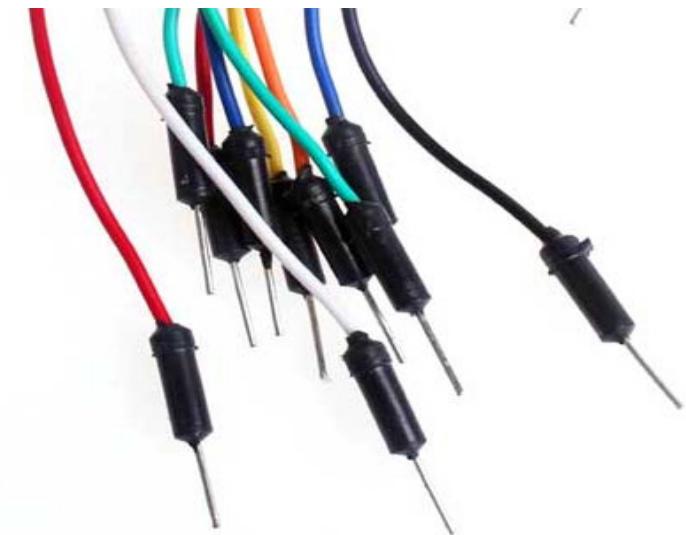


Primeiros Passos: O Código

Prototipagem: protoboard

- Para que serve?
 - Permite conectar componentes eletrônicos sem soldá-los!
 - Para isso são utilizados fios jumper (imagem ao lado);

Fonte da imagem: <http://www.digibay.in/>

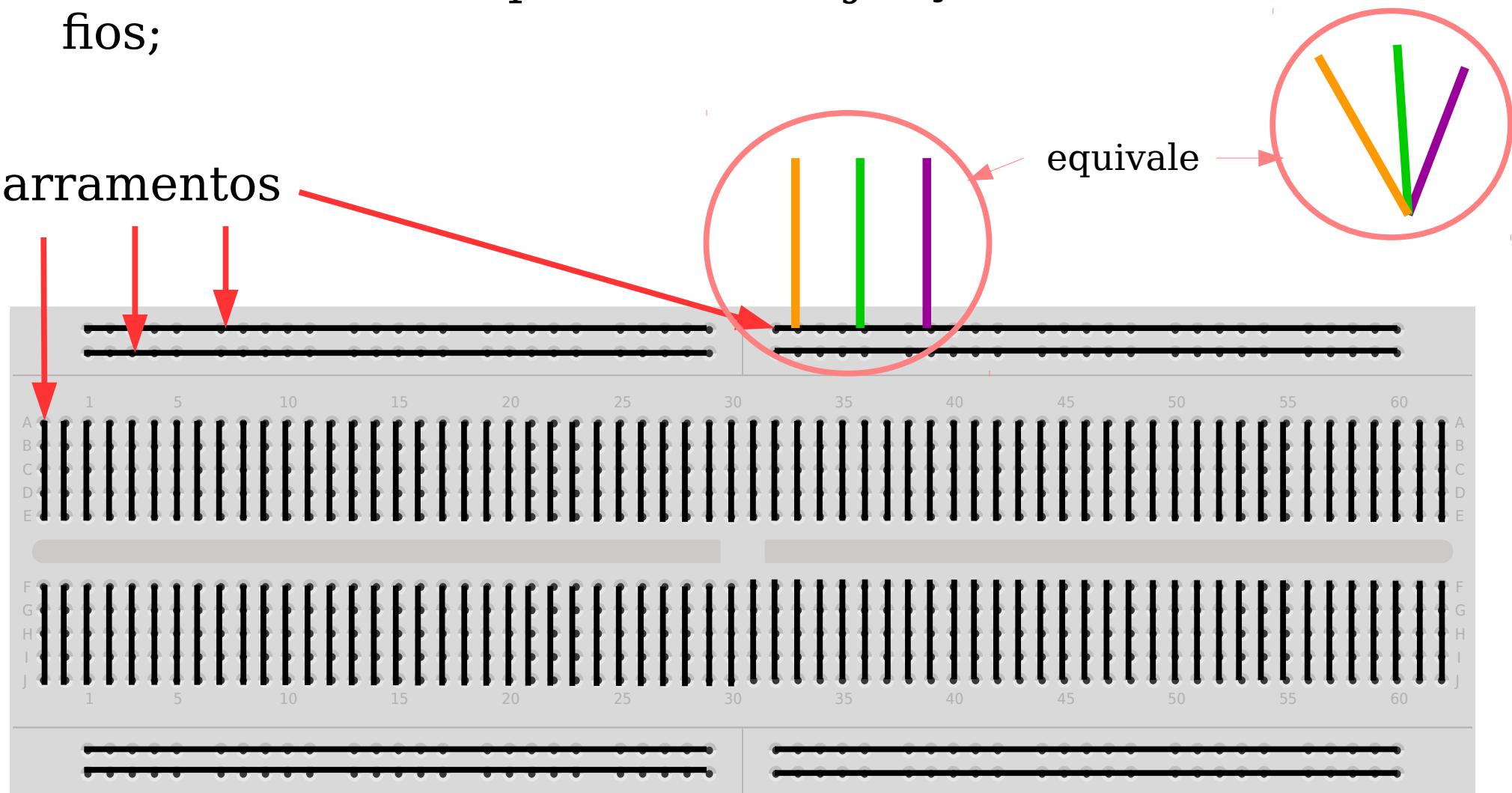


Fonte da imagem:
<http://fritzing.org/home/>

Protoboard: Como funciona?

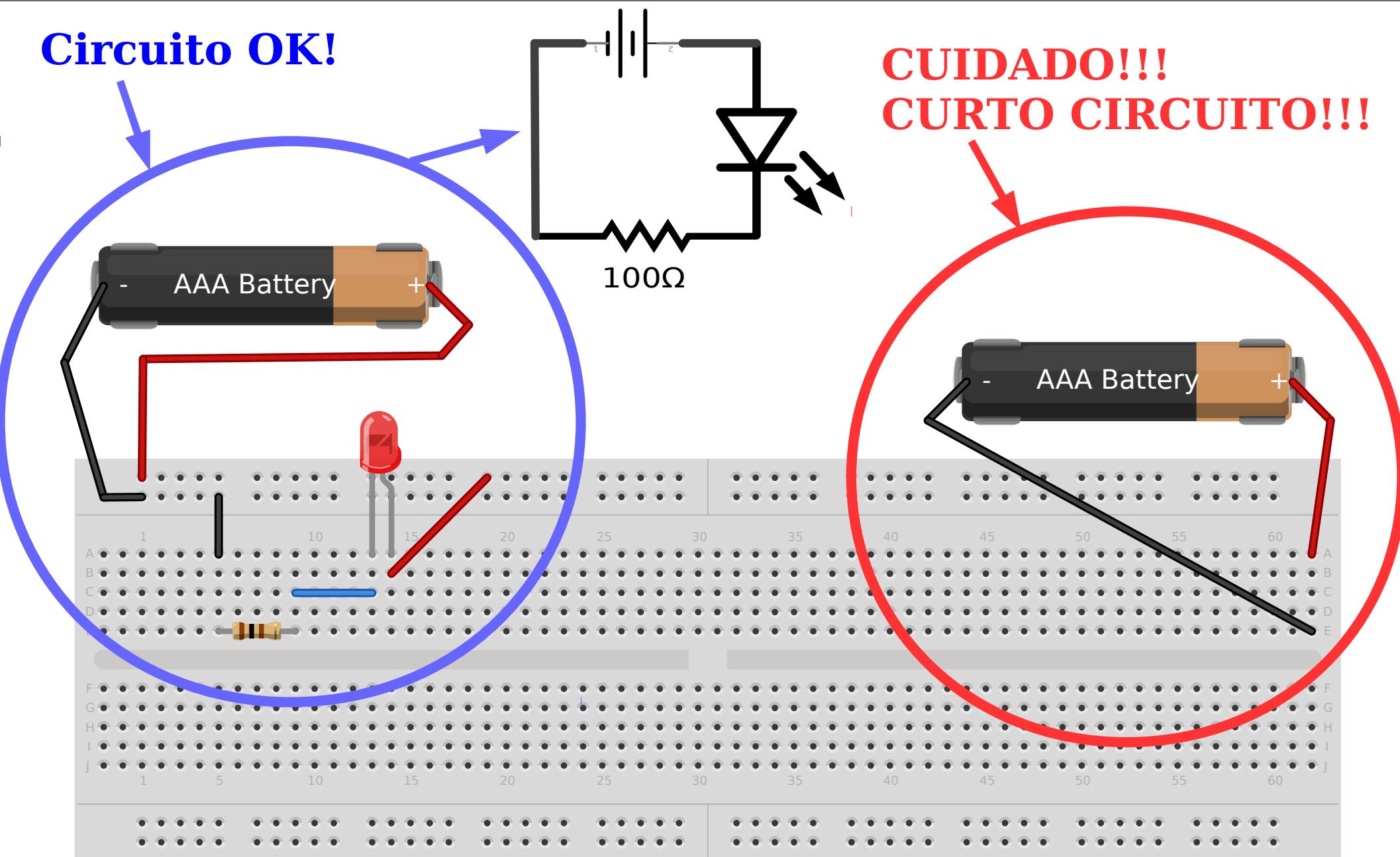
- Consiste num conjunto de barramentos isolados entre si;
- Um barramento equivale à uma junção de dois ou mais fios;

barramentos



Protoboard: Exemplos

Circuito OK!



