

Introdução a Plataforma Arduino

Marina de Freitas
Saulo Jacques

Autoras

Rafael Brandão

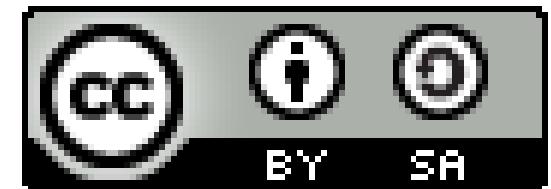
Renan B. Da Silva

Nelso Jost

Marina de Freitas

CTA - Centro de

Tecnologia Acadêmica



Estrutura da oficina

- Introdução
 - Conceitos básicos de eletricidade
 - Plataforma Arduino
 - Prototipagem em protoboard
- Projetos básicos
 - Conceitos básicos de eletrônica e programação em C
 - Montagem de projetos

CTA!

<https://cta.if.ufrgs.br/>

Estações
Meteorológicas
Laboratório
Física e Música

TropOS



cta.if.ufrgs.br



Material da Oficina

<https://cta.if.ufrgs.br/>

Material da Oficina de Hoje (clica aqui)

Sobre Princípios Projetos Canais **CONTATO** Entrar

IA
2017
Participação no Red Bull Basement - 08/07 a 03/09
CTA no CNI 2017 - 13/07
Oficina de Jekyll
Introdução ao Arduino B no Colégio Estadual José Loureiro da Silva - 10 de Maio
II Hackatona CTA - Incubadora de Sementes - 20 de Maio
Atividades CTA no Arduino Day 2017 - 1 de abril
CTA no Gathering for Open Science Hardware - 22 a 27 de Março

Participação no Red Bull Basement
Micro Aerogerador
Tecnologia Cidadã através de Estações Meteorológicas Modulares
Atividades

Os projetos **Micro Aerogerador** e **Estação Meteorológica Modular** participam da terceira Residência Hacker do Red Bull Basement. A residência acontece em São Paulo de julho a setembro de 2017 no Red Bull Station (Praça da Bandeira, 137).

O projeto das Estações Meteorológicas Modulares está na Residência Hacker com a etapa Tecnologia Cidadã através de Estações Meteorológicas Modulares.

Micro Aerogerador
Participantes na residência: Álisson e Cristthian.
Resumo da proposta:

Tecnologia Cidadã através de Estações Meteorológicas Modulares
Participantes na residência: Marina, Leonardo e Jan.
Resumo da proposta:

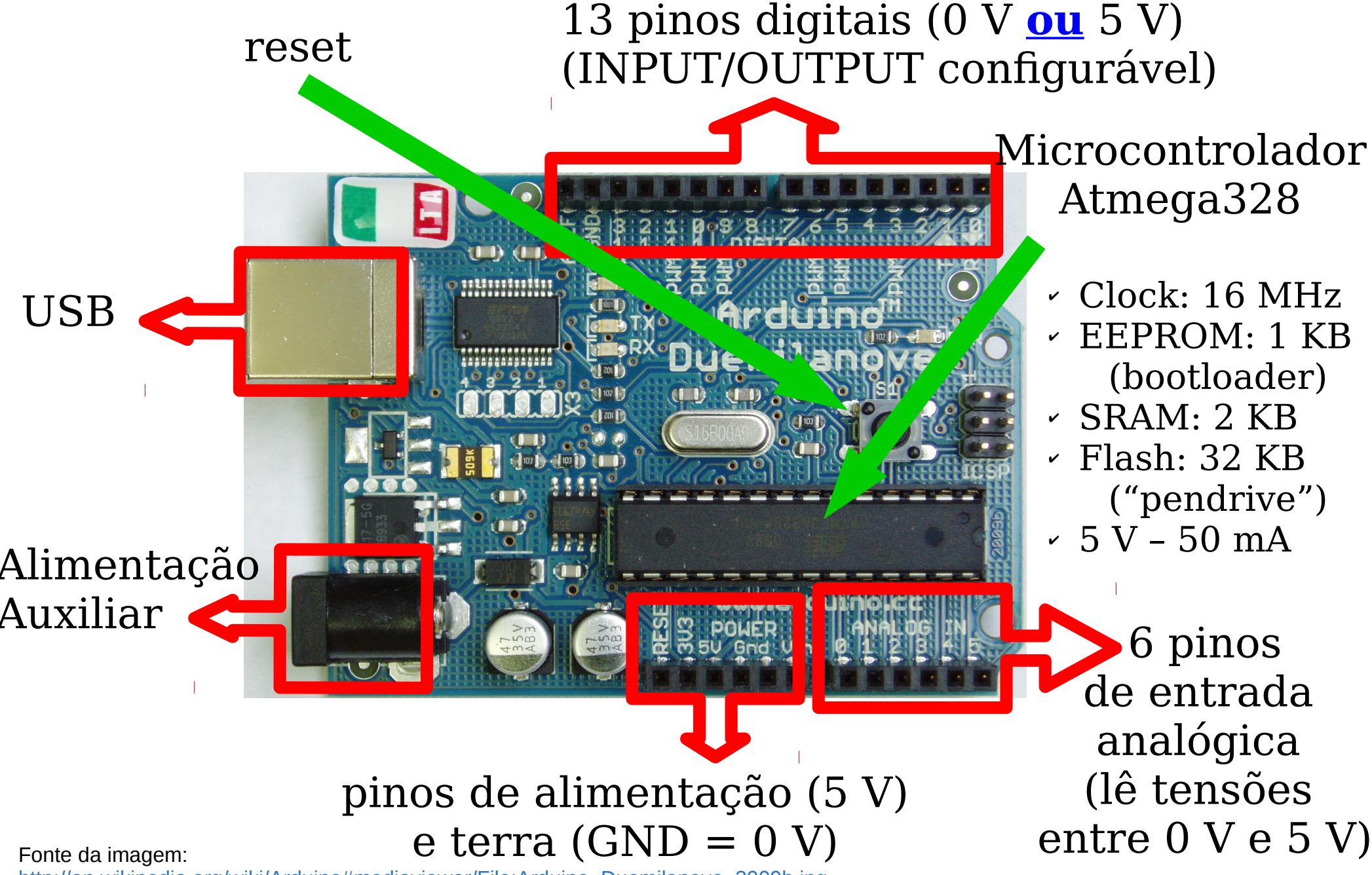
Atividades
Oficina documentação
Oficina de KiCAD
Uso de Arduino para Irrigação Automatizada de Hortas Urbanas



Eventos

Frequentemente conduzimos oficinas, cursos e participamos de eventos de ciúncio, ciência e tecnologia. Confira a nossa [página de eventos](#).

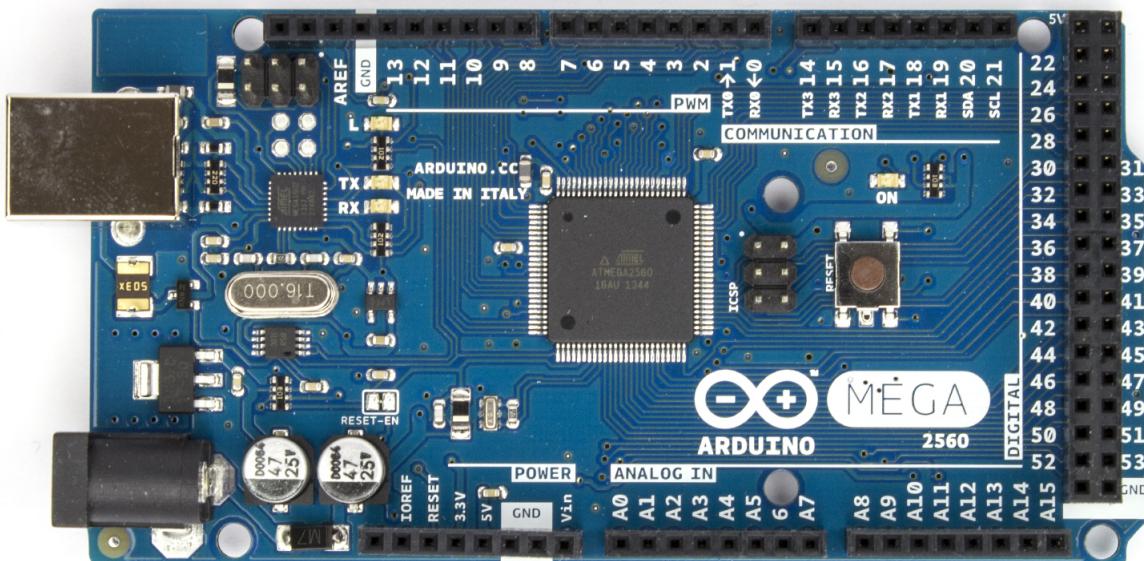
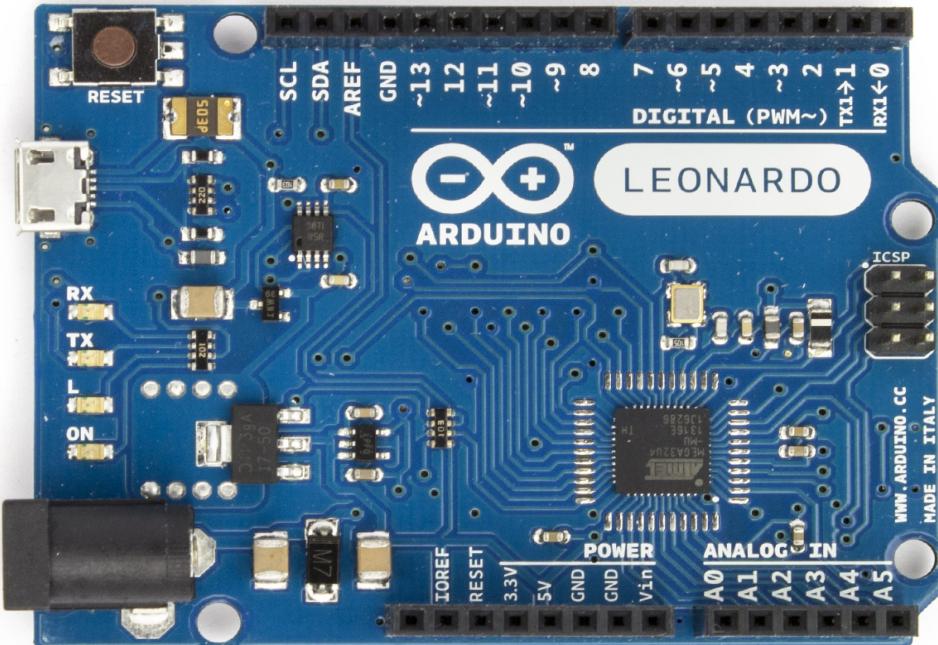
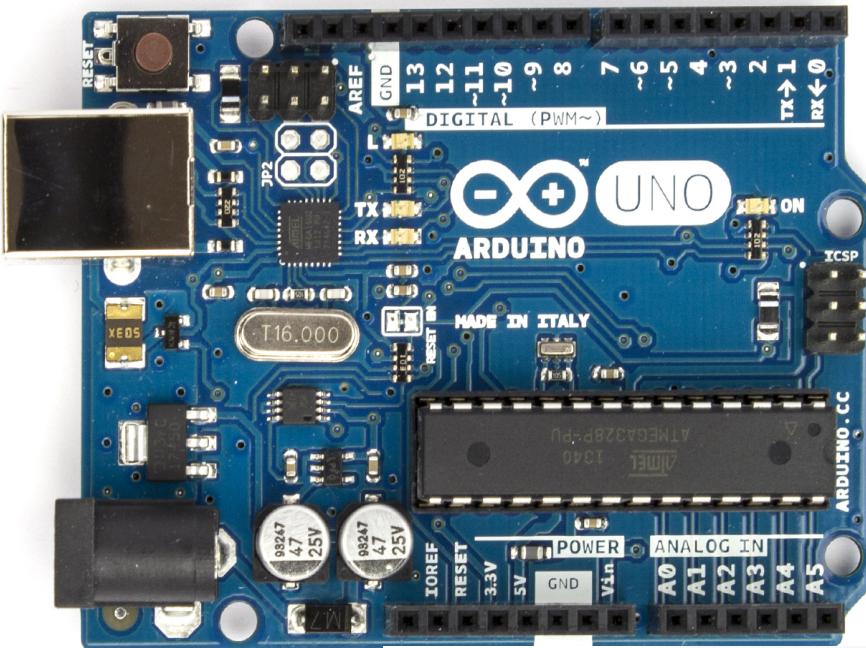
Arduino: Conheça a placa



