

GIZMODO

APRESENTA



ARDUINO

Professor: Luciano Ramalho

Agenda

- O que dá para fazer com Arduinos?
- Primeiro circuito: Blink 
- Coding Dojo com Arduino 
- ABCdE: ABC da Eletrônica
- Circuito semáforo 
- Fechamento

Logística

- Hashtag: #MakersBR
- Workshop: 14 a 19h
- Localização dos banheiros, fumódromo
- Café, água: sirvam-se quando quiserem
- Coffee break ≈ 16h

Apresentações

MAKERS

Luciano Ramalho

- Instrutor e palestrante internacional especializado na linguagem Python

The vanishing pattern: from iterator generators in Python

Luciano Ramalho (Oficinas Turing)
5:00pm Thursday, 07/25/2013
Python
Location: D136
Average rating: ★★★★☆ (4.33, 12 ratings)
[Rate This Session](#)

Luciano Ramalho

Luciano Ramalho was a Web developer before the Netscape IPO in 1995, and switched from Perl to Java to Python in 1998. Since then he worked on some of the largest news portals in Brazil using Python, and taught Python web development in the Brazilian media, banking and government sectors. His speaking credentials include [OSCON 2002](#), and 15 talks over the years at [PythonBrasil](#) (the Brazilian PyCon) and [FISL](#) (the largest FLOSS conference in the Southern Hemisphere). Ramalho is a member of the [PSF](#) and a founder of

Luciano Ramalho

- Programador auto-didata desde 1978
 - BASIC, ASM Z-80, Pascal, C, C++, Smalltalk, Perl, Java, Python, Ruby...
- Formado em Biblioteconomia na ECA/USP
 - Oficina de Programação e Arte (PSI-2615) na Poli/USP com Etienne Delacroix

Luciano Ramalho

- Dono e professor nas Oficinas Turing, escola virtual de computação
- Sócio-fundador do Garoa Hacker Clube



Apresentem-se

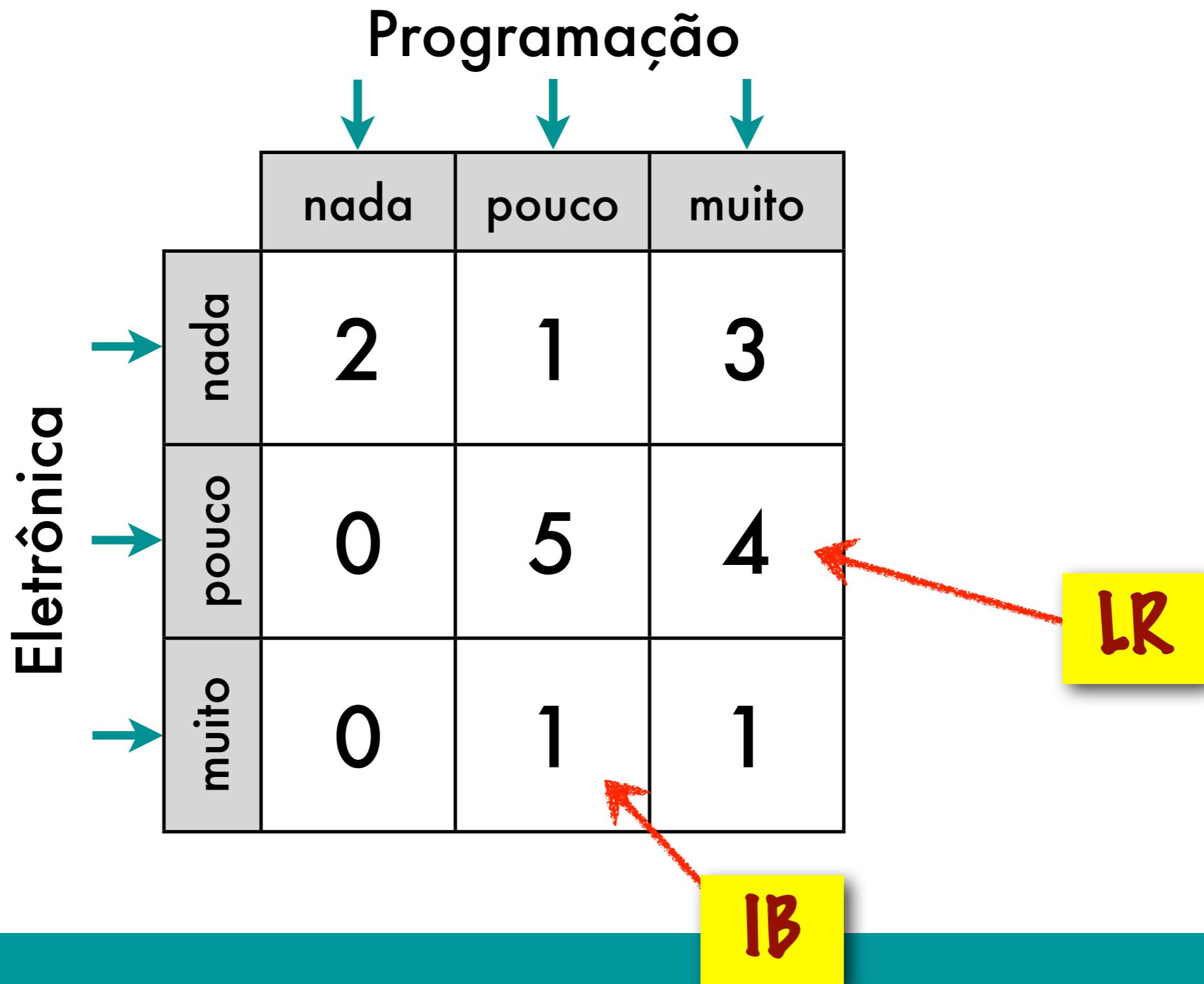
5
minutos

- Descubra o nome, profissão e hobbies de cada pessoa perto de você
- Pergunte e conte para seus vizinhos:
 - porque se interessou pelo Arduino?
 - já tem algum projeto em mente?

Censo

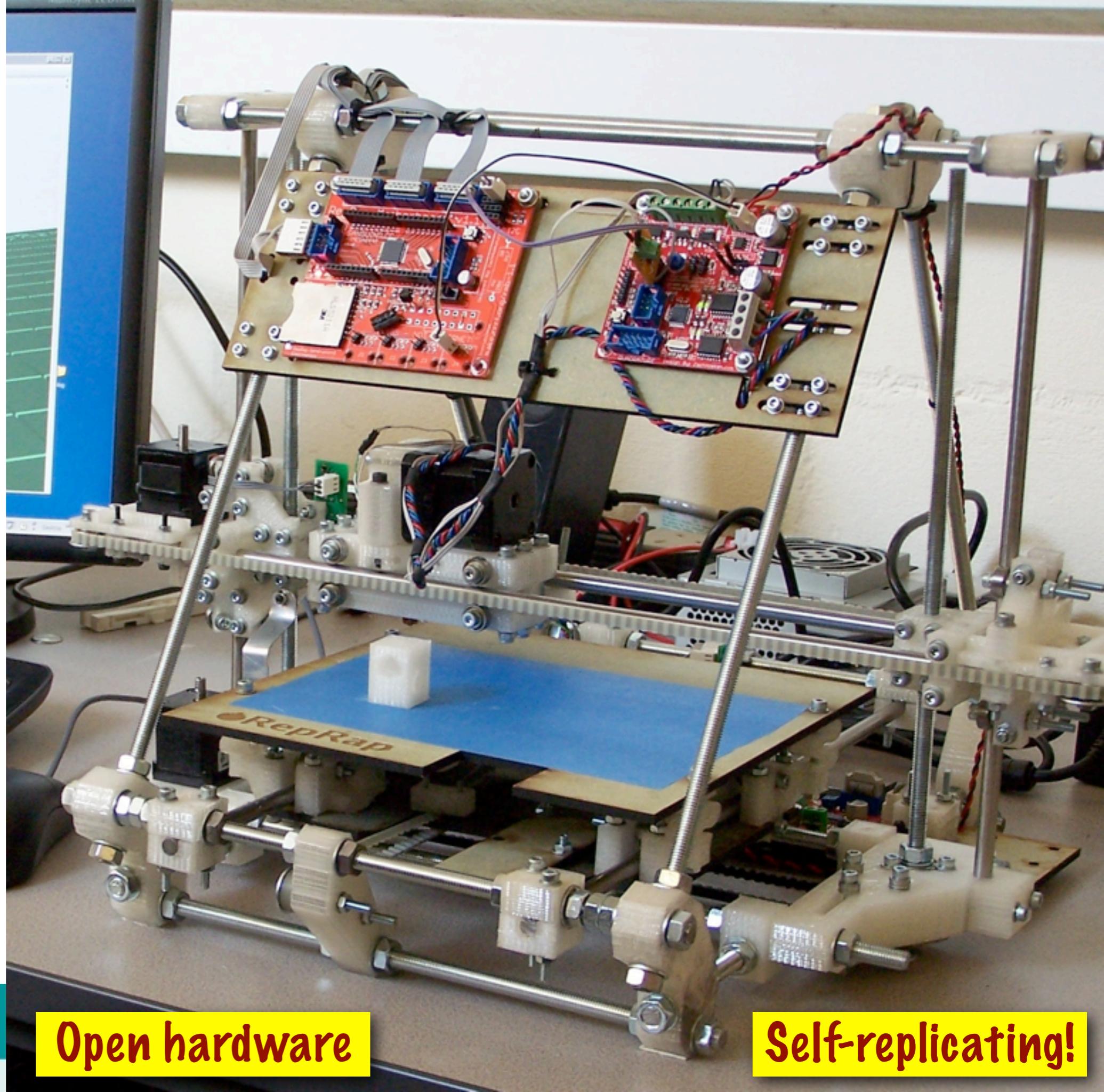
- Quem entende bem de eletrônica?
- Quem sabe um pouco de eletrônica?
- Quem entende bem de programação?
- Quem sabe um pouco de programação?

Censo



O que dá para fazer com Arduinos?

Reprap 3D printer



MAKERS

Open hardware

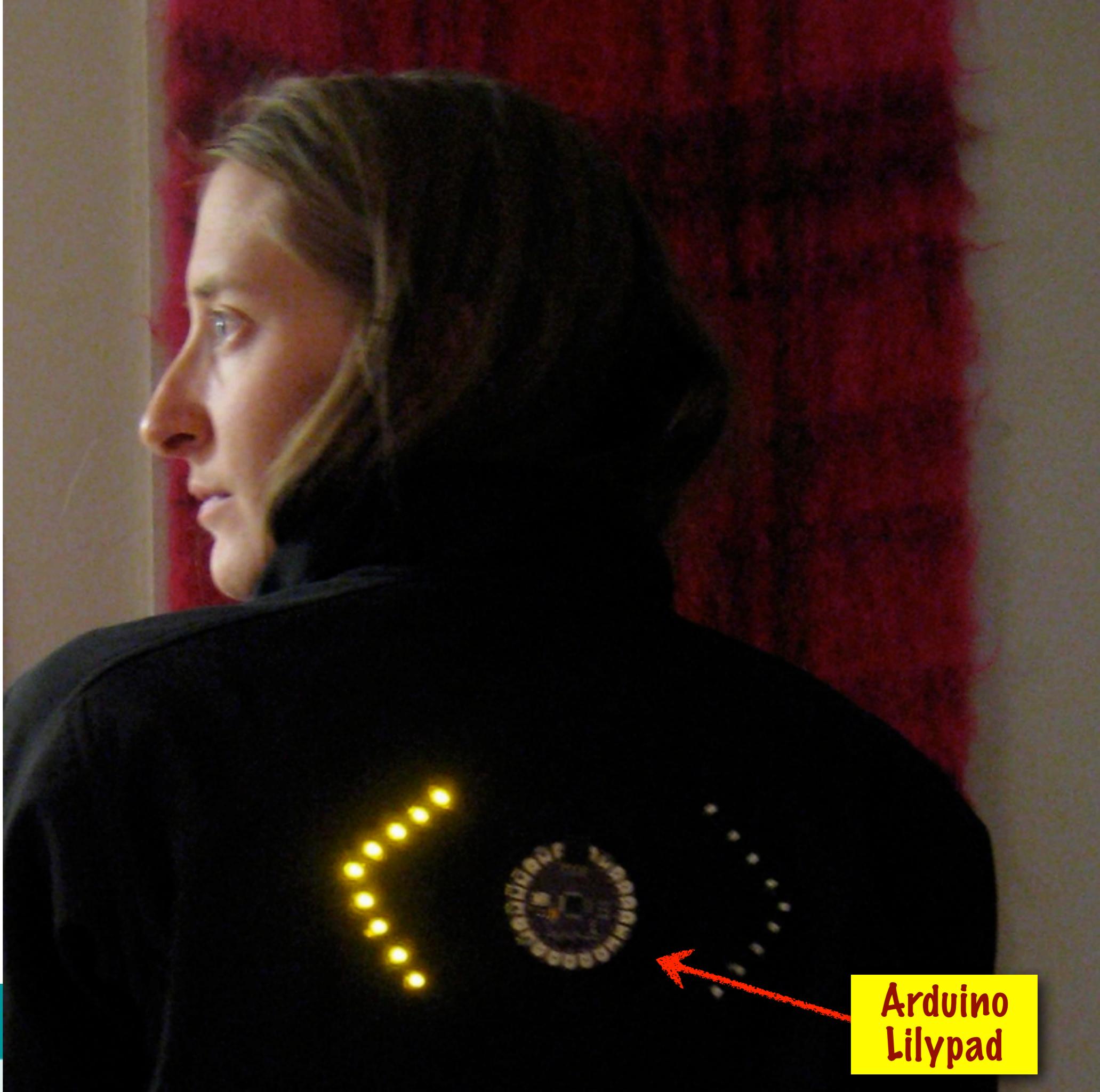
Self-replicating!

Open hardware

- Open hardware:
esquemas e software livres,
componentes “de prateleira”
ou fabricáveis em casa
- Arduino é open hardware:
copiar é legal
- Uso da marca **Arduino**: sob licença



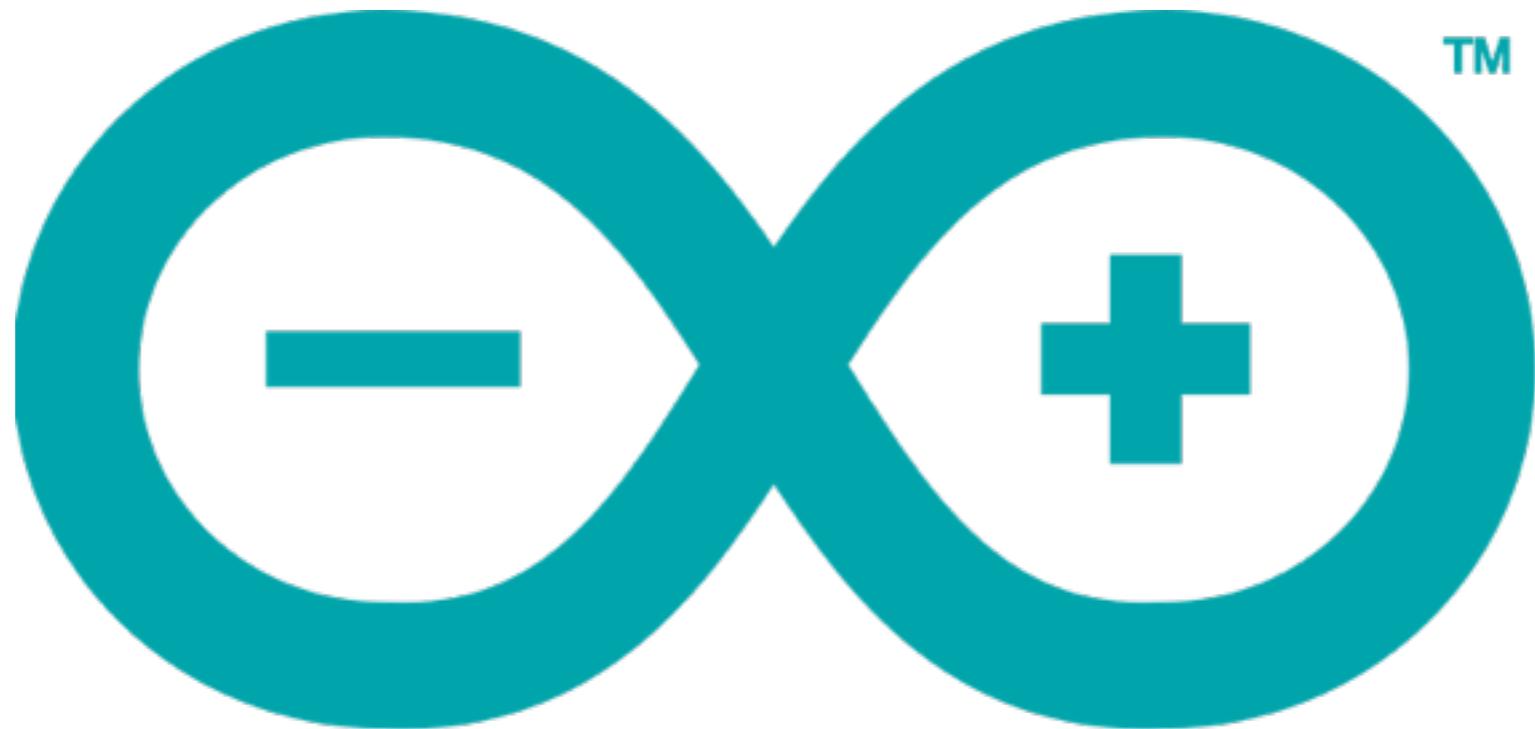
Pisca-pisca biker



MAKERS

Arduino
Lilypad

- **Microfone-bafômetro**
www.instructables.com/id/Breathalyzer-Microphone/
- **Pedal de guitarra**
www.instructables.com/id/Lo-fi-Arduino-Guitar-Pedal/
- **Robô equilibrista (inspirado no Segway)**
hacknmod.com/hack/make-a-mini-segway-using-the-arduino/
- **Fotografia de alta velocidade**
hacknmod.com/hack/high-speed-photography-how-to-trigger-using-arduino/
- **Automação residencial**
www.makeuseof.com/tag/how-to-build-home-automation-system-raspberry-pi-and-arduino/



Não por acaso, o logo é
 ∞

Abrindo o kit

Lista de peças do kit

Controlador

- 1 Placa compatível com Arduino UNO R3
- 1 Cabo USB AB

Componentes de entrada

- 10 Push-button 6x6 mm
- 2 Potenciômetro 100kΩ
- 2 Sensores ópticos reflexivos
- 1 Sensor de luminosidade LDR 3mm
- 1 Sensor de temperatura LM35

