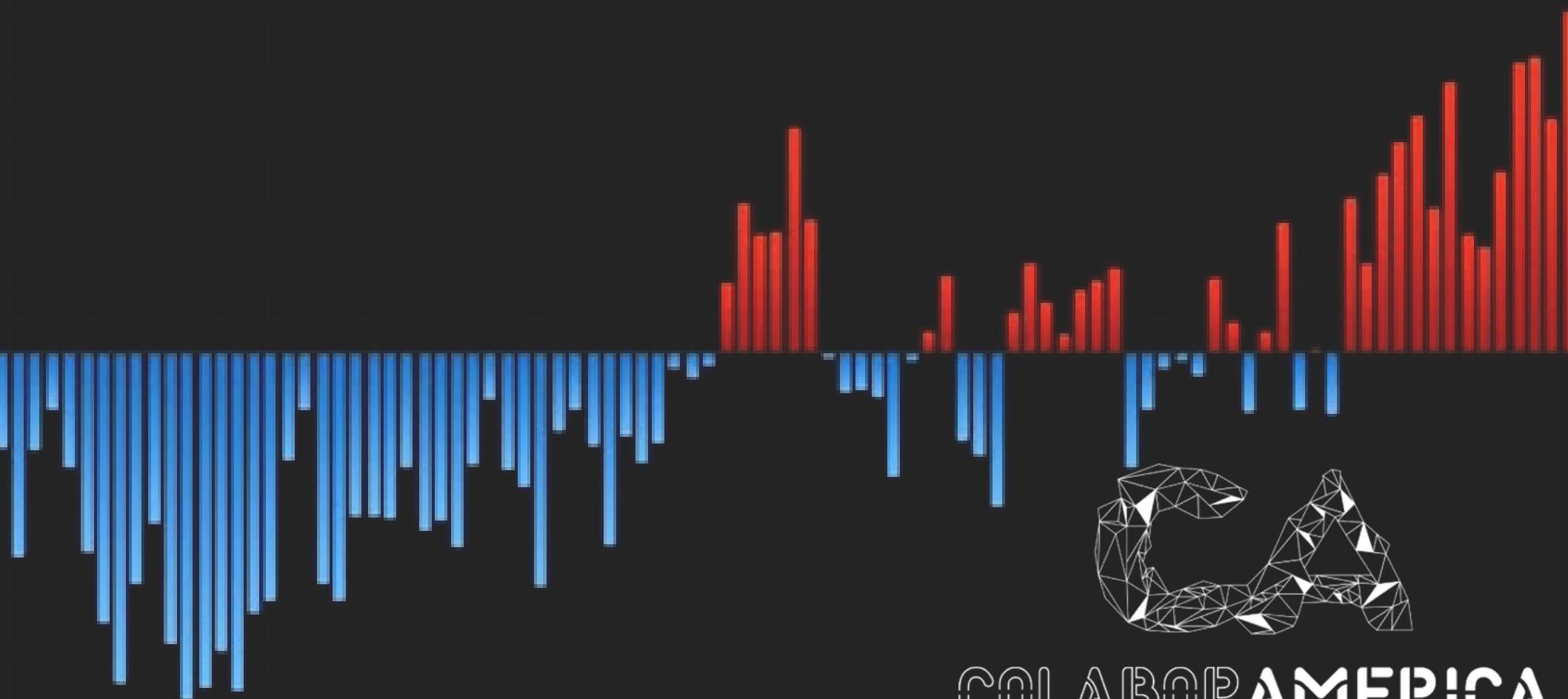


Hackeando Dados Ambientais



Material Disponível

Materiais do Curso com Acesso Livre

<https://github.com/smjacques/hackeando-dados-ambientais>

The screenshot shows the GitHub repository page for the user smjacques with the repository name hackeando-dados-ambientais. The page includes a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. Below the header, there's a search bar and a repository summary section with metrics like 1 commit, 1 branch, 0 releases, 1 contributor, and CC-BY-SA-4.0 license. The main content area displays the repository's contents, showing files like docs, esquema, lib, and pictures, each with a first commit message and timestamp.

This repository

Search

Pull requests Issues Marketplace Explore

smjacques / **hackeando-dados-ambientais**

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

material do workshop "Hackeando Dados Ambientais" no Colaboramerica 2017

Edit

Add topics

1 commit 1 branch 0 releases 1 contributor CC-BY-SA-4.0

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

File	Commit Message	Time
docs	first commit	12 minutes ago
esquema	first commit	12 minutes ago
lib	first commit	12 minutes ago
pictures	first commit	12 minutes ago

Livre ou Aberto?



Tecnologia Livre

- Liberdade dos usuários como uma questão de justiça.
- Caminho para uma nova sociedade.
- Promove as liberdades, a criação, a inovação com igualdade para a produção
- Código como patrimônio coletivo.
- Todos podem ser protagonistas.



Código Aberto (open source)

- Ponto de vista técnico
- Pragmático, evita questões éticas.
- Licenças não focam nas liberdades
- O ponto de partida não é mudanças sociais
- Conceitos alinhados à lógica de mercado

Ética ou Estética Hacker?

Steven Levy

- Compartilhamento
- Abertura
- Descentralização
- Livre acesso aos computadores
- Melhoria do mundo

Richard Stallman

- Alguém que gosta de desafios intelectuais
- Incômodo com restrições burocráticas
- Independente de compromisso ético com as outras pessoas.
- Estética no lugar da ética

Onde Estão os Hacker?

How to run a successful corporate hackathon

Ready to organize your **own**
hackathon?

Evangelize | *Train* | *Innovate* | *Transform*

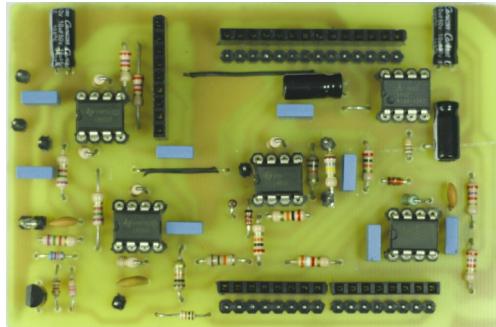
Onde Estão os Hacker?



Centro de Tecnologia Acadêmica

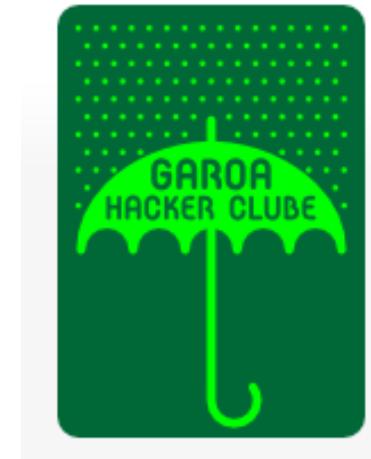


Nossos princípios

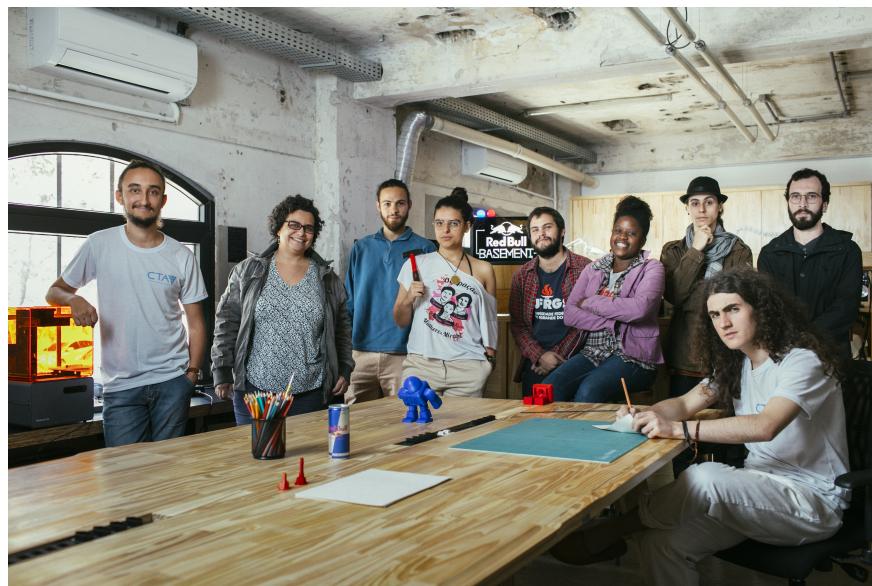


TropOS

Debian
GNU/Linux



Onde Estão os Hacker?



Fotos: Fabio Piva e Felibe Gabriel - Red Bull Content Pool

Onde Estão os Hacker?



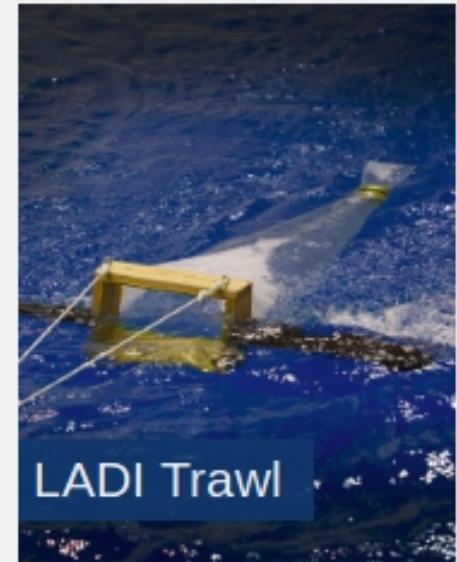
Have you built
or used a DIY
scientific
instrument?



CLEAR
guidelines for
designing
equitable
scientific tools



DIY
Microscopes



LADI Trawl

We want to hear from
you! We have a survey to
see how people are
building, using, and
improving the
technologies and
protocols we make
public.

When we design
scientific instruments, we
think of about users that
are scientists with
degrees in well-funded
institutions, but also rural
Newfoundlanders, who
also have research
questions and a right to
answer them. To this end,
we have several
guidelines for how we
design and build our
tools.

A list of different do-it-
yourself microscope
designs!

The LADI trawl is an
open source, scientific
surface trawl for
monitoring marine
plastics. You can build
your own for \$500 or less.

Onde Estão os Hacker?

Oceanography For Everyone



OpenCTD

The OpenCTD is a low-cost, open-source oceanographic instrument for measuring conductivity, temperature, and depth down a vertical profile. The CTD is the workhorse of oceanography, allowing scientist to generate water column profiles.

[OpenCTD GitHub Repository](#)

[Rockethub Crowdfunding Campaign](#)



Niskin3D

Niskin3D is a low-cost, 3D printable Niskin bottle that allows users to take discrete water samples at specific depths or in specific environmental conditions. Niskin3D has been designed to integrate with the OpenROV, but it is adaptable to a variety of platforms.

[Niskin3D GitHub Repository](#)

[Niskin3D Original Project Description](#)



BeagleBox

The BeagleBox is a tough, single-board-computer-powered field laptop designed to fit into a Pelican case. It's suitable for basic computing and adaptable to a variety of needs.

[BeagleBox GitHub Repository](#)

[BeagleBox Original Project Description](#)

Onde Estão os Hacker?



I. The Problem

Communities lack access to **the tools and techniques** needed to participate in decisions being made about their communities, especially

II. The Collaboration

We are an open network of **community organizers**, educators, technologists and researchers working to create low cost solutions for

III. The Solution

Join us today, as we work together to build and inspire a community of DIY activists and explorers using simple tools to build a **growing body**

Onde Estão os Hacker?



Tecnologias Livres e Emancipação



Fotos:

<http://www.agroecologia.org.br> - <https://litci.org/> - <https://www.carosamigos.com.br>

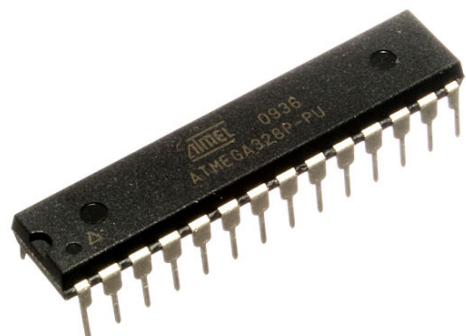
O Projeto Arduino

Microcontrolador - Circuito integrado (CI) que incorpora várias funcionalidades

“computador de um único chip”

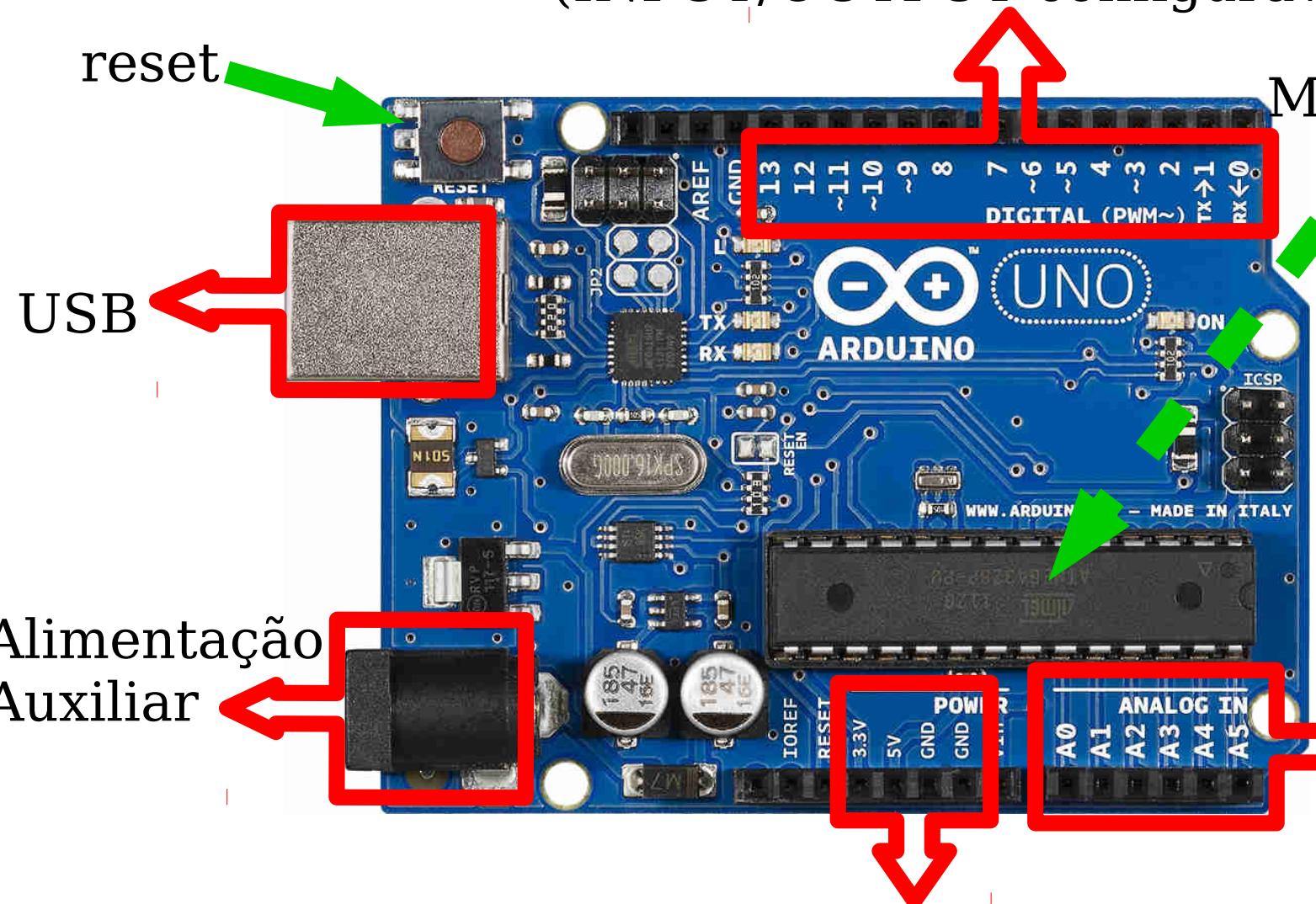
Utilizado com frequência em sistemas embarcados (carros, eletrodomésticos, automação residencial, etc)

Computação Física: Sistema que pode interagir com seu ambiente por meio de circuitos eletrônicos e softwares.