Projeto 0: Pisca LED externo

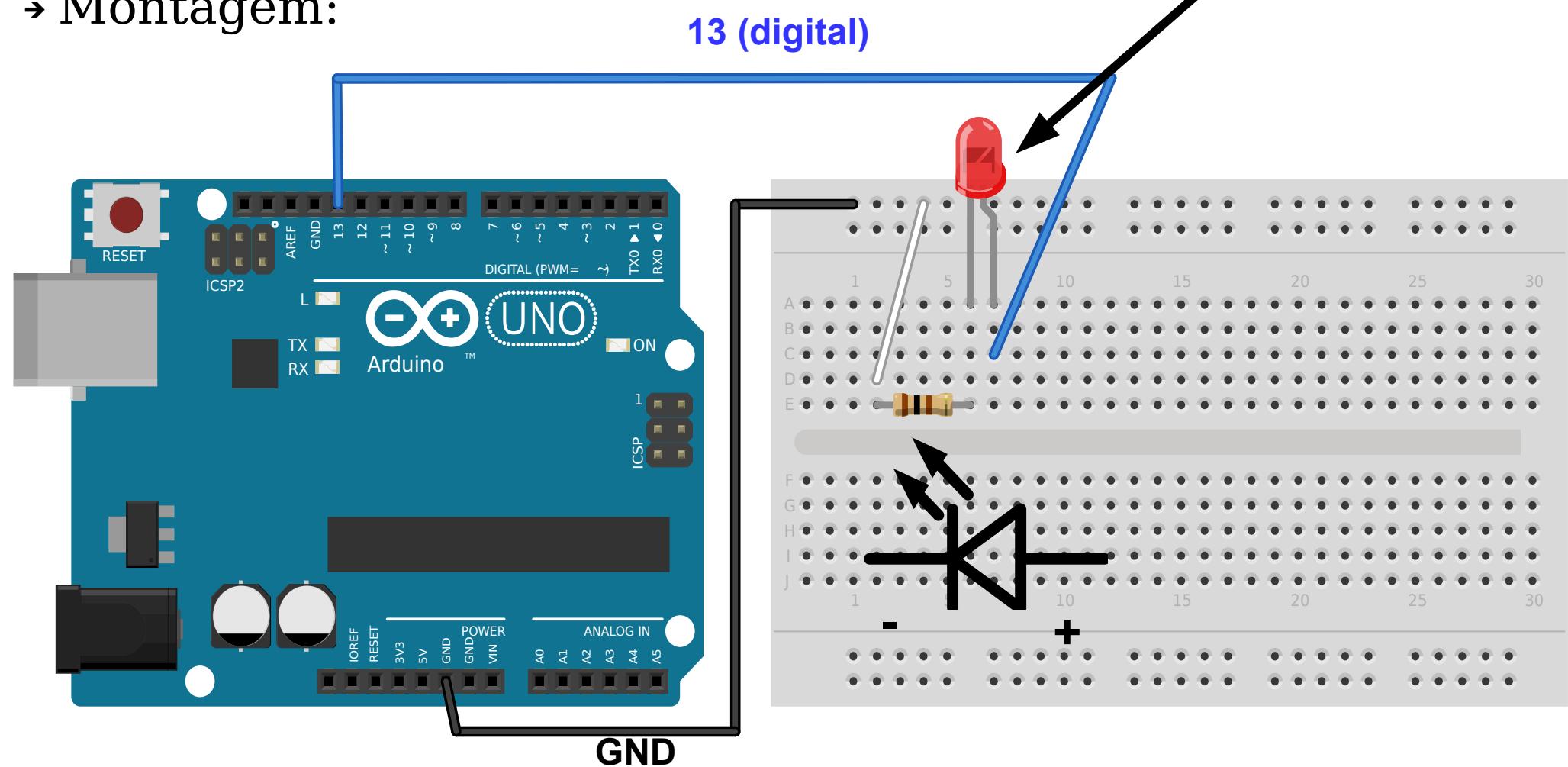
→ Materiais:

- ✓ 1 LED
- ✓ 1 resistor de 100 ohms

Componente polarizado:
perna mais longa deve ser ligada no 5 V (positivo)

→ Montagem:

13 (digital)



Projeto 0: Pisca LED externo

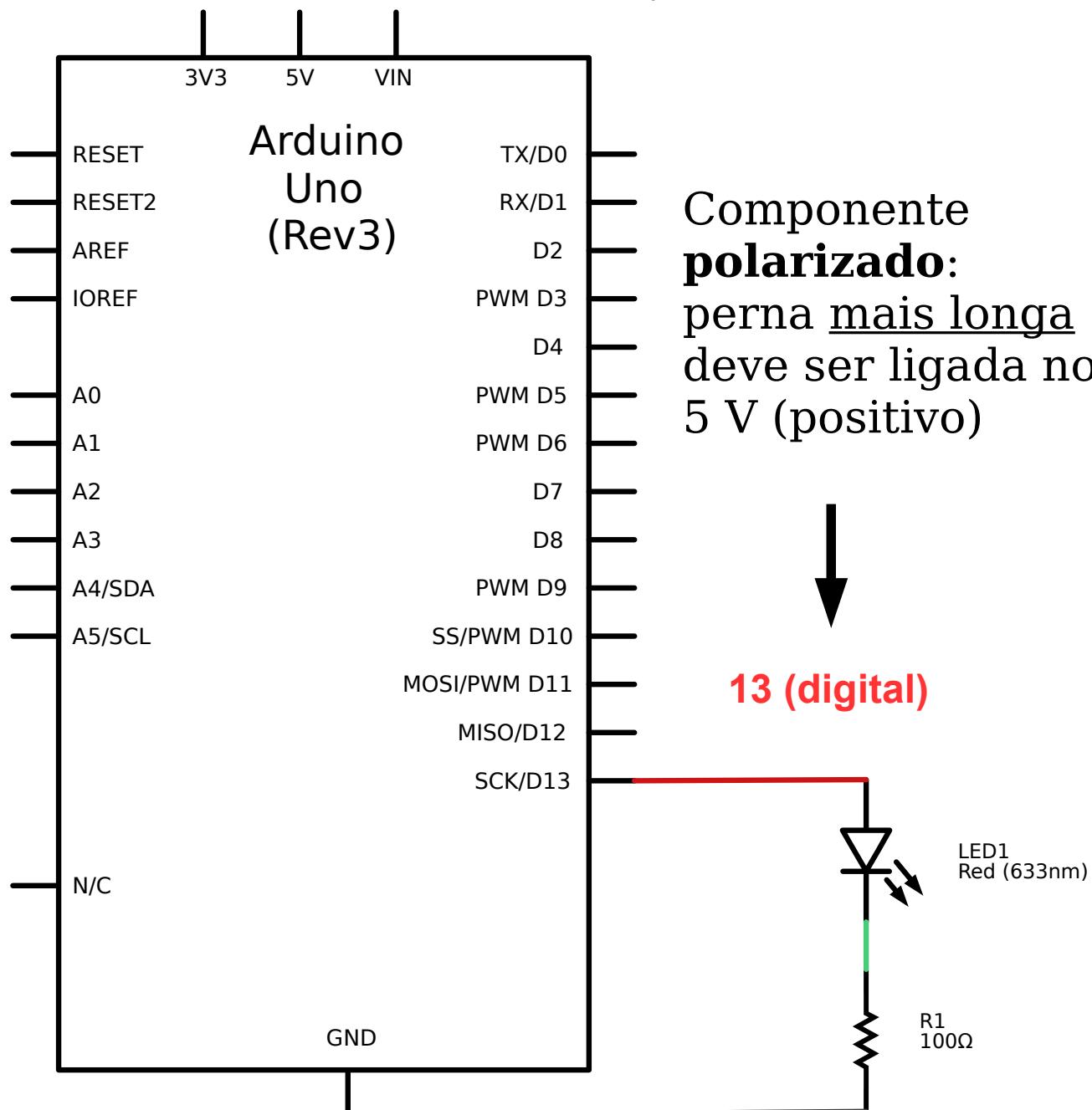
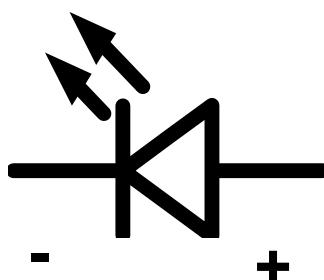
Part1

→ Materiais:

- ✓ 1 LED
- ✓ 1 resistor de 100 ohms
- ✓ fios jumper

→ Montagem:

Light Emitter Diode



Projeto 0: Pisca LED externo

→ Código:

```
#define PINO_LED    13      // pino digital  
  
int pausa = 1000;           // millisegundos  
  
// executada uma vez ao ligar  
void setup()  
{  
    pinMode(PINO_LED, OUTPUT);  
}  
  
// fica executando para sempre  
void loop()  
{  
    digitalWrite(PINO_LED, HIGH);  
    delay(pausa);  
    digitalWrite(PINO_LED, LOW);  
    delay(pausa);  
}
```

comentários
importantes

altere o intervalo de
pausa em um único
lugar!

serão substituídos
por “13” (sem aspas)