Componentes básicos

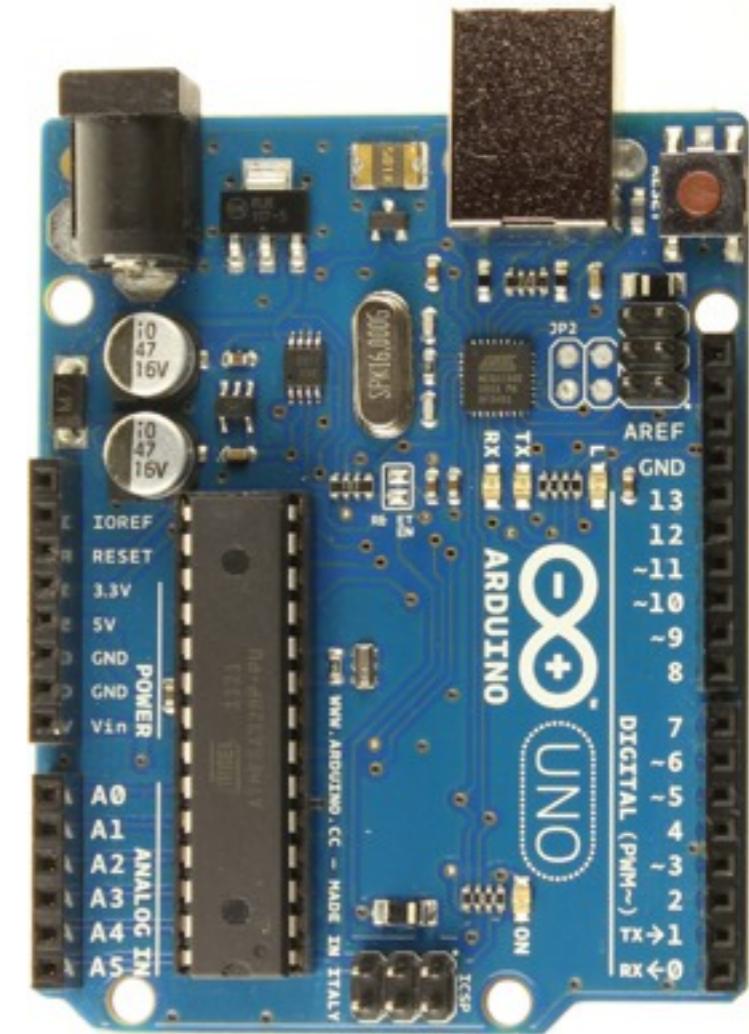
- 1 Breadboard 830 pontos
- 1 Placa wire-wrap 5x7 cm
- 65 Jumpers macho/macho
- 3 Cabos 3 vias Dupont
- 20 Resistores 1kΩ
- 6 Diodos 1N4006

Componentes de saída

- 12 LEDs 5mm, 2 de cada cor (verde, vermelho, amarelo, laranja, azul e branco)
- 2 Relés 5v
- 1 Display 7 segmentos 3 dígitos
- 1 Micro-servo 9g SG90 TowerPro
- 1 Buzzer contínuo

Arduino Uno R3

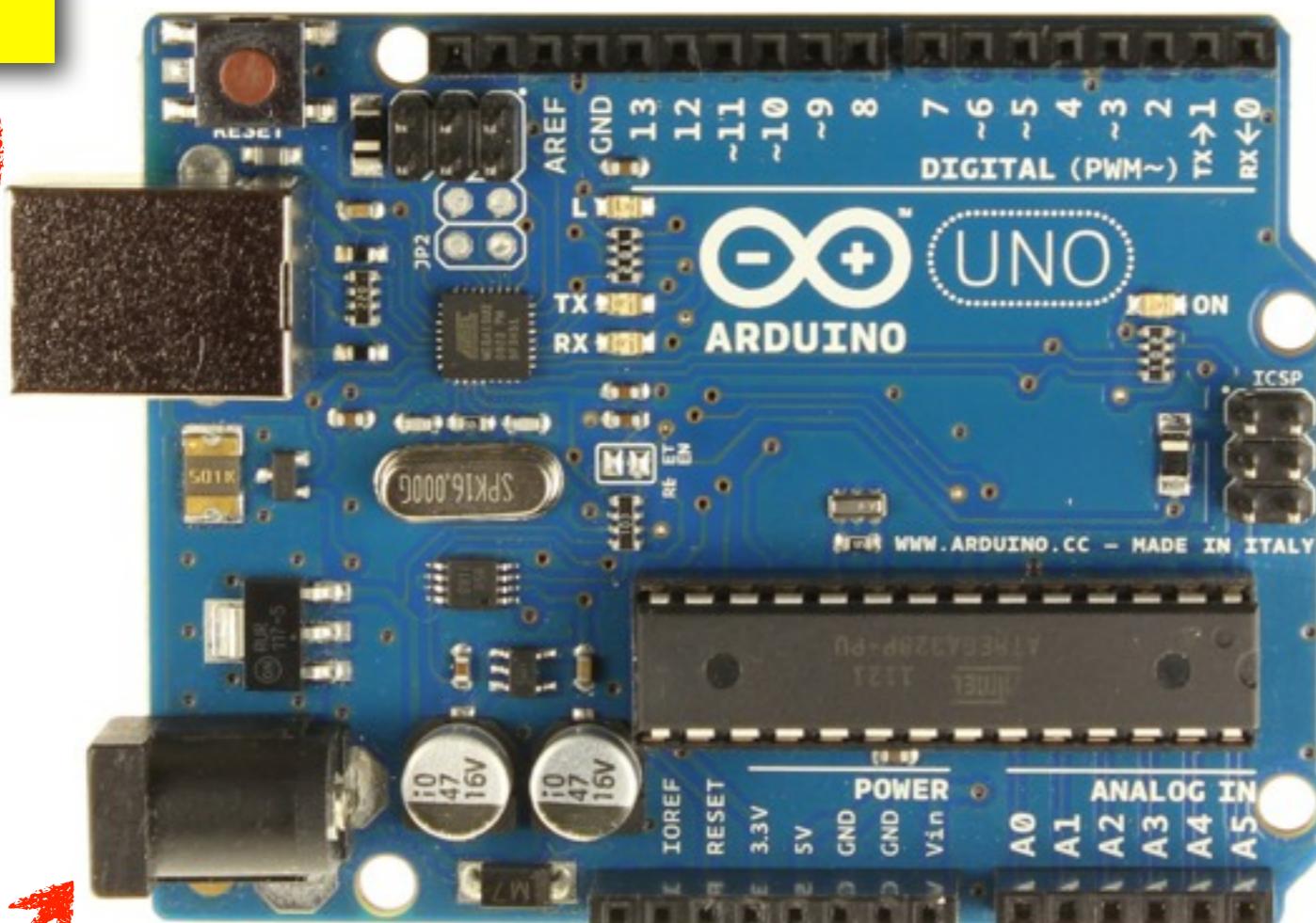
- Placa com microcontrolador ATmega328 e circuitos auxiliares
- interface USB (cliente)
- regulador de voltagem aceita 7-12V (recomendação)



Arduino: alimentação

1

5V via
USB

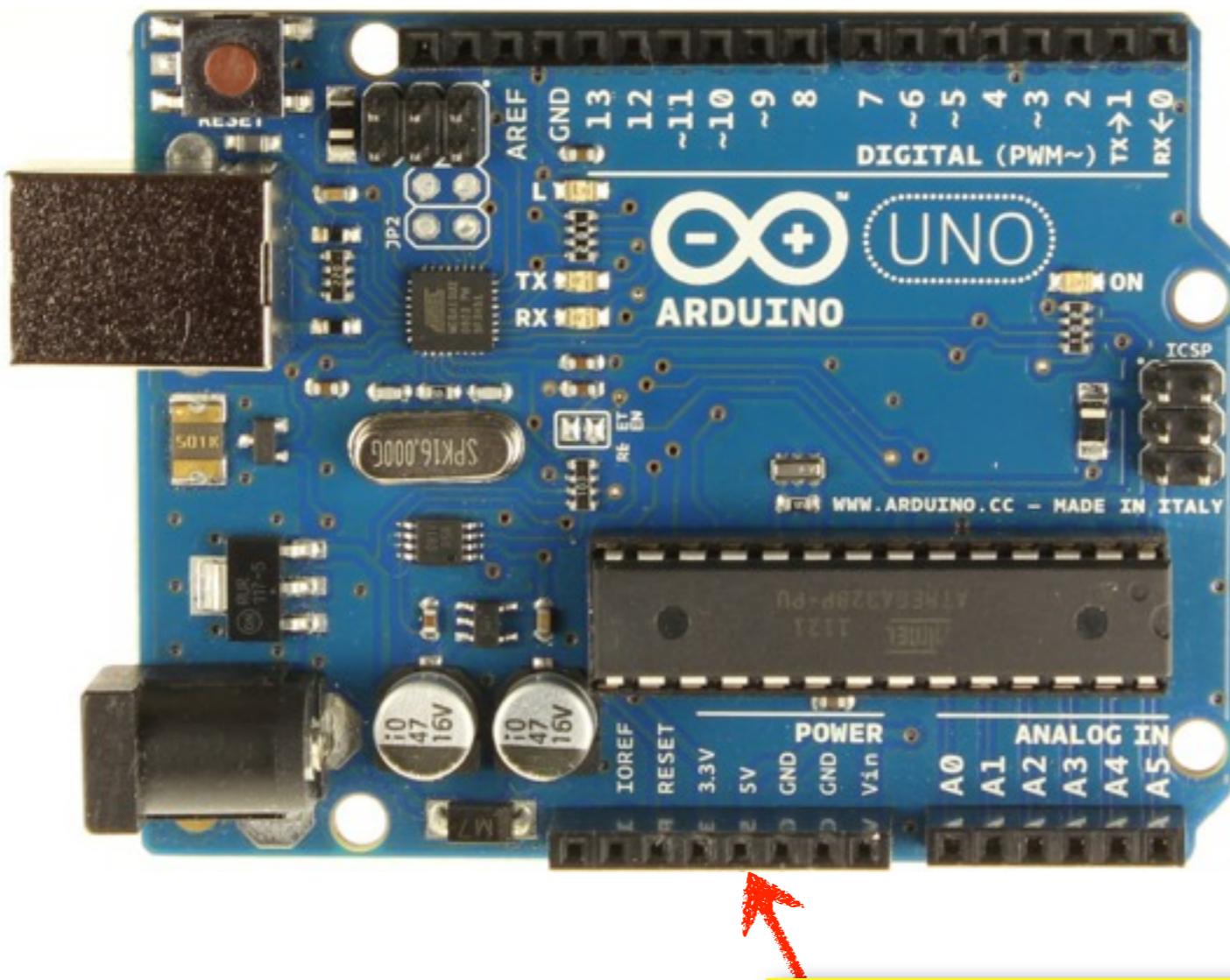


7 a 12V DC

GND, V_{in}

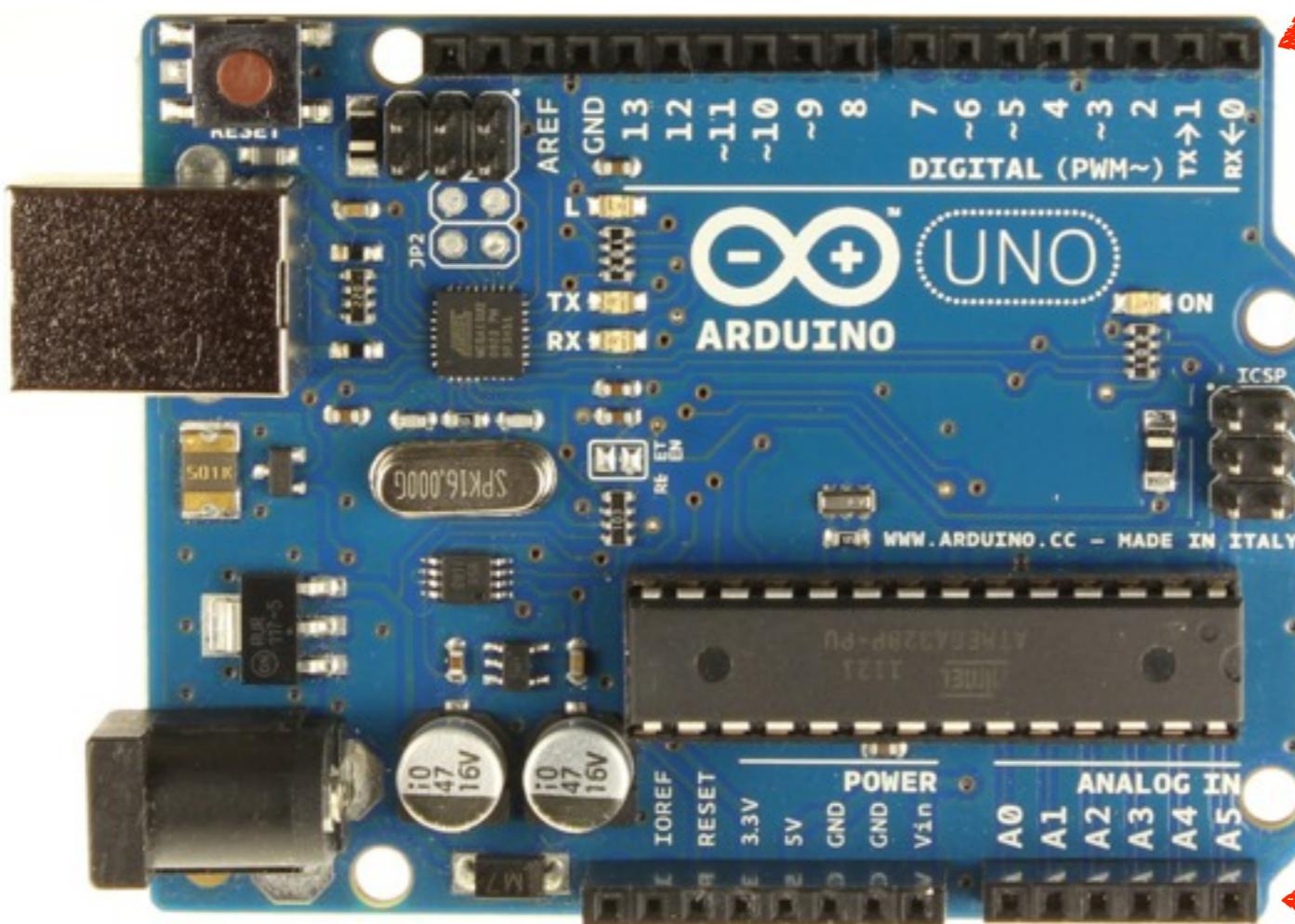
- Externas:
 - USB: 5V
 - adaptador DC de 7 a 12V (não incluído)
 - V_{in} e GND : 7 a 12V DC

Arduino: alimentação



- Internas:
 - 3.3 V e 5 V (+)
 - GND: terra (-)
 - V_{in} : direto da fonte externa 7 a 12 V (+)

Pinos de entrada ou saída



14 pinos de entrada e saída digital: 0 a 13

6 deles com Pulse Width Modulation: PWM ~

6 pinos de entrada analógica: A0 a A5

Entradas × saídas

- Entradas:
**chaves, sensores
etc.**
- Saídas:
**luzes, motores,
mostradores, etc.**

Componentes de entrada

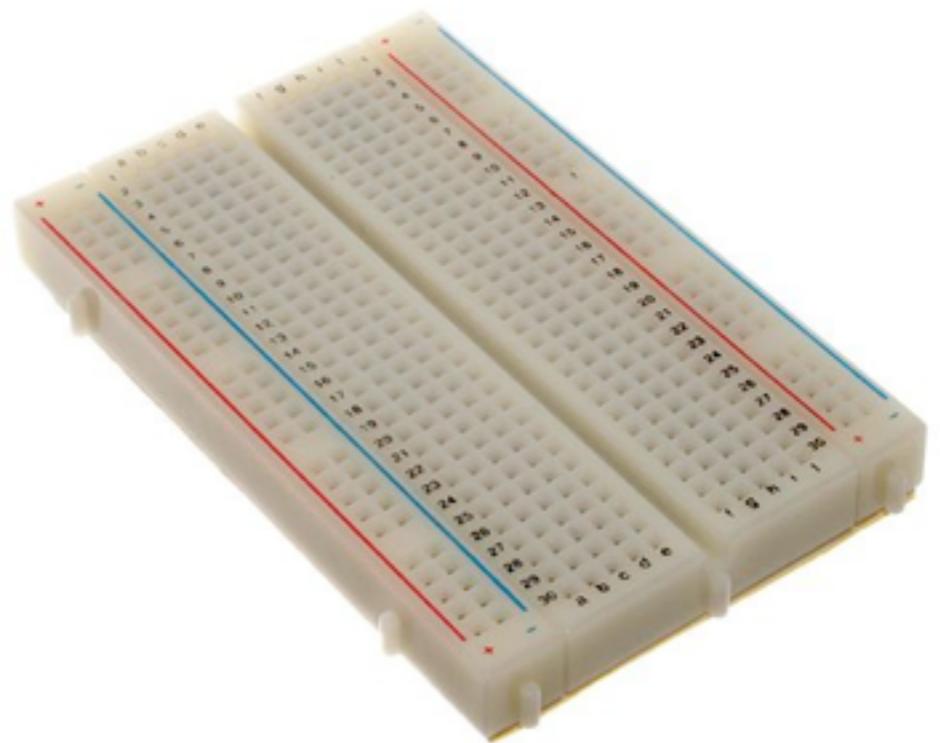
- 10 Push-button 6x6 mm
- 2 Potenciômetro 100kΩ
- 2 Sensores ópticos reflexivos
- 1 Sensor de luminosidade LDR 3mm
- 1 Sensor de temperatura LM35

Componentes de saída

- 12 LEDs 5mm, 2 de cada cor (verde, vermelho, amarelo, laranja, azul e branco)
- 2 Relés 5v
- 1 Display 7 segmentos 3 dígitos
- 1 Micro-servo 9g SG90 TowerPro
- 1 Buzzer contínuo