A Placa Arduino

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



Microcontrolador
Atmega328

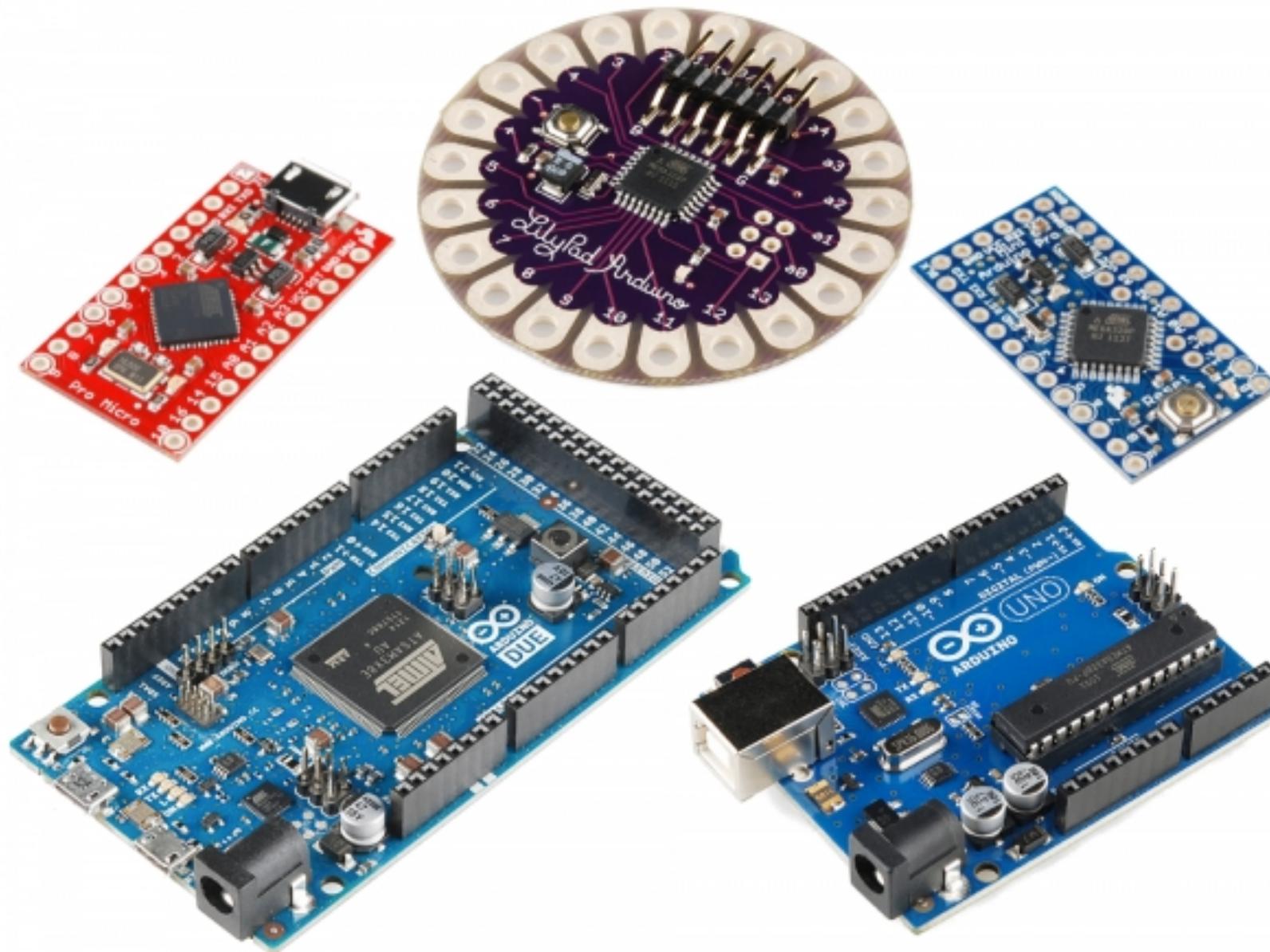
- ✓ Clock: 16 MHz
- ✓ EEPROM: 1 KB (bootloader)
- ✓ SRAM: 2 KB
- ✓ Flash: 32 KB ("pendrive")
- ✓ 5 V - 50 mA

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.



Ambiente de Desenvolvimento Instalando e Usando a IDE

Intervace de Programação (IDE)

Download the Arduino IDE



ARDUINO 1.8.5

The open-source Arduino Software ([IDE](#)) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows Installer

Windows ZIP file for non admin install

Windows app

Mac OS X 10.7 Lion or newer

Linux 32 bits

Linux 64 bits

Linux ARM

[Release Notes](#)

[Source Code](#)

[Checksums \(sha512\)](#)

Disponível em: <https://www.arduino.cc/en/Main/Software>

Fazendo download do software (IDE)

Download the Arduino IDE

- Escrever o código do programa
- Salvar o código do programa
- Compilar um programa
- Transportar o código compilado para a placa do Arduino



ARDUINO 1.8.5

The Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other

open source software. This software can be used with any Arduino Board. Refer to the [Getting Started](#) page for Installation instructions.

[Windows Installer](#)

[Windows ZIP file for non admin install](#)

[Windows app](#)

[Mac OS X 10.7 Lion or newer](#)

[Linux 32 bits](#)

[Linux 64 bits](#)

[Linux ARM](#)

[Release Notes](#)

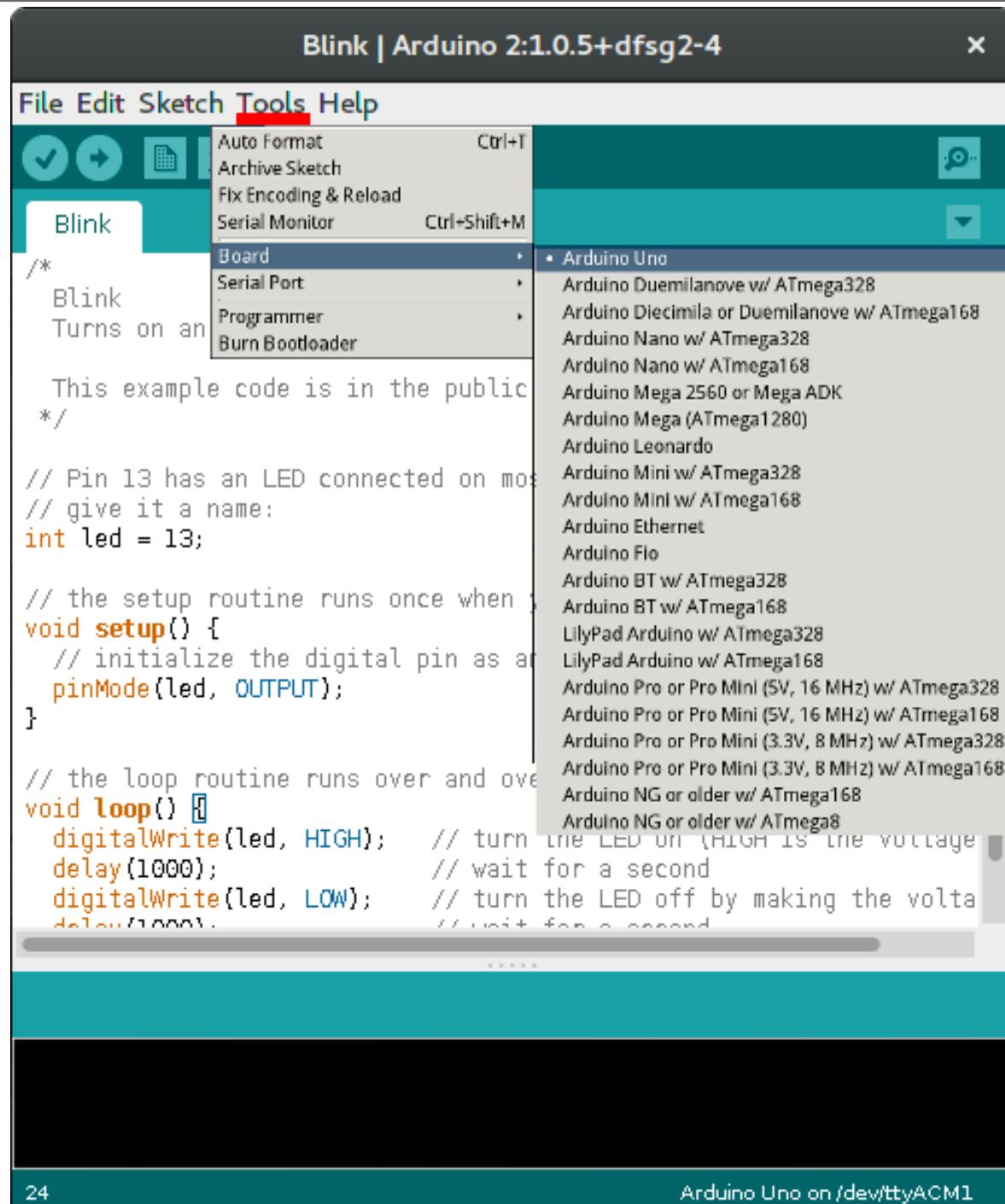
[Source Code](#)

[Checksums \(sha512\)](#)

Selecionando Modelo da Placa

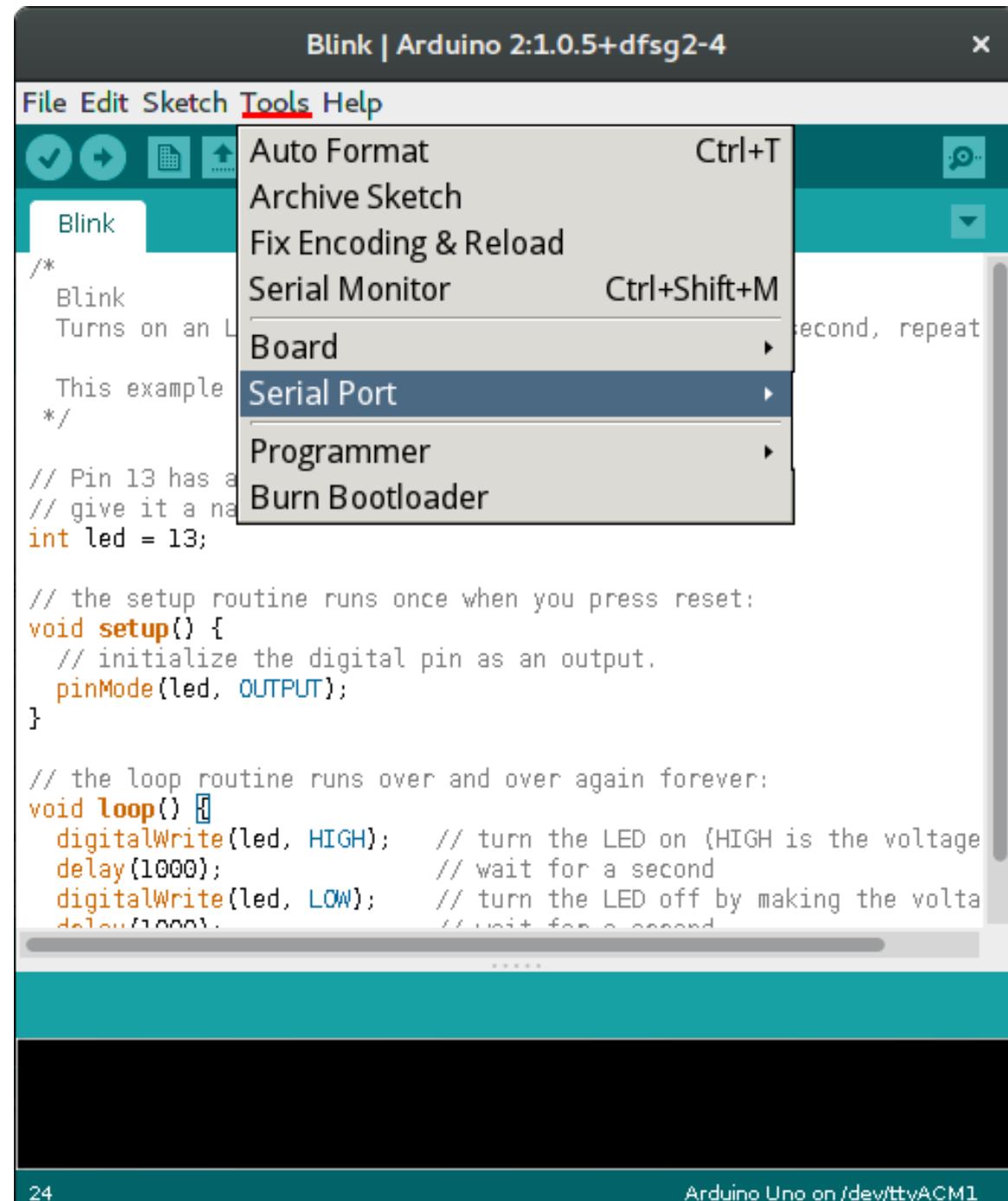
Antes de programar a placa é preciso selecionar o modelo usado usando os seguintes passos:

Tools → Board → Modelo do Arduino



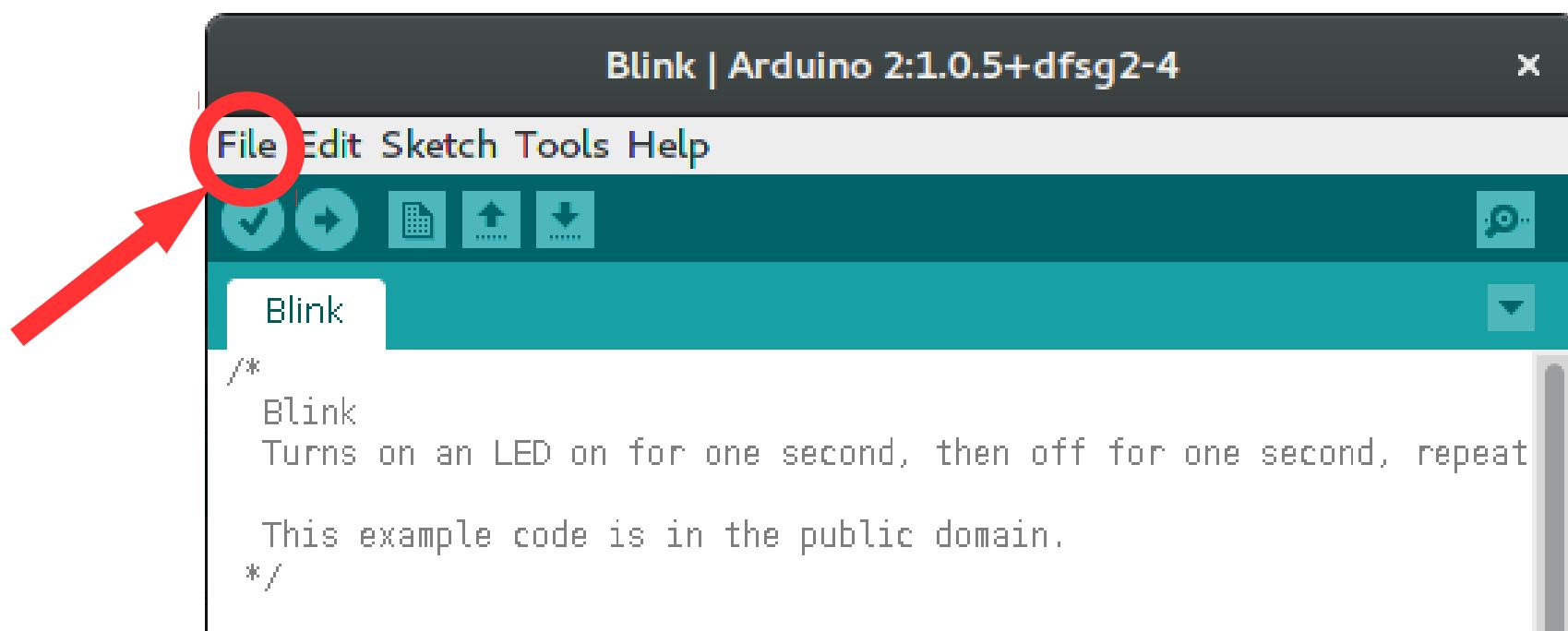
Selecionando a Porta Serial

Nome e número da Porta Serial pode variar de acordo com o modelo de Arduino e/ou quando o cabo USB é reconectado ao computador