Fonte da imagem:

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

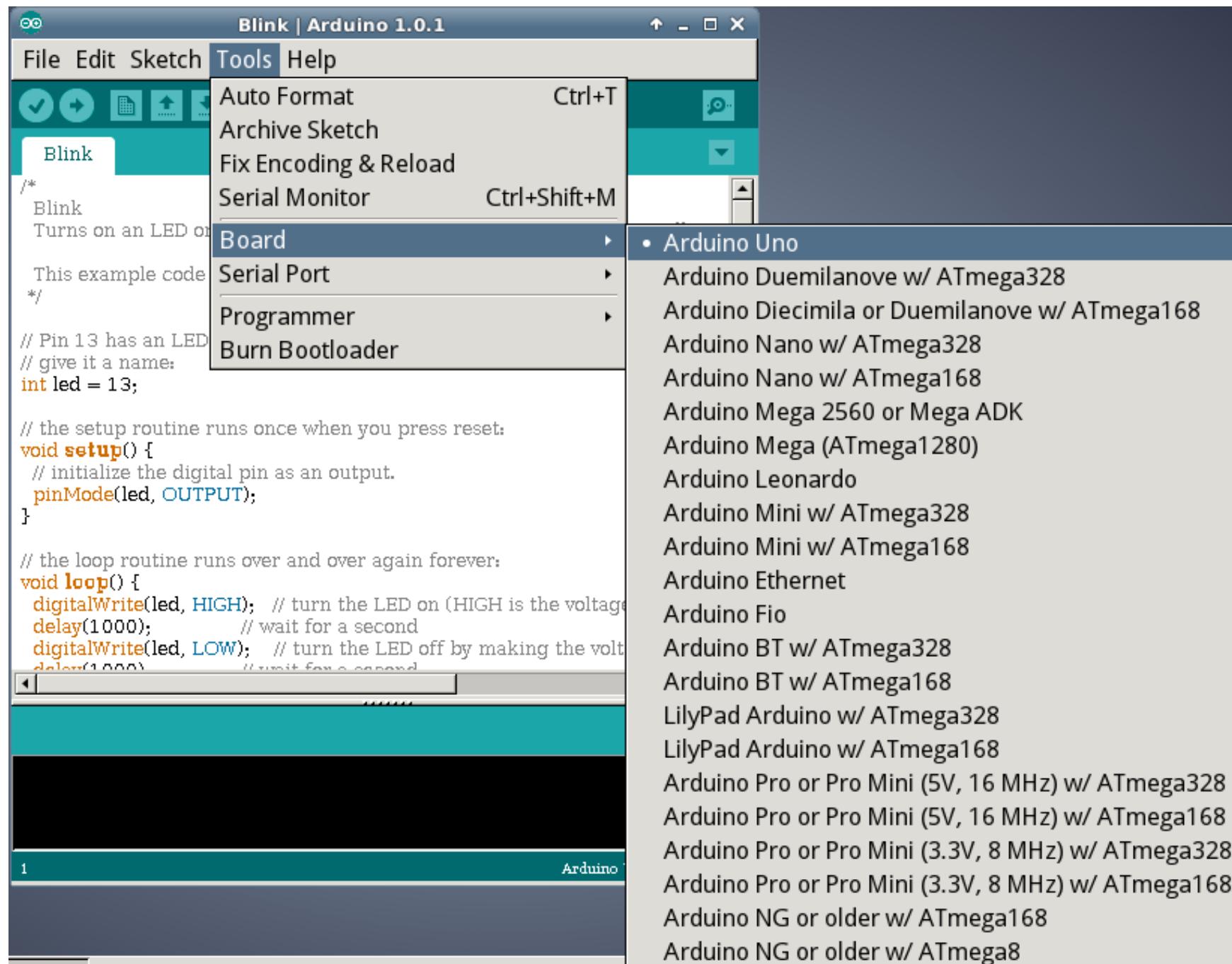
Variantes: Uno, Mega, Leonardo, etc.



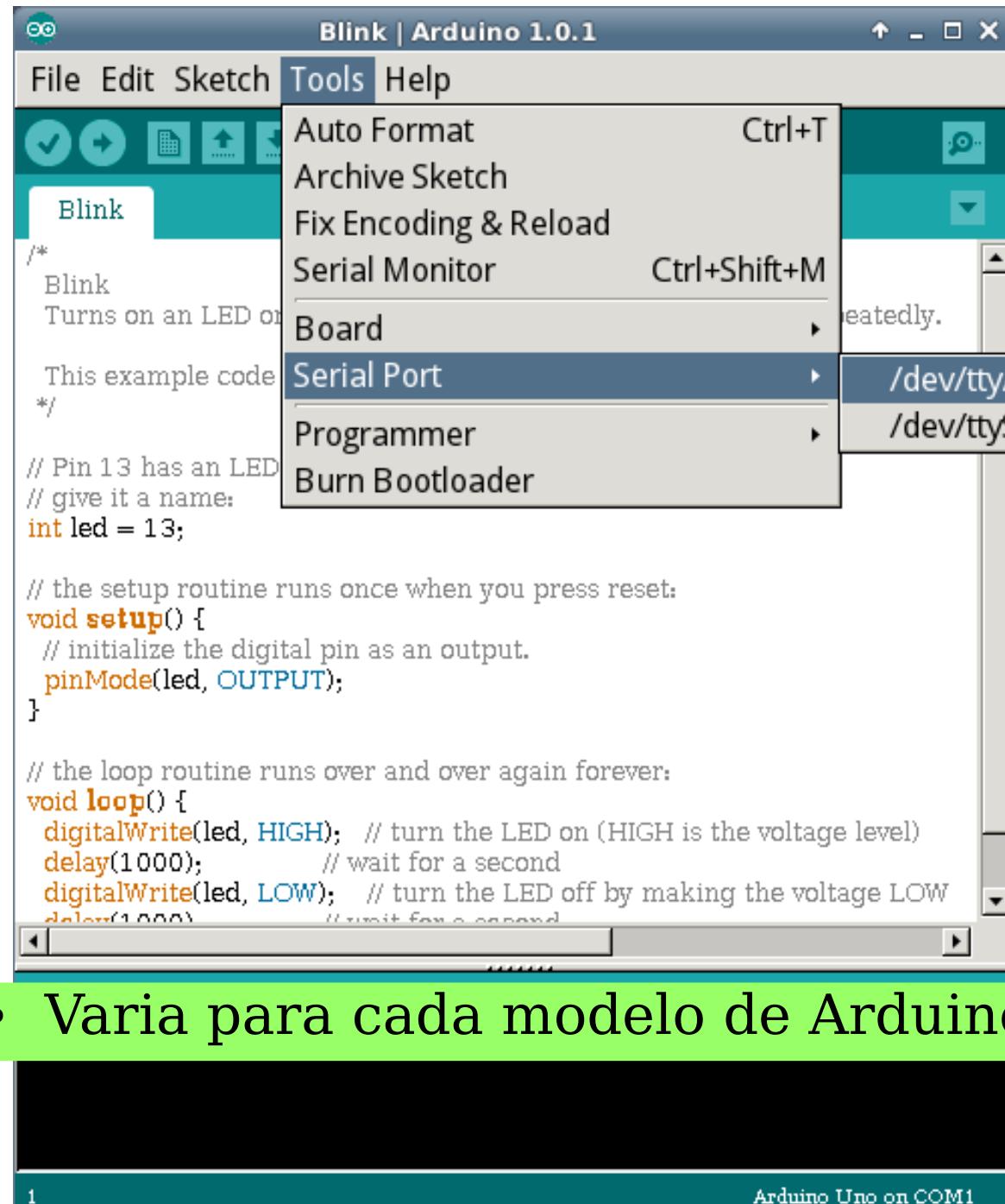
Fonte das imagens:

<http://www.arduino.cc/en/Main/Products>

Testando: Configure o modelo da placa



Testando: Configure a porta serial



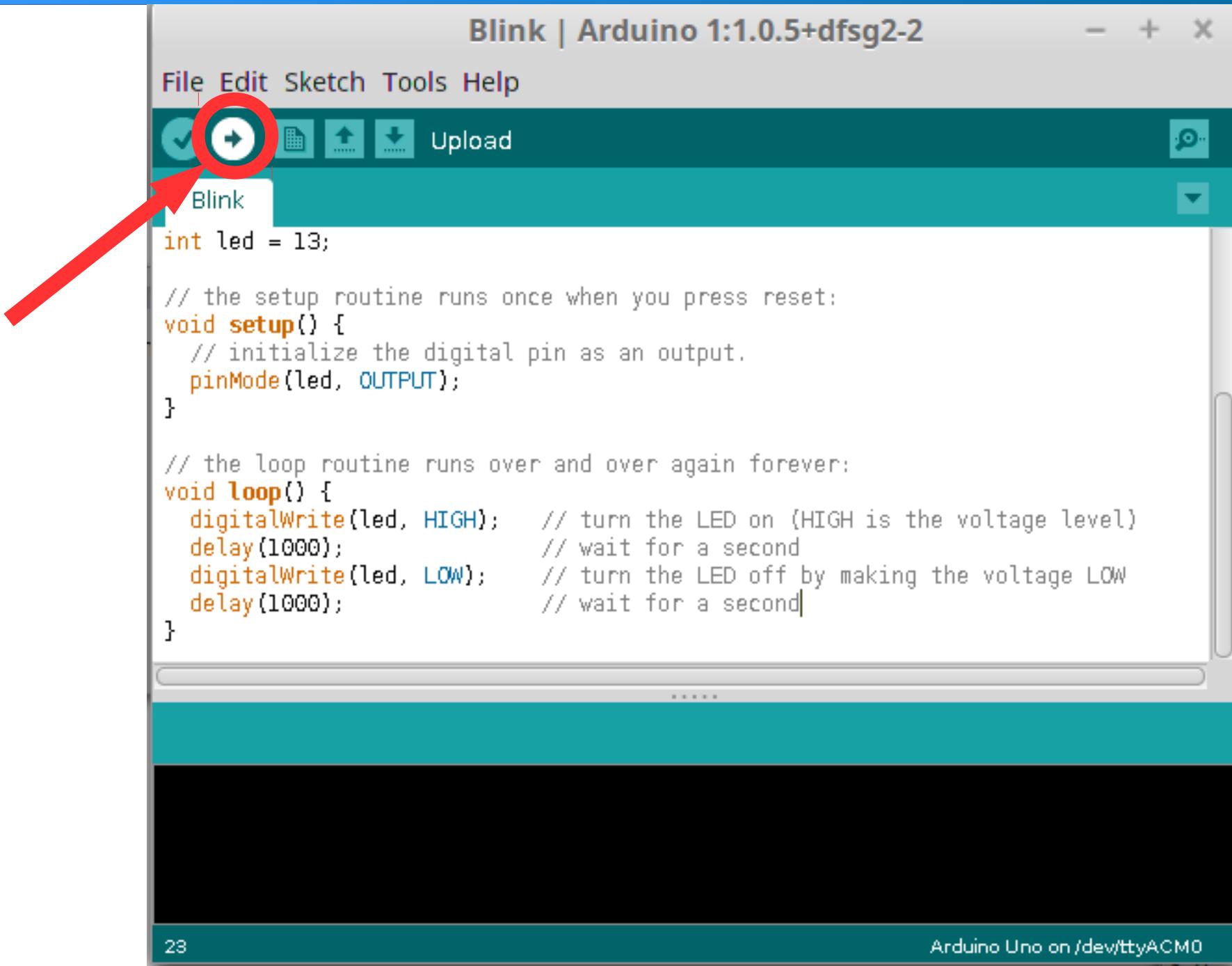
- Varia para cada modelo de Arduino

Testando: Carregue o exemplo Blink



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

Testando: Faça upload para placa



Blink | Arduino 1:1.0.5+dfsg2-2

File Edit Sketch Tools Help

Upload

Blink

```
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
}
```

23

Arduino Uno on /dev/ttyACM0

Código: Estrutura principal

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

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

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

Código: 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);
```

Código: 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!

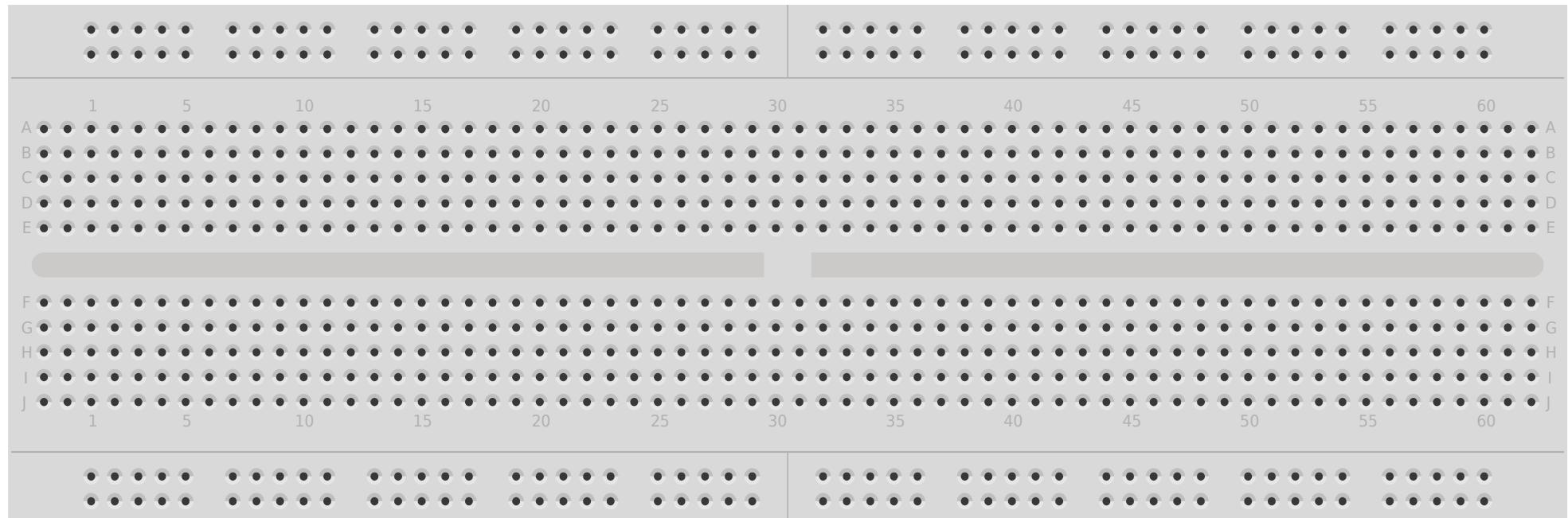
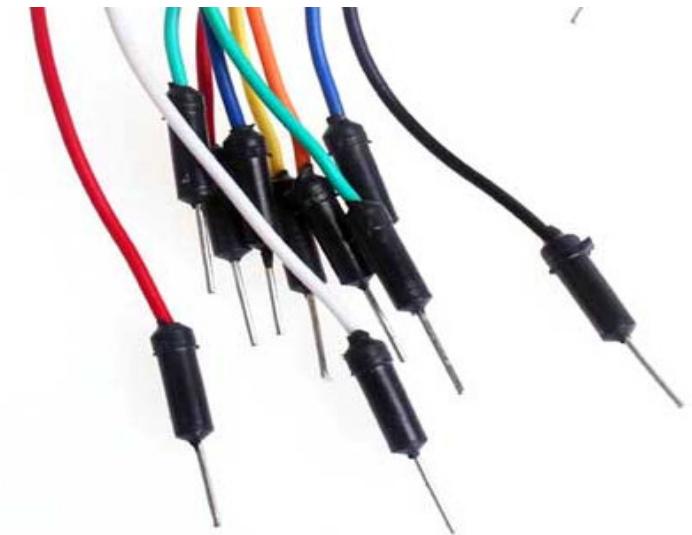
The screenshot shows the Arduino Reference homepage at www.arduino.cc/en/Reference/HomePage. The page is divided into three main sections:

- Structure**: Contains links to `setup()` and `loop()`.
- Control Structures**: Contains links to `if`, `if...else`, `for`, `switch case`, `while`, `do... while`, `break`, `continue`, `return`, and `goto`.
- Further Syntax**: Contains links to `;` (semicolon) and `{}` (curly braces).
- Variables**: Contains sections for **Constants** (links to `HIGH`, `LOW`, `INPUT`, `OUTPUT`, `INPUT_PULLUP`, `LED_BUILTIN`, `true`, `false`, `integer constants`, and `floating point constants`) and **Data Types** (links to `void`, `boolean`, `char`, `unsigned char`, `byte`, `int`, `unsigned int`, `word`, and `long`).
- Functions**: Contains sections for **Digital I/O** (links to `pinMode()`, `digitalWrite()`, and `digitalRead()`), **Analog I/O** (links to `analogReference()`, `analogRead()`, and `analogWrite()` - *PWM*), **Due only** (links to `analogReadResolution()` and `analogWriteResolution()`), and **Advanced I/O** (links to `tone()`, `noTone()`, and `shiftOut()`).

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/>

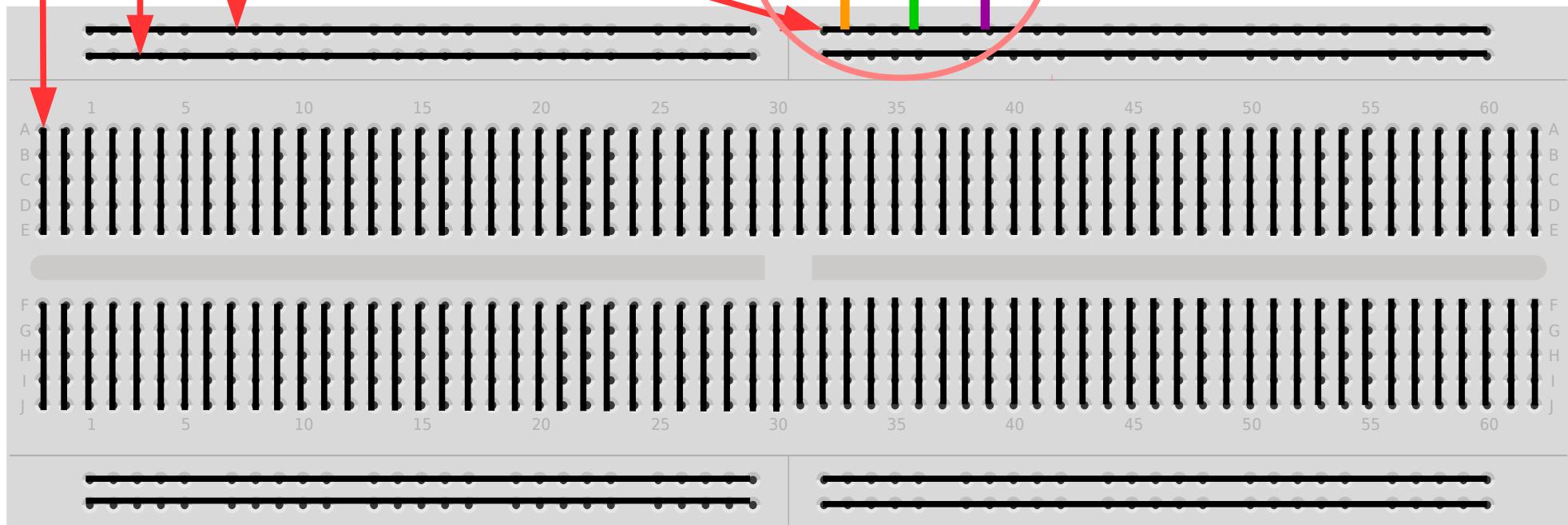


Fonte da imagem:
<http://fritzing.org/home/>