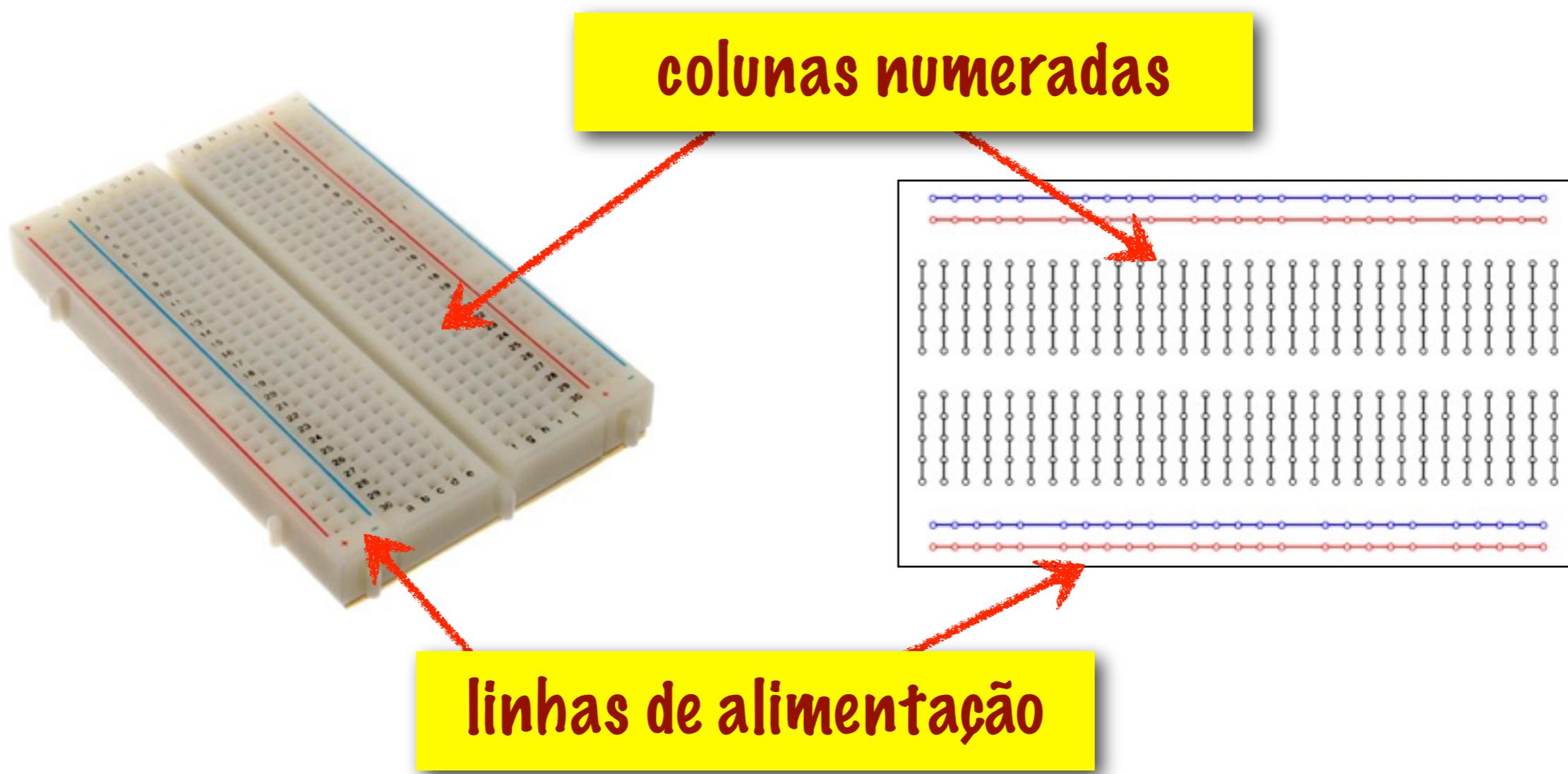
Breadboard

- Conhecido no Brasil como “protoboard”
- No kit: breadboard de 830 furos

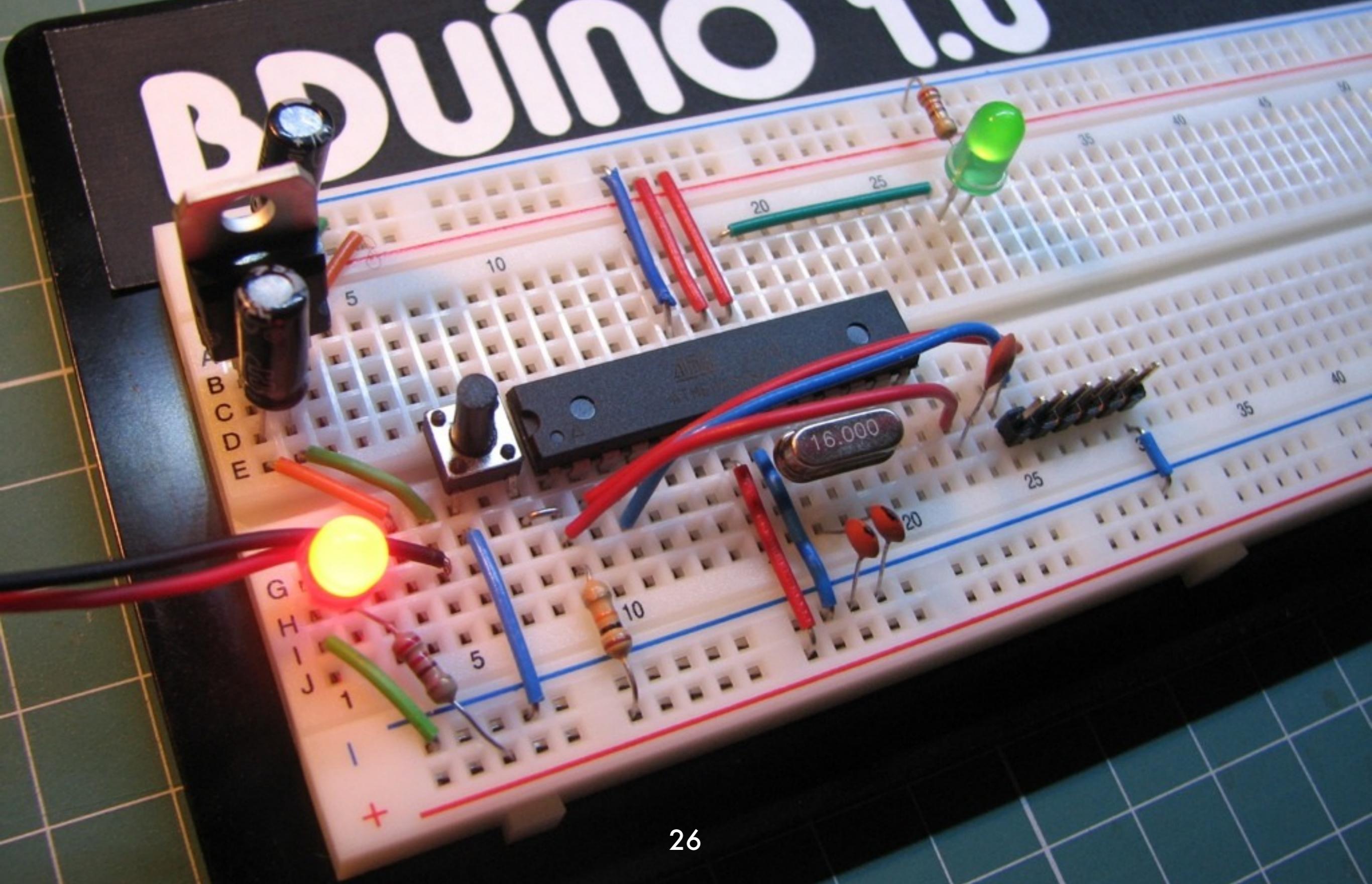


na foto: 400 furos

Esquema elétrico de um breadboard comum

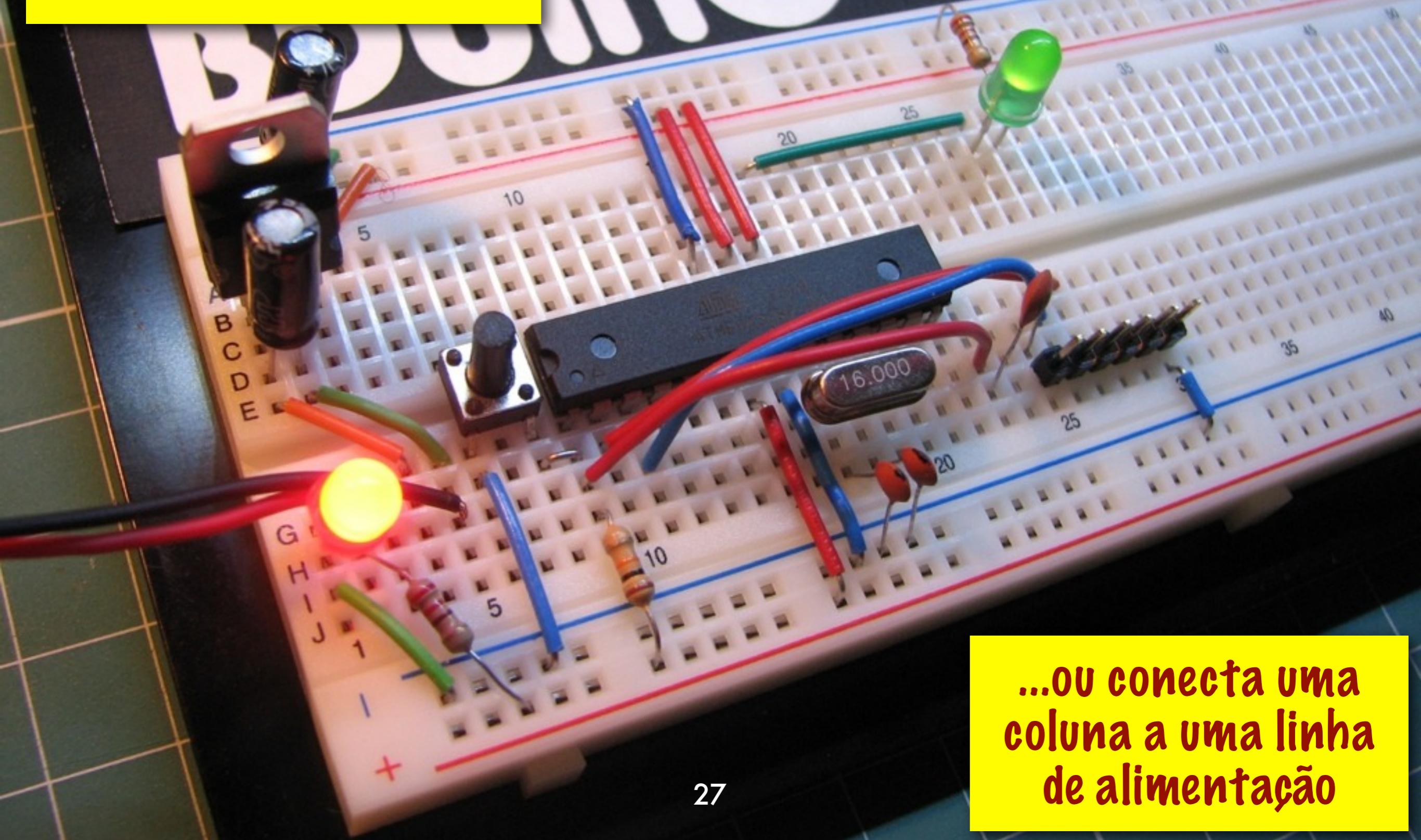


BDUINO 1.0

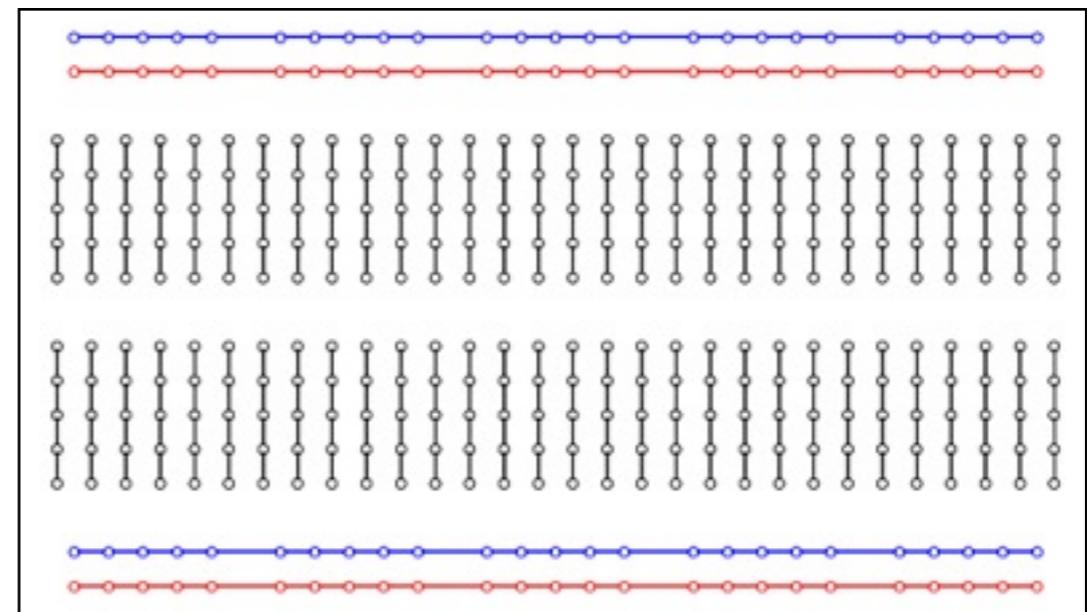
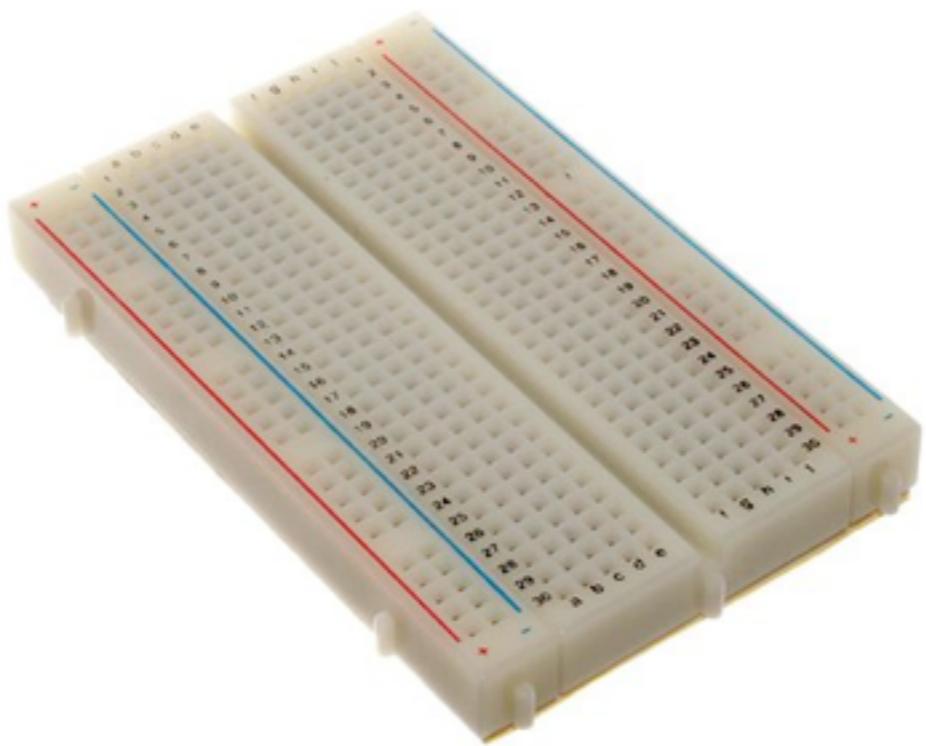


Note: cada componente se liga a duas ou mais colunas diferentes

Breadboard 1.0



Teste: posso ligar assim?



exemplos ao vivo...

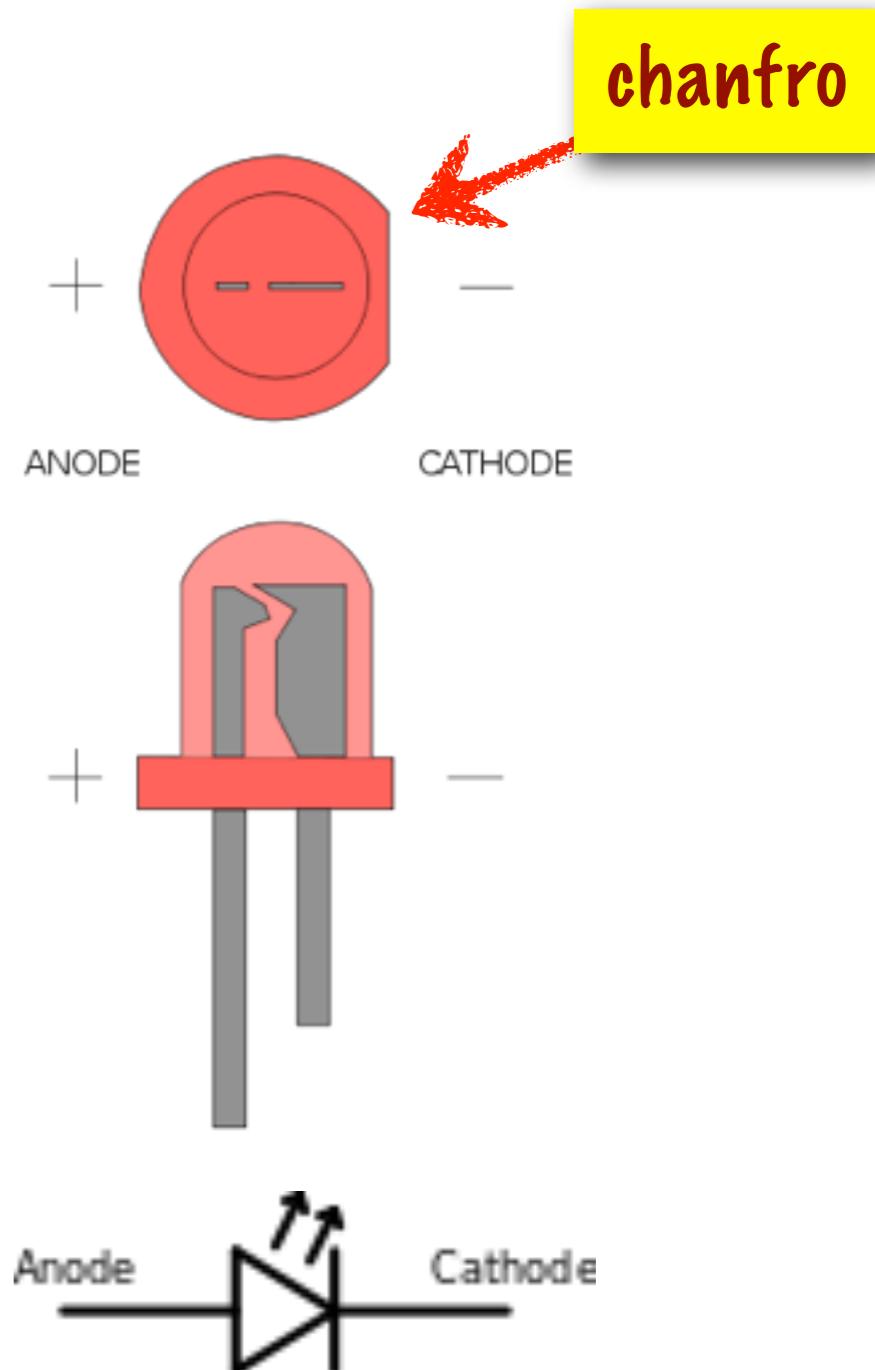
LED: light-emitting diode

- Diodo emissor de luz
- Componente polarizado:
tem direção certa para
ligar



LED: light-emitting diode

- Pino + (mais longo) ligado na fonte
- Pino - ligado no terra
- Ligar com resistor em série para proteção