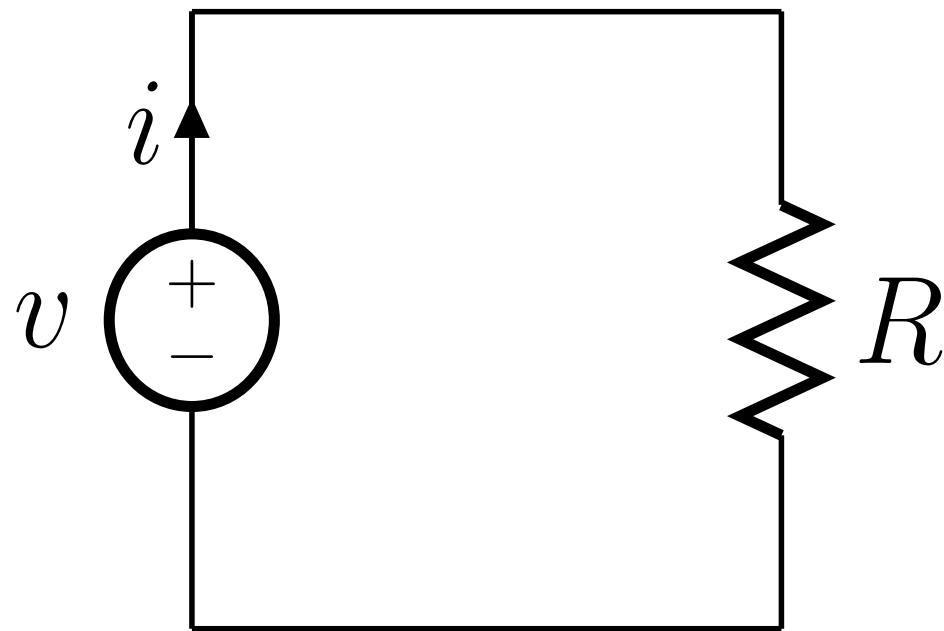
Vantagens de
constantes:

- (sintaxe) Definindo constantes:

```
#define <nome> valor
```

- Não ocupam memória;
- Facilita manutenção;

Circuitos elétricos

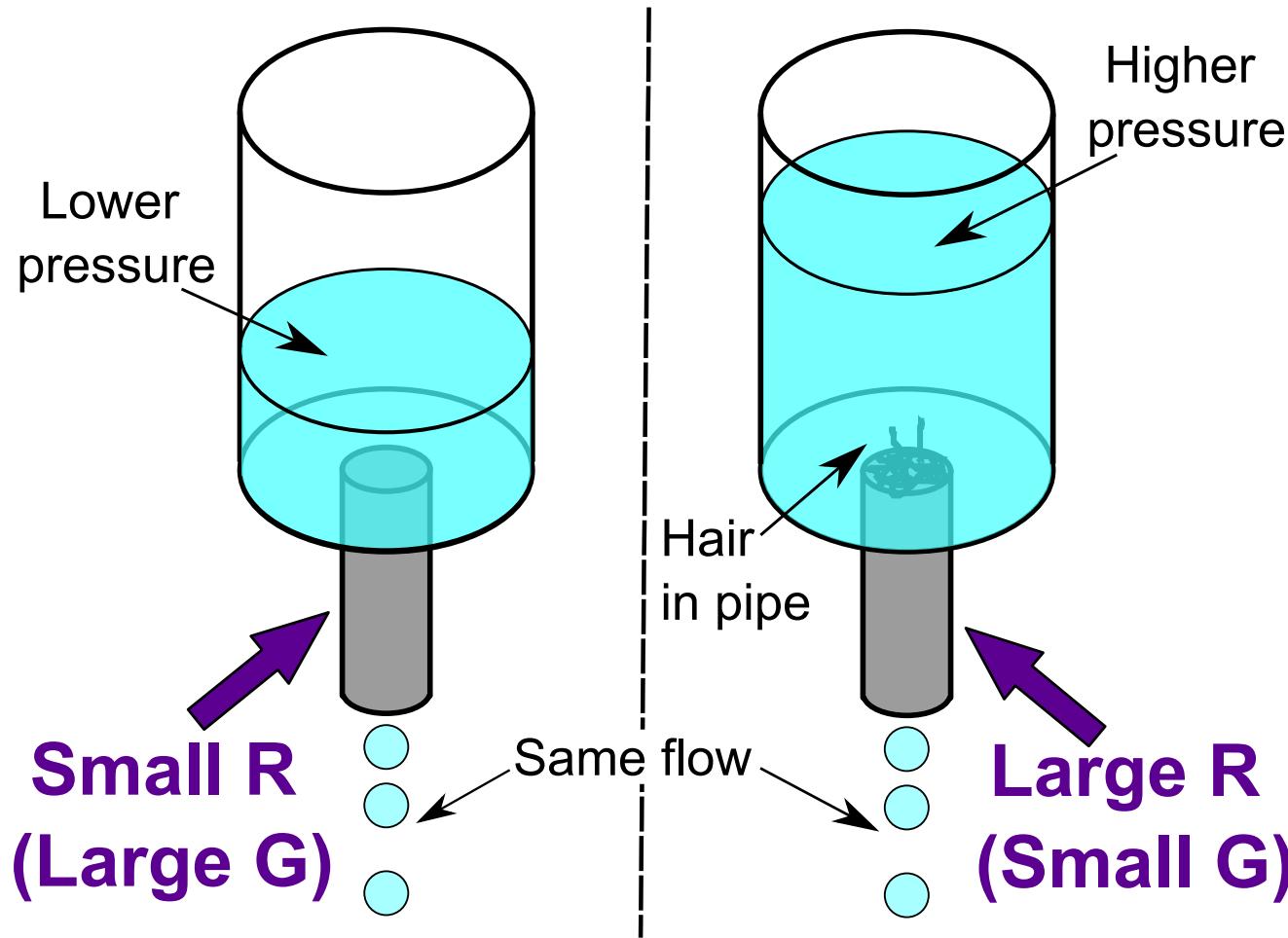


i : Corrente elétrica

V : Tensão elétrica

R : Resistência elétrica

Analogia elétrico-hidráulica



- A pressão é análoga à força eletromotriz
- O escoamento é análogo à corrente elétrica
- A obstrução do cano é análoga à resistência elétrica

Fonte da imagem:

<https://commons.wikimedia.org/wiki/File:ResistanceHydraulicAnalogy.svg>

Referência Arduino: analogRead()

analogRead: Lê o valor do pino analógico especificado. A placa Arduino contém um conversor analógico-digital; com isso, ela pode mapear tensões elétricas de entrada de entre 0 e 5 volts para valores inteiros entre 0 e 1023.

Sintaxe:

`analogRead(pino)`

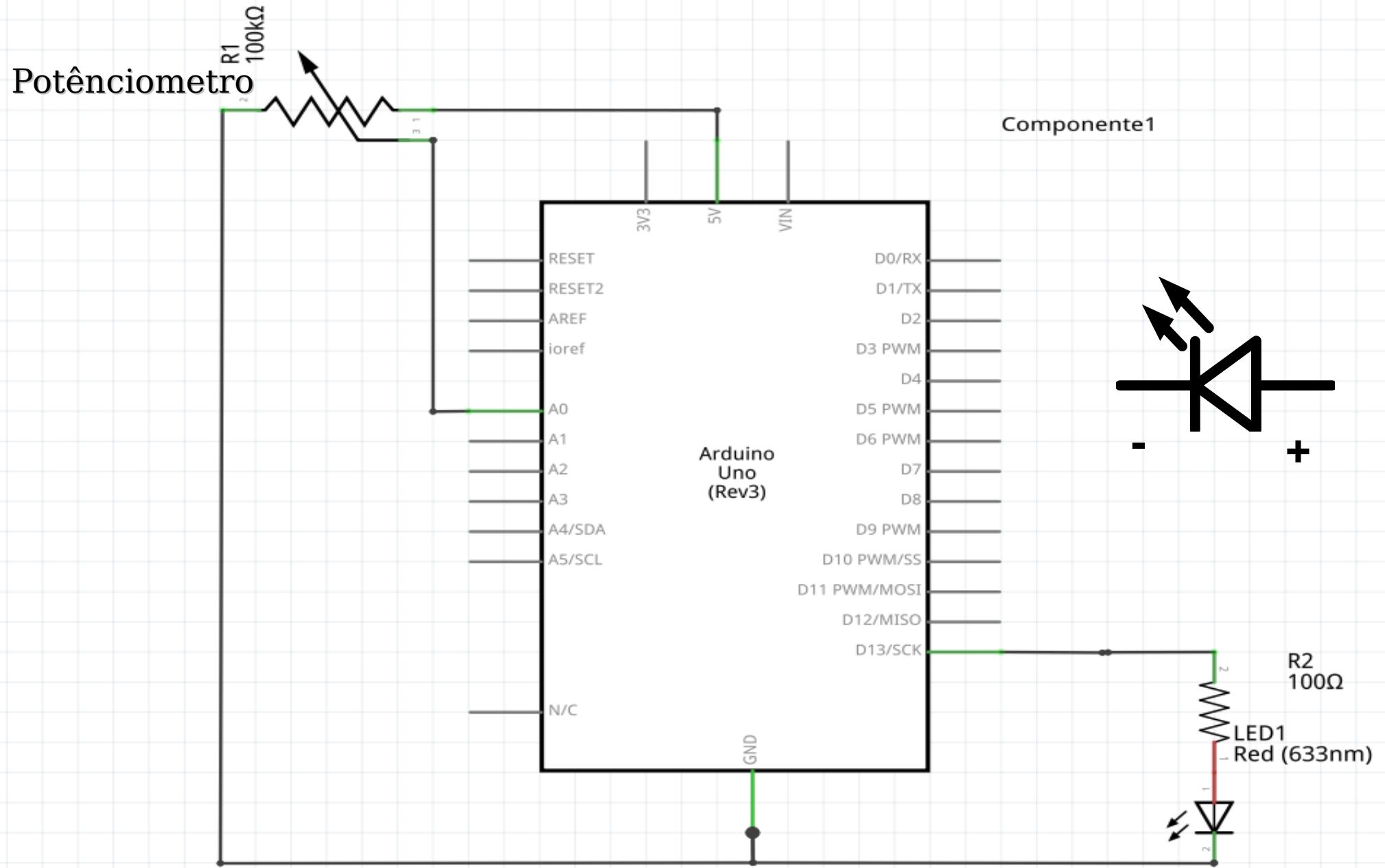
Parâmetros:

- `pino`: o número do pino.