Protoboard: Como funciona?

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

barramentos

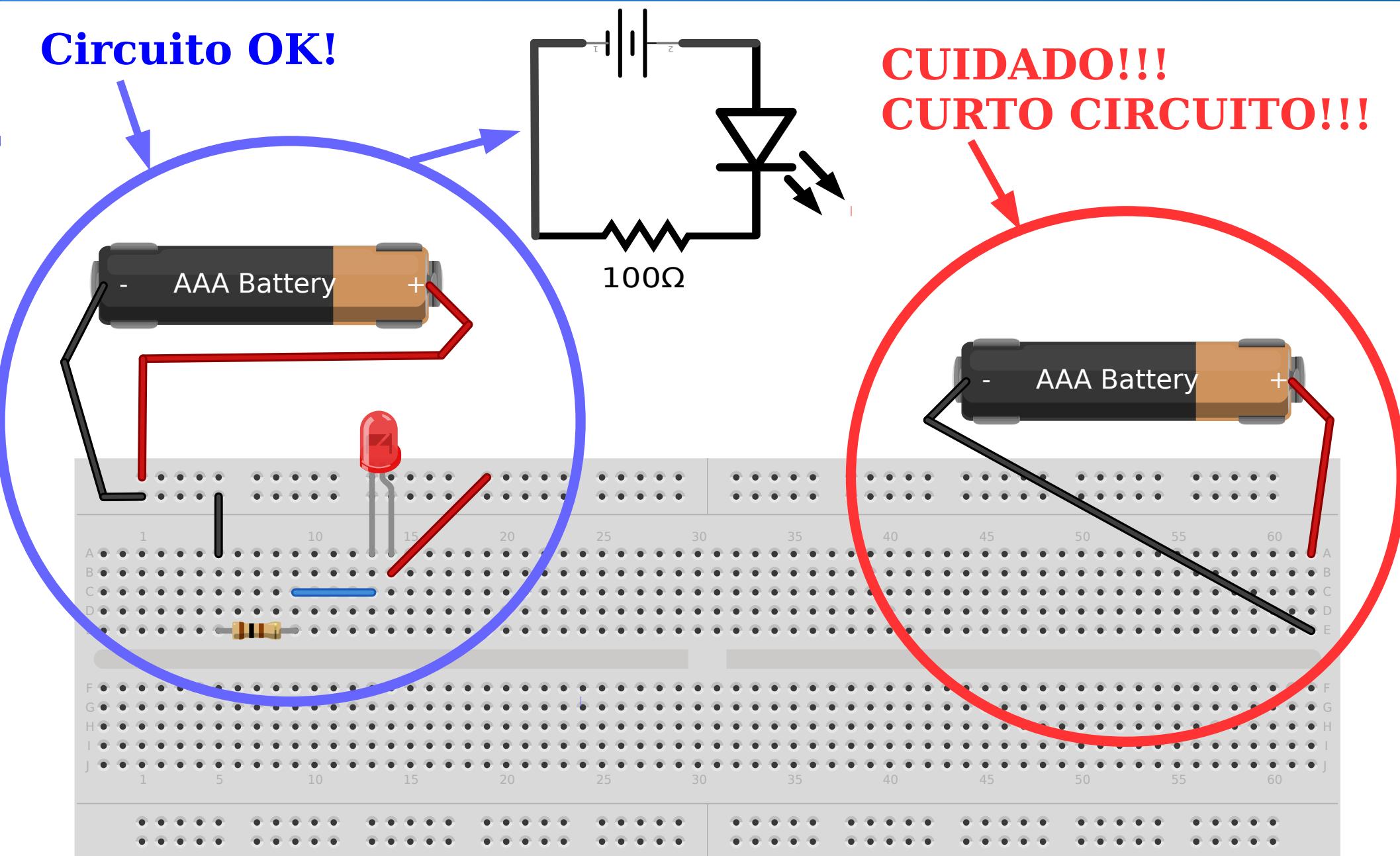


Fonte da imagem:

<http://fritzing.org/home/>

Protoboard: Exemplos

Circuito OK!



Projeto 0: Pisca LED externo

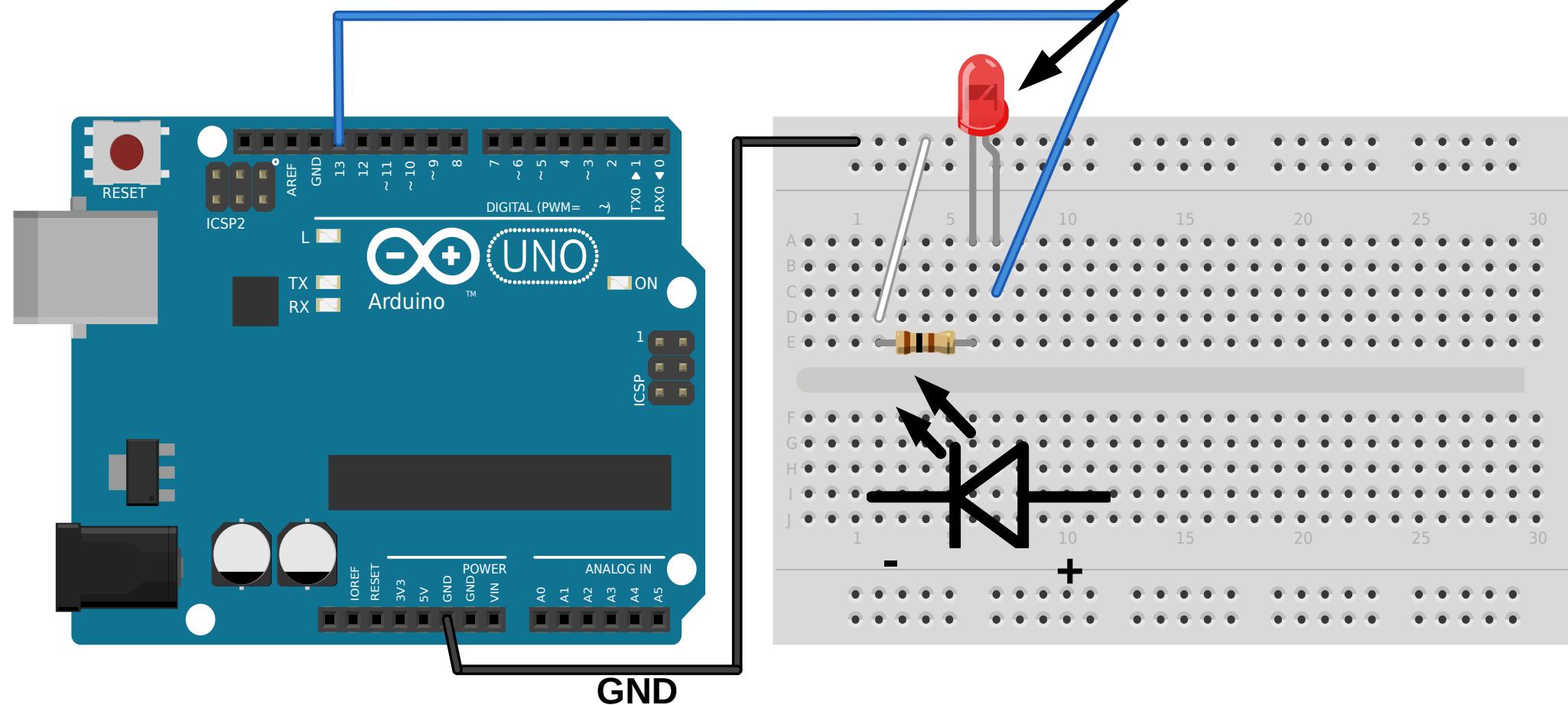
→ Materiais:

- ✓ 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

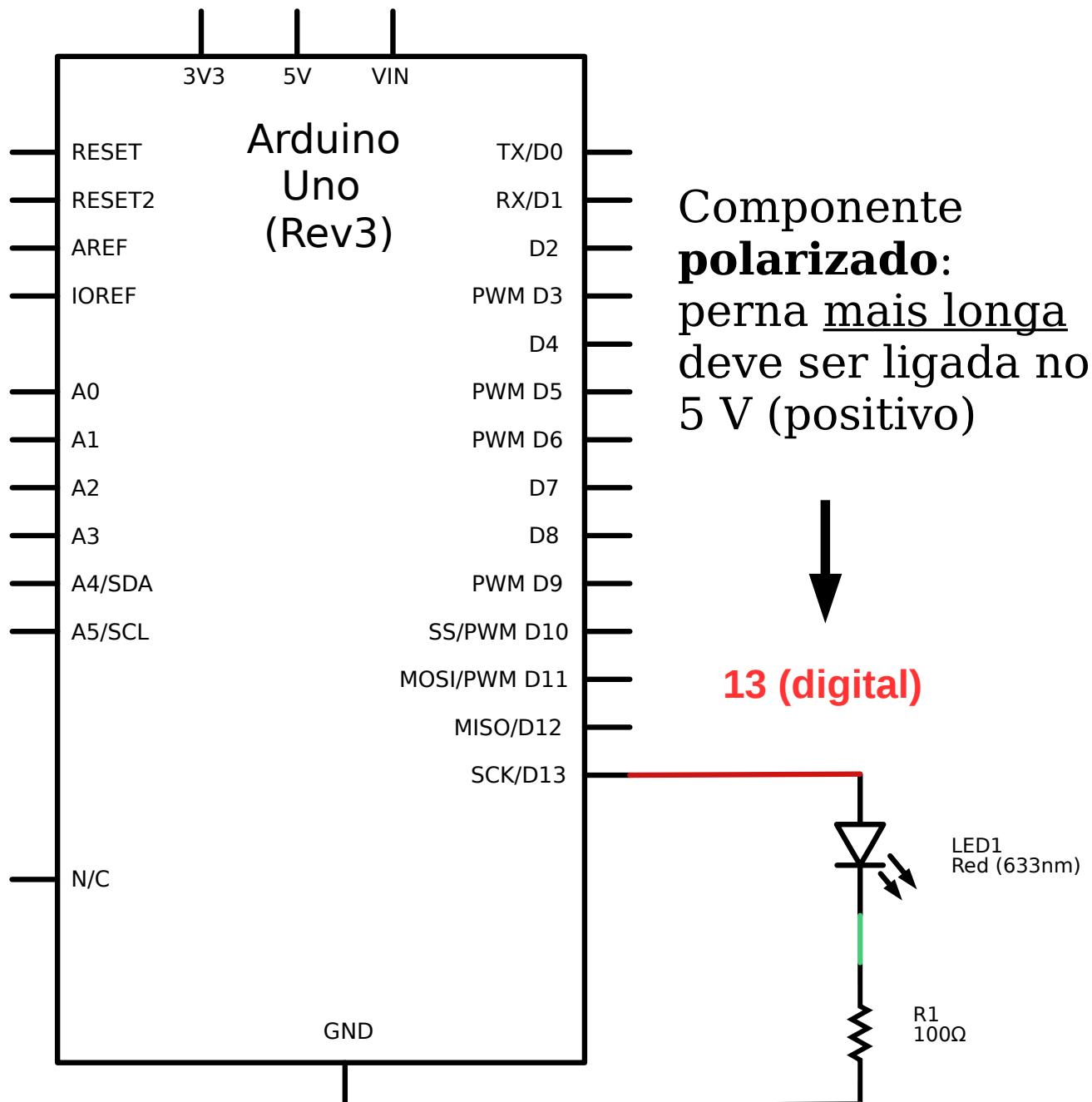
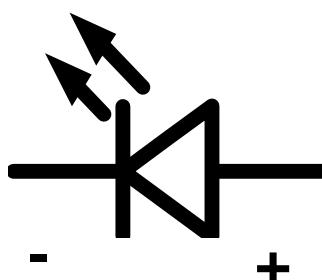
Part1

→ 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)

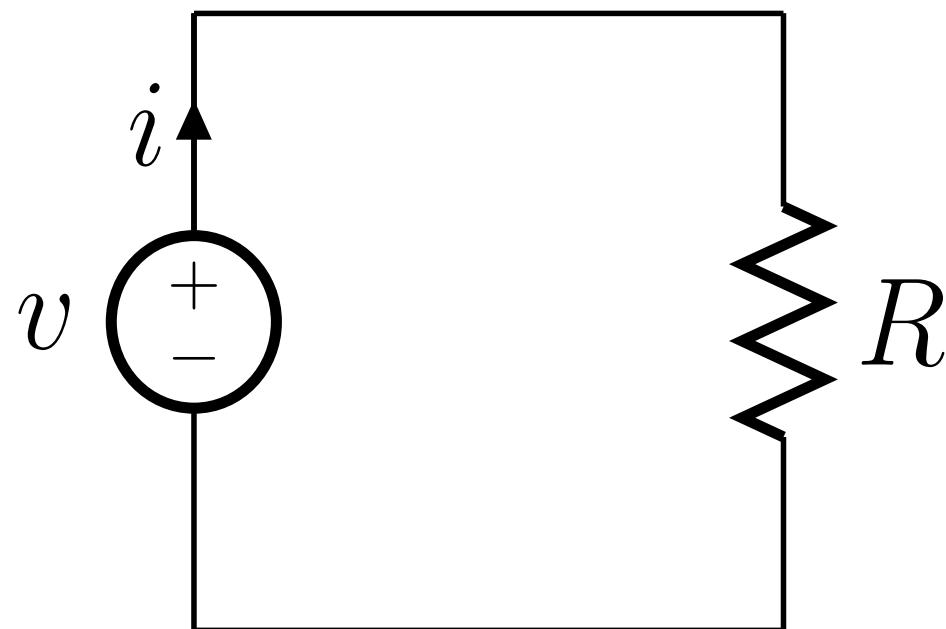
Vantagens de
constantes:

- (sintaxe) Definindo constantes:

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

- Não ocupam memória;
- Facilita manutenção;

Circuitos elétricos

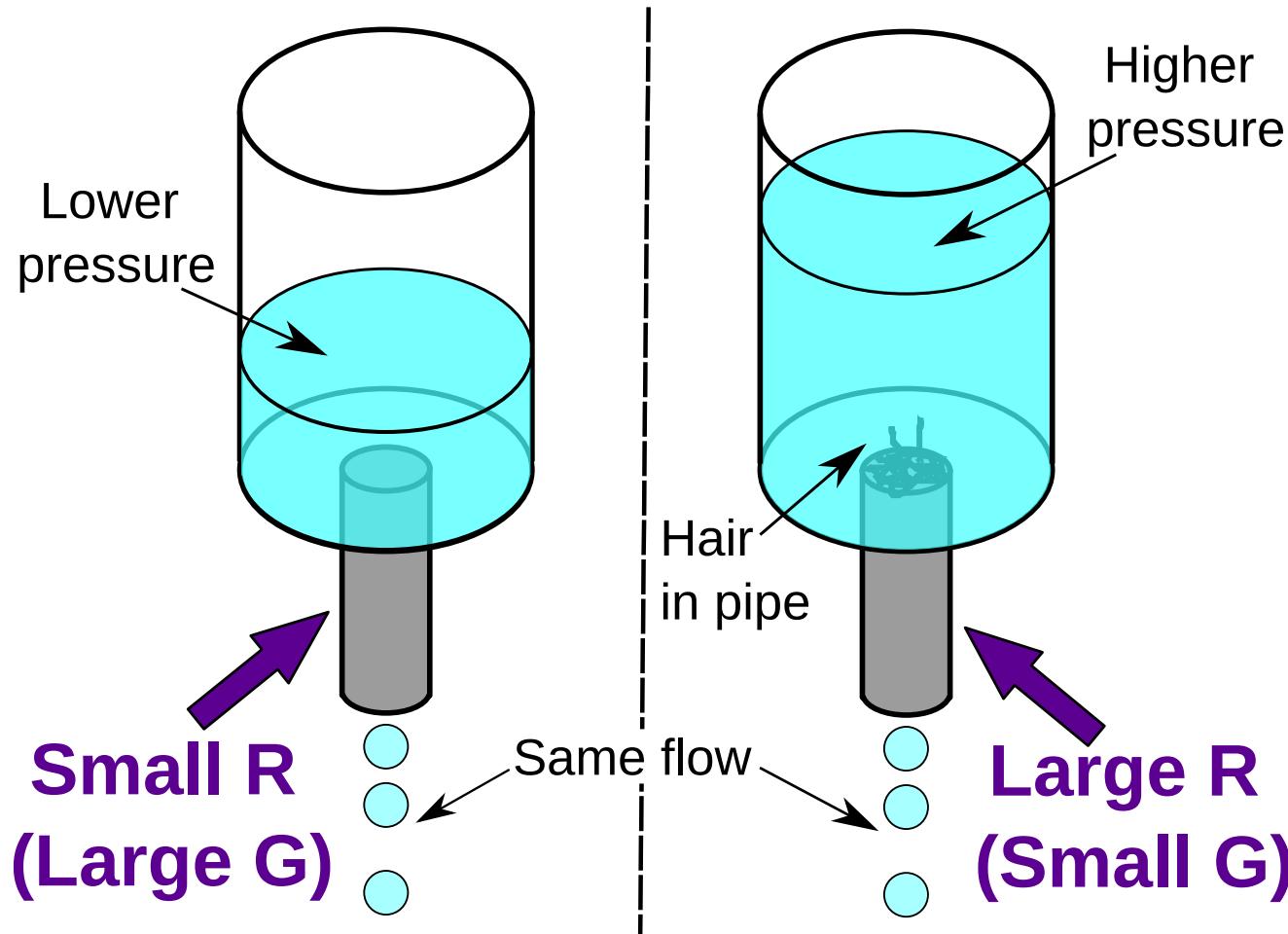


i : Corrente elétrica

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()

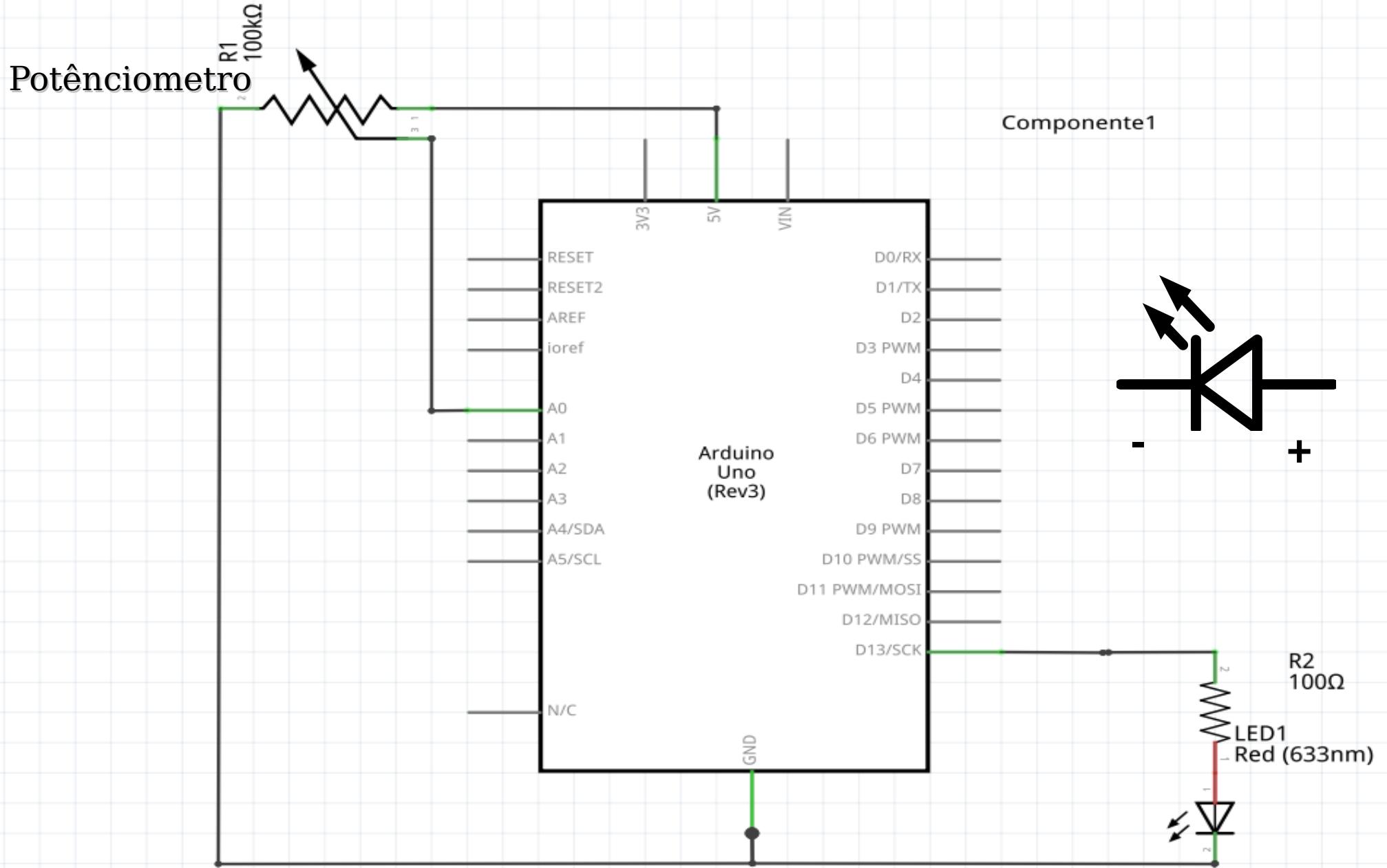
analogRead: Lê o valor do pino analógico especificado. A placa Arduino contém um conversor analógico-digital; com isso, ela pode mapear tensões elétricas de entrada de entre 0 e 5 volts para valores inteiros entre 0 e 1023.

Sintaxe:
`analogRead(pino)`

Parâmetros:
• `pino`: o número do pino.

Retorno:
• `int (0 a 1023)`

Projeto 1: Controle Pisca LED

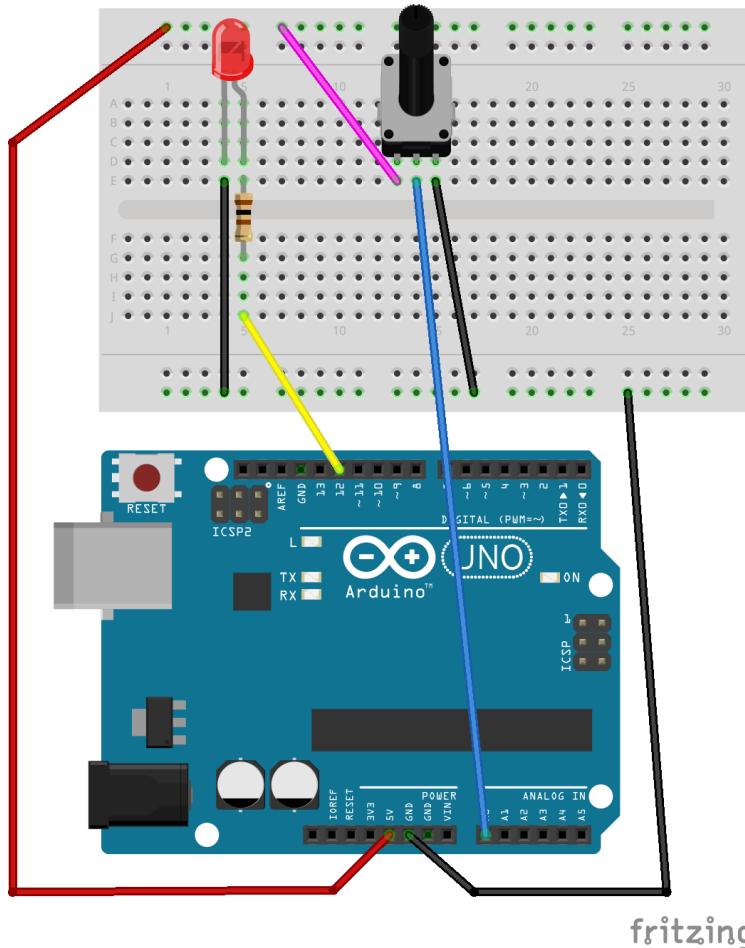


Projeto 1: Controle Pisca LED

→ 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

Atividades Locais Arquivos Ter, 9 de Mai, 22:08

< > Pasta pessoal Downloads oficina_IA_AD2017 Projeto_1

Recentes Pasta pessoal Área de trabalho Documentos

Projeto_1 Projeto_2 Projeto_3 Projeto_4 Projeto_5 slides.odp

Atividades Locais Arquivos Ter, 9 de Mai, 22:08

< > Pasta pessoal Downloads oficina_IA_AD2017 Projeto_2

Recentes Pasta pessoal Área de trabalho Documentos Downloads Imagens

Projeto_2.ino

Atividades Locais processing-app-Base Ter, 9 de Mai, 22:08

< > Pasta pessoal Downloads oficina_IA_AD2017 Projeto_2

Recentes Pasta pessoal Área de trabalho Documentos Downloads Imagens Música Vídeos Lixeira Computador Navegar na rede x-nautilus-desktop:/// Conectar a servidor

Projeto_2.ino

Projeto_2 | Arduino 2:1.0.5+dfsg2-4

File Edit Sketch Tools Help

Projeto_2