LED RGB

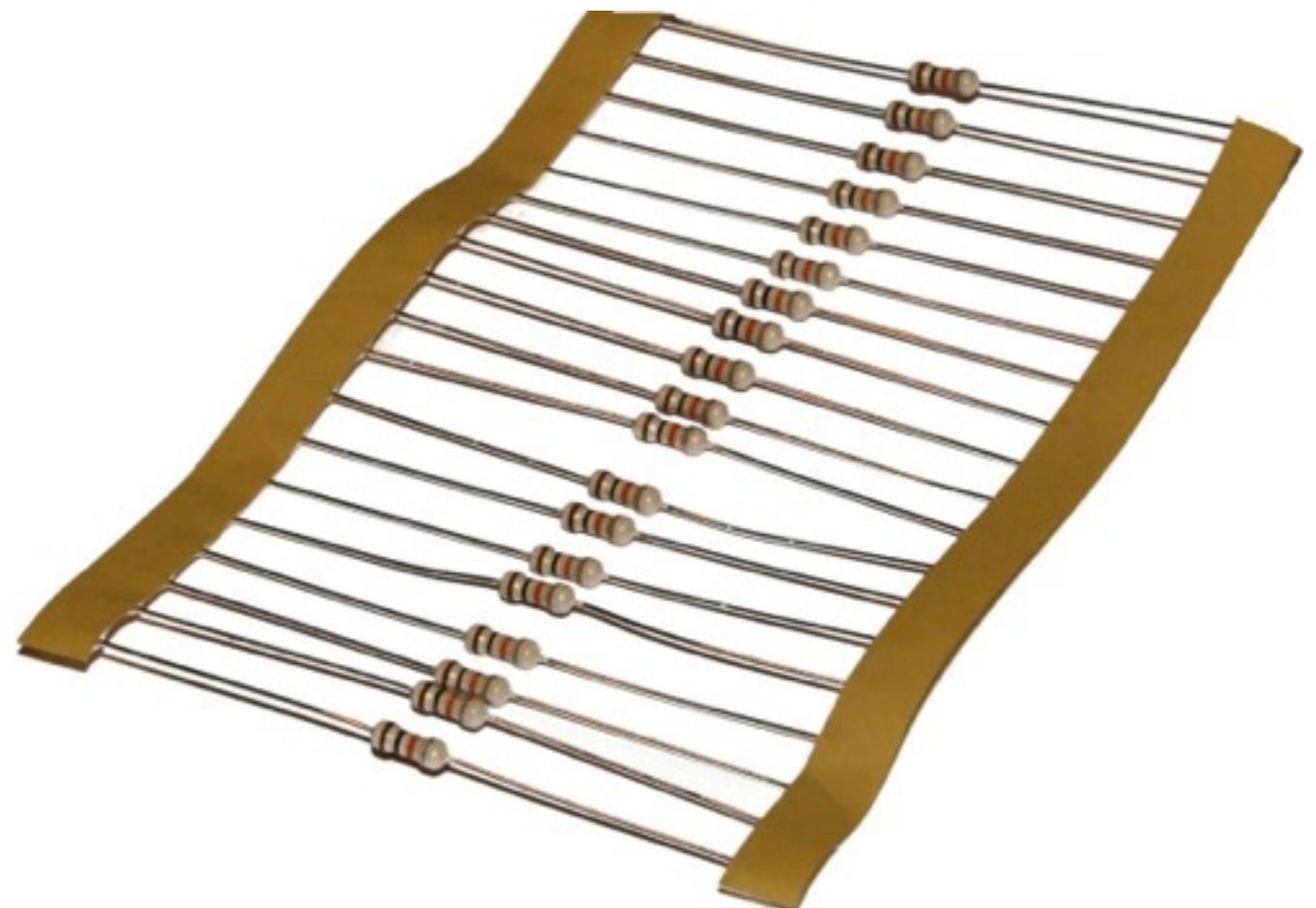
- 3 LEDs (vermelho, verde, azul) em um único componente



**não incluído
no kit**

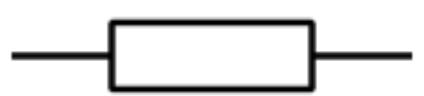
Resistores

- Kit: $20 \times 1\text{K } \Omega$



O que é um resistor

- Reduz a corrente
- Sem polaridade:
pode ser montado
em qualquer direção
- Valor em Ohms (Ω)
- Símbolos em esquemas:



Pequeno desvio
para entender o
resistor...

Tensão, corrente & resistência

Sentindo na pele

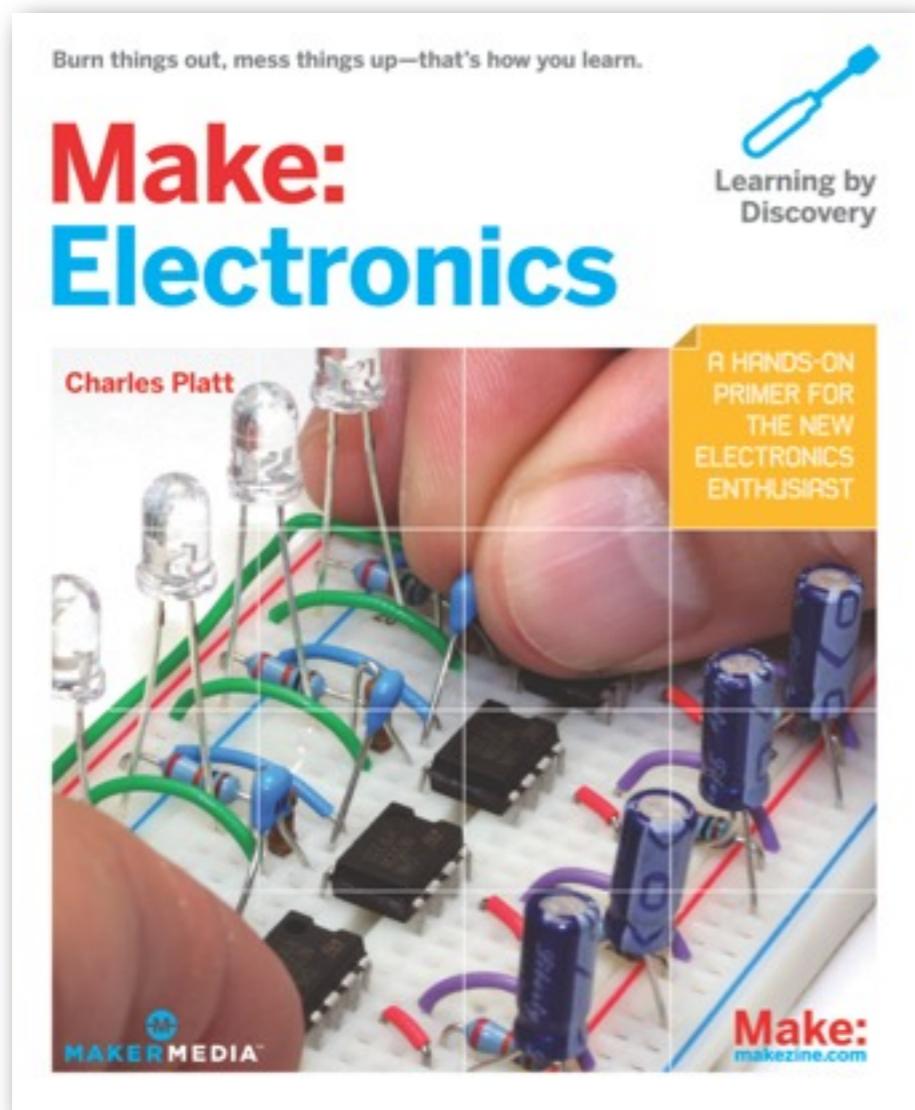
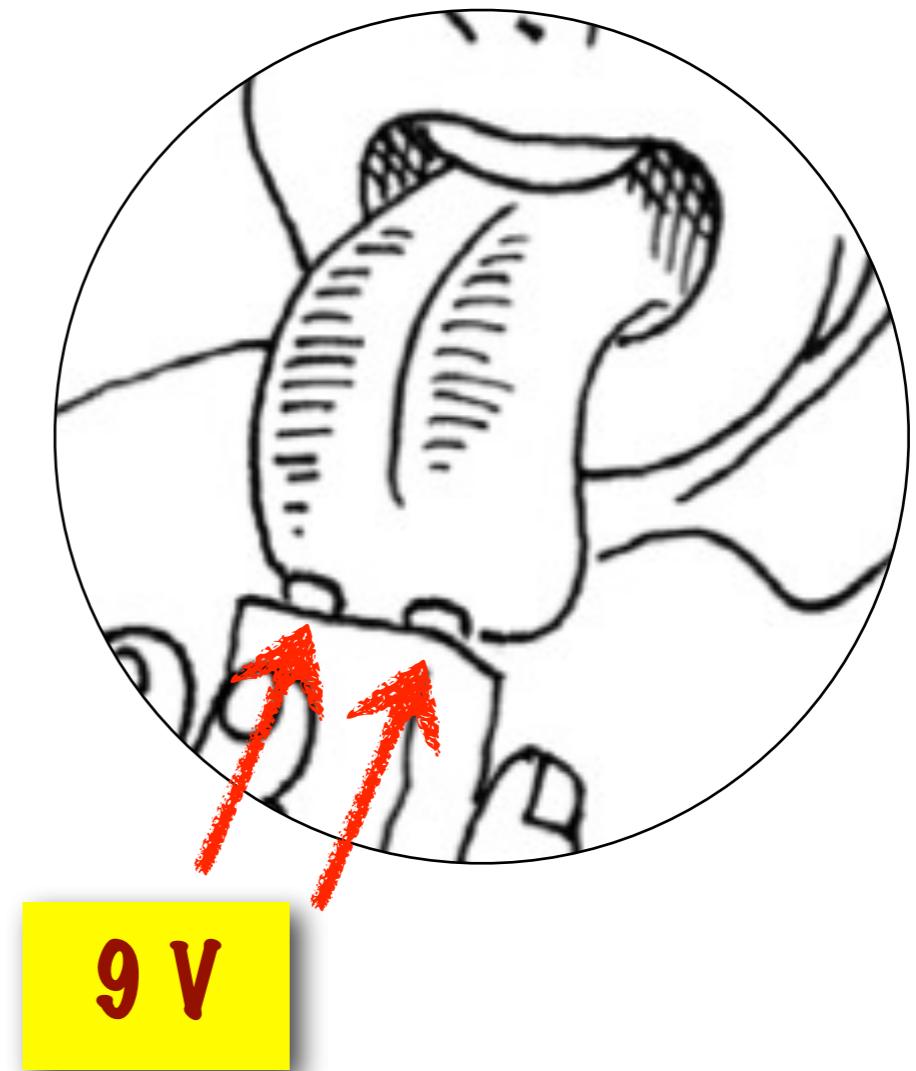


Figure 1-16. Step 1 in the process of learning by discovery: the 9-volt tongue test.

Cap. 1, p. 5

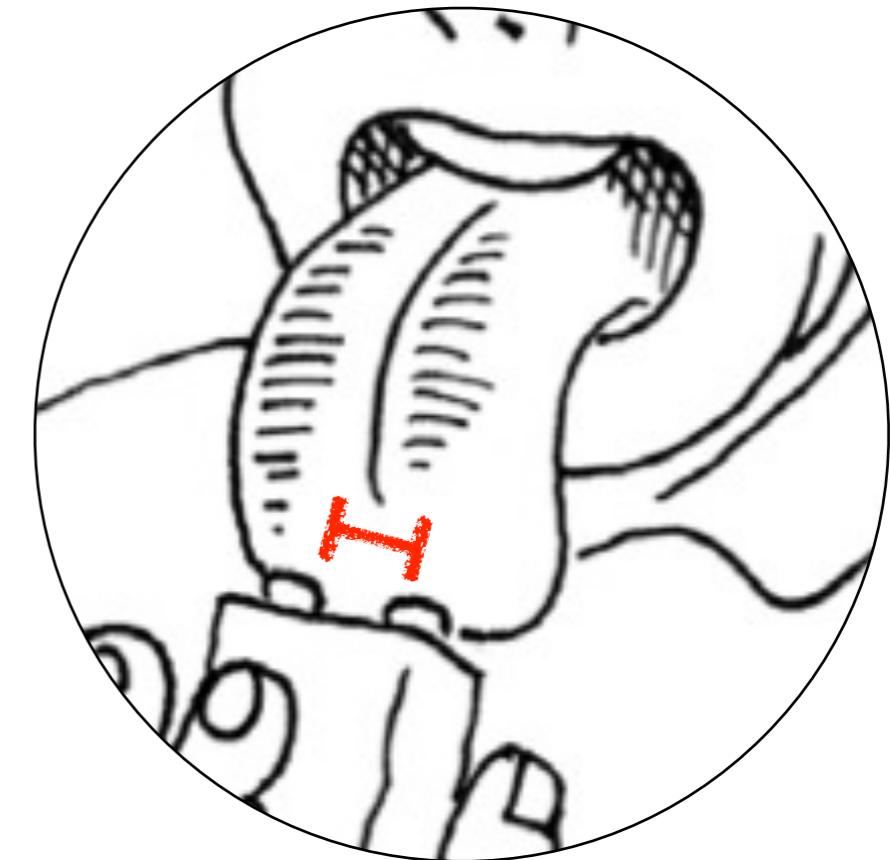
Tensão

- Medida: Volt (V)
- Símbolo em fórmulas: V



Resistência

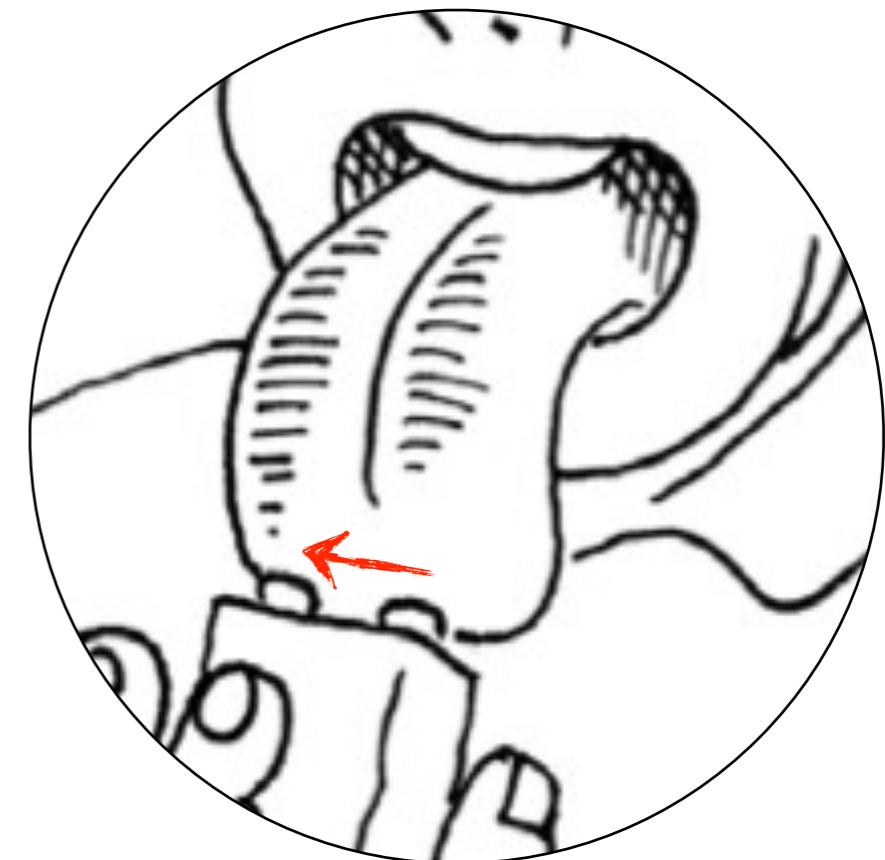
- Medida: Ohm (Ω)
- Símbolo em fórmulas: R



7mm de língua $\approx 330 \Omega$

Corrente

- Medida: Ampère (A)
- Símbolo em fórmulas: I
- 6.241×10^{18} elétrons por segundo

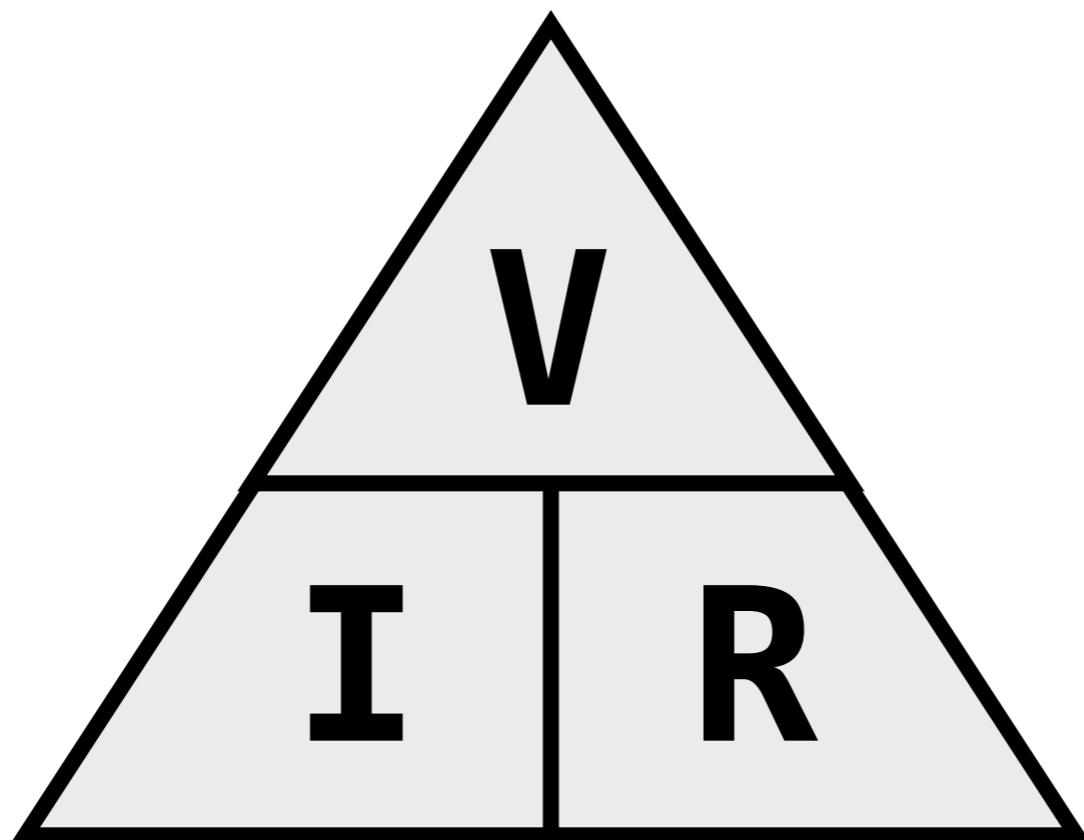


$$9\text{ V} \div 330\text{ }\Omega \approx 0.027\text{ A}$$
$$27\text{ mA}$$

Exemplos de corrente

Aparelho auditivo	0.7 mA	0.0007 A
Carregador celular simples	500 mA	0.5 A
Carregador tablet	2.1 A	2.1 A
Torradeira (120V)	16 A	16 A
Motor de arranque	120 A	120 A