Retorno:

- `int (0 a 1023)`

Projeto 1: Controle Pisca LED

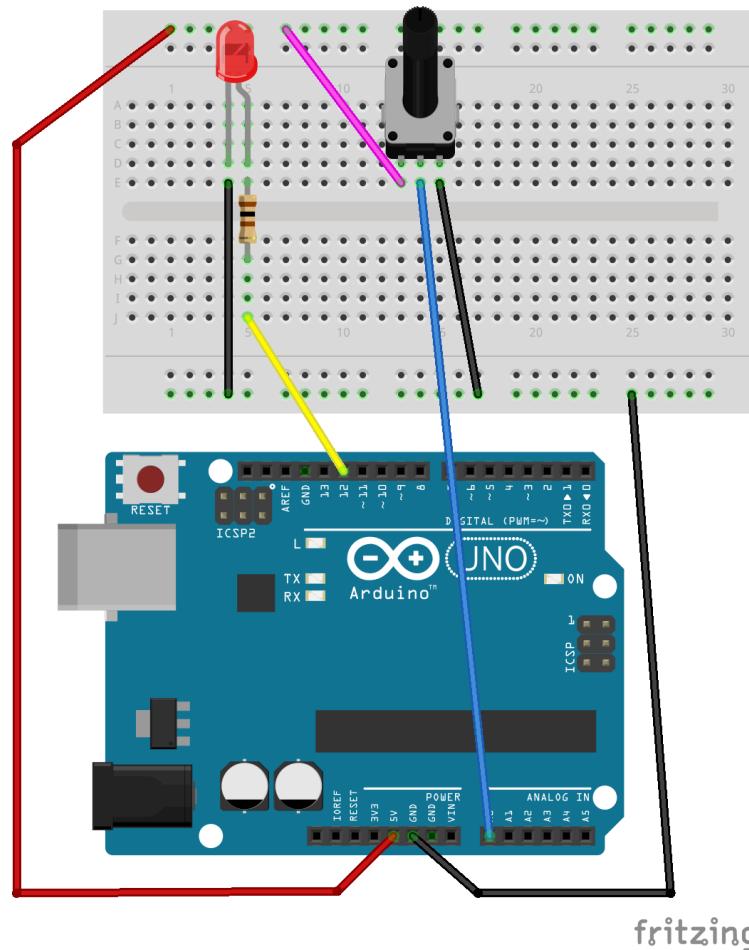


Projeto 1: Controle Pisca LED

→ Materiais:

- ✓ 1 Potenciômetro (resistor variável)

→ Montagem:



→ Código:

```
#define PINO_LED      12      // pino digital
#define PINO_POT       0       // pino analogico

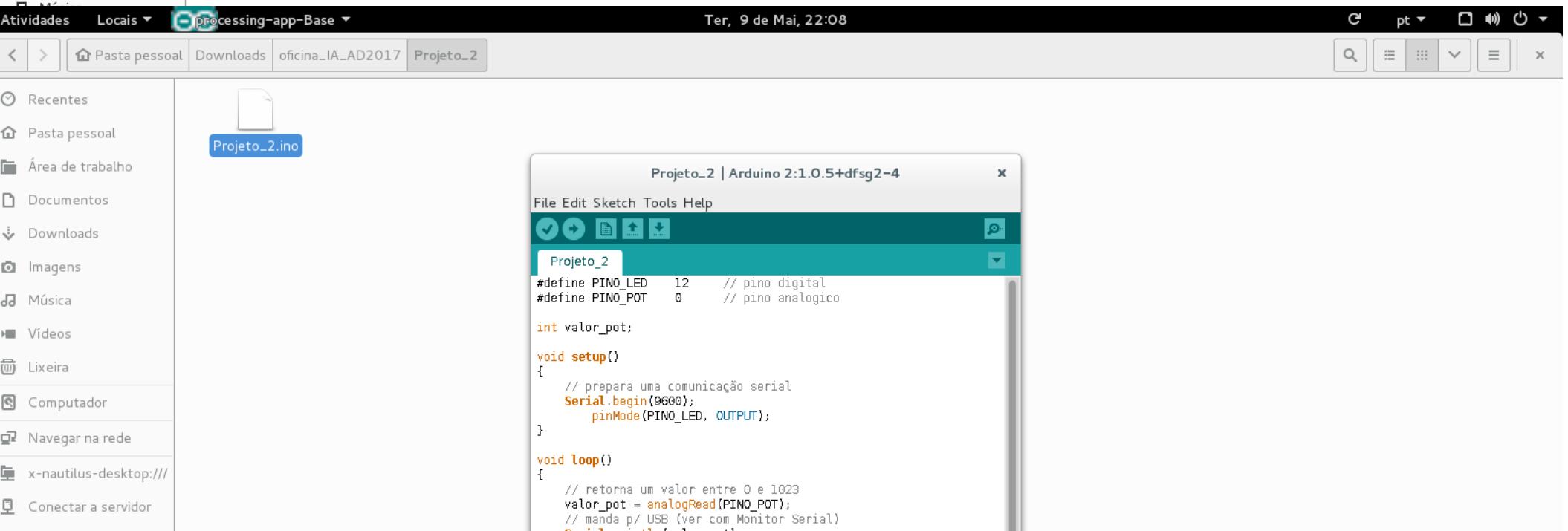
int valor_pot;

void setup()
{
    // prepara uma comunicação serial
    Serial.begin(9600);
    pinMode(PINO_LED, OUTPUT);
}

void loop()
{
    // retorna um valor entre 0 e 1023
    valor_pot = analogRead(PINO_POT);
    // manda p/ USB (ver com Monitor Serial)
    Serial.println(valor_pot);

    digitalWrite(PINO_LED, HIGH);
    delay(valor_pot);
    digitalWrite(PINO_LED, LOW);
    delay(valor_pot);
}
```

Abrindo o arquivo do projeto 2



LDR: Resistência Dependente de Luz

- Sua resistência varia conforme a intensidade da luz.
- A medida que a Intensidade Aumenta, a resistência Diminui.

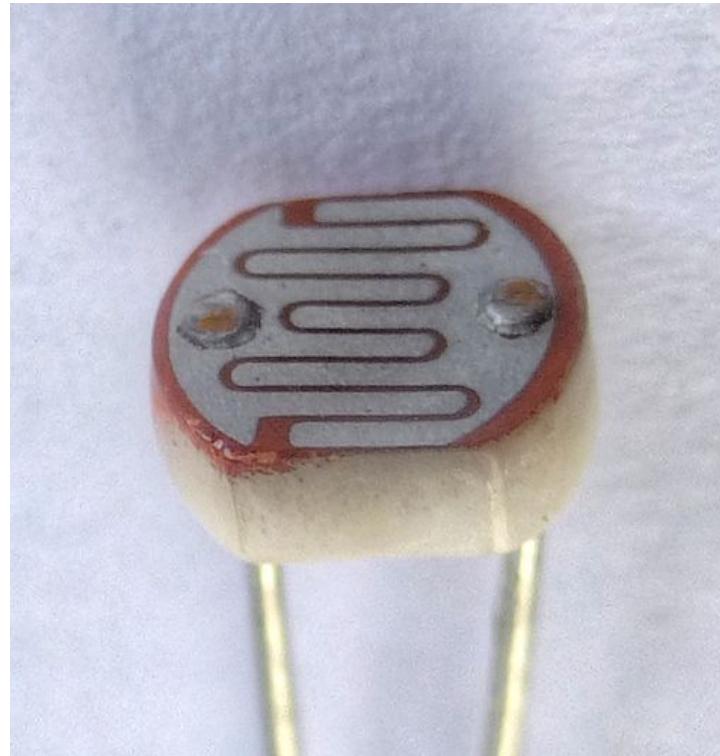
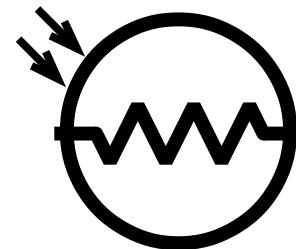


Foto de © Nevit Dilmen [CC BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0>) or GFDL (<http://www.gnu.org/copyleft/fdl.html>)], via Wikimedia Commons
Retirado de https://commons.wikimedia.org/wiki/File%3ALDR_1480405_6_7_HDR_Enhancer_1.jpg

Referência Arduino: Serial.begin() e Serial.print()

Serial.begin: configura a taxa de troca de dados em bits por segundo entre o computador e o Arduino (transmissão serial).

Serial.print: imprime dados na porta serial em um formato de texto que pode ser lido por humanos.