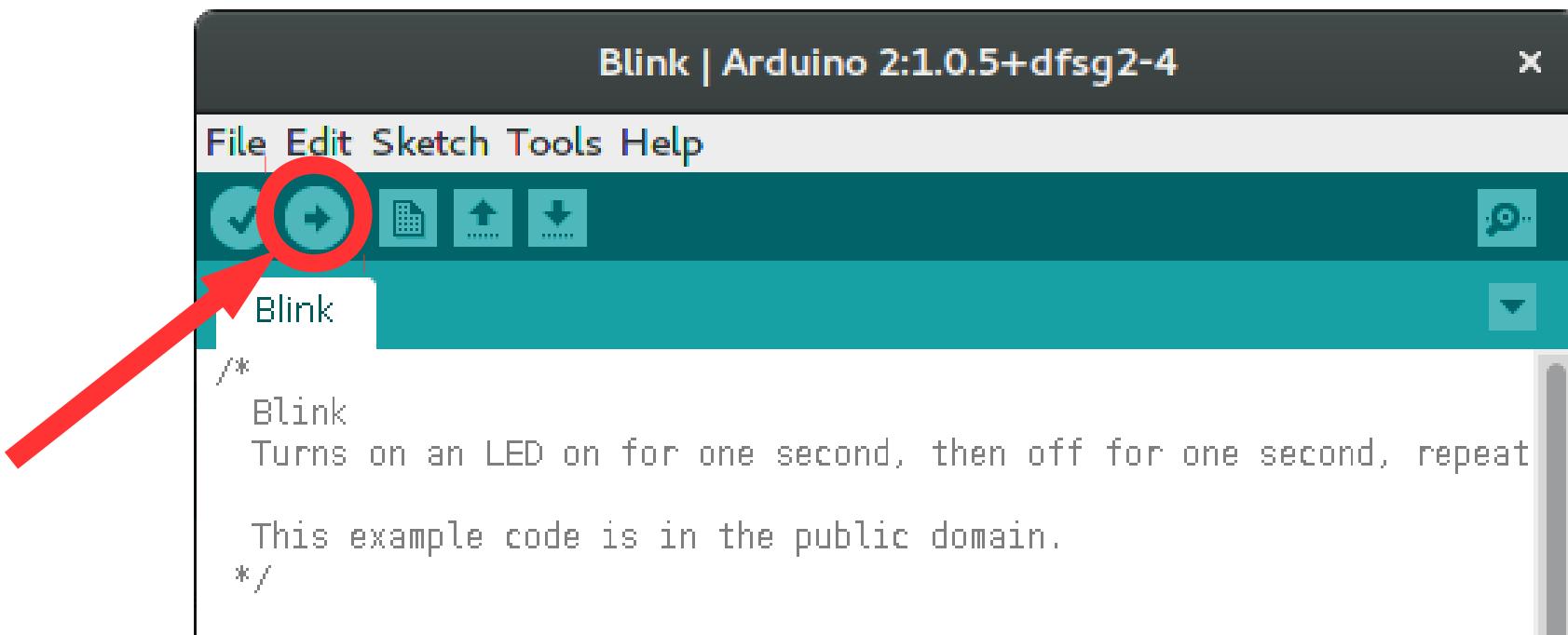


Testando: Carregue o exemplo Blink

File → Examples → Basics → Blink



Fazendo upload para placa



```
sInput,  
ngth, iN;  
dblTemp;  
ain = true;  
  
again) {  
= -1;  
in = false;  
:line(cin, sInput);  
stem("cls");  
ingstream(sInput) >> dblTemp;  
ength = sInput.length();  
(iLength < 4) {  
again = true;  
continue;  
else if (sInput[iLength - 3] != '.') {  
again = true;  
continue;  
while (++iN < iLength) {  
if (isdigit(sInput[iN])) {  
continue;  
else if (iN == (iLength - 3)) {  
continue;
```

O Código

Estrutura principal

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

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

- Em alguns SO a IDE informa Sketch Arduino vazio!
- Já está sintaticamente correto (pode compilar).
- Precisa de comandos dizendo o que fazer.

Estrutura principal

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

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

setup(): onde devem ser definidas algumas configurações iniciais do programa. Executa uma única vez.

loop(): função principal do programa. Fica executando indefidamente.

Variáveis e Funções

Declarando variáveis (sintaxe):

```
int      led    =    13;  
         ↓     ↓     ↓  
<tipo> <nome>;  
<tipo> <nome> = <valor atribuído a pino>;
```

Tipos de Variáveis

int - Armazena números inteiros e ocupa 16 bits de memória (2 bytes). A faixa de valores é de -32.768 a 32.767.

unsigned int - O mesmo que *int*, porém a faixa de valores válidos é de 0 a 65.535.

void - Usado geralmente para informar que uma função não retorna nenhum valor. Indica tipo indefinido.

float - Armazena valores de ponto flutuante (com vírgula) e ocupa 32 bits (4 bytes) de memória. A faixa de valores é de -3.4028235E+38 até 3.4028235E+38

boolean - valor 1 (true) ou 0 (false). Ocupa um byte de memória.

char - Pode ser uma letra ou um número. A faixa de valores válidos é de -128 a 127. Ocupa 1 byte de memória.

unsigned char - O mesmo que o char, porém a faixa de valores válidos é de 0 a 255.

Tipos de Variáveis

long - Armazena números de até 32 bits (4 bytes). Valores de -2.147.483.648 à 2.147.483.647.

unsigned long - O mesmo que o long, mas a faixa de valores é de 0 até 4.294.967.295.

short - Armazena número de até 16 bits (2 bytes). A faixa de valores é de -32.768 até 32.767.

byte - Ocupa 8 bits de memória. A faixa de valores é de 0 a 255.

word - O mesmo que um *unsigned int*.

Tipos de Funções

Chamando/executando funções disponíveis (sintaxe):

pinMode(led, OUTPUT);



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

Tipos de Funções

```
// 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  
}
```

Tipos de Funções

- Funções básicas mais comuns do Arduino:

Digitais

pinMode(**num_do_pino**, INPUT | OUTPUT)
digitalWrite(**num_do_pino**, LOW | HIGH)
digitalRead(**num_do_pino**)

Analógicas

analogRead(**num_do_pino**)
analogWrite(**num_do_pino**, **valor_PWM**)

delay(**milliseconds**)

+info: consulte a documentação!

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

- Structure** (Green header):
 - `setup()`
 - `loop()`
- Variables** (Blue header):
 - 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** (Orange header):
 - Digital I/O**
 - `pinMode()`
 - `digitalWrite()`
 - `digitalRead()`
 - Analog I/O**
 - `analogReference()`
 - `analogRead()`
 - `analogWrite() - PWM`
 - Due only**
 - `analogReadResolution()`
 - `analogWriteResolution()`
 - Advanced I/O**
 - `tone()`
 - `noTone()`
 - `shiftOut()`
 - `shiftIn()`

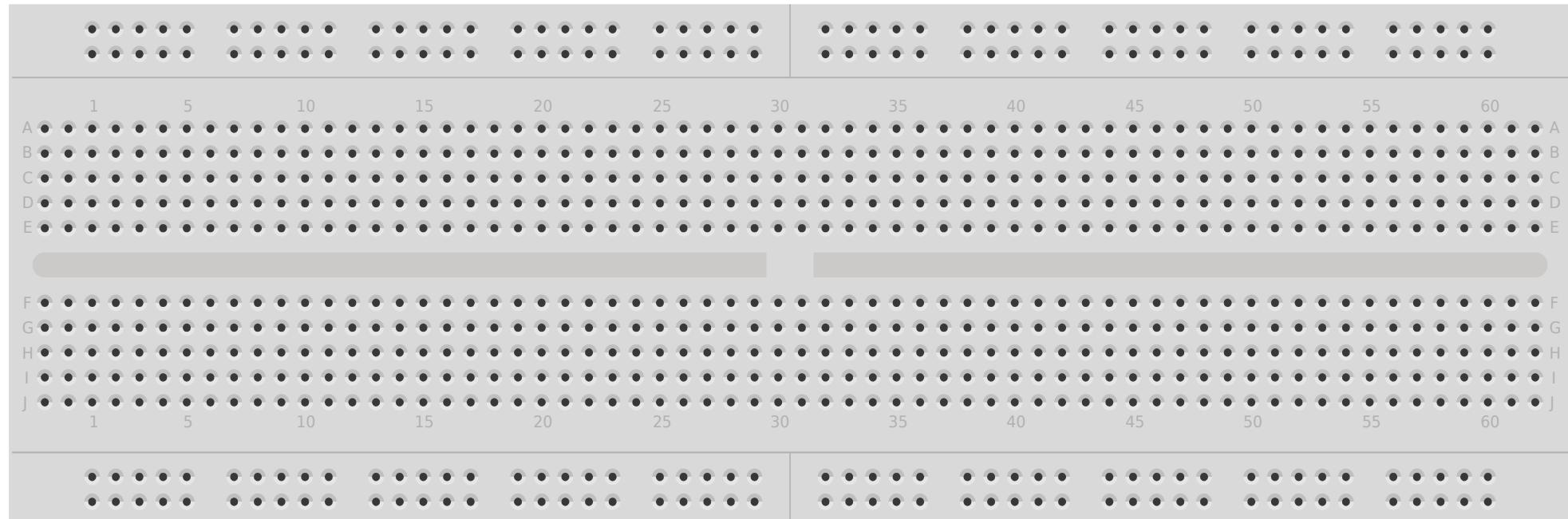
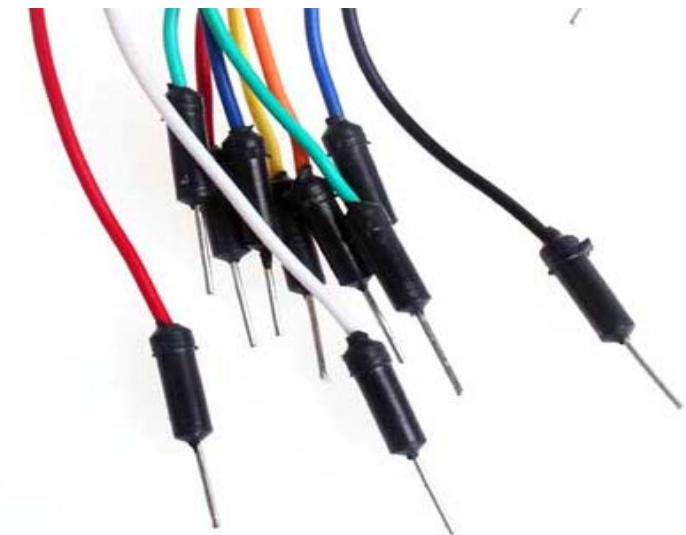
Prototipagem

```
sInput,  
ngth, iN;  
dblTemp;  
ain = true;  
  
again) {  
= -1;  
in = false;  
line(cin, sInput);  
tem("cls");  
ingstream(sInput) >> dblTemp;  
ength = sInput.length();  
(iLength < 4) {  
again = true;  
continue;  
else if (sInput[iLength - 3] != '.') {  
again = true;  
continue;  
while (++iN < iLength) {  
if (isdigit(sInput[iN])) {  
continue;  
else if (iN == (iLength - 3)) {  
continue;
```

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/>

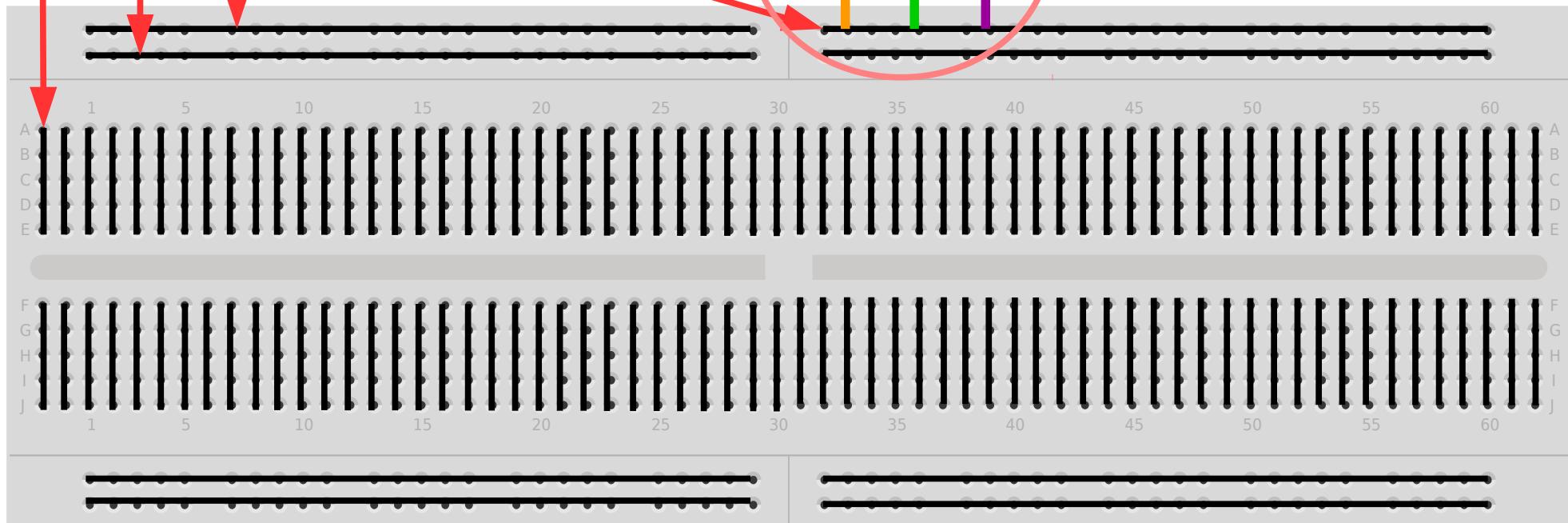


Fonte da imagem:
<http://fritzing.org/home/>

Protoboard: Como funciona?