```
#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);
}
```

LDR: Resistência Dependente de Luz

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

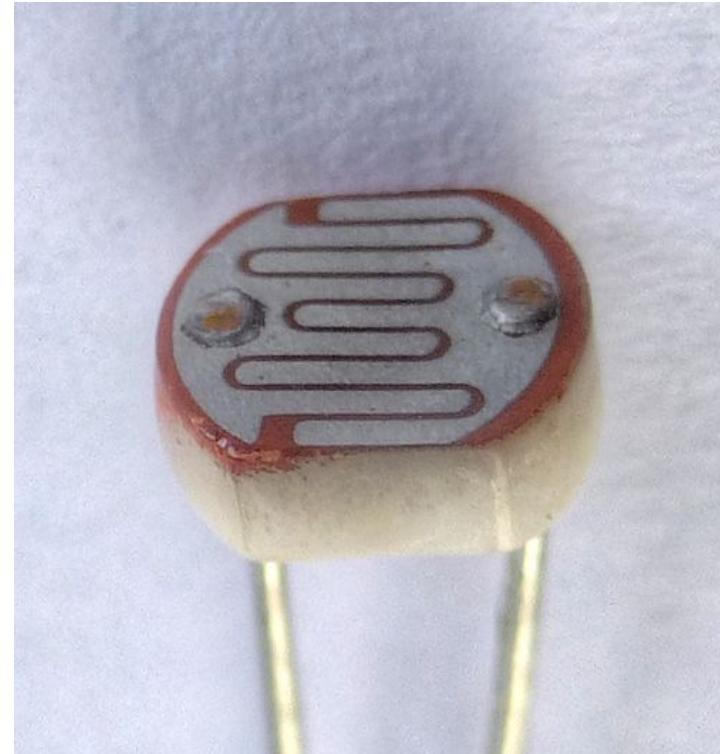
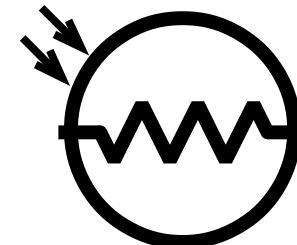


Foto de © Nevit Dilmen [CC BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0>) or GFDL (<http://www.gnu.org/copyleft/fdl.html>)], via Wikimedia Commons
Retirado de https://commons.wikimedia.org/wiki/File%3ALDR_1480405_6_7_HDR_Enhancer_1.jpg

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)

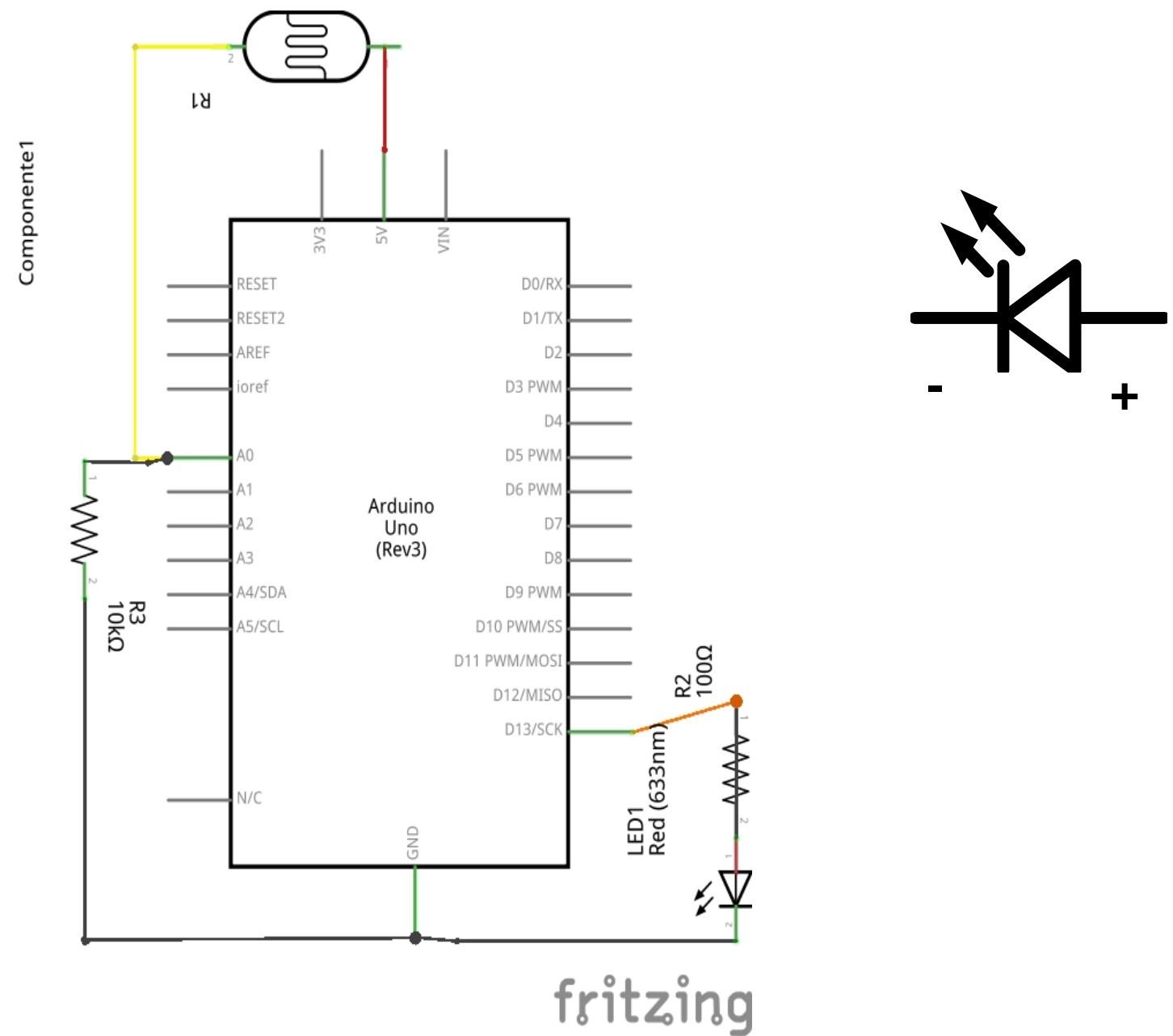
Parâmetros:

- Valor: qualquer número, texto ou variável.

Retorno:

- size_t (long)

Projeto 2: Controle LED com LDR

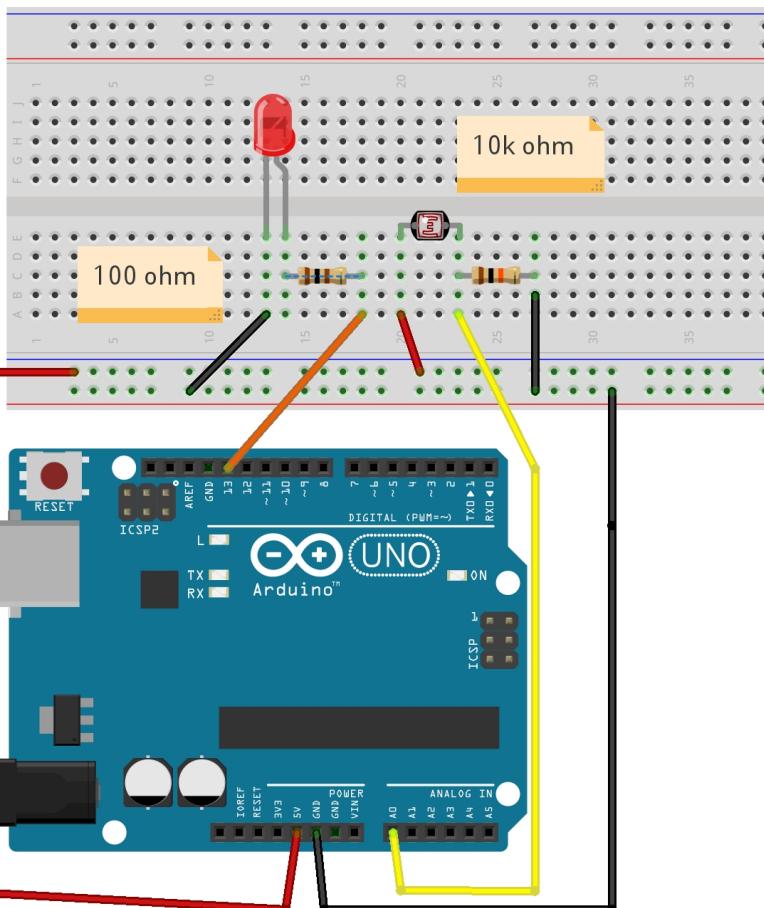


Projeto 2: Controle LED com LDR

→ Materiais adicionais:

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

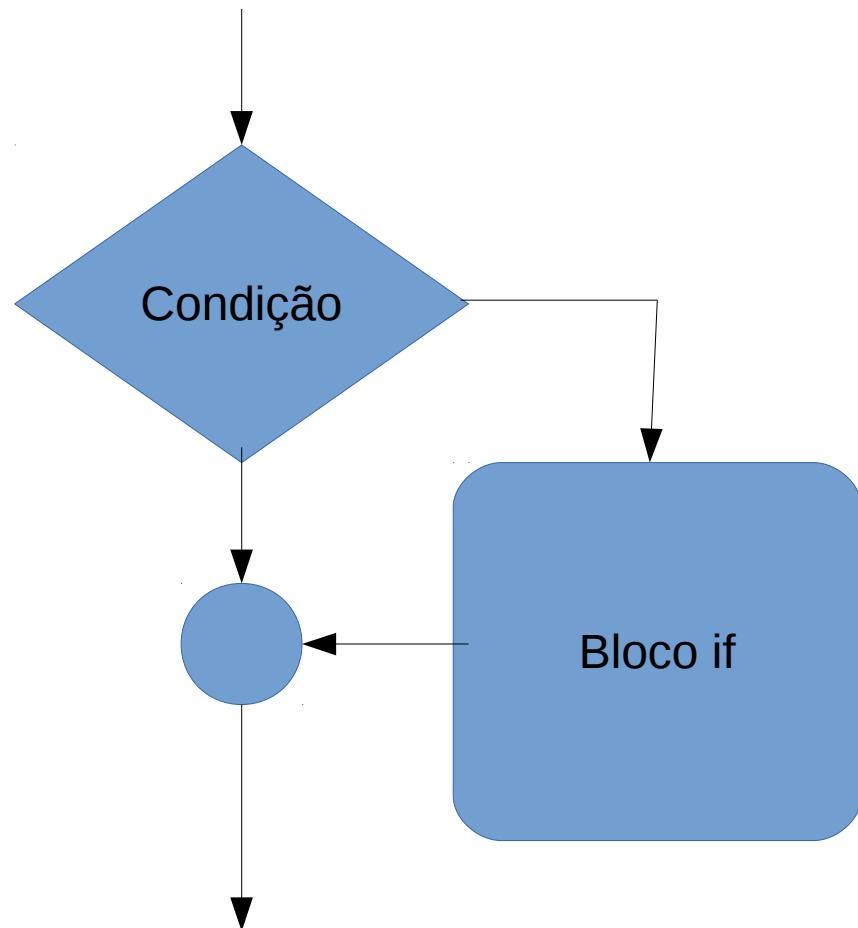
→ Montagem:



→ 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);  
}
```