Sintaxe:

Serial.begin(velocidade)

Parâmetros:

- Velocidade: 9600 bit/segundo

Retorno:

- nada

Sintaxe:

Serial.print(valor)

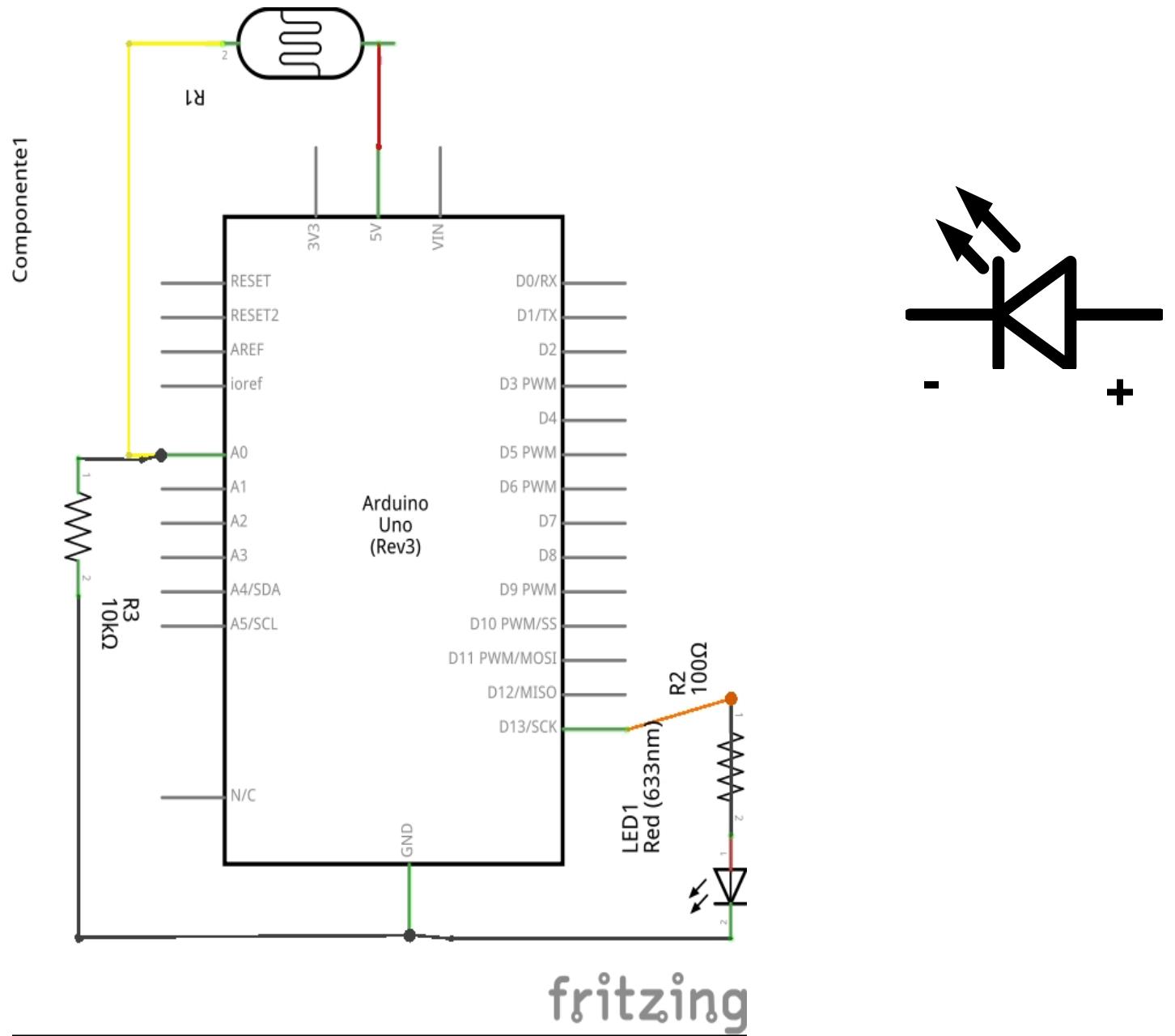
Parâmetros:

- Valor: qualquer número, texto ou variável.

Retorno:

- size_t (long)

Projeto 2: Controle LED com LDR

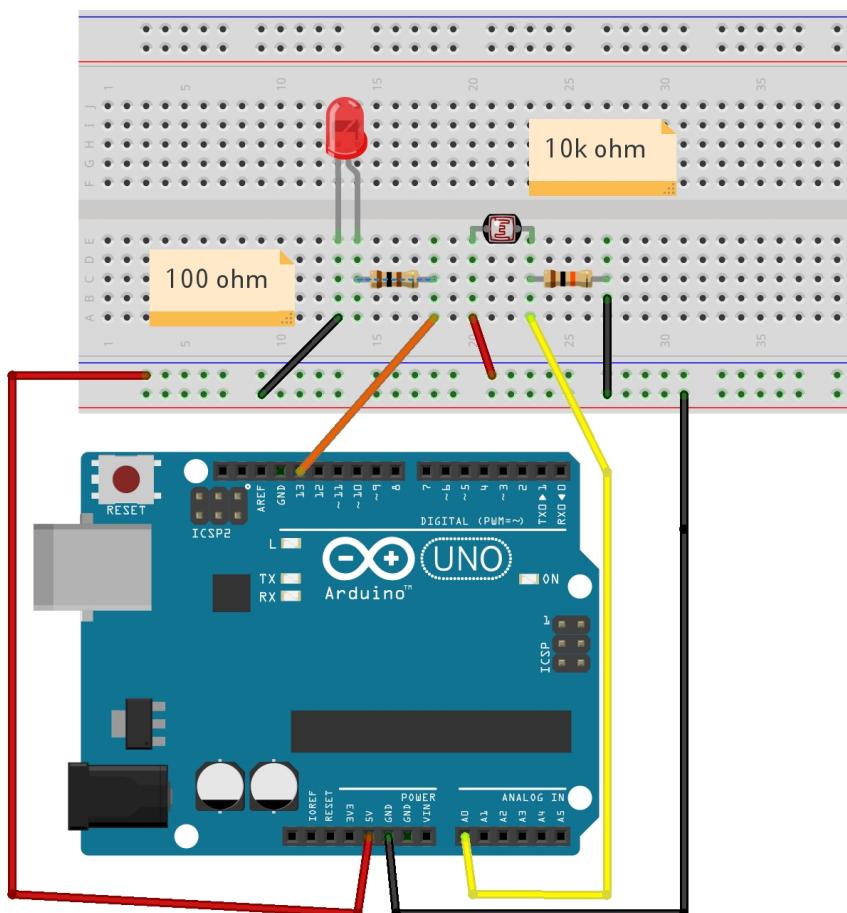


Projeto 2: Controle LED com LDR

→ Materiais adicionais:

- ✓ 1 LED
- ✓ 1 resistores de 100 ohms
- ✓ 1 resistor de 10k ohm
- ✓ 1 LDR

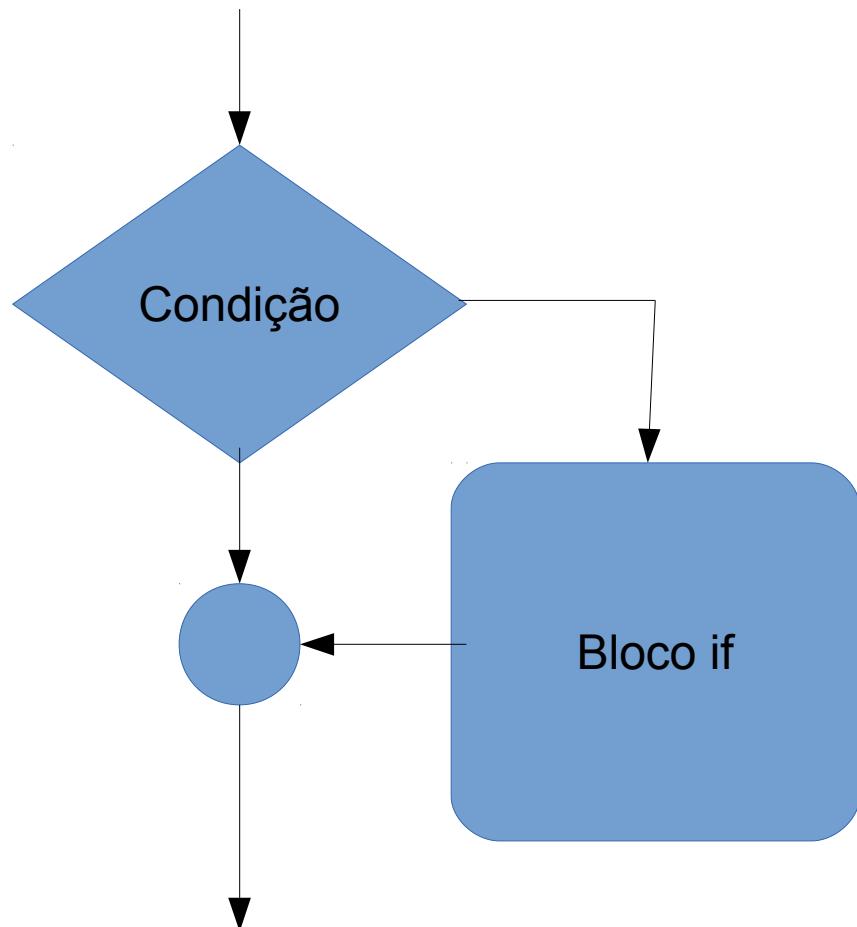
→ Montagem:



→ Código

```
float val_lum;  
int luminosidade;  
byte LDRpin=A0;  
byte LED = 13;  
  
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    val_lum = analogRead(LDRpin);  
    luminosidade = val_lum*0.0977;  
    Serial.write("Luminosidade: ");  
    Serial.println(luminosidade);  
  
    if (luminosidade < 50)  
    {  
        digitalWrite(LED, HIGH);  
    }  
    else  
    {  
        digitalWrite(LED, LOW);  
    }  
    delay(1000);  
}
```