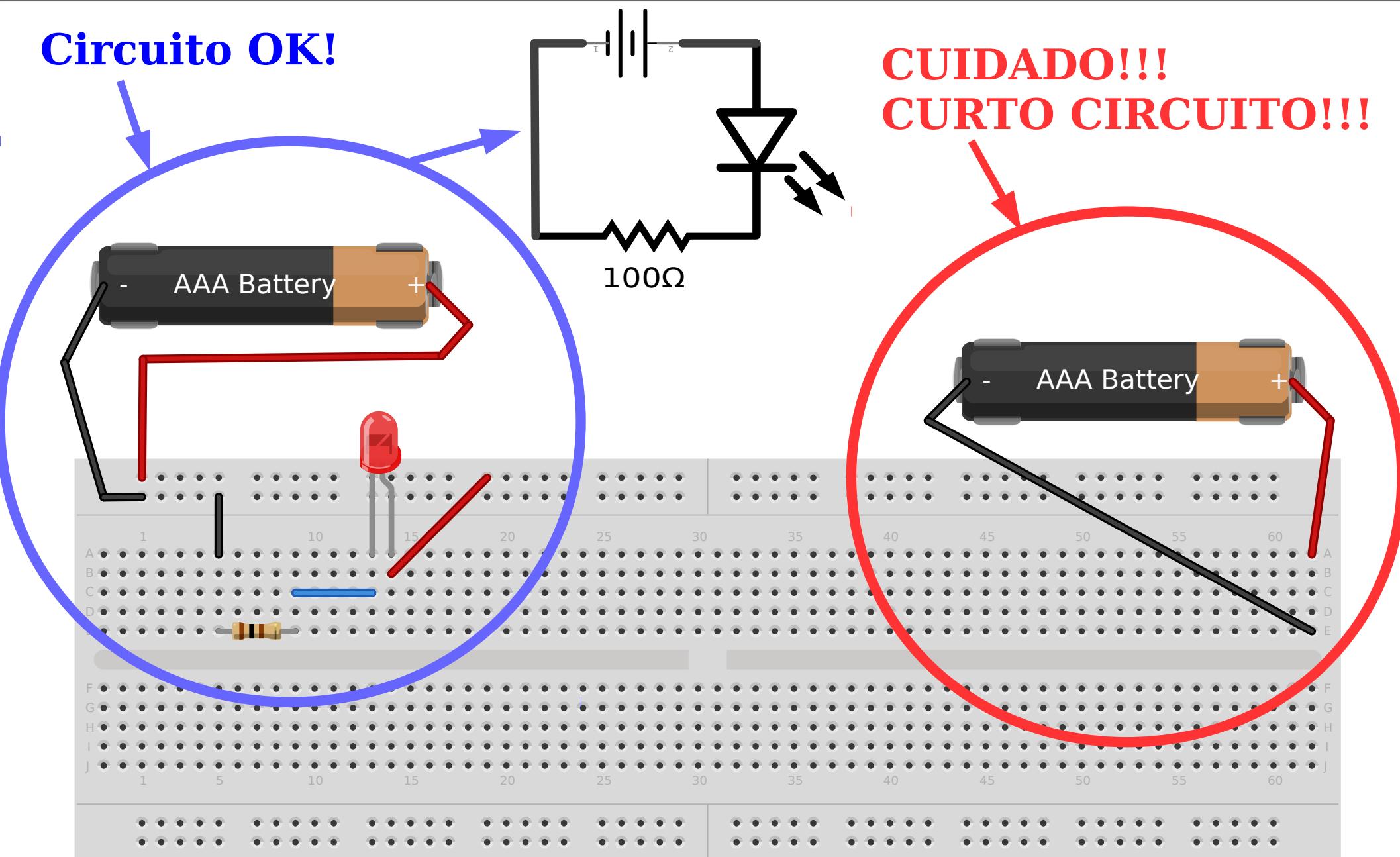
- Consiste num conjunto de barramentos isolados entre si;
- Um barramento equivale à uma junção de dois ou mais fios;

barramentos



Protoboard: Exemplos

Circuito OK!





“Mão na Massa”

Projeto 0: Pisca LED externo

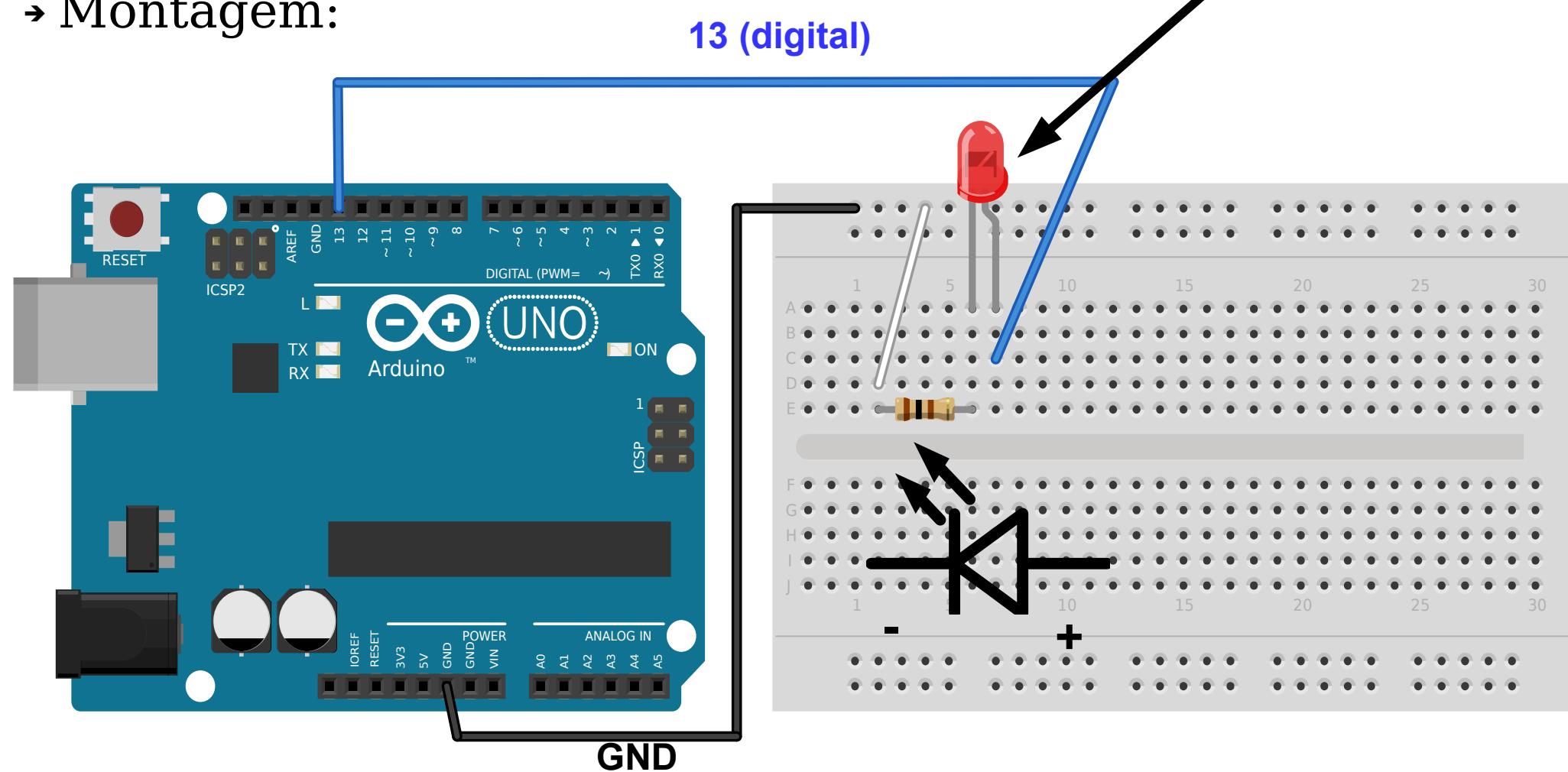
→ Materiais:

- ✓ 1 LED
- ✓ 1 resistor de 100 ohms

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

→ Montagem:

13 (digital)



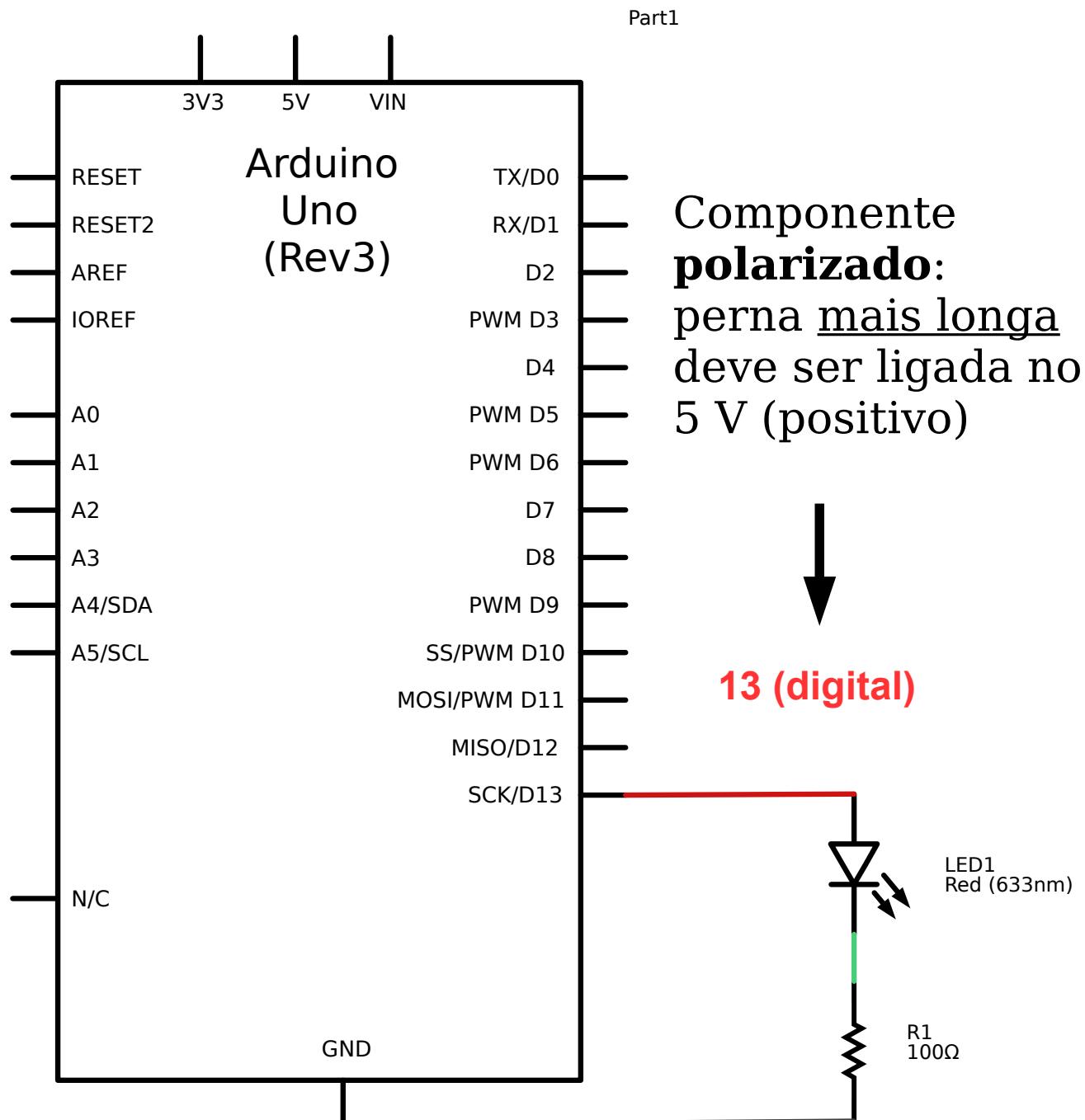
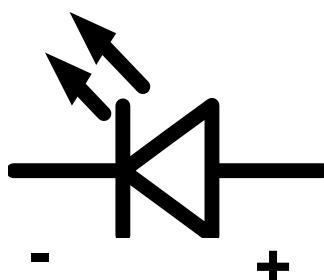
Projeto 0: Pisca LED externo

→ Materiais:

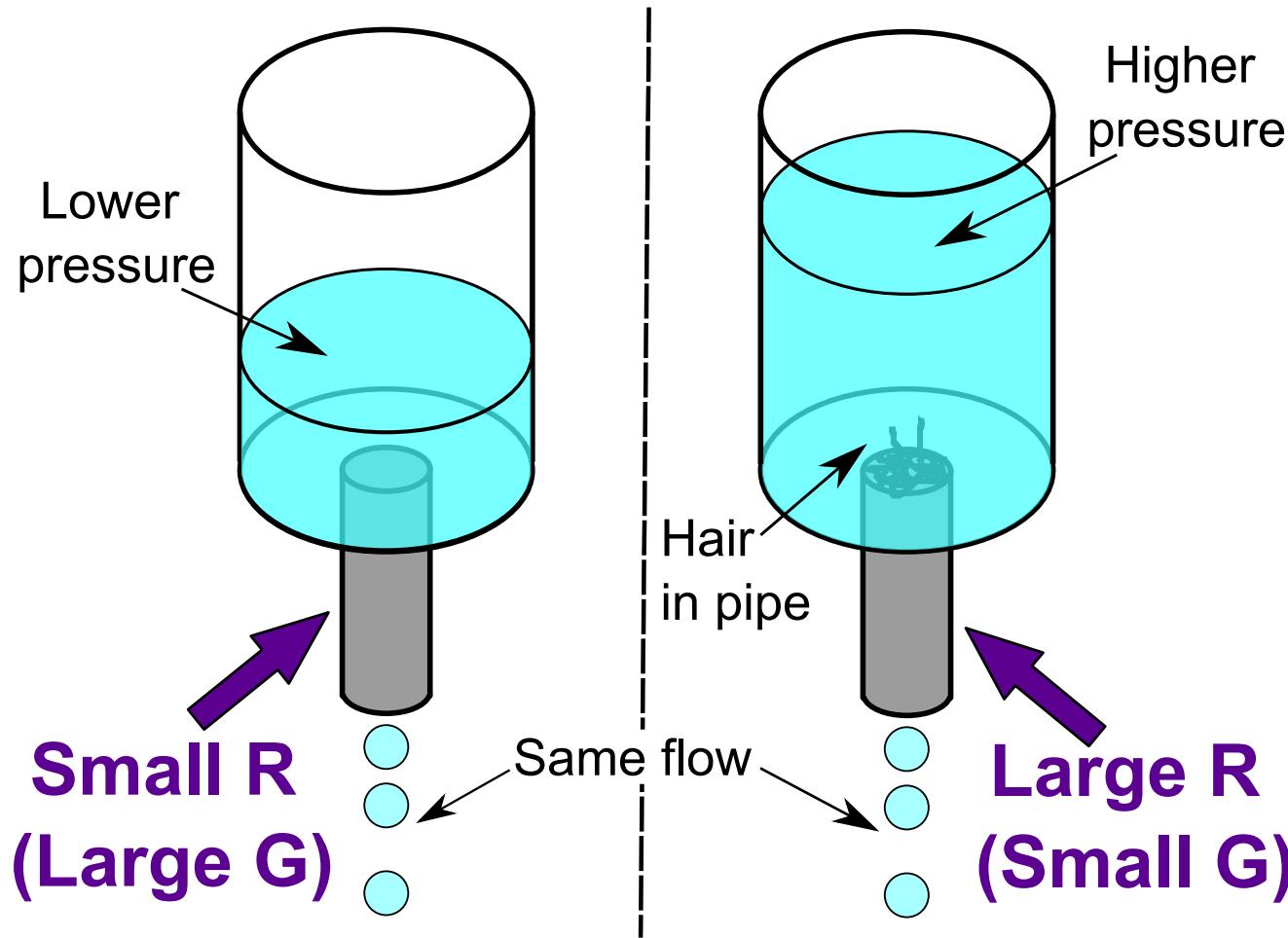
- ✓ 1 LED
- ✓ 1 resistor de 100 ohms
- ✓ fios jumper

→ Montagem:

Light Emitter Diode



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>

Circuitos elétricos

Lei de Ohm (Ω)

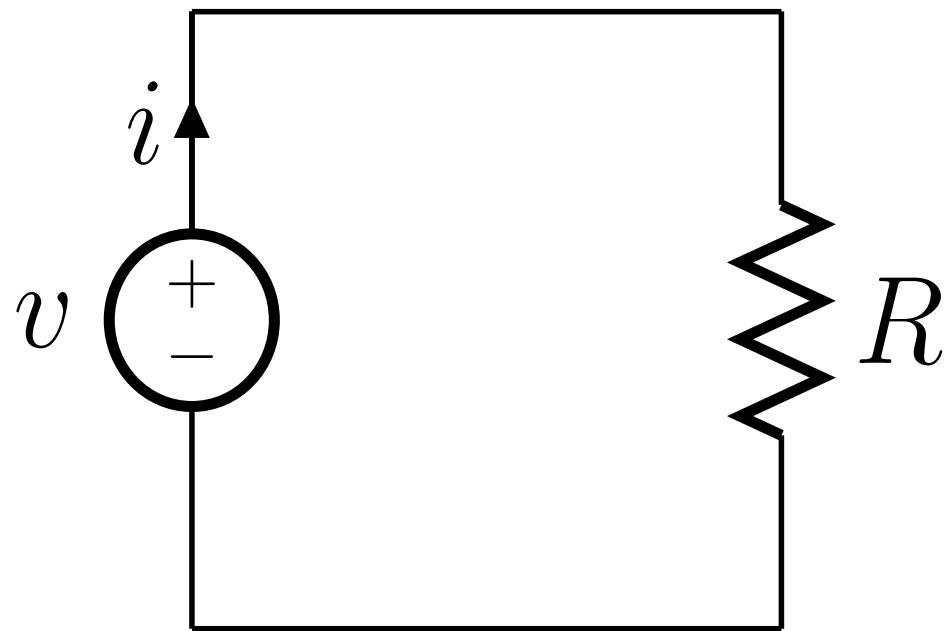
Como medimos a resistência oferecer à passagem da corrente elétrica.

$$R = \frac{V}{I}$$

Resistores causam uma queda de tensão na região do circuito em questão, mas nunca uma queda de corrente.