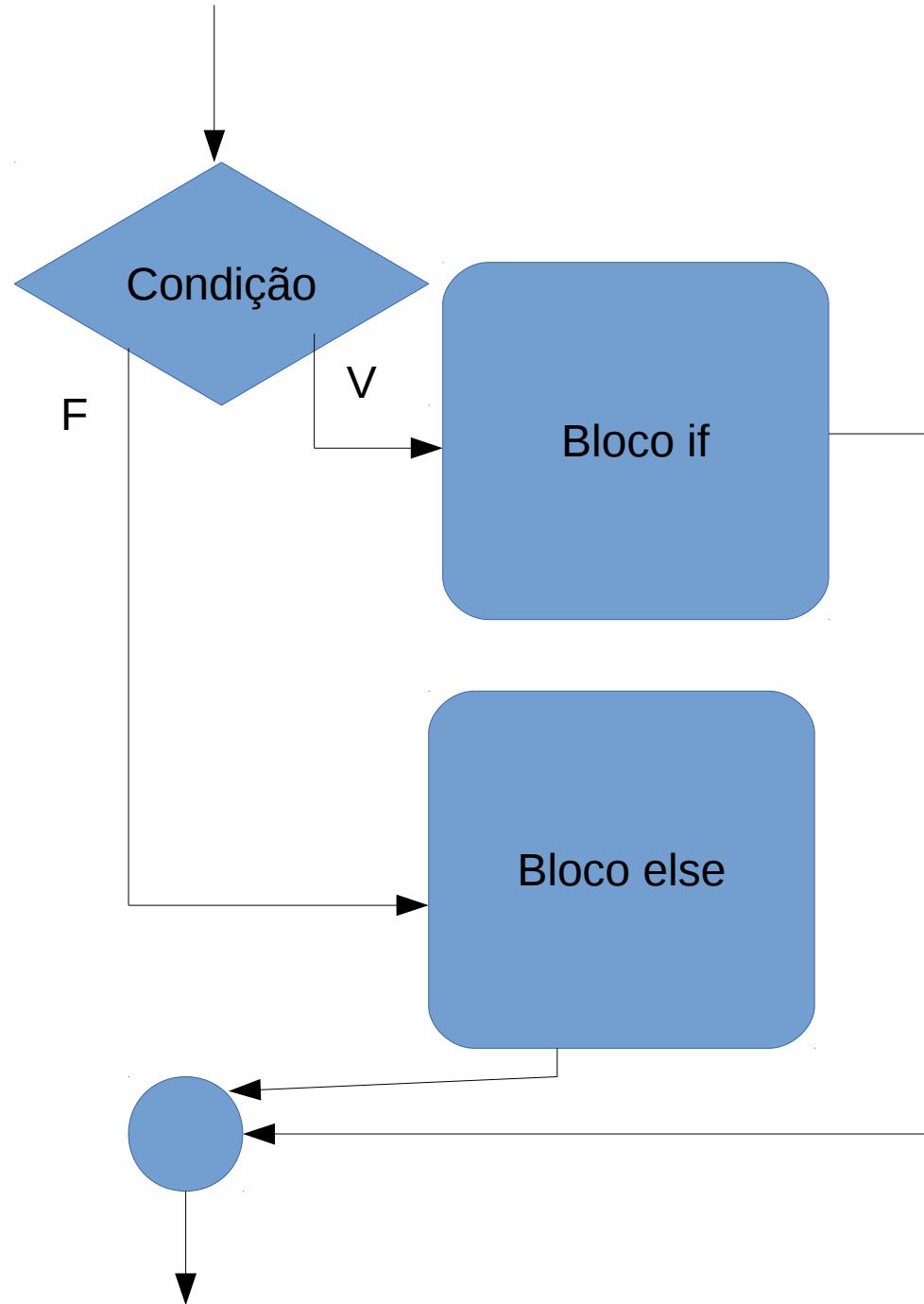
Desvio condicional: if



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

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

Desvio condicional: if-else



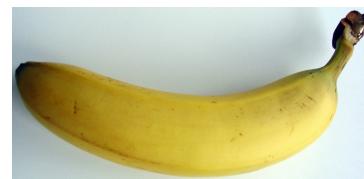
```
if (<condição>)
{
    <comandos V>;
}
else
{
    <comandos F>;
}
```