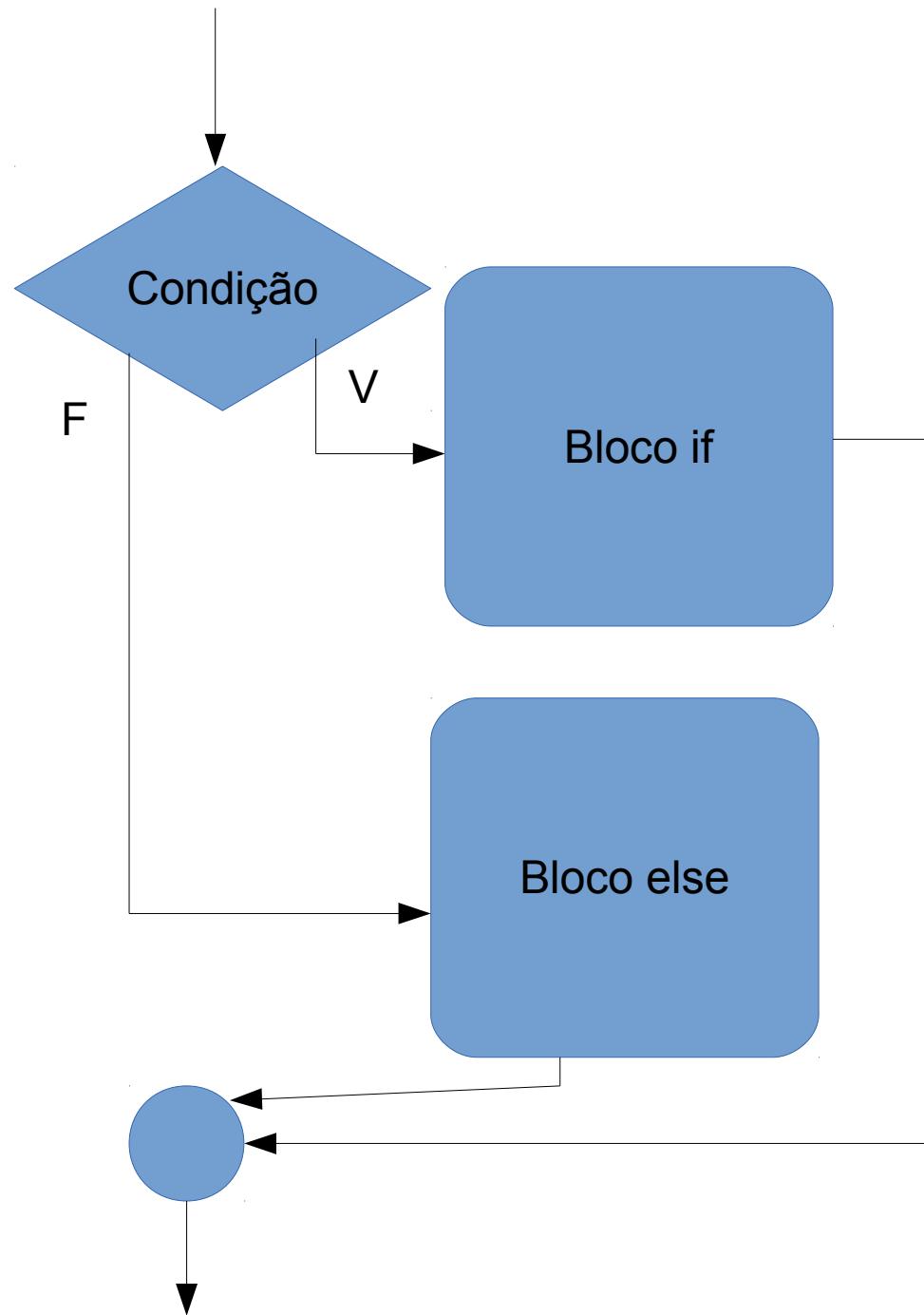
Desvio condicional: if



```
if (<condição>)
{
    <comando 1>;
    <comando 2>;
    ...
    <comando n>;
}
```

```
se <condição> então:
    <comando 1>;
    <comando 2>;
    ...
    <comando n>;
fim-se
```

Desvio condicional: if-else



```
if (<condição>)
{
    <comandos V>;
}
else
{
    <comandos F>;
}
```

```
se <condição> então:
    <comandos V>;
senão
    <comandos F>;
fim-se
```

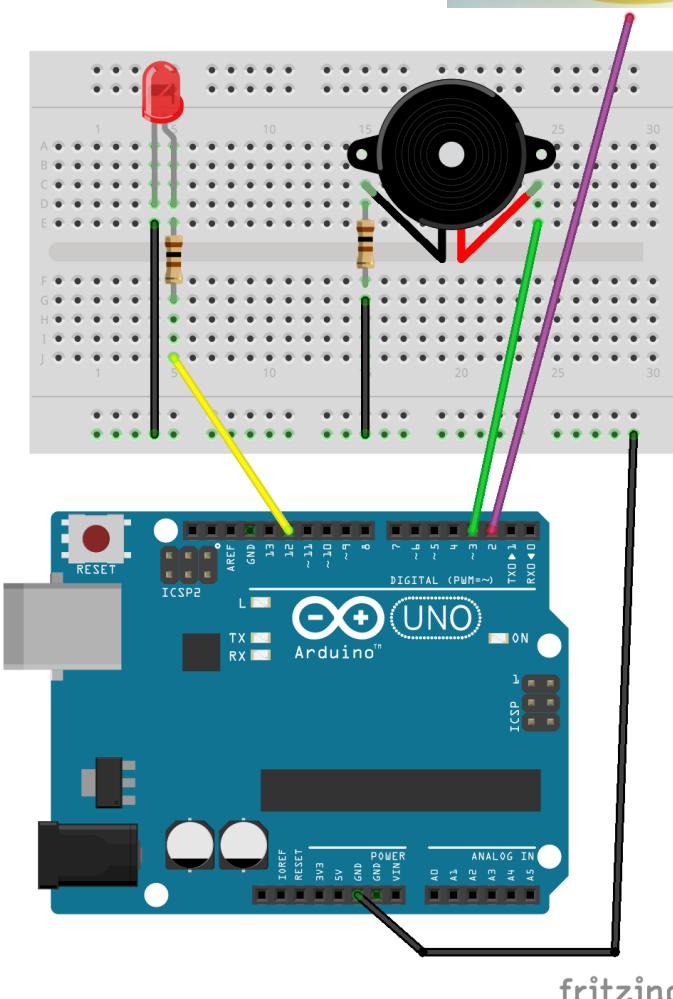
Projeto 3: Banana CapSense

→ Materiais :

- ✓ 1 LED
 - ✓ 1 Buzzer (Piezzo)
 - ✓ 2 resistores de 100 ohms
 - ✓ 1 banana



→ Montagem:



→ Código:

```
#include <pincapsense.h>
#define PINO_LED      12 // pino digital
#define PINO_BUZZER   3 // pino digital PWM
#define PINO_CAP       2 // pino sensor cap.
#define NOTA_E7        2637 // Hz
int valor_cap;

void setup() {
    pinMode(PINO_LED, OUTPUT);
    pinMode(PINO_BUZZER, OUTPUT);
}

void loop() {
    valor_cap = readCapacitivePin(PINO_CAP);
    Serial.println(valor_cap);

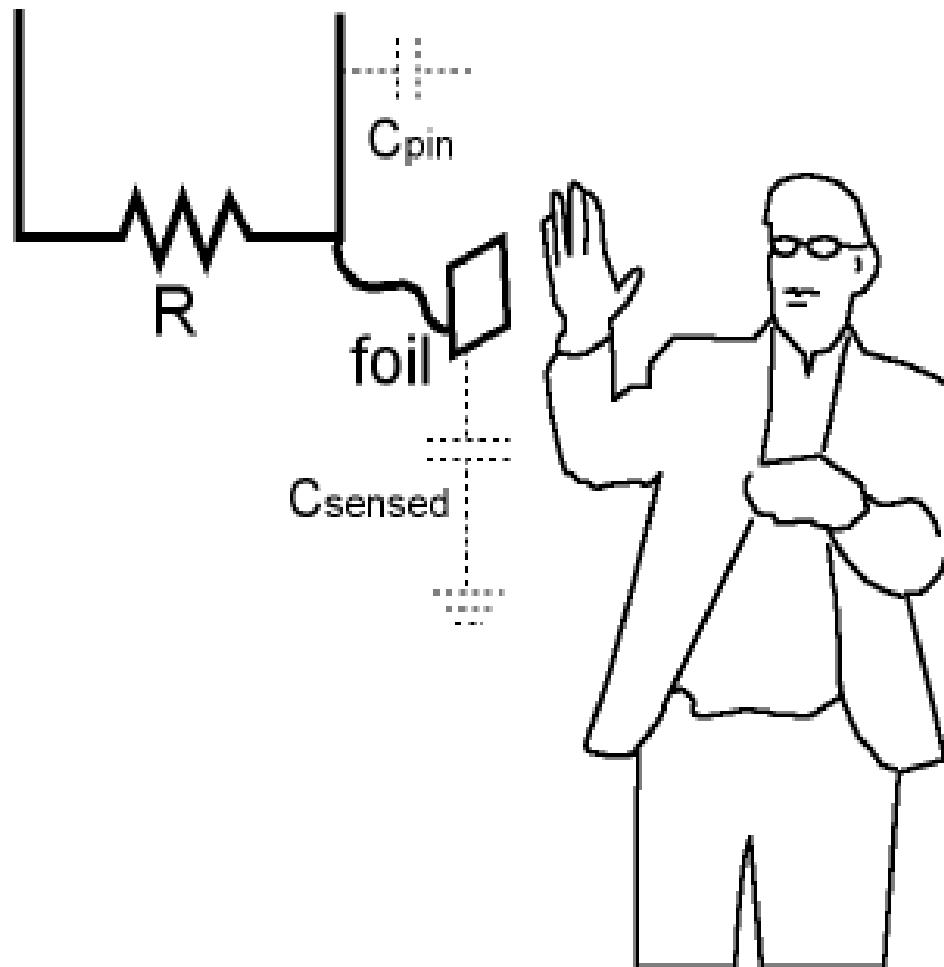
    if (valor_cap > 6) { // tocou na banana?
        digitalWrite(PINO_LED, HIGH);
        tone(PINO_BUZZER, NOTA_E7);
    }
    else {
        digitalWrite(PINO_LED, LOW);
        noTone(PINO_BUZZER);
    }
    delay(200);}
```