A corrente elétrica que entra em um terminal do resistor é a mesma corrente que sai pelo outro terminal, porém existe uma queda de tensão.

Circuitos elétricos



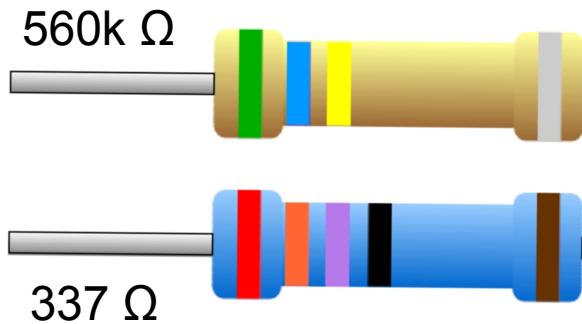
i : Corrente elétrica

V : Tensão elétrica

R : Resistência elétrica

Circuitos elétricos

Código de cores dos resistores



Cor	Dígito	Multiplicador	Tolerância
Prata	-	$\times 0,01$	$\pm 10\%$
Dourado	-	$\times 0,1$	$\pm 5\%$
Preto	0	$\times 1$	-
Marrom	1	$\times 10$	$\pm 1\%$
Vermelho	2	$\times 100$	$\pm 2\%$
Laranja	3	$\times 1K$	-
Amarelo	4	$\times 10K$	-
Verde	5	$\times 100K$	$\pm 0,5\%$
Azul	6	$\times 1M$	$\pm 0,25\%$
Violeta	7	$\times 10M$	$\pm 0,1\%$
Cinza	8	-	$\pm 0,05\%$
Branco	9	-	-

Fonte da imagem:

https://pt.wikipedia.org/wiki/Resistor#/media/File:Tabela_de_cores_de_um_resistor.jpg

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;

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)`

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)

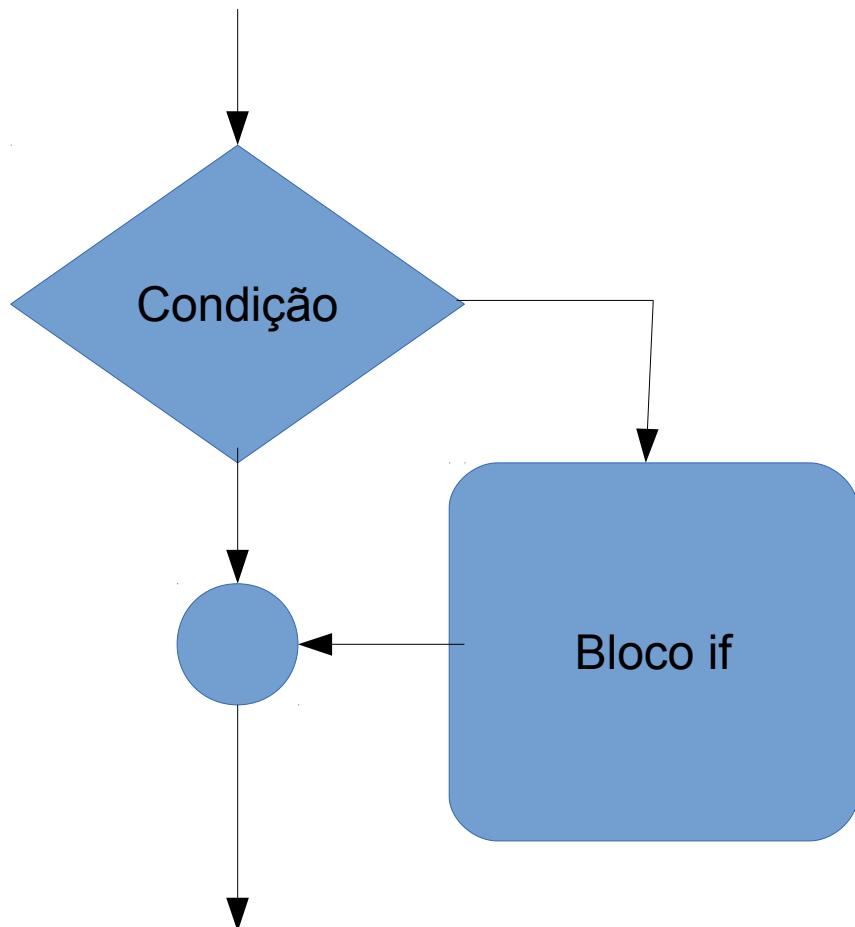
Parâmetros:

- Valor: qualquer número, texto ou variável.

Retorno:

- size_t (long)

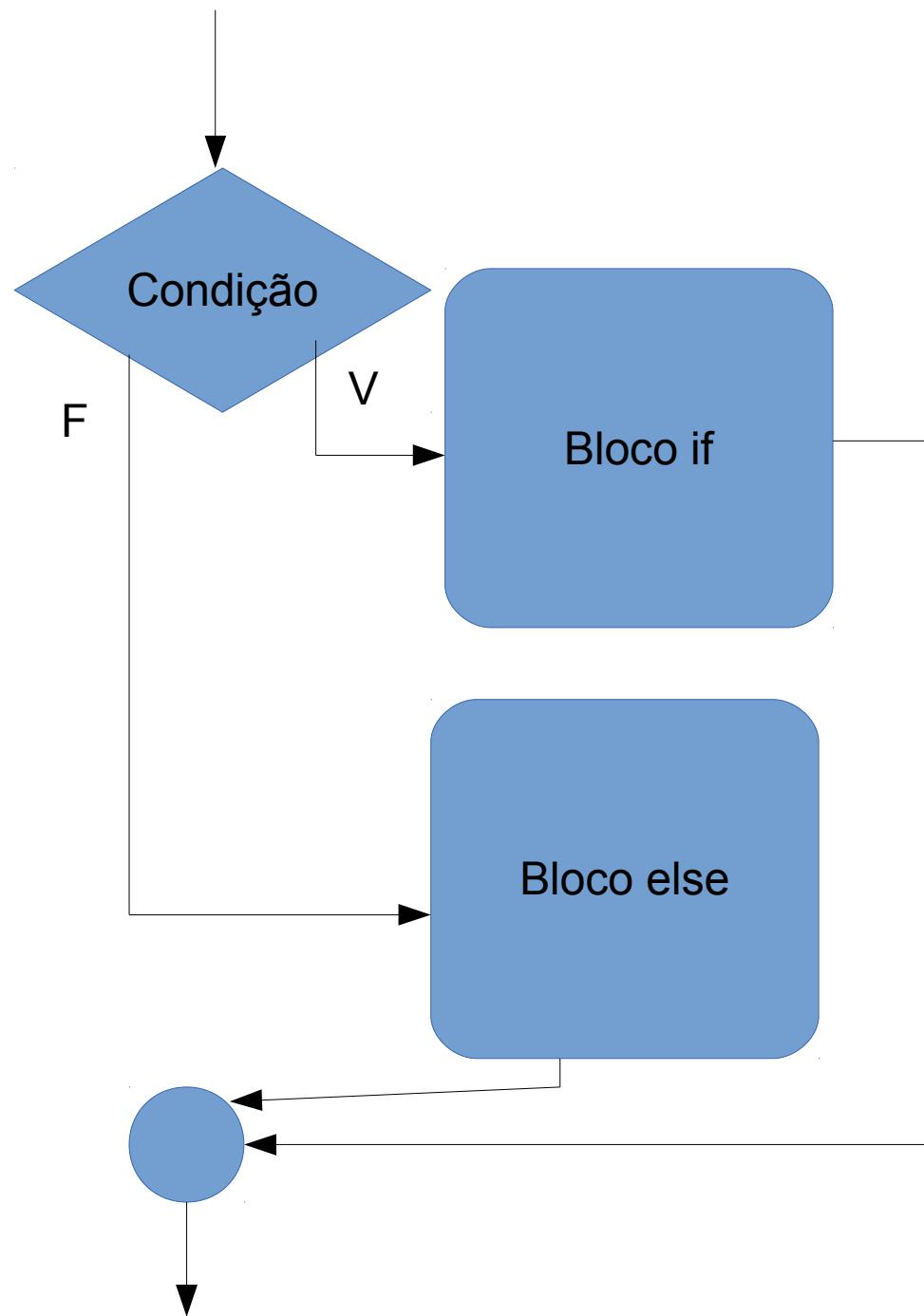
Desvio condicional: if



```
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>;
}
```

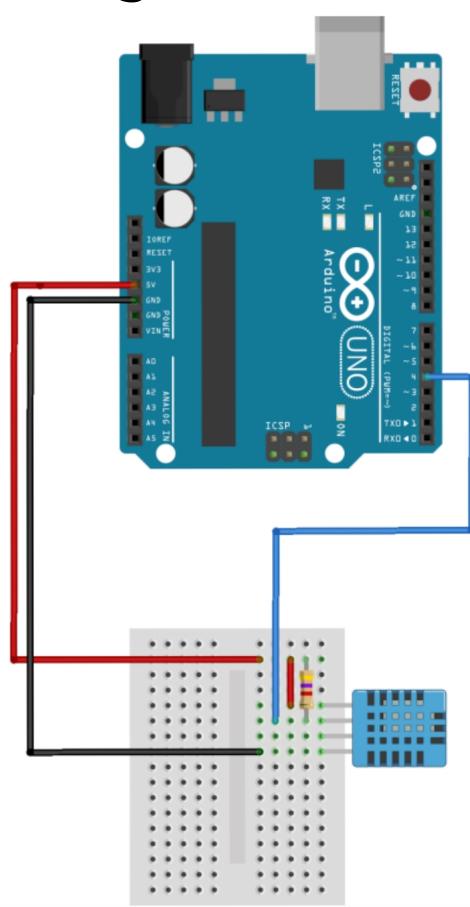
```
se <condição> então:
    <comandos V>;
senão
    <comandos F>;
fim-se
```

Projeto 1: Medindo Temperatura e Umidade

→ Componentes

- ✓ Sensor DHT11
- ✓ Resistor 4.7kOhms

→ Montagem



Made with Fritzing.org

→ Código:

```
#include "DHT.h"

#define DHTPIN A1 // pino que estamos conectado
#define DHTTYPE DHT11 // DHT 11 // Definimos que
tipo de sensor vamos usar
/*
Definimos que tipo de sensor vamos usar (no caso de
uso de DHT22 se troca DHT11 por DHT22)
*/

//Criamos um objeto de tipo DHT e recebemos os
parametros que inserimos acima.
DHT dht(DHTPIN, DHTTYPE);

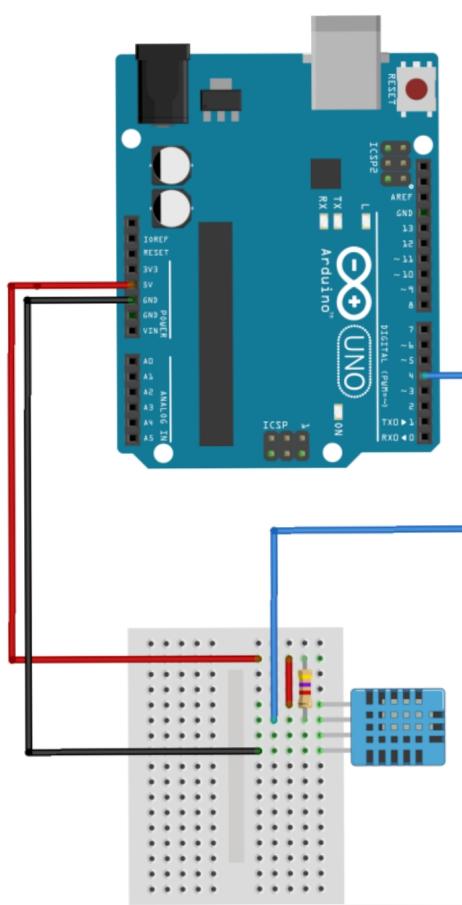
void setup()
{
    Serial.begin(9600);
    Serial.println("Testando DHT11");
    dht.begin();
}
```

Projeto 1: Medindo Temperatura e Umidade

→ Componentes

- ✓ Sensor DH11
- ✓ Resistor 4.7kOhms

→ Montagem



→ Código:

```
void loop()
{
    // 0 atraso do sensor pode chegar a 2 segundos.
    float h = dht.readHumidity();
    float t = dht.readTemperature();

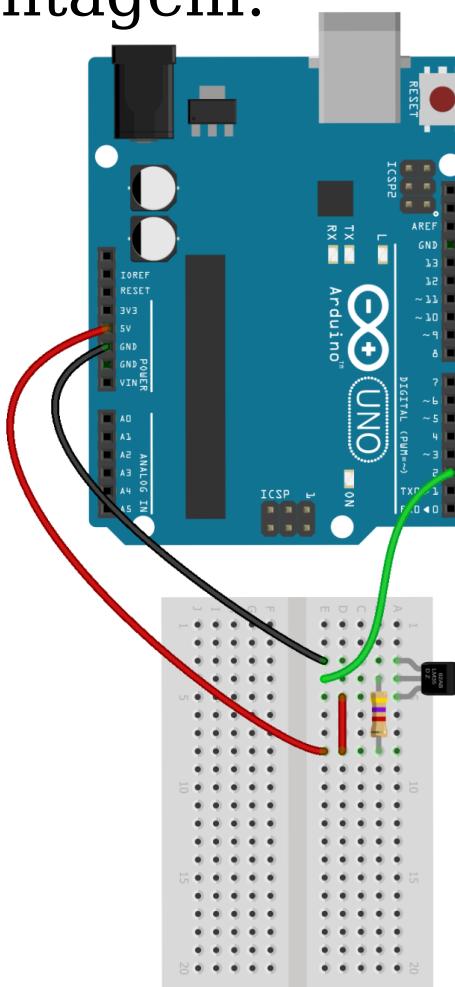
    // testar se retorno é valido, caso contrário algo
    // está errado.
    if (isnan(t) || isnan(h))
    {
        Serial.println("Falha ao ler do DHT");
    }
    else
    {
        Serial.print("Umidade: ");
        Serial.print(h);
        Serial.print(" %t");
        Serial.print("Temperatura: ");
        Serial.print(t);
        Serial.println(" *C");
    }
}
```