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

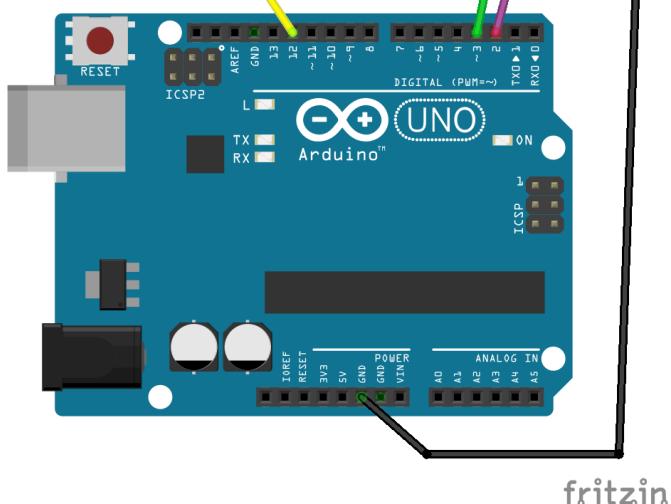
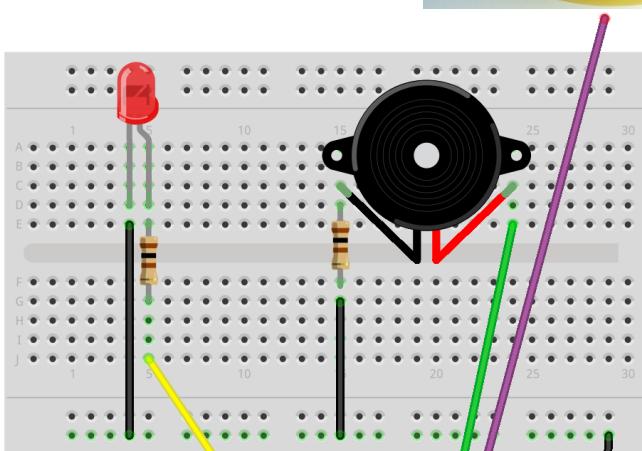
Projeto 3: Banana CapSense

→ 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);

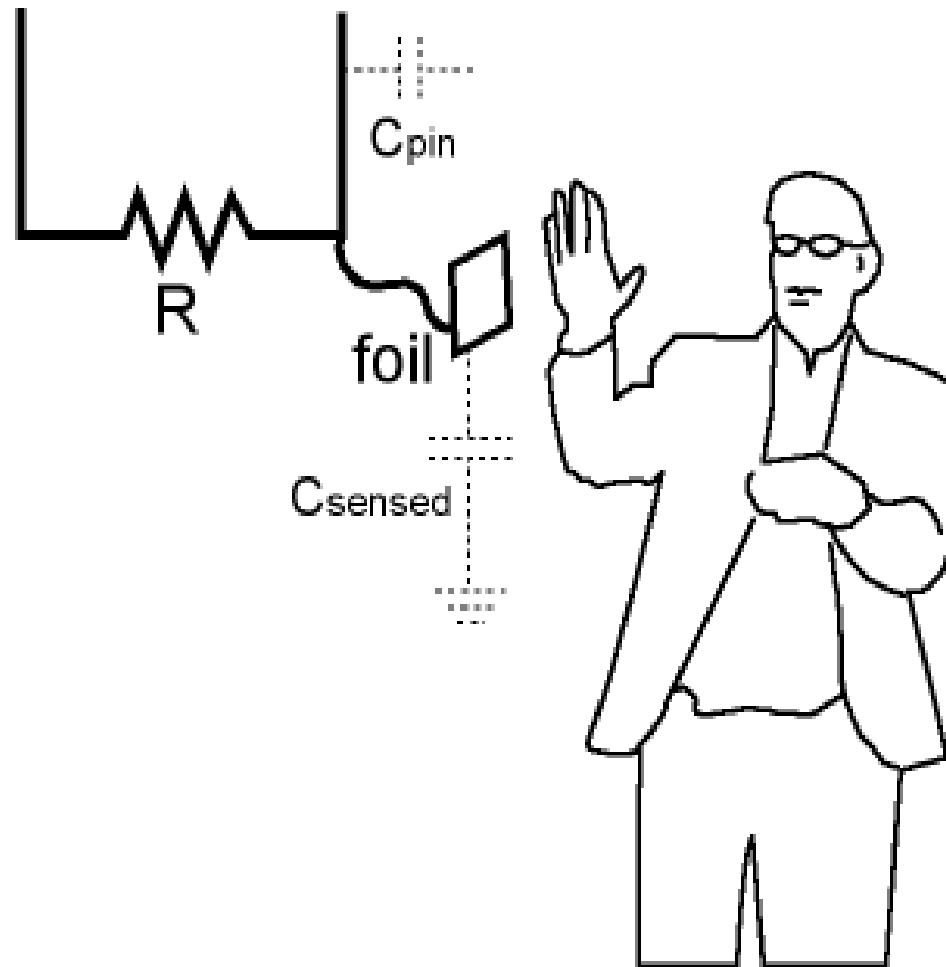
    if (valor_cap > 6) { // tocou na banana?
        digitalWrite(PINO_LED, HIGH);
        tone(PINO_BUZZER, NOTA_E7);
    }
    else {
        digitalWrite(PINO_LED, LOW);
        noTone(PINO_BUZZER);
    }
    delay(200);}
```

Fonte da imagem da banana:

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

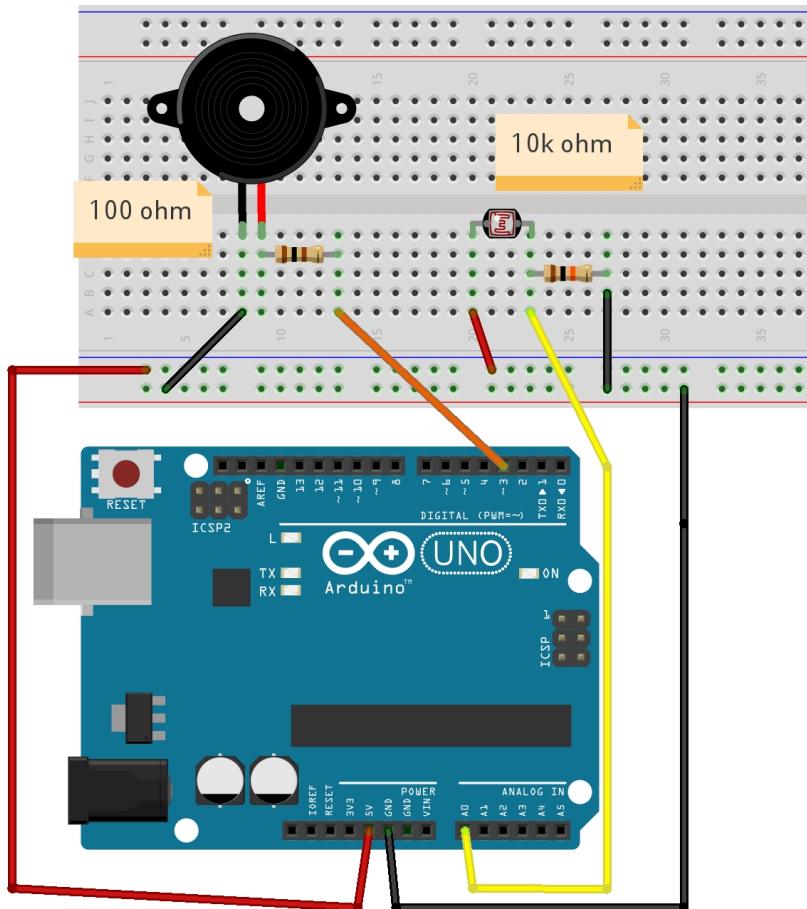
→ Leitura CapSense:
Estima-se **ciclos de
leitura** para quedas e
subidas de tensão

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;

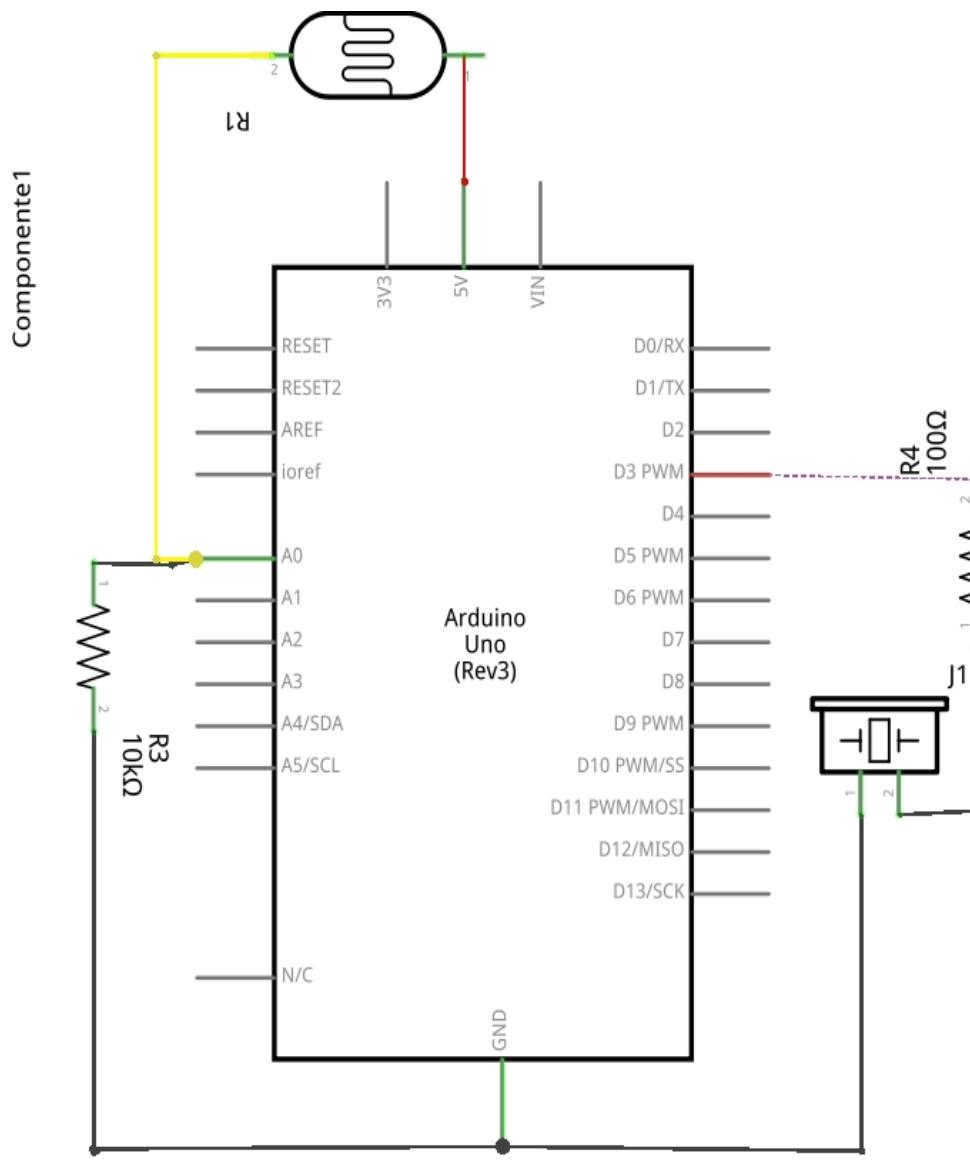
void setup() {
    pinMode(PINO_BUZZER, OUTPUT);
    pinMode(LDRpin, INPUT);
    Serial.begin(9600);
}

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

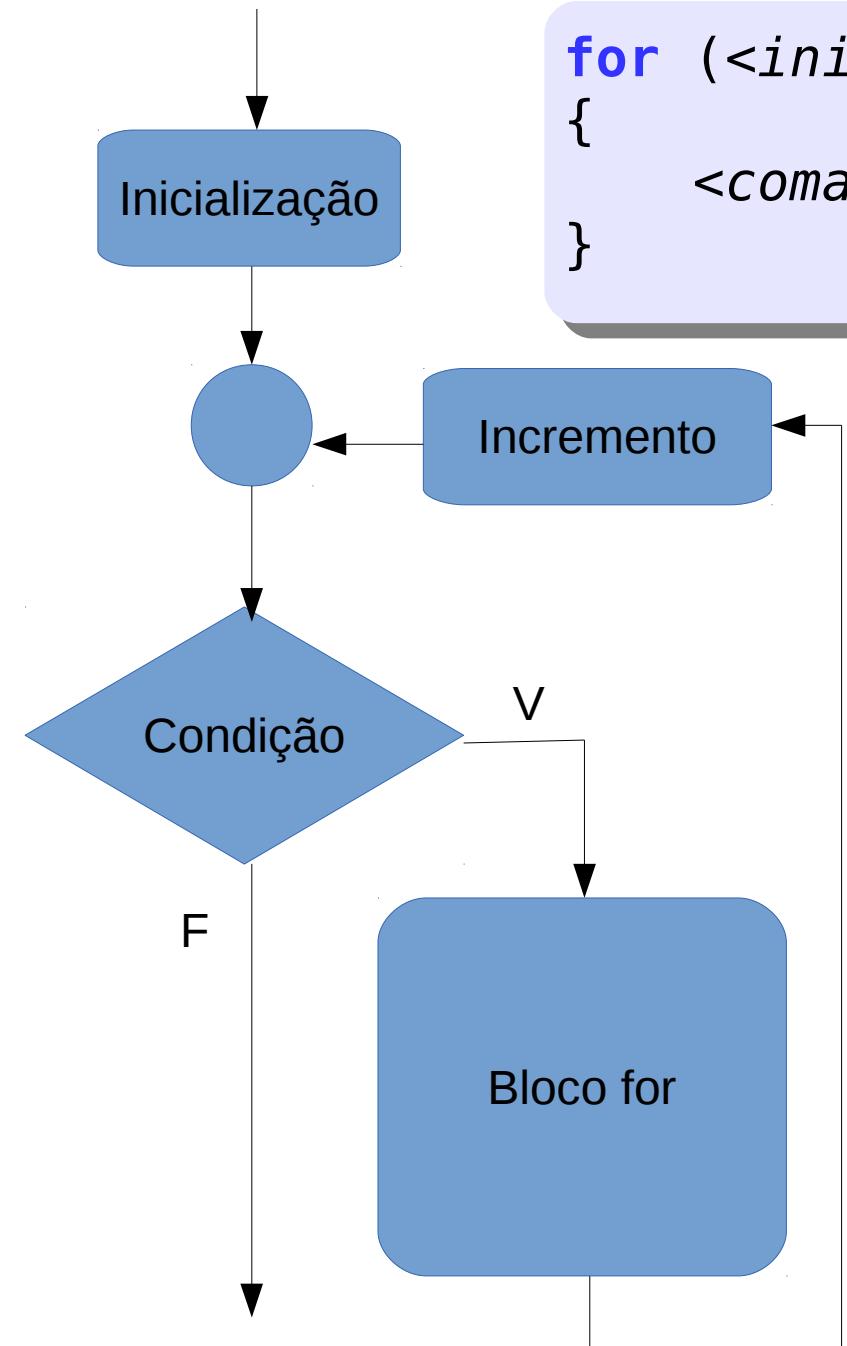
O que aprendemos?

- 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>;
}
```