Fonte da imagem da banana

http://commons.wikimedia.org/wiki/Banana#mediaviewer/File:Banane_Frucht.jpg

Sensor capacitivo: princípio

Send pin Receive pin



→ **Capacitor:**
componente que
acumula cargas

→ Pode ser
descarregado:
basta ligar as placas
por um fio + resistor

→ Pinos do Arduino já
possuem um **resistor**
interno associado

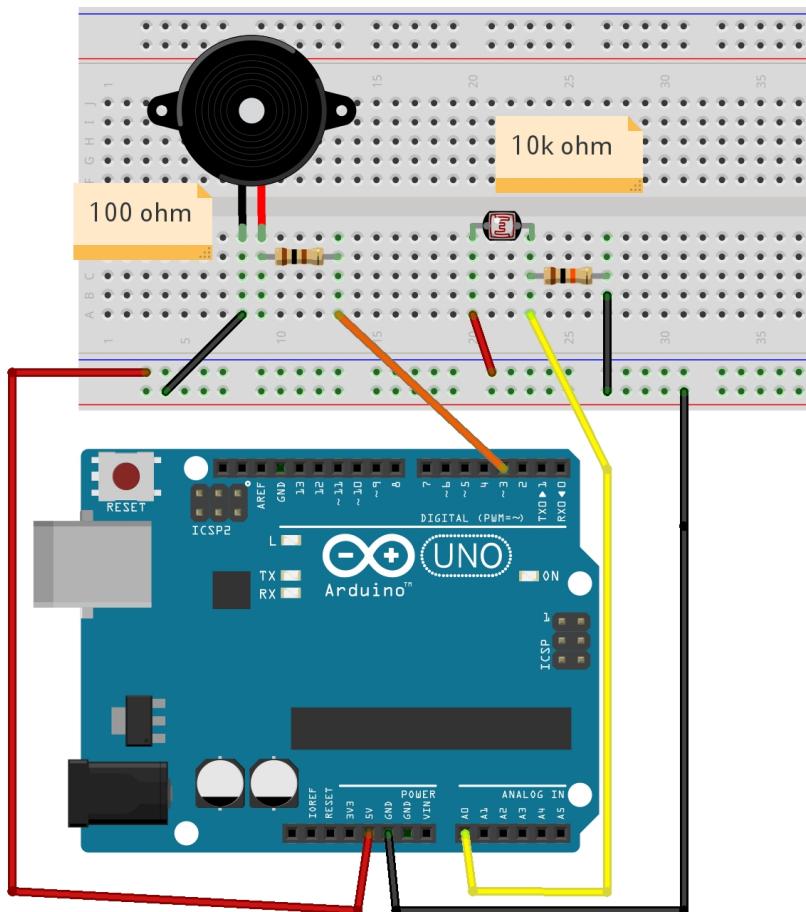
→ Leitura CapSense:
Estima-se **ciclos de
leitura** para quedas e
subidas de tensão

Projeto 4: Buzzer de Luz

→ Materiais :

- ✓ 1 Buzzer (Piezzo)
- ✓ 1 resistor de 100 ohms
- 1 resistor de 10k ohm
- 1 LDR

→ Montagem:



```
#define PINO_BUZZER 3 // pino digital PWM
#define LDRpin A0

float val_lum, periodo = 200;
int luminosidade;

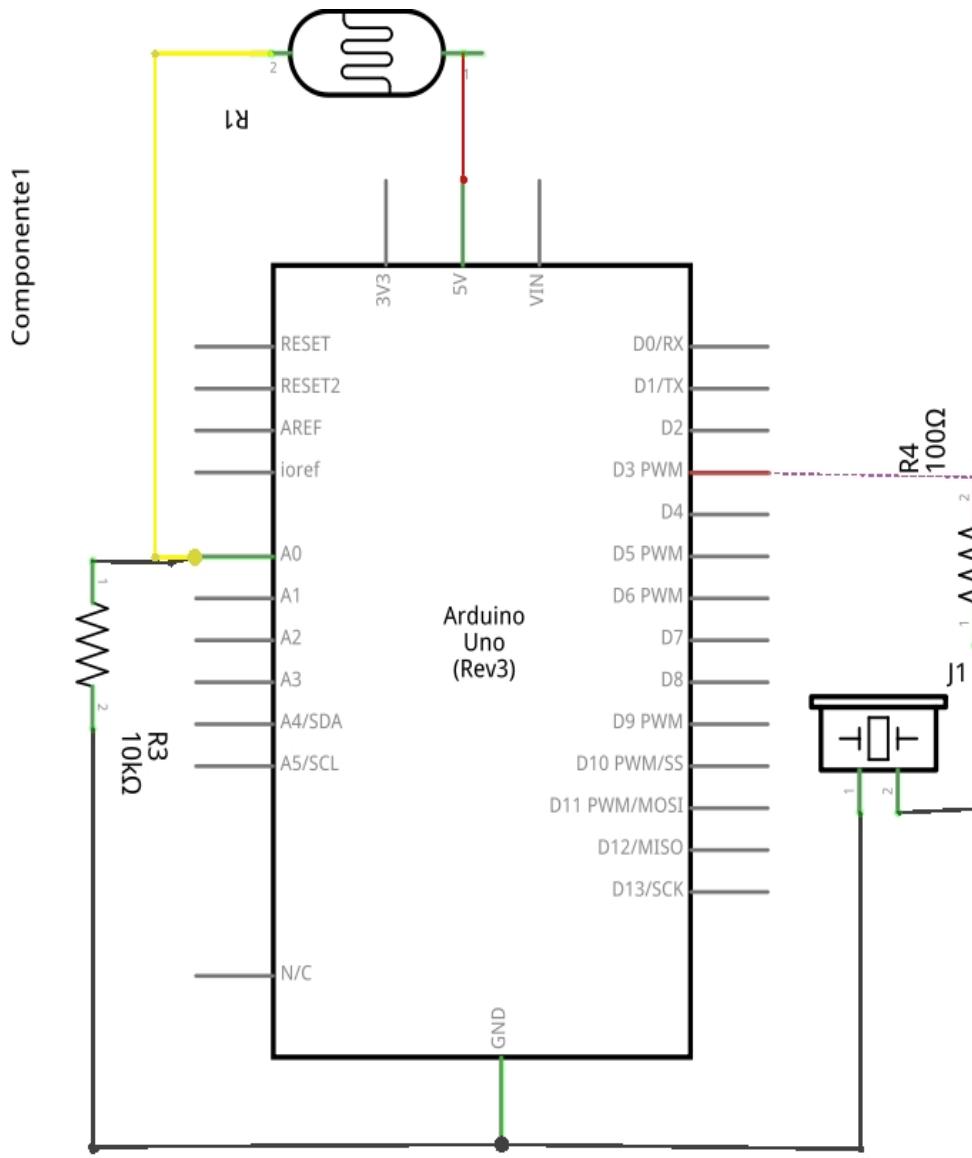
void setup() {
    pinMode(PINO_BUZZER, OUTPUT);
    pinMode(LDRpin, INPUT);
    Serial.begin(9600);
}

void loop() {
    val_lum = analogRead(LDRpin);
    luminosidade = val_lum*9.77;

    Serial.write("Luminosidade: ");
    Serial.println(luminosidade);

    tone(PINO_BUZZER, luminosidade);
    delay(periodo);
}
```

Projeto 4: Buzzer de Luz



fritzing

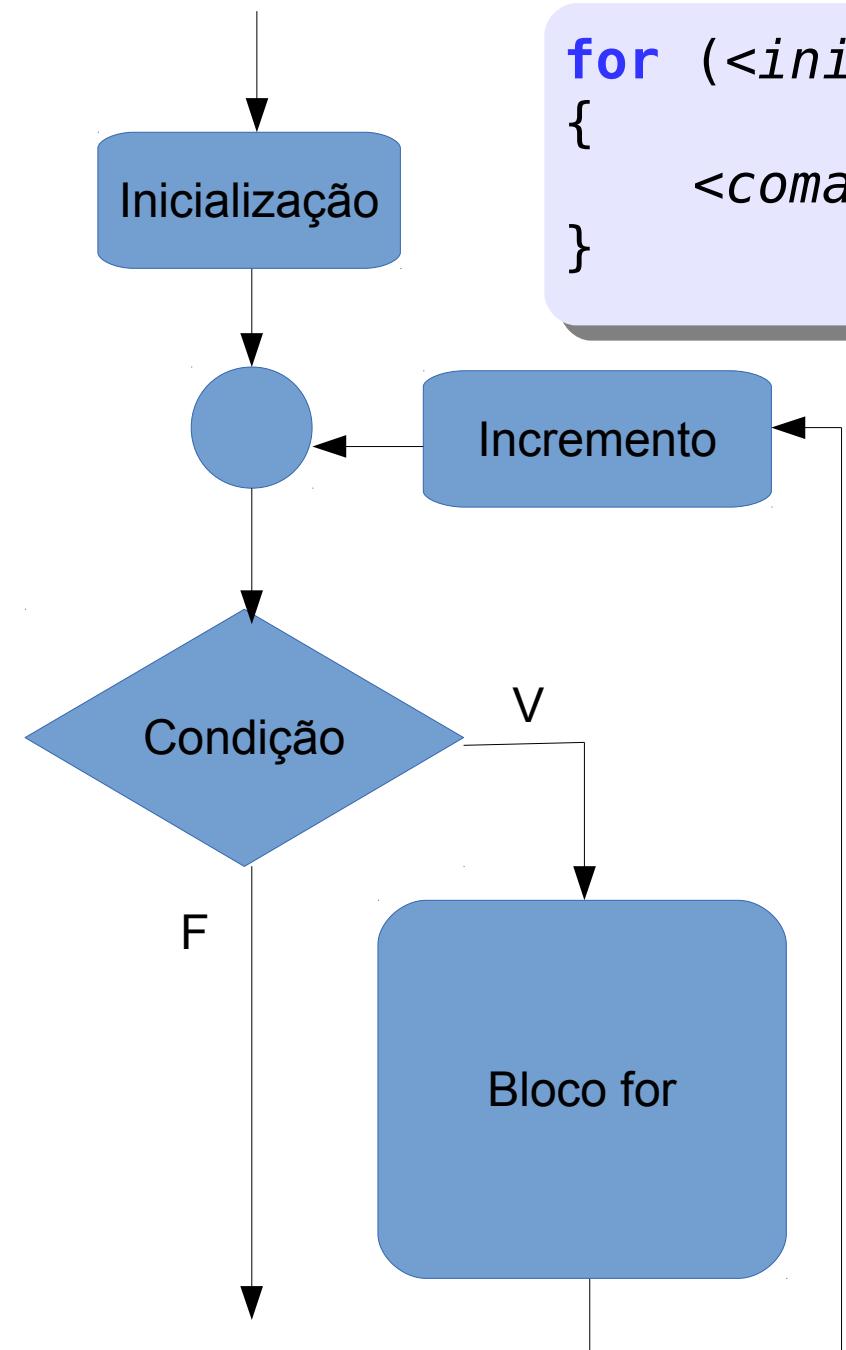
O que aprendemos?