Lei de Ohm



$$V = I \times R$$

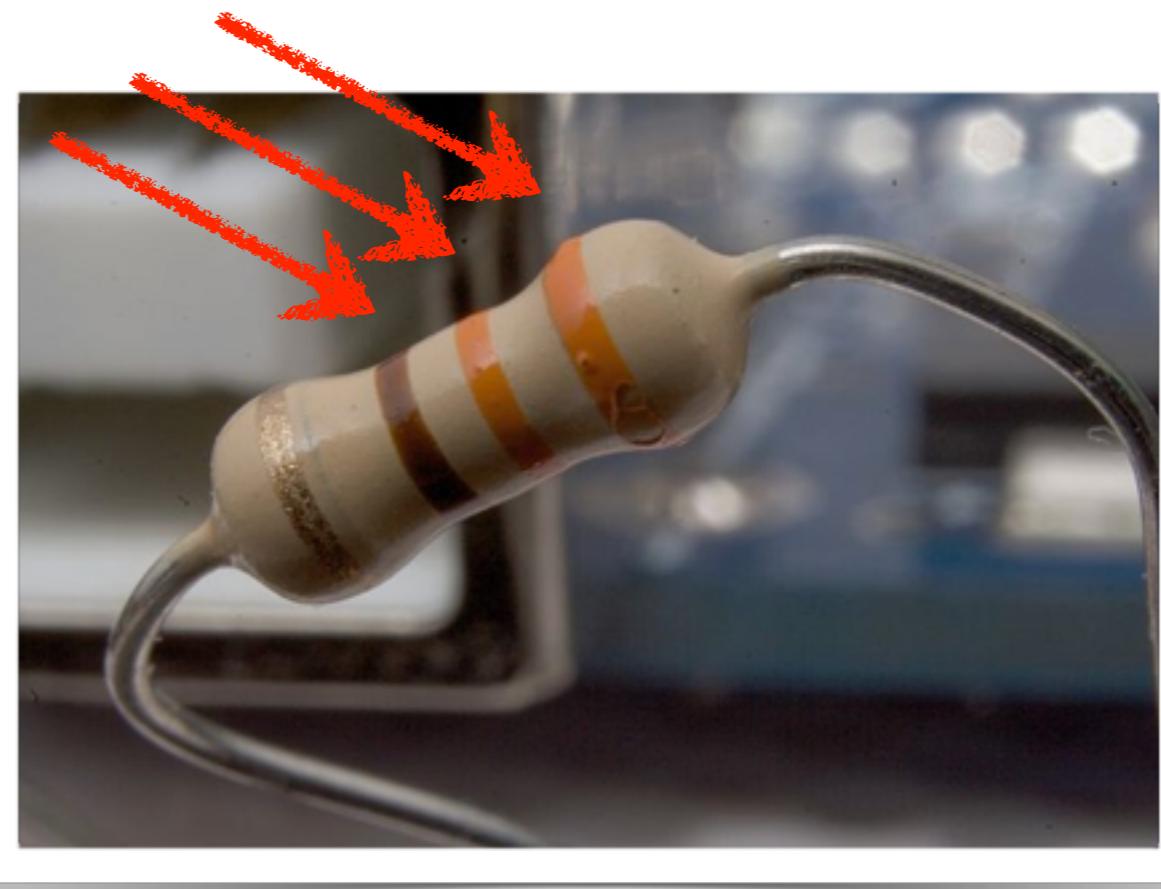
$$I = V / R$$

$$R = V / I$$

Voltando ao resistor...

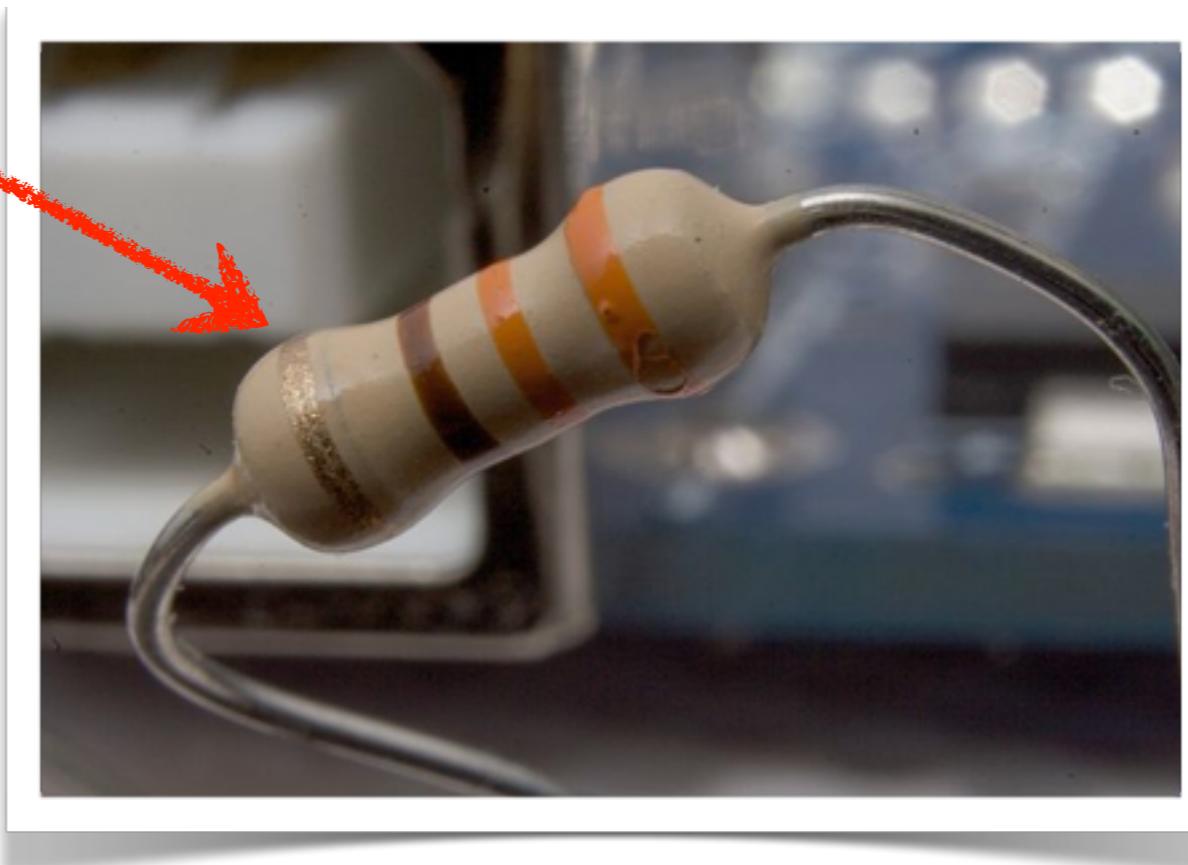
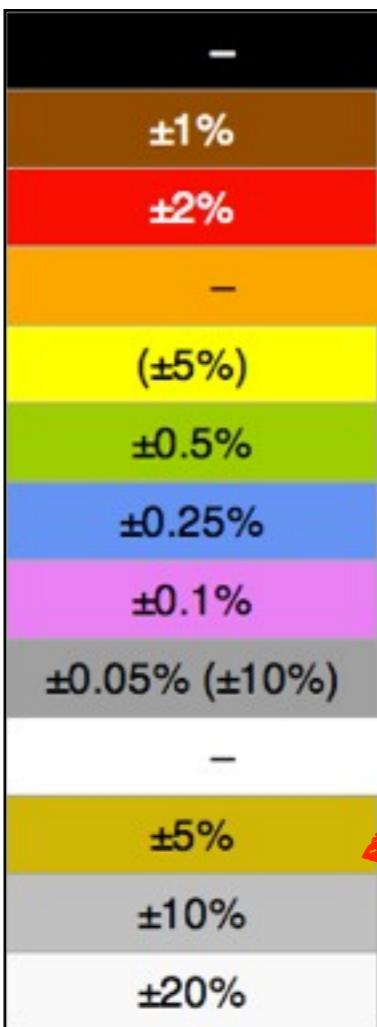
Resistor: código de cores

0	$\times 10^0$
1	$\times 10^1$
2	$\times 10^2$
3	$\times 10^3$
4	$\times 10^4$
5	$\times 10^5$
6	$\times 10^6$
7	$\times 10^7$
8	$\times 10^8$
9	$\times 10^9$



$$33 \times 10^1 = 330 \Omega$$

Resistor: código de cores

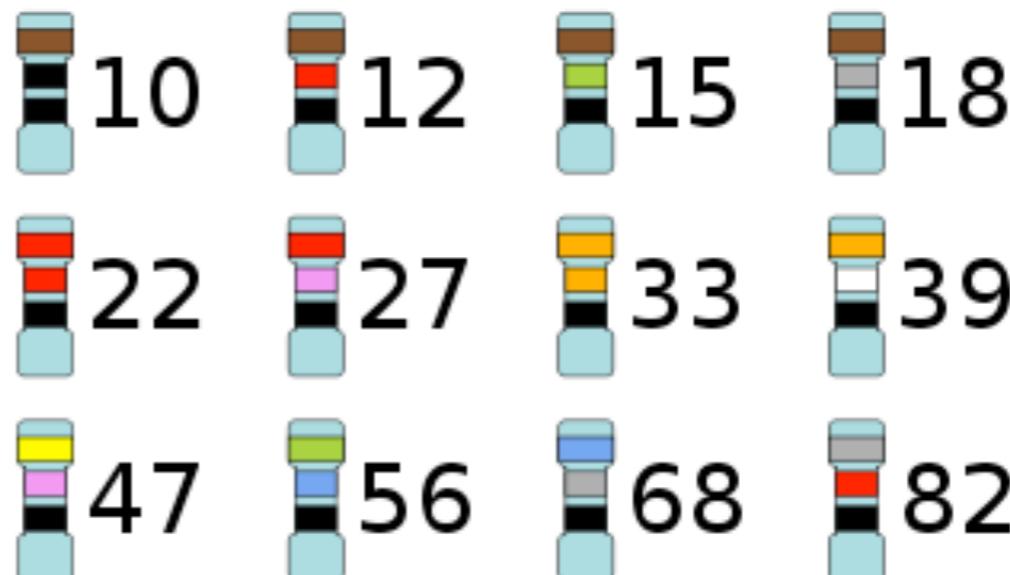


330Ω 5% de tolerância: de 313Ω a 346Ω

Resistor: valores padrão

0	$\times 10^0$
1	$\times 10^1$
2	$\times 10^2$
3	$\times 10^3$
4	$\times 10^4$
5	$\times 10^5$
6	$\times 10^6$
7	$\times 10^7$
8	$\times 10^8$
9	$\times 10^9$

Série E12 da Norma IEC 60063
(para tolerância 10%)



Resistor: exemplos

0	$\times 10^0$
1	$\times 10^1$
2	$\times 10^2$
3	$\times 10^3$
4	$\times 10^4$
5	$\times 10^5$
6	$\times 10^6$
7	$\times 10^7$
8	$\times 10^8$
9	$\times 10^9$

330Ω



$1 \text{ k}\Omega$



$10 \text{ k}\Omega$

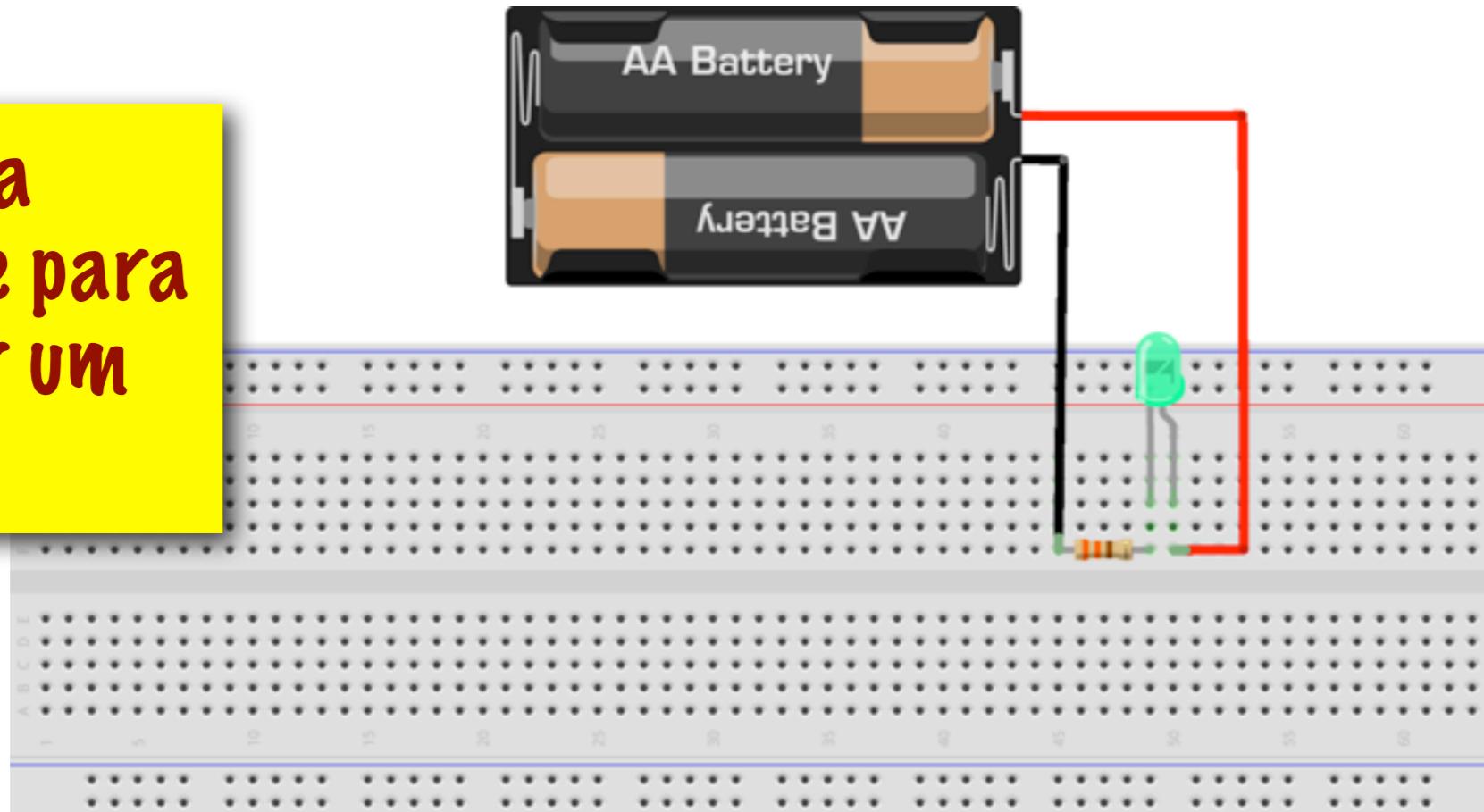


$1 \text{ M}\Omega$



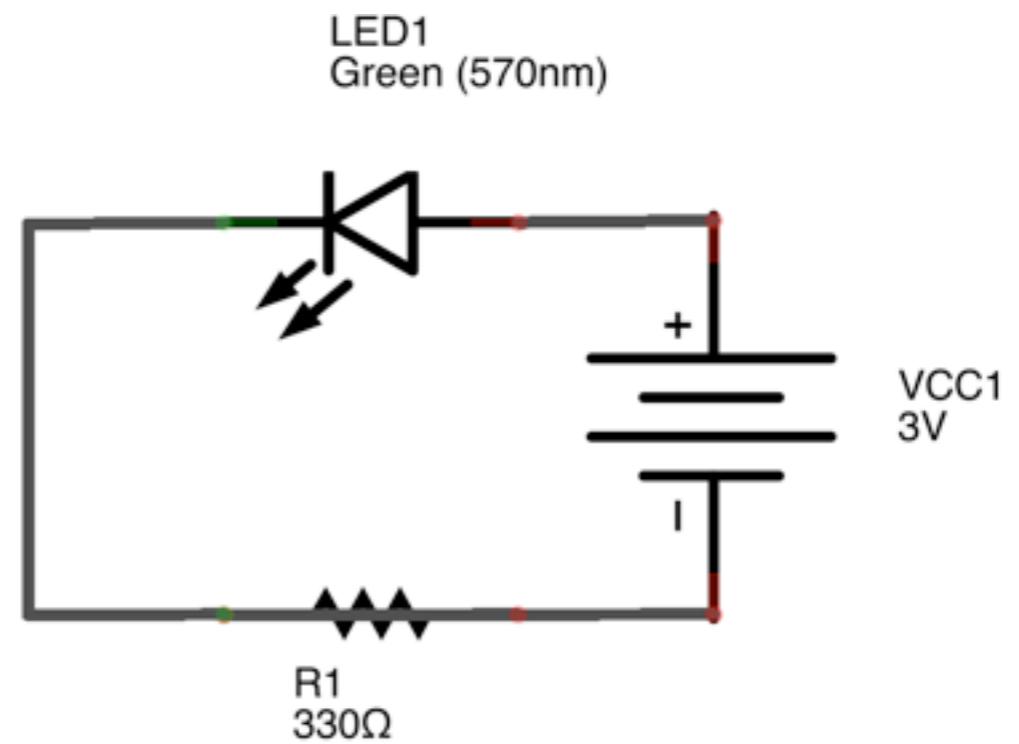
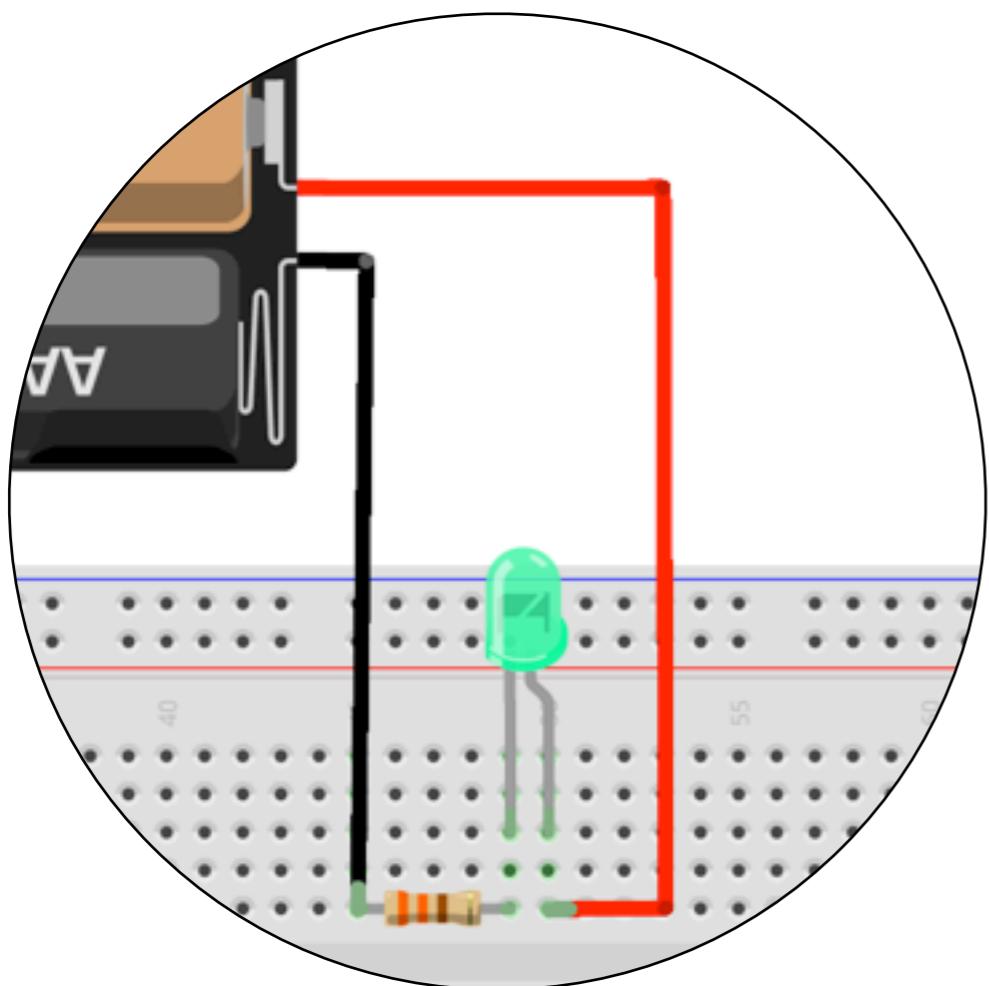
Resistor: exemplo de uso