Projeto 2: Sensor de Temperatura

→ Materiais :

- 1 Sensor de Temperatura
- 1 resistor de 4.7k ohm

→ Montagem:



```
#include <OneWire.h>  
#include <DallasTemperature.h>
```

```
// Porta do pino de sinal do sensor  
#define ONE_WIRE_BUS 3
```

```
// Define uma instancia do OneWire para comunicacao  
com o sensor
```

```
OneWire oneWire(ONE_WIRE_BUS);
```

```
// Armazena temperaturas minima e maxima
```

```
float tempMin = 999;
```

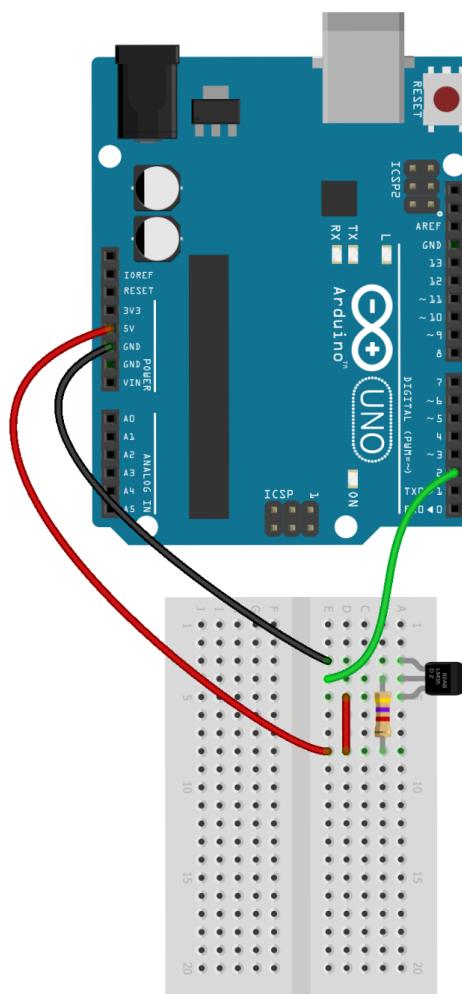
```
float tempMax = 0;
```

```
DallasTemperature sensors(&oneWire);
```

```
DeviceAddress sensor1;
```

Projeto 2: Sensor de Temperatura

→ Montagem



fritzing

```
void loop() {  
    sensors.requestTemperatures();  
  
    float tempC = sensors.getTempC(sensor1);  
  
    if (tempC < tempMin) {  
        tempMin = tempC;  
    }  
    if (tempC > tempMax) {  
        tempMax = tempC;  
    }  
    Serial.print("Temp C: ");  
    Serial.print(tempC);  
    Serial.print(" Min : ");  
    Serial.print(tempMin);  
    Serial.print(" Max : ");  
    Serial.println(tempMax);  
}
```



DIY!

Sensor de Umidade e Condutividade

Sensor de Umidade de Solo DYI

Materiais

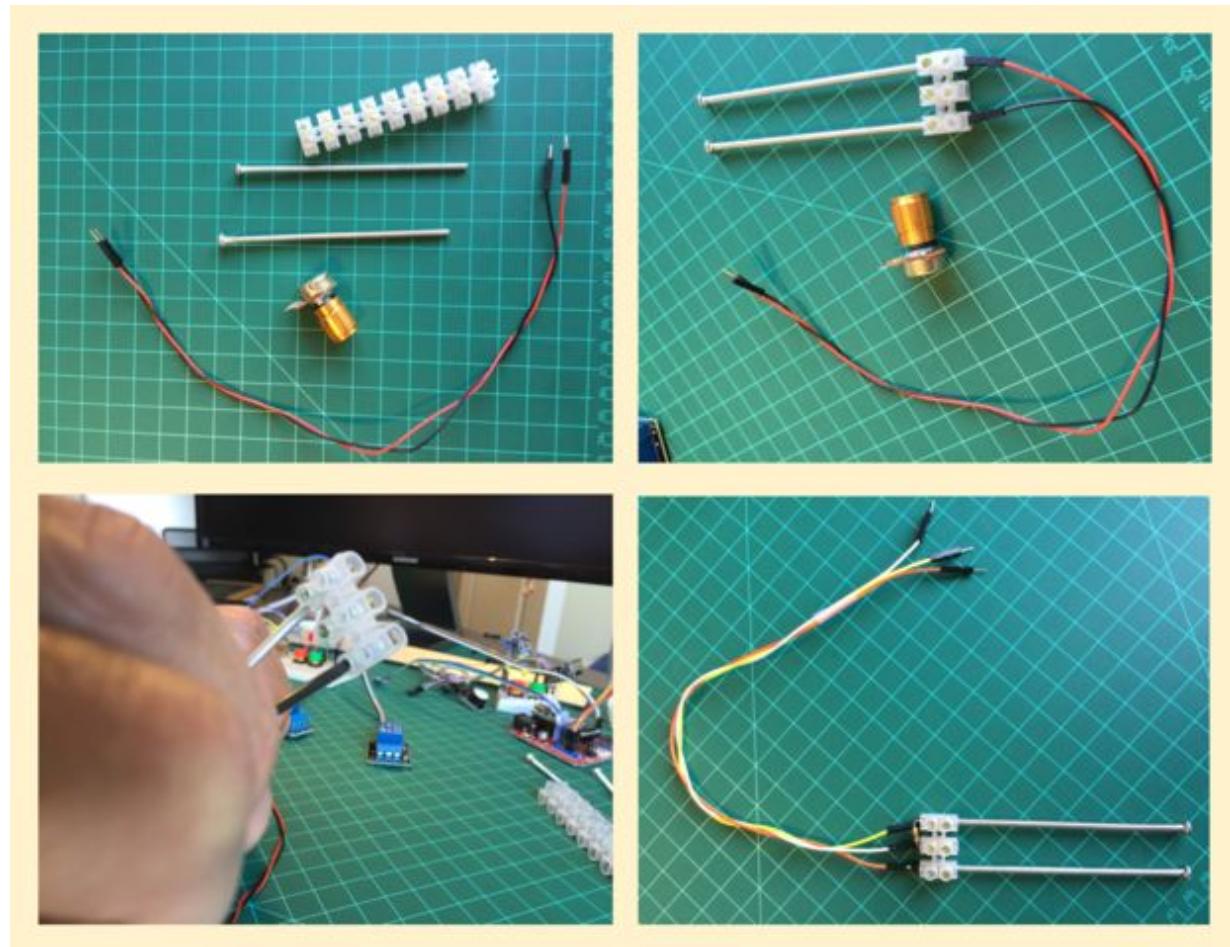
- Fios de tamanho adequado
- Pregos ou arames de material condutivo, mas difícil de oxidar (geralmente galvanizados)
- Material isolante para suporte (Isopor/ EVA)
- Conectores de Fios ou Material para soldagem
- Cola de silicone

Sensor de Umidade de Solo DYI

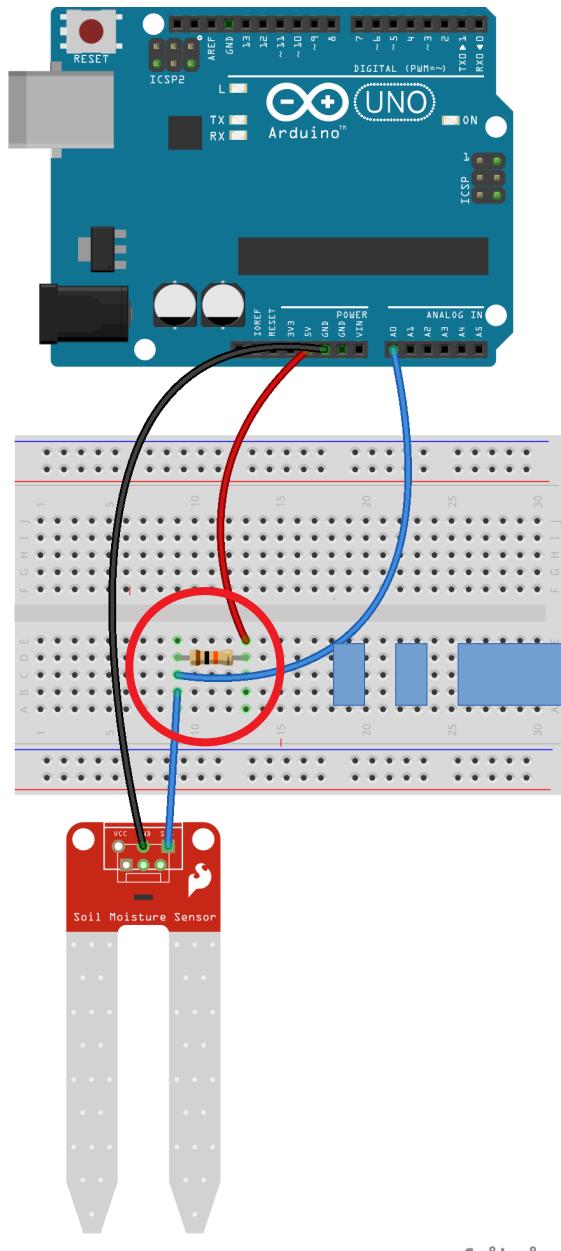
Passos

Instalação:

- Instalar as duas extremidades em uma estrutura firme não condutiva e que evite contato;
- Desencapar dois fios de mesmo comprimento;
- Conectar cada fio na região superior de cada extremidade

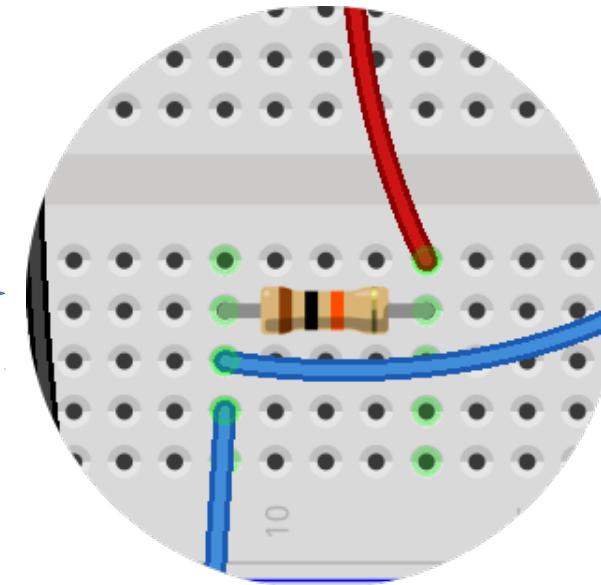


Sensor de Umidade de Solo DYI



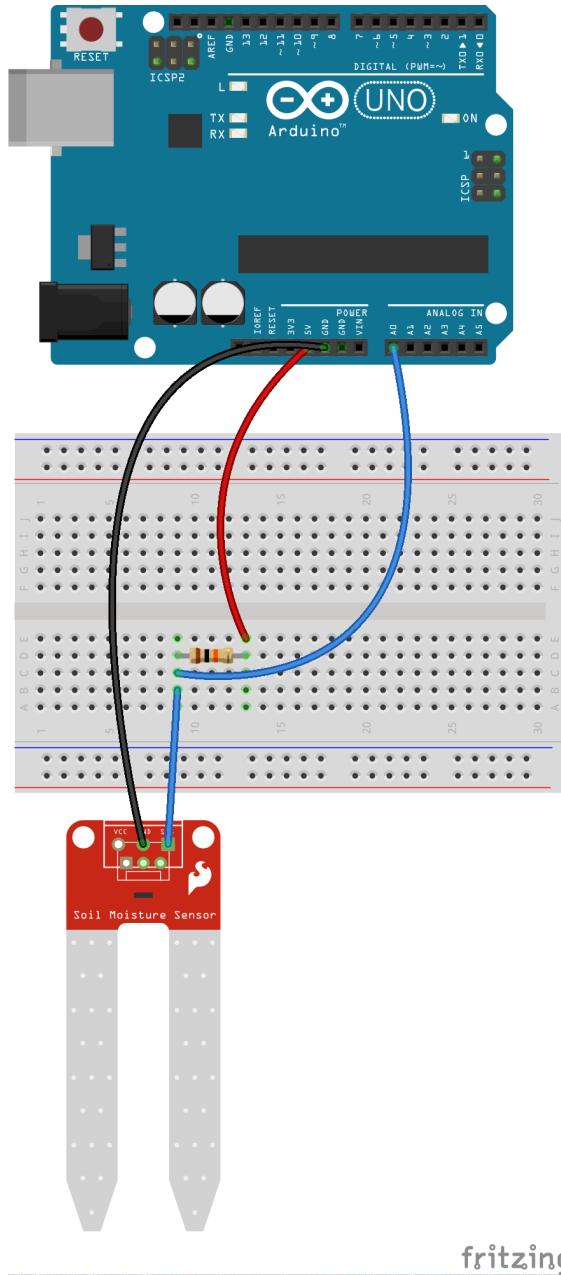
Conexão ao Arduino

- Usar resistor de 10K ohms entre o pino de sinal e 5V;