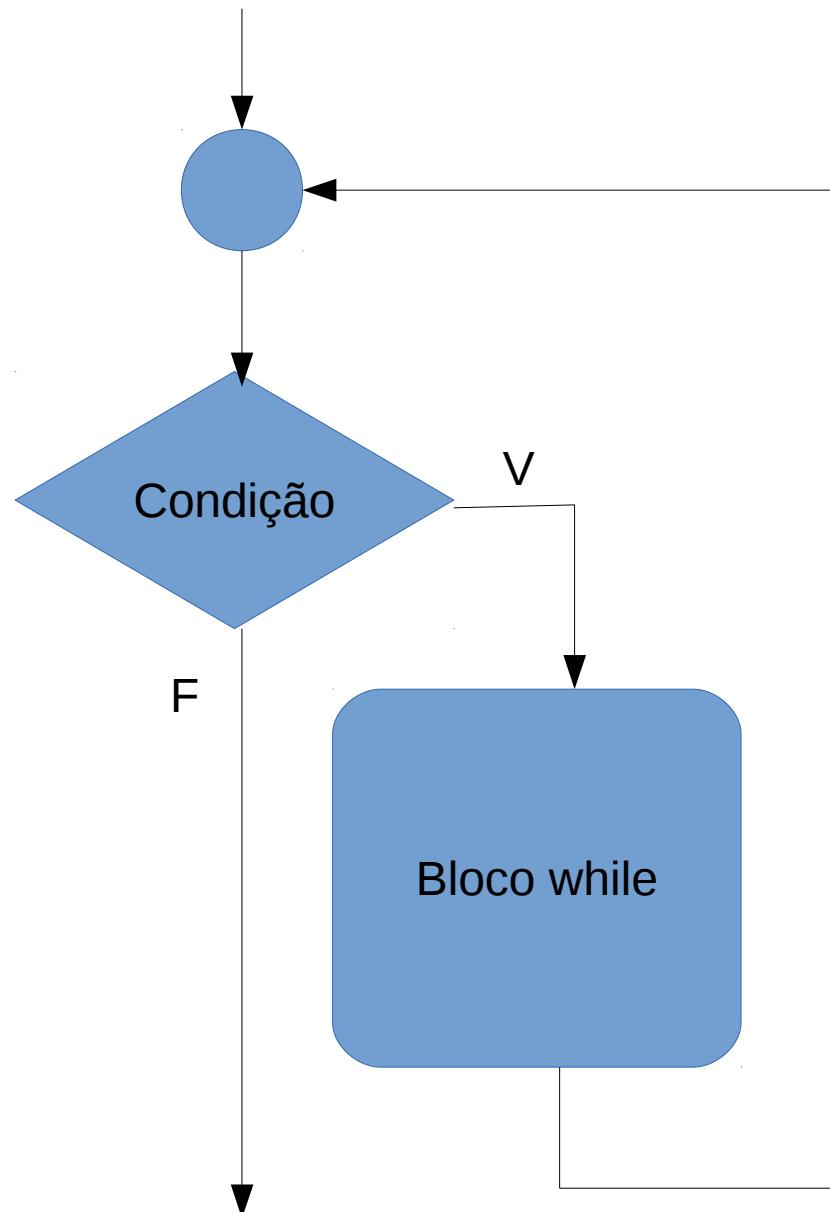
```
para <valor_inicial>
até <valor_final> faça:
    <comandos>;
fim-para
```

→ Melhor p/ contar repetições:

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

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

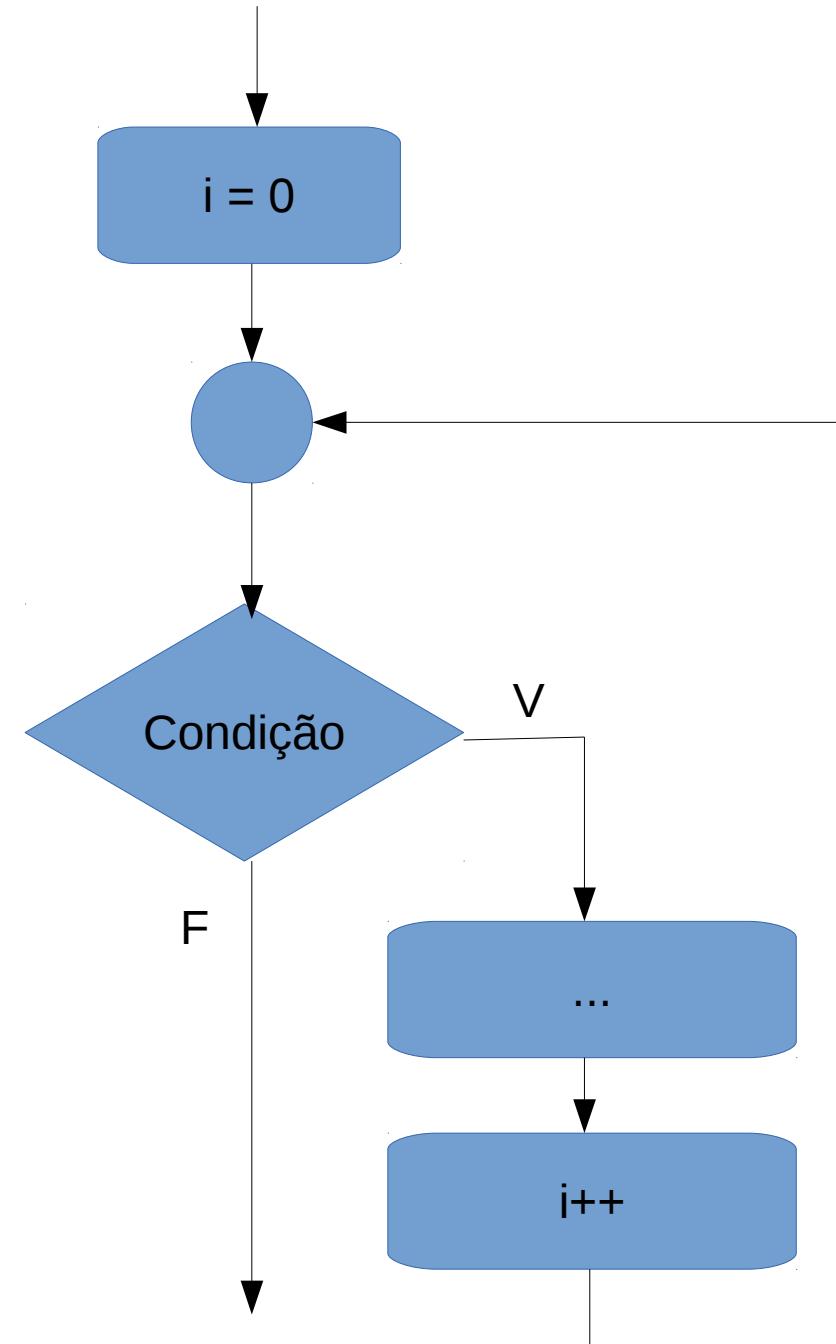
Laço de repetição: while



```
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++;  
}
```