Limitar a corrente para proteger um LED

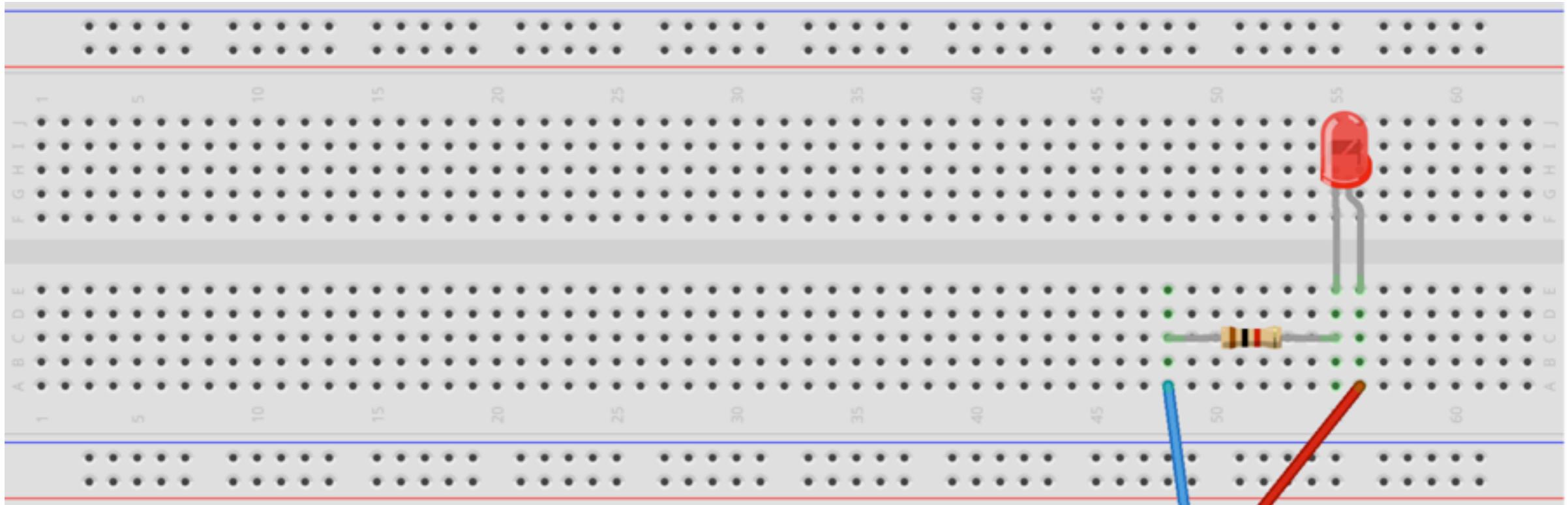


Made with Fritzing.org

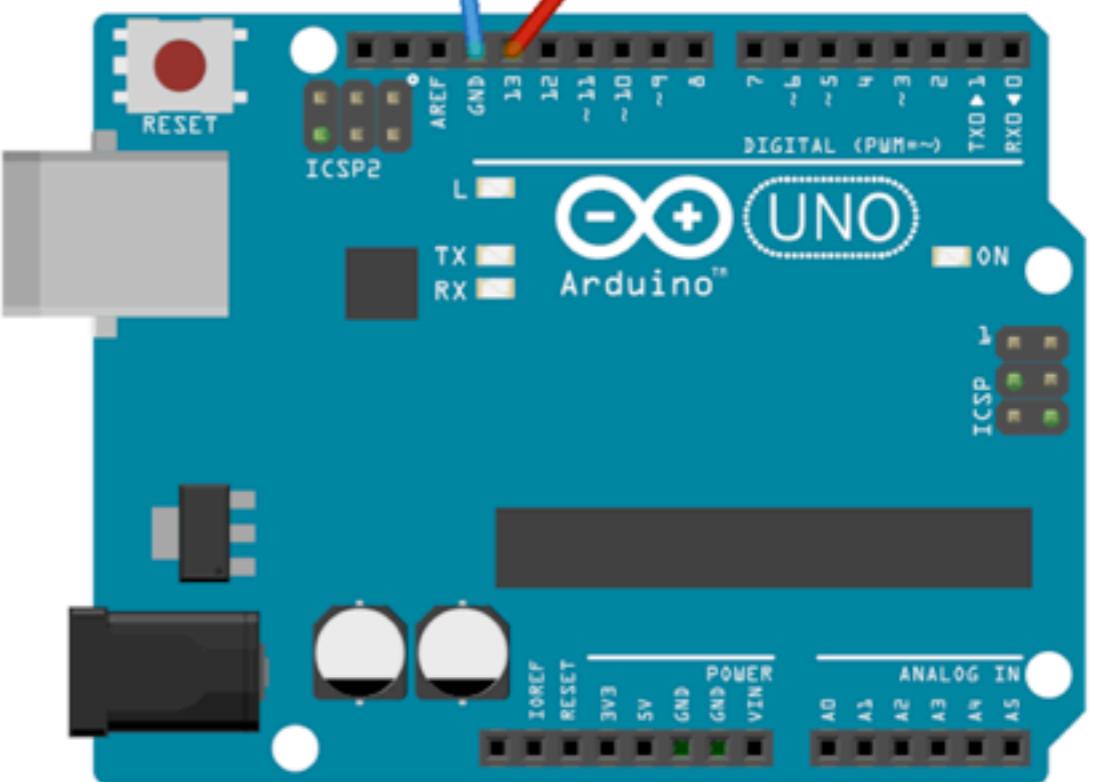
Resistor: exemplo de uso



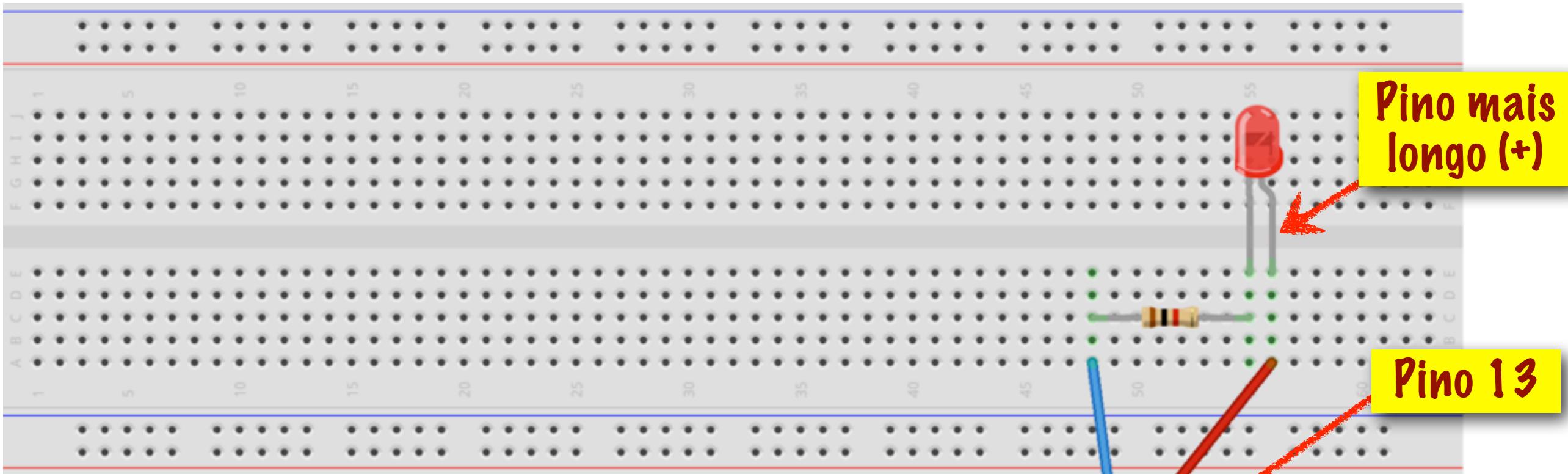
Primeiro circuito



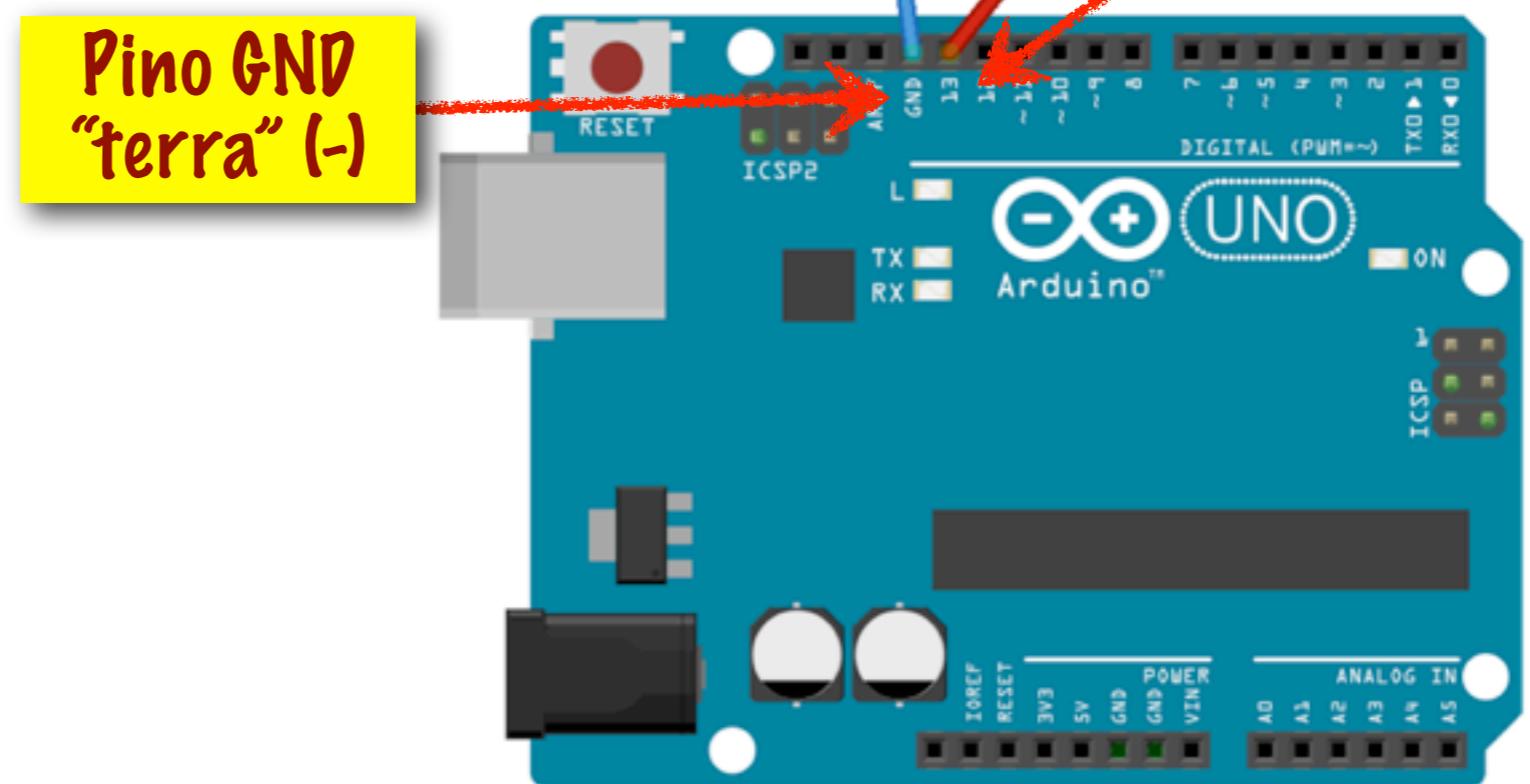
Primeiro circuito: blink



Made with  Fritzing.org

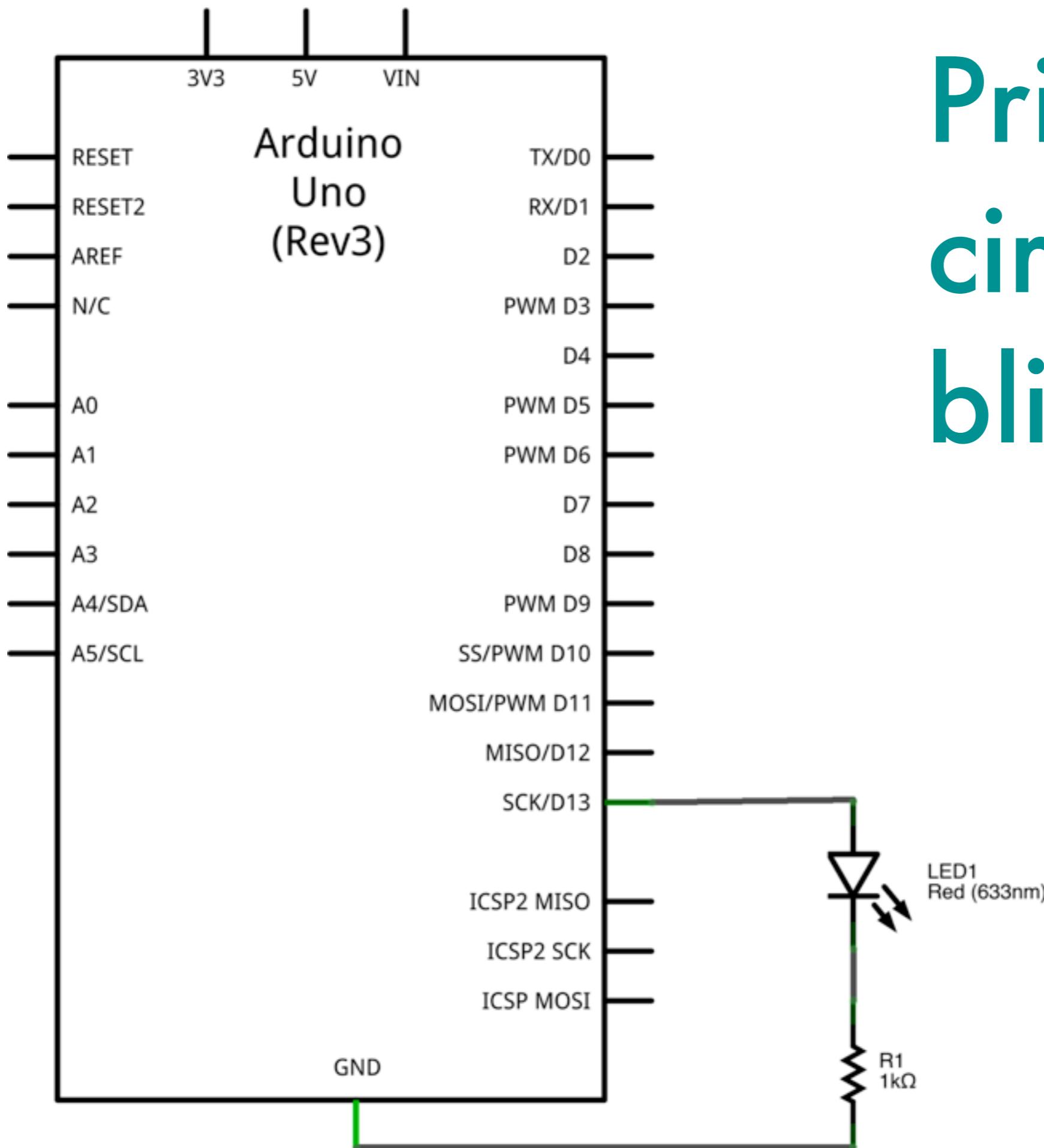


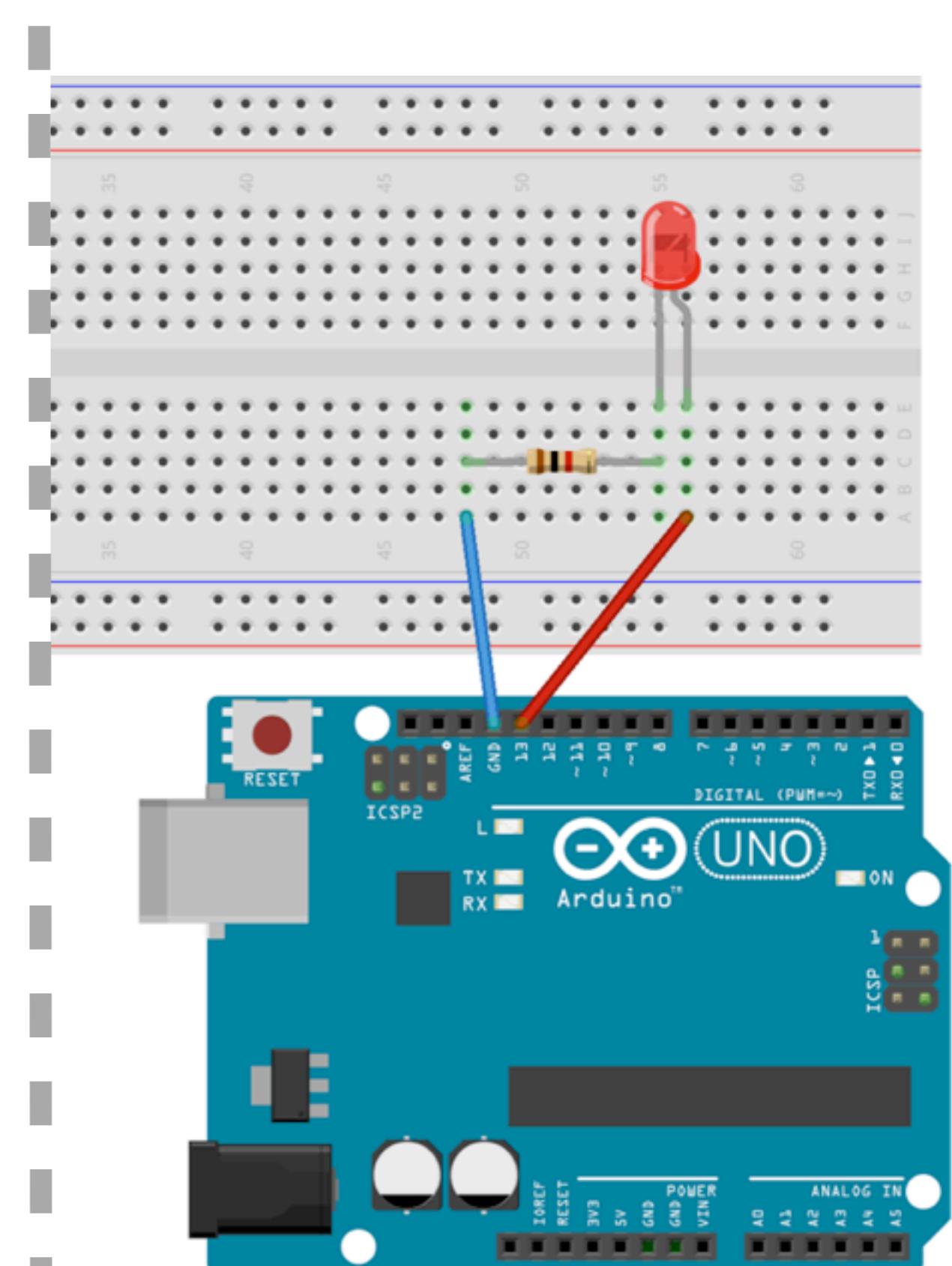
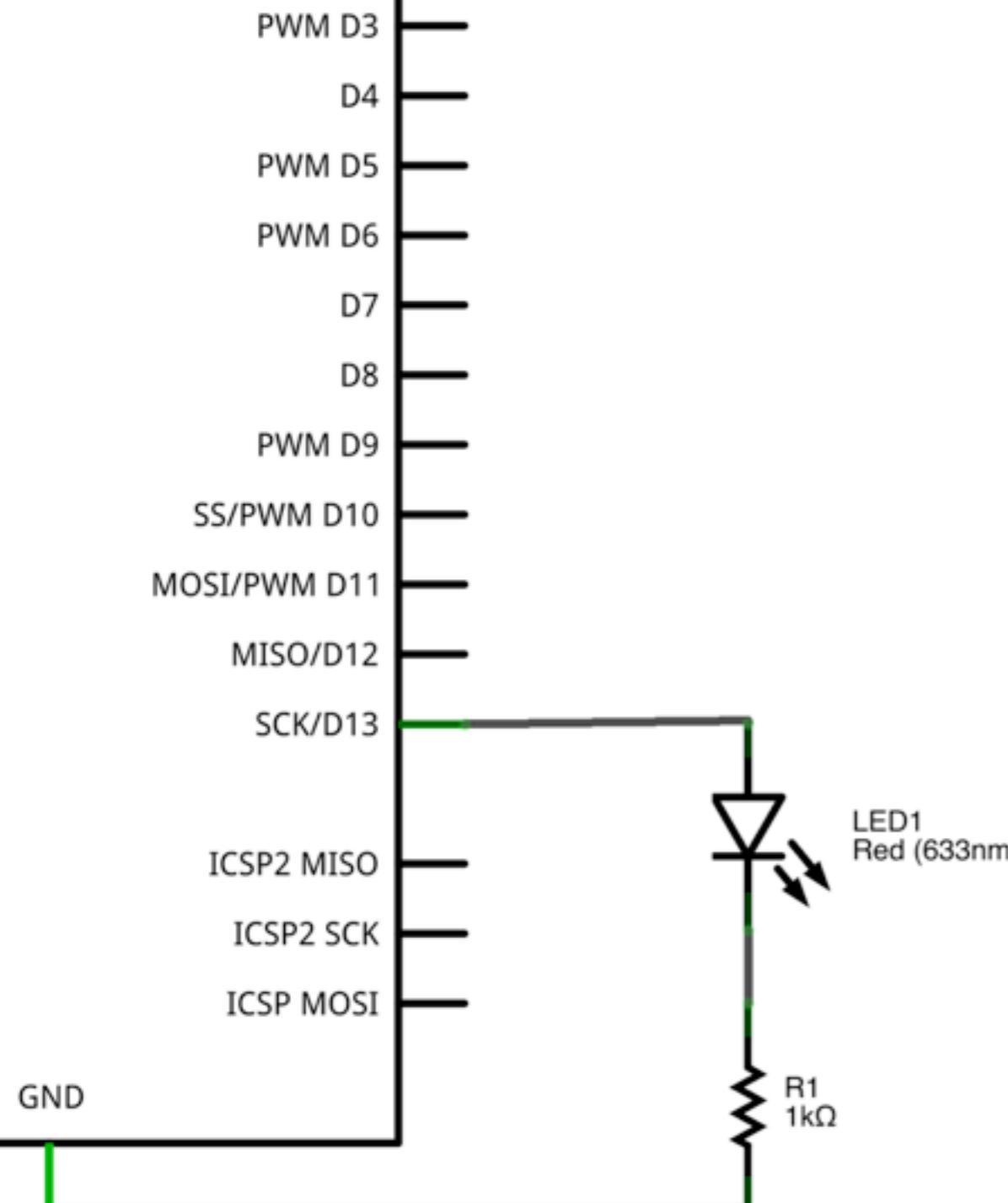
Primeiro circuito: blink



Made with  Fritzing.org

Primeiro círculo: blink



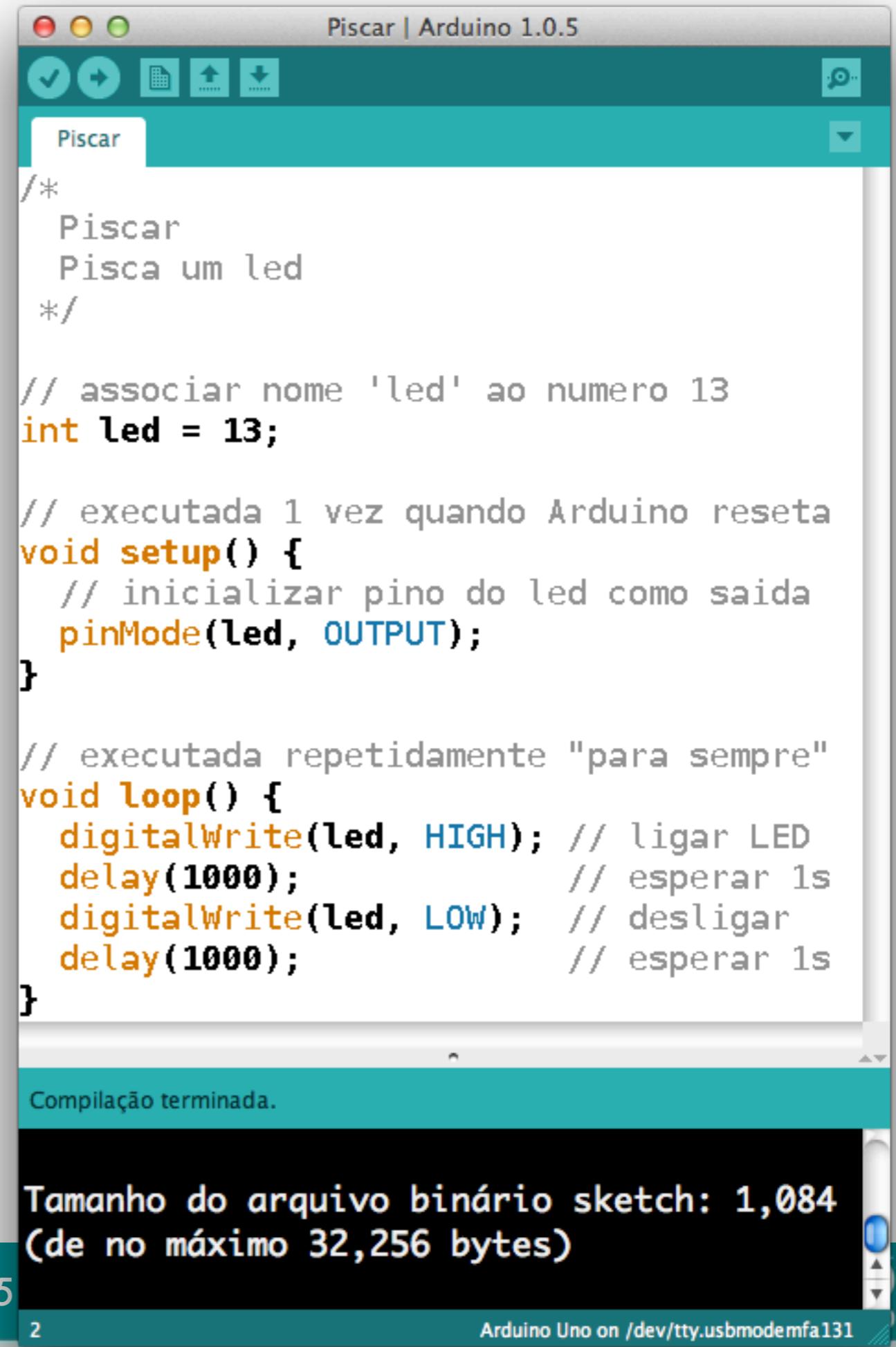


Programação

Arduino IDE

- Editor fácil,
baseado no
Processing, feito
para artistas

arduino.cc/en/Main/Software



The screenshot shows the Arduino IDE interface with a sketch titled "Piscar". The code is as follows:

```
/*
  Piscar
  Pisca um led
 */

// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
  // inicializar pino do led como saida
  pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
  digitalWrite(led, HIGH); // ligar LED
  delay(1000);           // esperar 1s
  digitalWrite(led, LOW); // desligar
  delay(1000);           // esperar 1s
}
```

The status bar at the bottom indicates "Compilação terminada." (Compilation finished) and "Tamanho do arquivo binário sketch: 1,084 (de no máximo 32,256 bytes)" (Size of binary sketch file: 1,084 (of maximum 32,256 bytes)).

Linguagem

- C++, com simplificações

```
/*
  Piscar
  Pisca um led
*/

// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
    // inicializar pino do led como saida
    pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
    digitalWrite(led, HIGH); // ligar LED
    delay(1000);           // esperar 1s
    digitalWrite(led, LOW); // desligar
    delay(1000);           // esperar 1s
}
```

Estrutura

comentários



```
/*
  Piscar
  Pisca um led
*/
// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
    // inicializar pino do led como saida
    pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
    digitalWrite(led, HIGH); // ligar LED
    delay(1000);           // esperar 1s
    digitalWrite(led, LOW); // desligar
    delay(1000);           // esperar 1s
}
```

Estrutura

comentários

declaração
de variável

```
/*
  Piscar
  Pisca um led
*/
// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
  // inicializar pino do led como saída
  pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
  digitalWrite(led, HIGH); // ligar LED
  delay(1000);           // esperar 1s
  digitalWrite(led, LOW); // desligar
  delay(1000);           // esperar 1s
}
```

Estrutura

comentários

declaração
de variável

definições
de funções

```
/*
  Piscar
  Pisca um led
*/
// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
  // inicializar pino do led como saída
  pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
  digitalWrite(led, HIGH); // ligar LED
  delay(1000);           // esperar 1s
  digitalWrite(led, LOW); // desligar
  delay(1000);           // esperar 1s
}
```

Estrutura

comentários

declaração
de variável

definições
de funções

chamada
de função

```
/*
  Piscar
  Pisca um led
*/
// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
  // inicializar pino do led como saída
  pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
  digitalWrite(led, HIGH); // ligar LED
  delay(1000);           // esperar 1s
  digitalWrite(led, LOW); // desligar
  delay(1000);           // esperar 1s
}
```

Sintaxe

blocos de
instruções
delimitadas
por {...}

```
/*
  Piscar
  Pisca um led
*/

// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
  // inicializar pino do led como saida
  pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
  digitalWrite(led, HIGH); // ligar LED
  delay(1000);           // esperar 1s
  digitalWrite(led, LOW); // desligar
  delay(1000);           // esperar 1s
}
```

Sintaxe

```
/*
  Piscar