- Conectar fio terra diretamente

Sensor de Umidade de Solo DYI

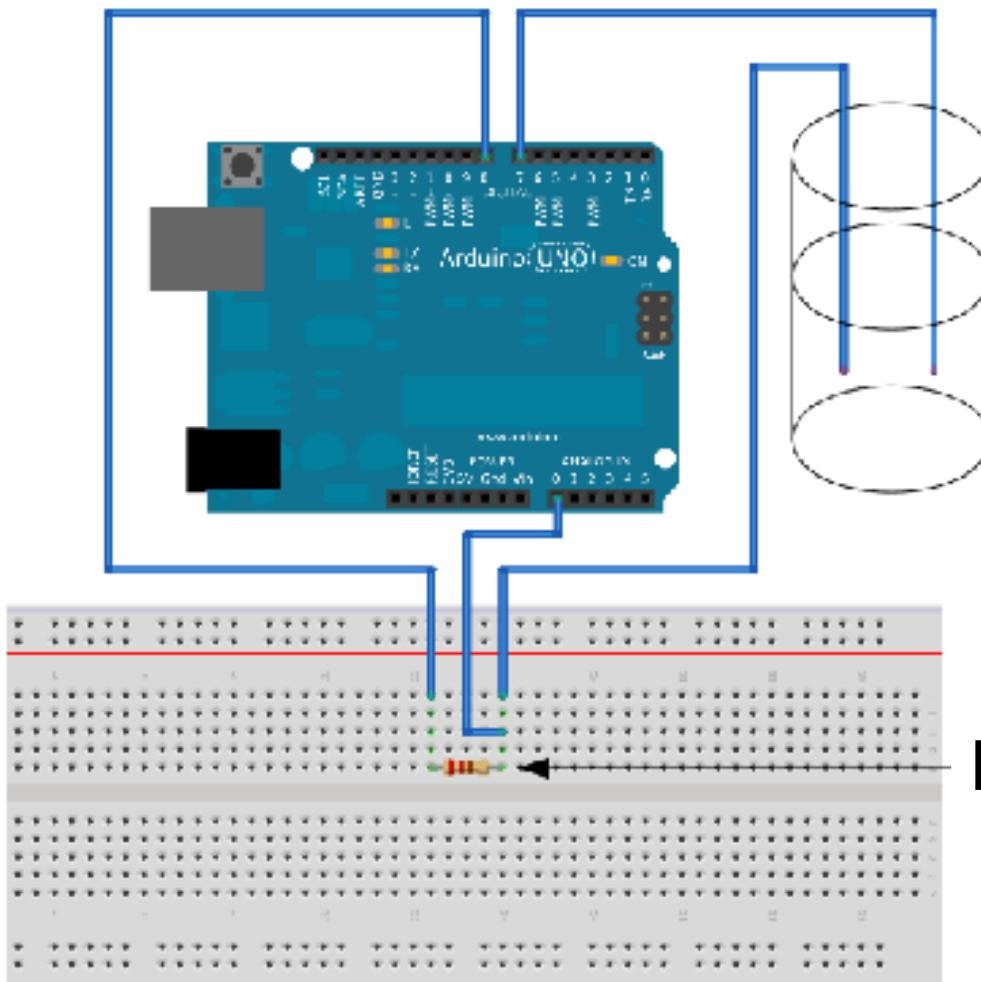


```
int val_umid = 0; // sensor de umidade

void setup() {
    Serial.begin(9600);
}

void loop() {
    int umidade_volt = analogRead(val_umid);
    int umidade = umidade_volt;
    Umidade = constrain(umidade_volt,400,1023);
    umidade = map(umidade,400,1023,100,0);
    Serial.println("Umidade (valor bruto)");
    Serial.println(umidade_volt);
    Serial.println("Umidade (%)");
    Serial.println(umidade);
}
```

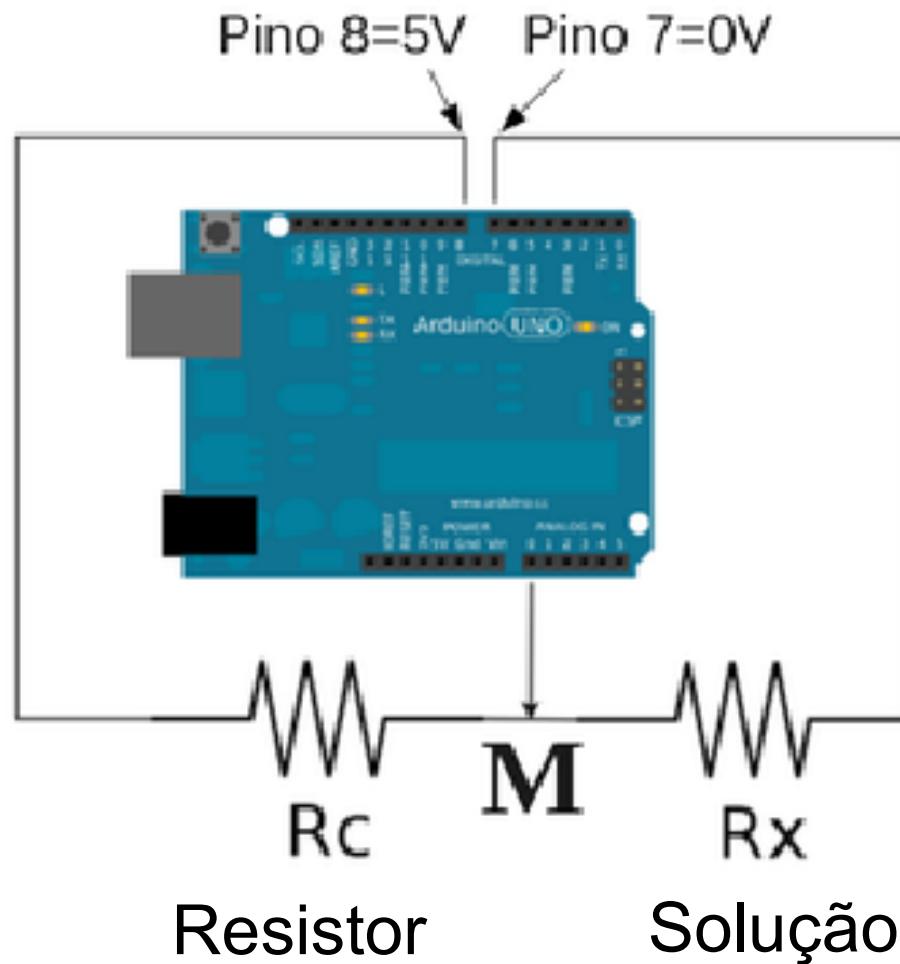
Sensor de Condutividade



Solução

Resistor

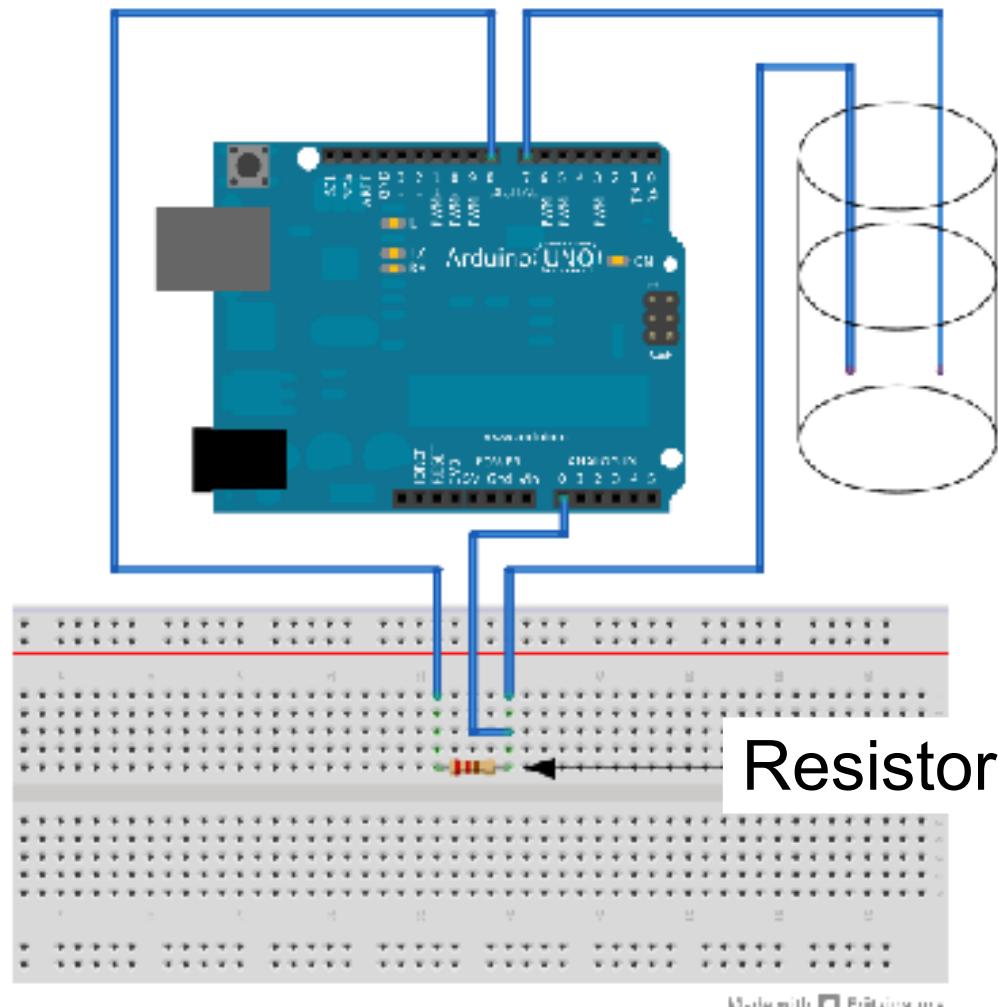
Sensor de Condutividade



Solução a ser analisada exerce o papel de uma resistência “variável”

“M” é o parâmetro usado para quantificar a resistência da solução e a partir daí a condutividade.

Sensor de Condutividade



```
byte electrode_1 = 7; //Pino eletrodo  
byte electrode_2 = 8; //Pino eletrodo  
int reading_pin = 0; //Pino analógico para leitura  
int reading; //Variável das leituras
```

```
void setup()
```

```
{
```

```
Serial.begin(9600);
```

```
pinMode(electrode_1, OUTPUT);  
pinMode(electrode_2, OUTPUT);
```

```
}
```

```
void loop()
```

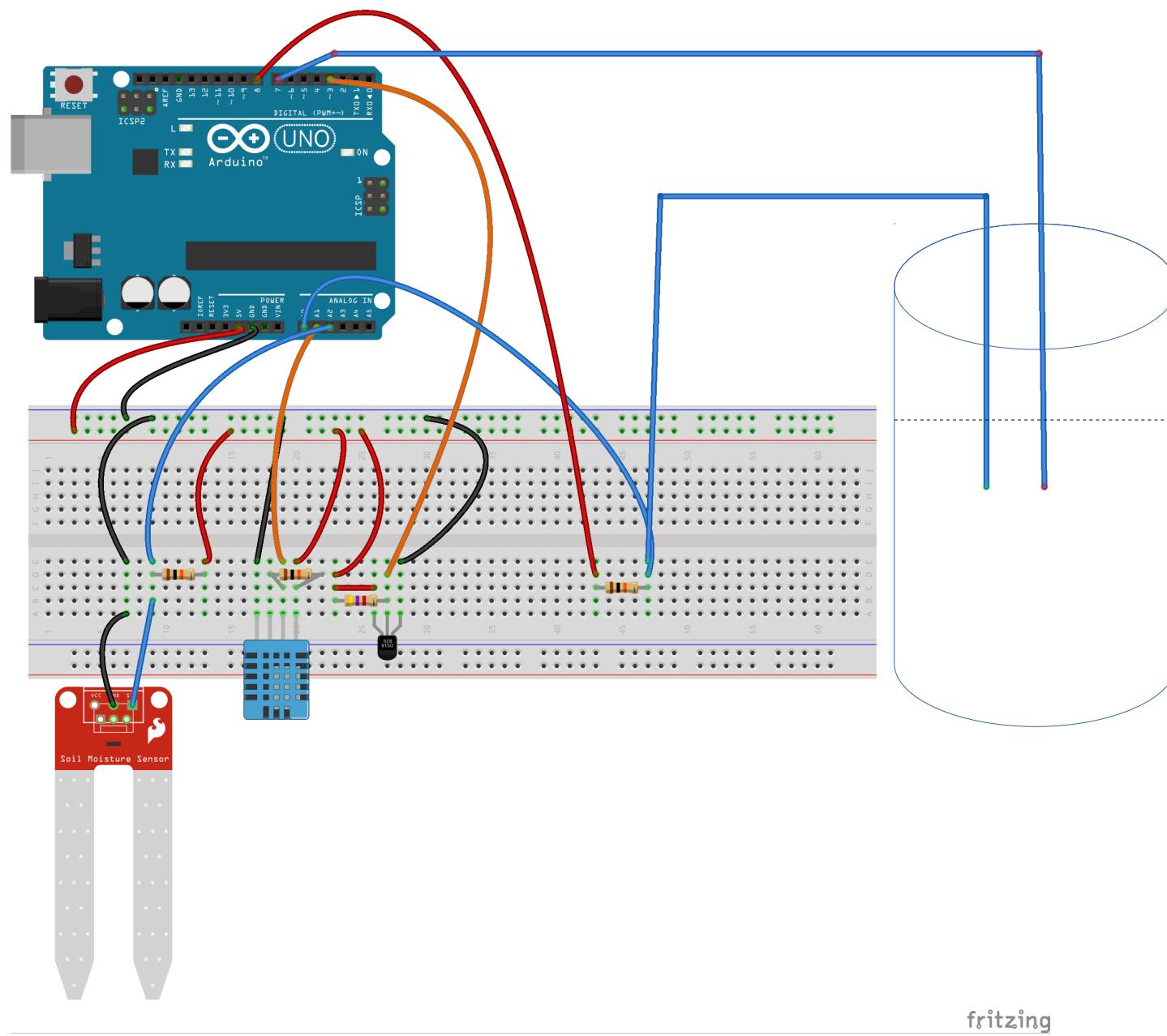
```
{
```

```
digitalWrite(electrode_2, HIGH );  
digitalWrite(electrode_1, LOW);  
delay(100);
```

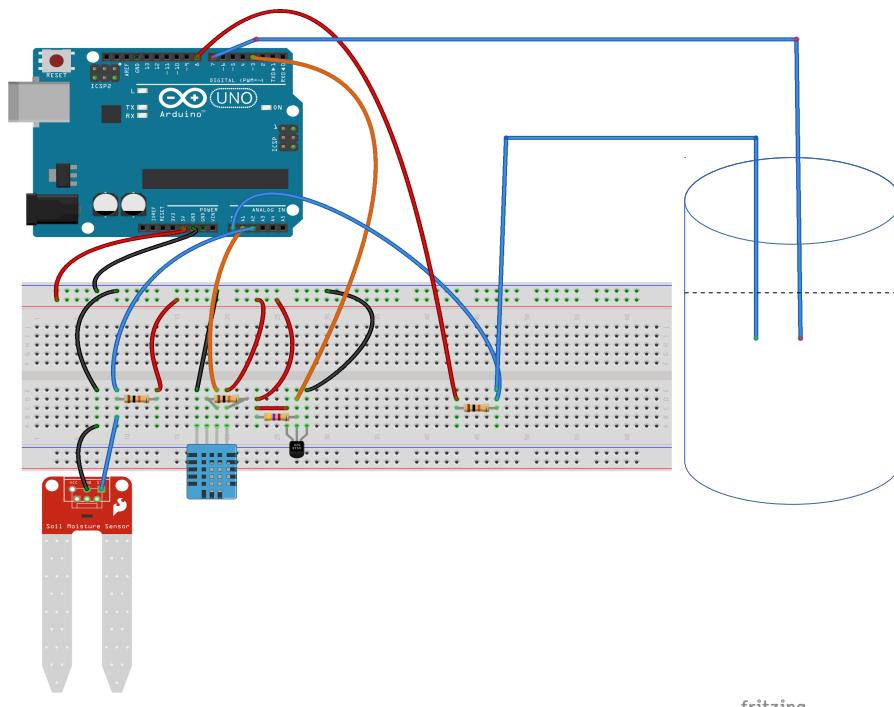
```
reading = analogRead(reading_pin);  
digitalWrite(electrode_2, LOW);  
Serial.println(reading);  
delay(500);
```

```
}
```