- Conceitos básicos de eletricidade;
- o que é o projeto Arduino;
- sobre as placas Arduino;
- como usar a IDE Arduino;
- como usar uma protoboard;
- montar circuitos com Arduino;
- conceitos básicos de eletrônica e
programação em C;

Informações Complementares

- Conceitos básicos de eletricidade;
- o que é o projeto Arduino;
- sobre as placas Arduino;
- como usar a IDE Arduino;
- como usar uma protoboard;
- montar circuitos com Arduino;
- conceitos básicos de eletrônica e programação em C;

Laço de repetição: for



```
for (<inicialização>; <condição>; <incremento>)
{
    <comandos>;
}
```

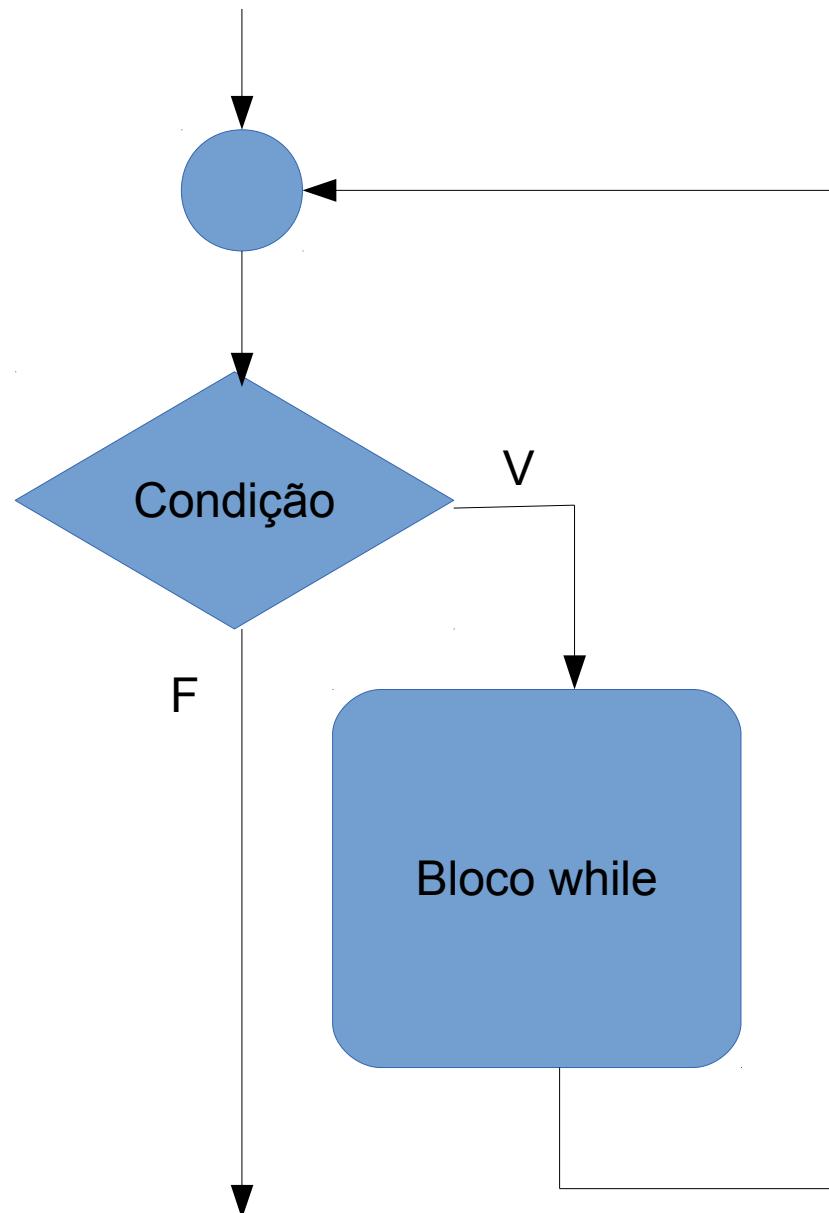
```
para <valor_inicial>
até <valor_final> faça:
    <comandos>;
fim-para
```

→ Melhor p/ contar repetições:

```
para i = 0 até 10 faça:
    <comandos>;
fim-para
```

```
for (i = 0; i < 10; i++)
{
    <comandos>;
}
```

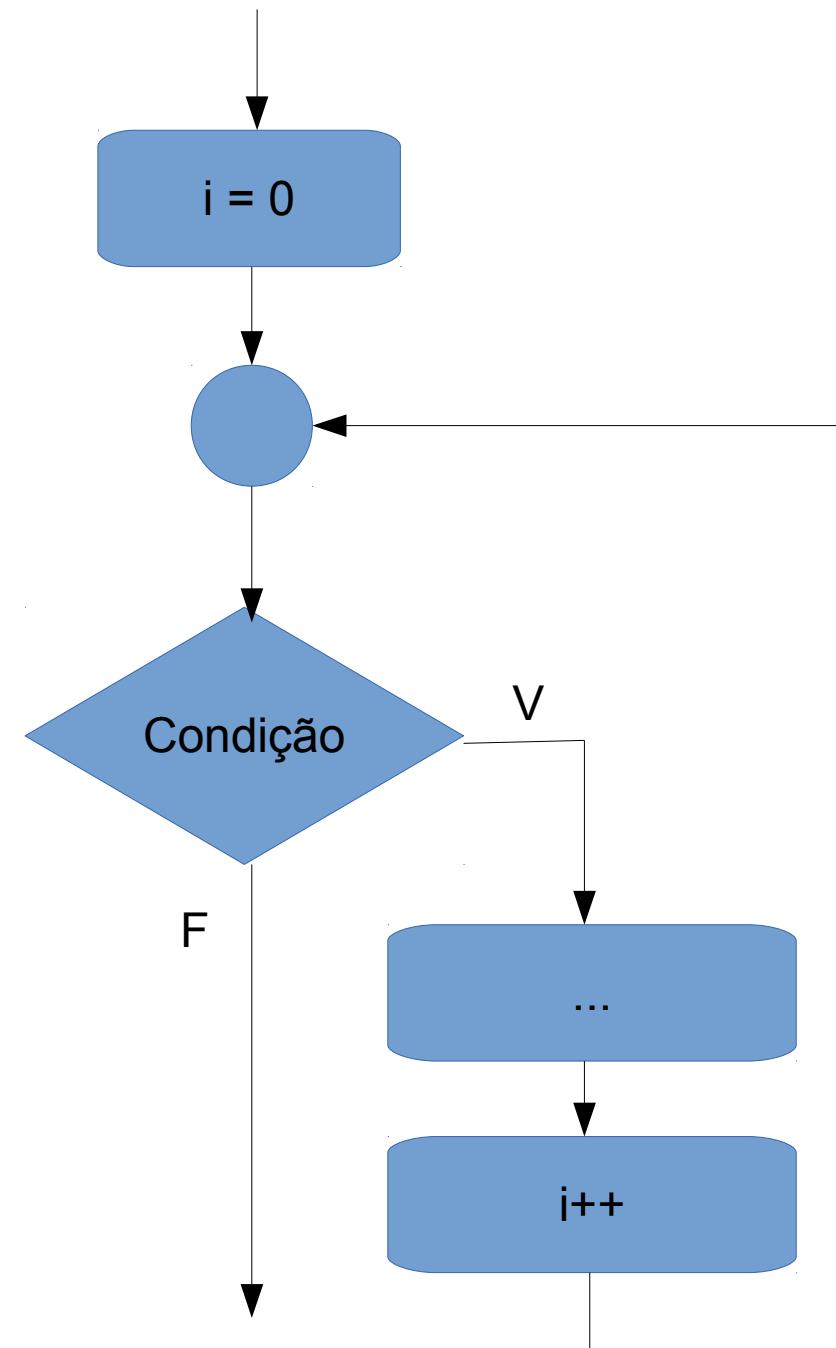
Laço de repetição: while



```
while (<condição>)
{
    <comando 1>;
    <comando 2>;
    ...
    <comando n>;
}
```

equanto <condição>
faça:
 <comando 1>;
 <comando 2>;
 ...
 <comando n>;
fim-enquanto

loop while: padrão bastante comum



→ Quando usado para contar repetições:

```
i = 0;  
equanto i < 10 faça:  
    ...  
    i ← i + 1;  
fim-enquanto
```

```
i = 0;  
while (i < 10)  
{  
    ...  
    i++;  
}
```