  Pisca um led
*/

// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
    // inicializar pino do led como saida
    pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
    digitalWrite(led, HIGH); // ligar LED
    delay(1000);           // esperar 1s
    digitalWrite(led, LOW); // desligar
    delay(1000);           // esperar 1s
}
```

editor
assinala pares
de chaves {...}

Sintaxe

editor marca
código com
cores para
ajudar na
leitura e
revisão

```
/*
  Piscar
  Pisca um led
 */

// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
    // inicializar pino do led como saida
    pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
    digitalWrite(led, HIGH); // ligar LED
    delay(1000);           // esperar 1s
    digitalWrite(led, LOW); // desligar
    delay(1000);           // esperar 1s
}
```

Dicas de sintaxe para C++

- Cada vírgula conta, mas espaços não
- Maiúscula ≠ minúscula
 - ex. Alfa ≠ alfa ≠ ALFA
- Todo comando termina com ;
- O editor assinala os pares de (), {}, e []

Lógica

Arduino
executa a
função
setup() uma
vez após
resetar



```
/*
  Piscar
  Pisca um led
*/

// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino resetar
void setup() {
  // inicializar pino do led como saida
  pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
  digitalWrite(led, HIGH); // ligar LED
  delay(1000);           // esperar 1s
  digitalWrite(led, LOW); // desligar
  delay(1000);           // esperar 1s
}
```

Lógica

aqui o pino
do led é
configurado
para saída

```
/*
  Piscar
  Pisca um led
 */

// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
  // inicializar pino do led como saída
  pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
  digitalWrite(led, HIGH); // ligar LED
  delay(1000);           // esperar 1s
  digitalWrite(led, LOW); // desligar
  delay(1000);           // esperar 1s
}
```

Lógica

Arduino
executa a
função `loop()`
repetidamente



```
/*
  Piscar
  Pisca um led
*/

// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
  // inicializar pino do led como saida
  pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
  digitalWrite(led, HIGH); // ligar LED
  delay(1000);           // esperar 1s
  digitalWrite(led, LOW); // desligar
  delay(1000);           // esperar 1s
}
```

Lógica

digitalWrite
serve para
mudar o
estado de um
pino digital

```
/*
  Piscar
  Pisca um led
*/

// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
  // inicializar pino do led como saida
  pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
  digitalWrite(led, HIGH); // ligar LED
  delay(1000);           // esperar 1s
  digitalWrite(led, LOW); // desligar
  delay(1000);           // esperar 1s
}
```

Lógica

Onde você
mudaria o
código para
acionar um
LED ligado ao
pino 8?

```
/*
  Piscar
  Pisca um led
 */

// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
    // inicializar pino do led como saida
    pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
    digitalWrite(led, HIGH); // ligar LED
    delay(1000);           // esperar 1s
    digitalWrite(led, LOW); // desligar
    delay(1000);           // esperar 1s
}
```

Lógica

Onde você
mudaria o
código para
mudar a
frequência
das piscadas?

```
/*
  Piscar
  Pisca um led
 */

// associar nome 'led' ao numero 13
int led = 13;

// executada 1 vez quando Arduino reseta
void setup() {
    // inicializar pino do led como saida
    pinMode(led, OUTPUT);
}

// executada repetidamente "para sempre"
void loop() {
    digitalWrite(led, HIGH); // ligar LED
    delay(1000);           // esperar 1s
    digitalWrite(led, LOW); // desligar
    delay(1000);           // esperar 1s
}
```

Coding Dojo com Arduino

Slides do Garoa Hacker Clube

Arduino & cia.

MAKERS

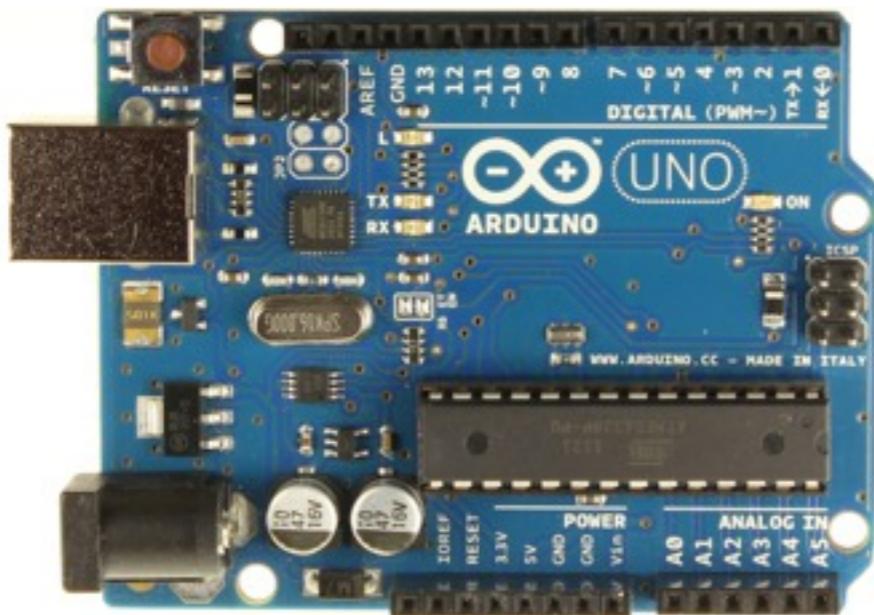
72

GIZMODO APRESENTA

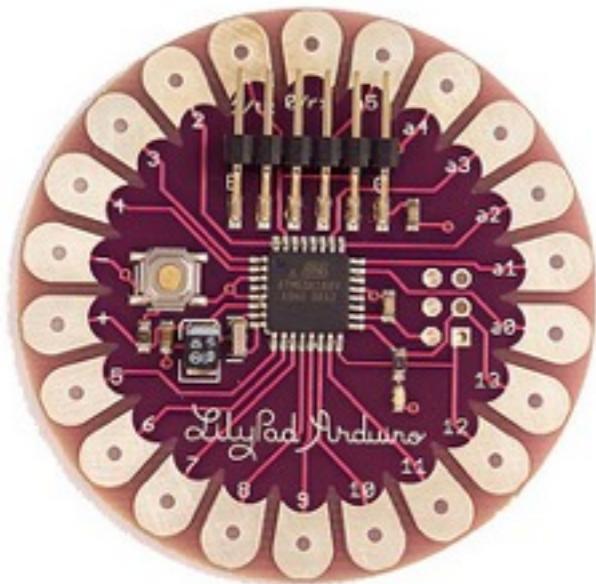
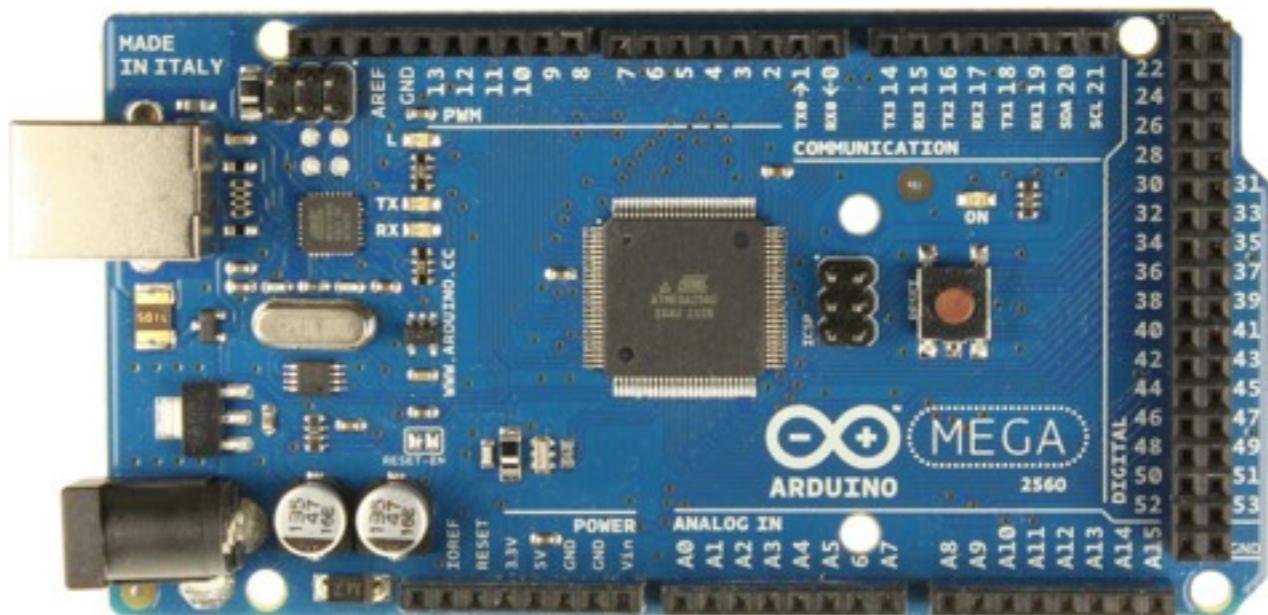