Integrando Sensores



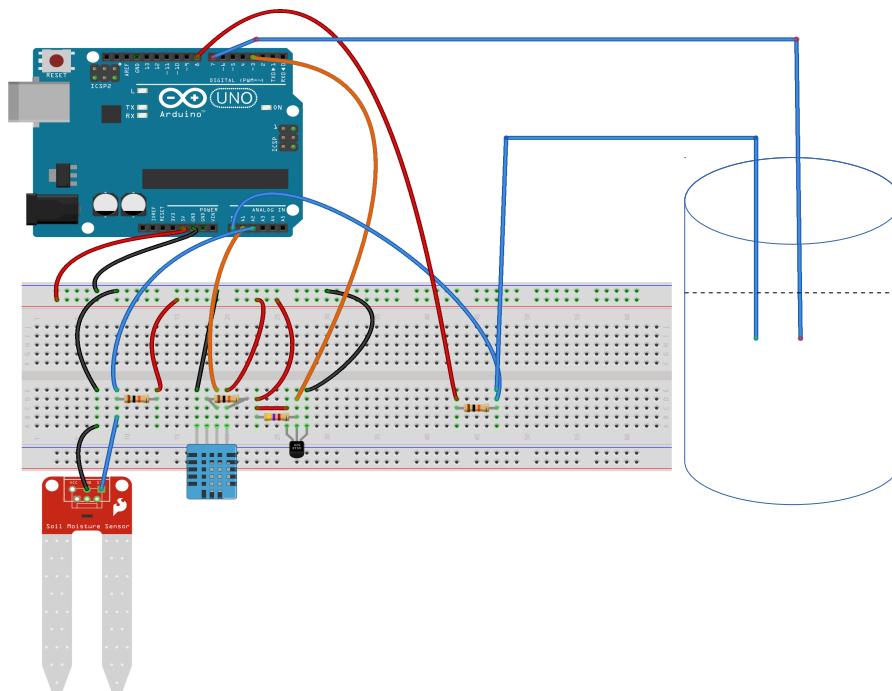
Sistema Integrado



```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <DHT.h>
#include <Adafruit_Sensor.h>

#define DHTPIN A1 // pino do dht
#define DHTTYPE DHT11 // Temperatura e umidade
DHT dht(DHTPIN, DHTTYPE); //definir dht com o que foi criado acima
```

Sistema Integrado

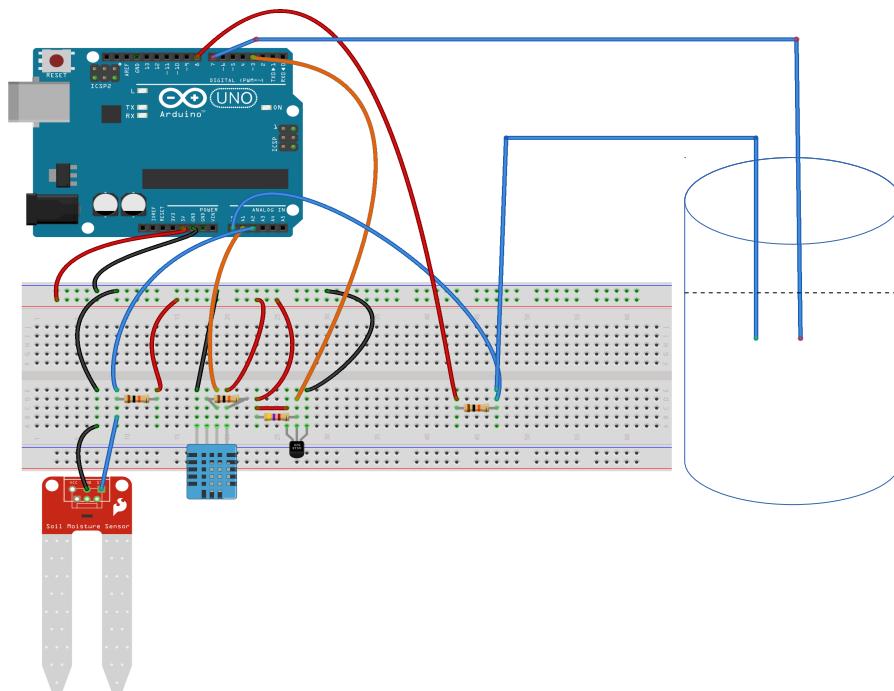


```
#define ONE_WIRE_BUS 3  
OneWire oneWire(ONE_WIRE_BUS);  
  
float tempMin = 999; // t minima e maxima  
float tempMax = 0;
```

```
DallasTemperature sensors(&oneWire);  
DeviceAddress sensor1;
```

```
int val_umid = A2; // sensor de umidade  
  
byte electrode_1 = 7; //Pino eletrodo  
byte electrode_2 = 8; //Pino eletrodo  
int reading_pin = 0; //Pino analógico para leitura  
int reading; //Variável que armazena leituras
```

Sistema Integrado

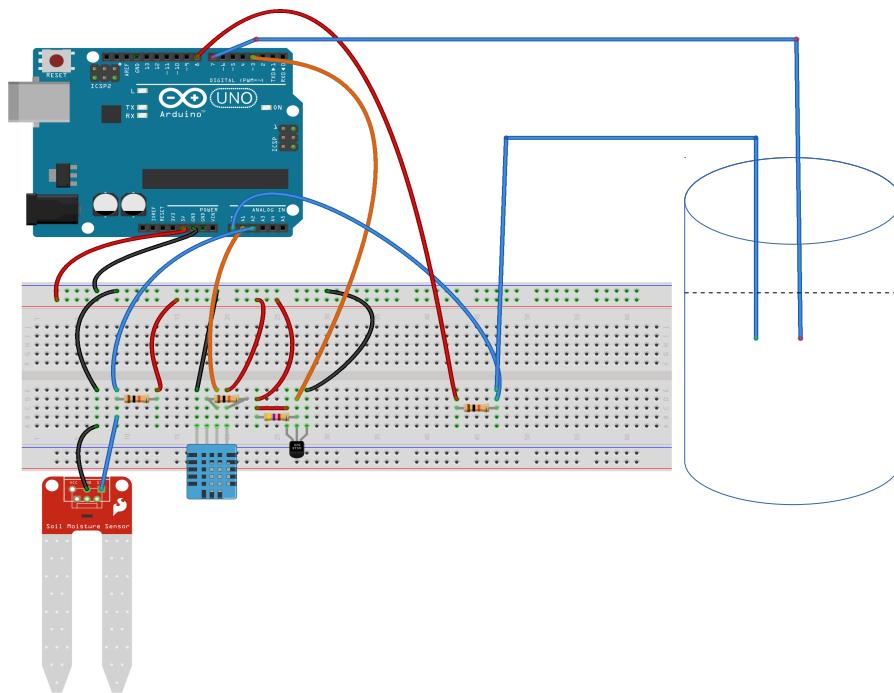


```
void setup()
{
    Serial.begin(9600);
    pinMode(electrode_1, OUTPUT);
    pinMode(electrode_2, OUTPUT);

    // Localiza e mostra enderecos dos sensores
    Serial.println("Testando DHT11");
    Serial.println("Localizando sensores DS18B20...");
    Serial.print("Foram encontrados ");
    Serial.print(sensors.getDeviceCount(), DEC);
    Serial.println(" sensores.");
    Serial.println();

    dht.begin();
    sensors.begin();
}
```

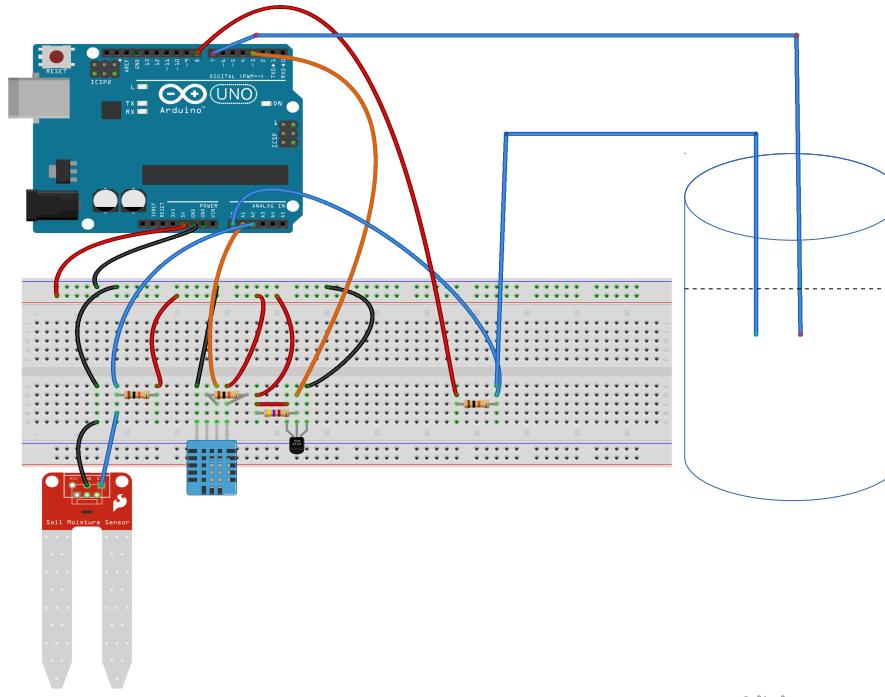
Projeto 5: Sistema Integrado



```
void loop() {
    float h = dht.readHumidity(); // O atraso do
    sensor DHT11 pode chegar a 2 segundos.
    float t = dht.readTemperature();

    sensors.requestTemperatures();
    float tempC = sensors.getTempC(sensor1);
    // Atualizar temperaturas minima e maxima
    if (tempC < tempMin)
    {
        tempMin = tempC;
    }
    if (tempC > tempMax)
    {
        tempMax = tempC;
    }
}
```

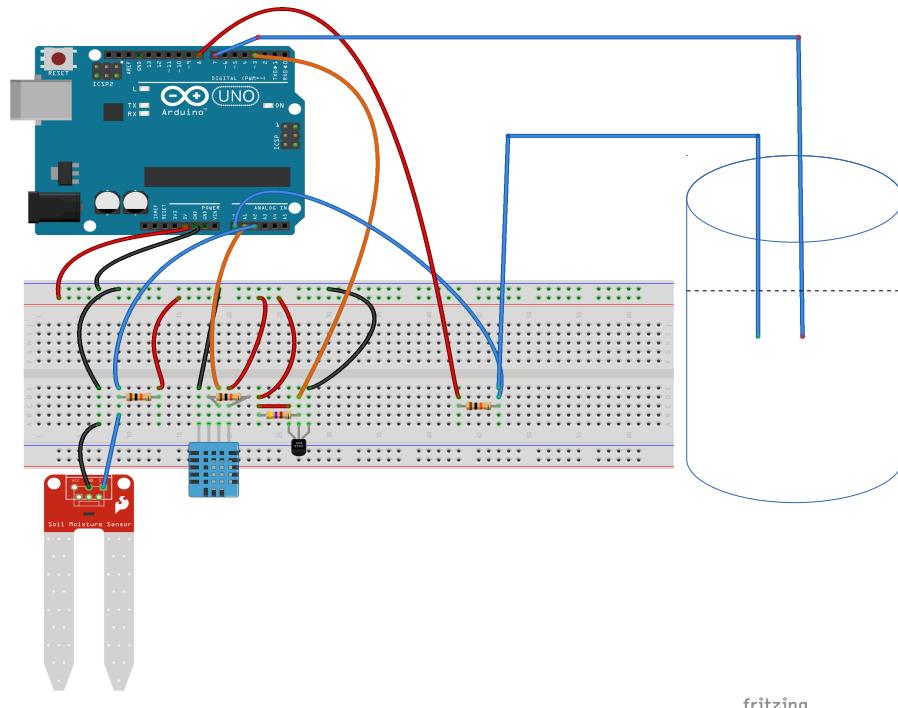
Projeto 5: Sistema Integrado



```
int umidade_volt = analogRead(val_umid);
int umidade = umidade_volt;
umidade = constrain(umidade_volt,400,1023);
umidade = map(umidade,400,1023,100,0);

digitalWrite(electrode_2, HIGH ); //polo positivo (5V)
digitalWrite(electrode_1, LOW); //polo negativo (0V)
delay(100); //Aguardar estabilização
reading = analogRead(reading_pin); // tensão entre o divisor de voltagem
digitalWrite(electrode_2, LOW); //eletrodo 1
```

Projeto 5: Sistema Integrado



```
Serial.print("Umidade: ");
Serial.print(h);
Serial.print(" %t");
Serial.print("Temperatura: ");
Serial.print(t);
Serial.println(" *C");
Serial.print("Temp C: ");
Serial.print(tempC);
Serial.print(" Min : ");
Serial.print(tempMin);
Serial.print(" Max : ");
Serial.println(tempMax);
Serial.println("Umidade (%)");
Serial.println(umidade);
Serial.println("Condutividade (V)");
Serial.println(reading);
delay(500);
```

E depois? Visualizando os dados

No computador



Libreoffice – <https://libreoffice.org>
Python - <https://www.python.org/>
R - <https://cran.r-project.org/>

Online

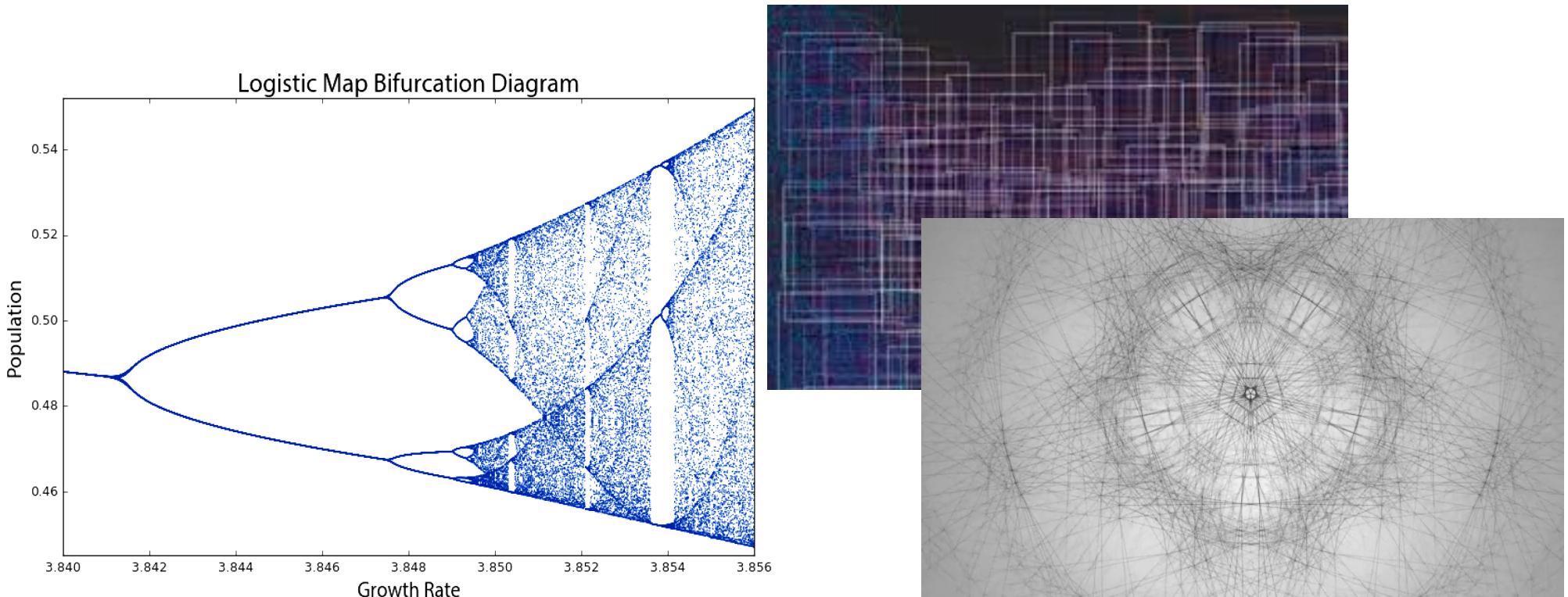
CHARTED
Beautiful, automatic charts

dygraphs

Data-
wrapper

Datawrapper - <https://www.datawrapper.de/>
Charter - <https://wwwcharted.co/>
Dygraphs - <http://dygraphs.com/>

E Além: Computadores Fazem Arte



SuperCollider

SuperCollider - <https://superollider.github.io/>
Processing - <https://processing.org/>

Contatos



saulojacques@protonmail.com

smjacques.github.io

signal/telegram: +5511995194881