Família Arduino

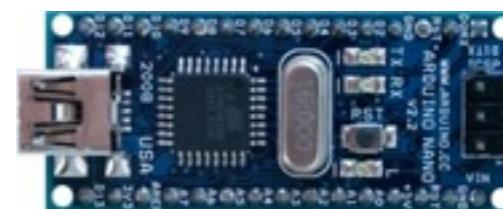
Uno



Mega



Lilypad

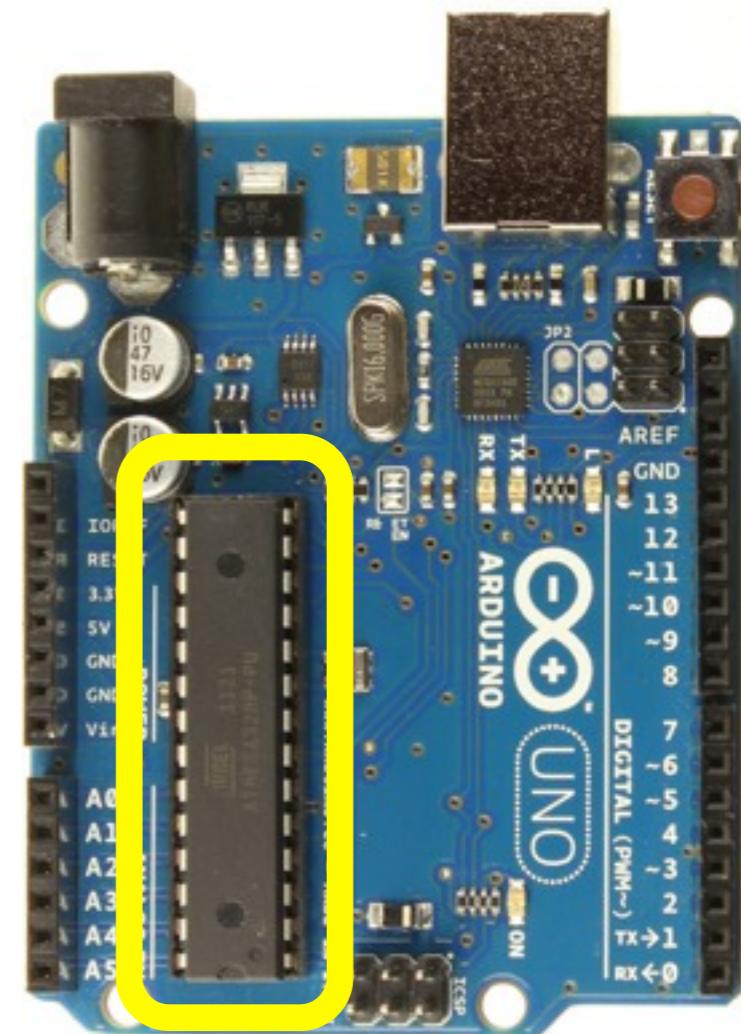


Nano

etc...

Microcontrolador do UNO: Atmel ATmega328

- família “AVR”
- clock: 16 MHz
- SRAM: 2 KB
- EEPROM: 1 KB
- Flash: 32 KB



Entradas e saídas no ATmega328

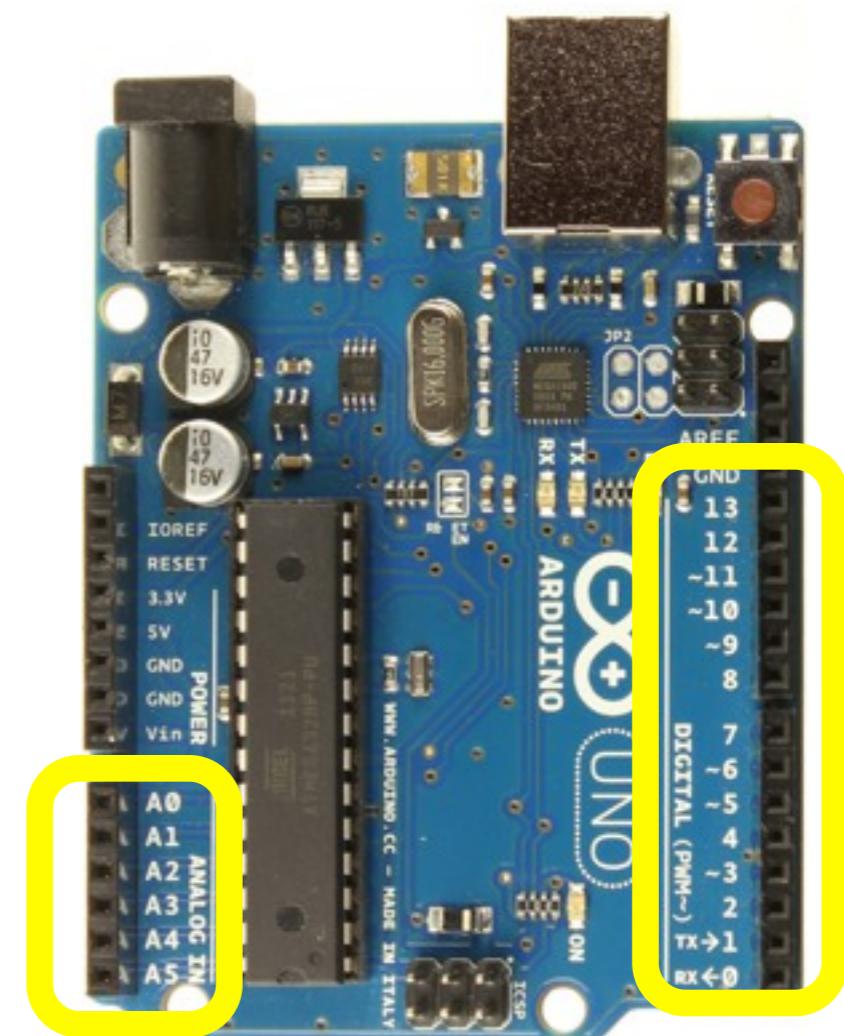
- 28 pinos
- 23 pinos multi-funcionais

(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)
(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)
VCC	7	22	GND
GND	8	21	AREF
(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC
(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)
(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)

diagrama do datasheet

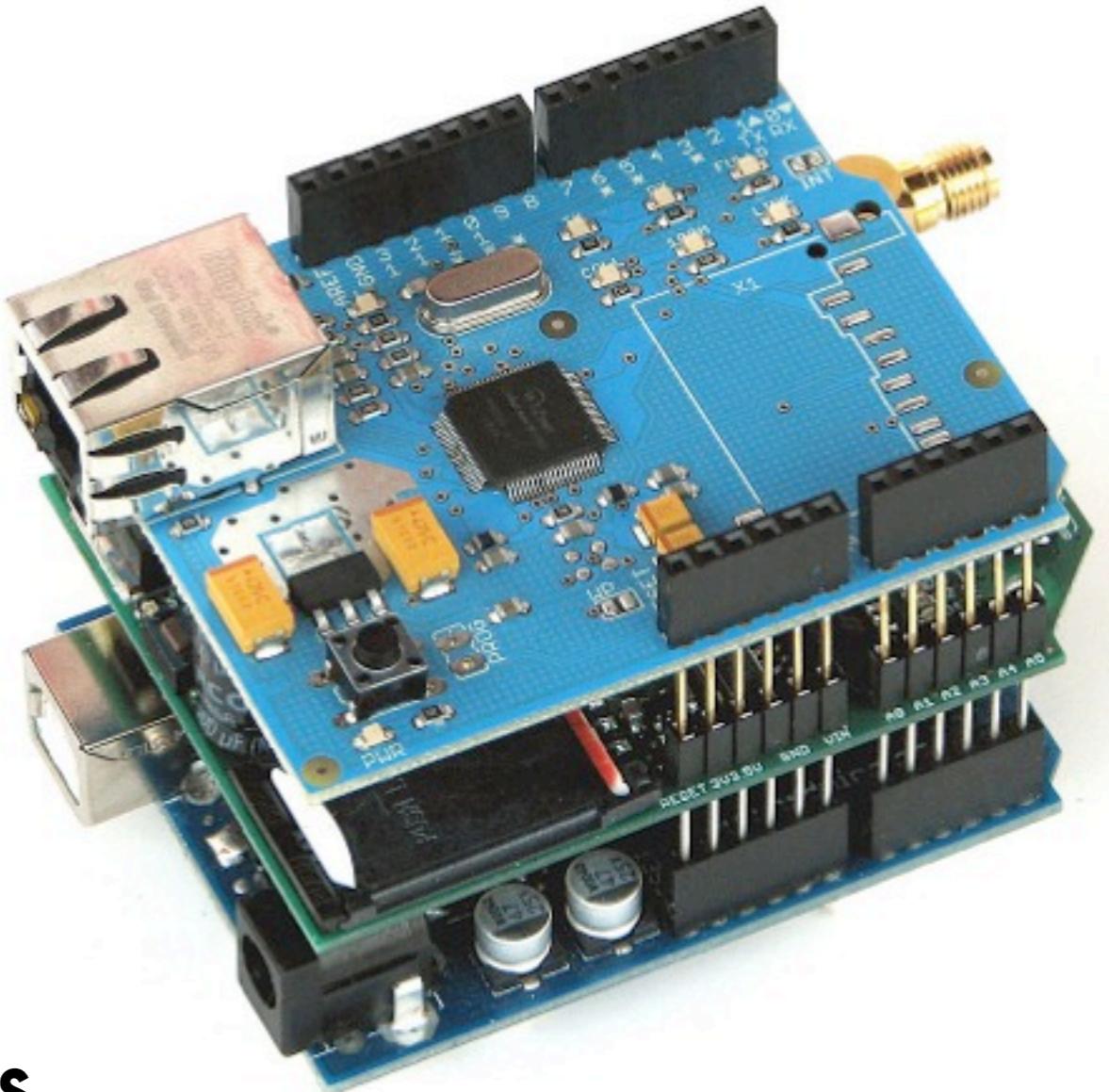
Entradas e saídas no Arduino UNO

- Função dos pinos: padronizadas e simplificadas
- Mais fácil de aprender
- Mais fácil de expandir



Shields

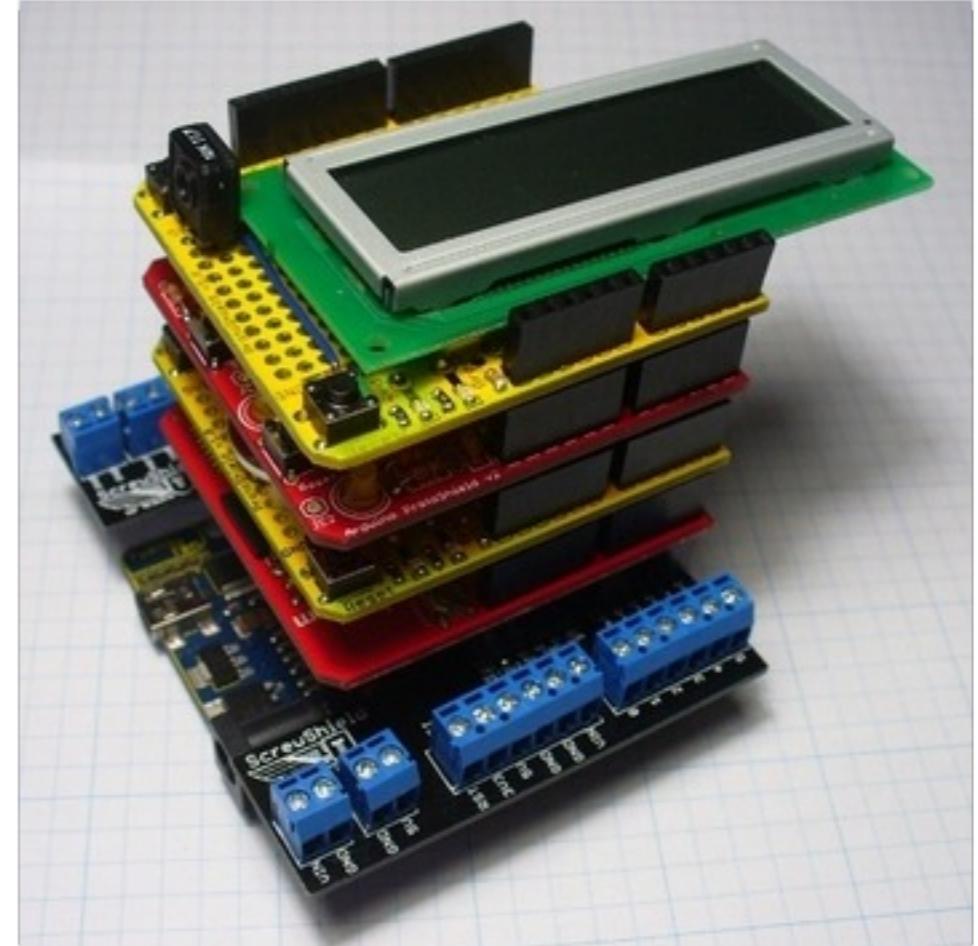
- Placas de expansão
- Alguns exemplos:
 - Ethernet, Wi-Fi, controle de motores, acelerômetro, GPS, tela LCD touch...



Arduino com dois
shields empilhados

Shields

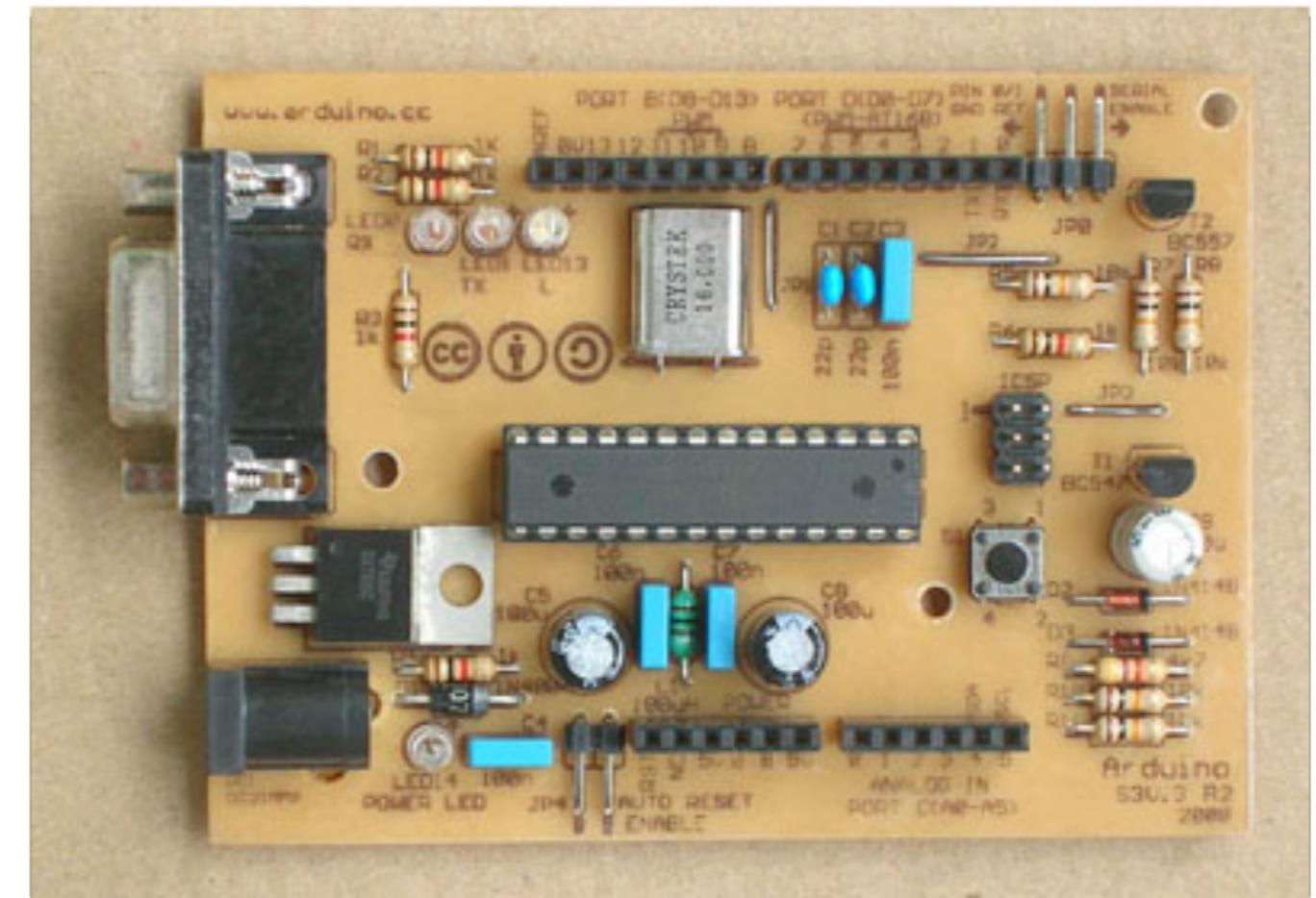
- Mais exemplos:
 - GPRS, NFC/RFID, MIDI sequencer, MP3 decoder, controle de câmera fotográfica, XBee radio...
- Imperdível: shieldlist.org



5
shields
empilhados!

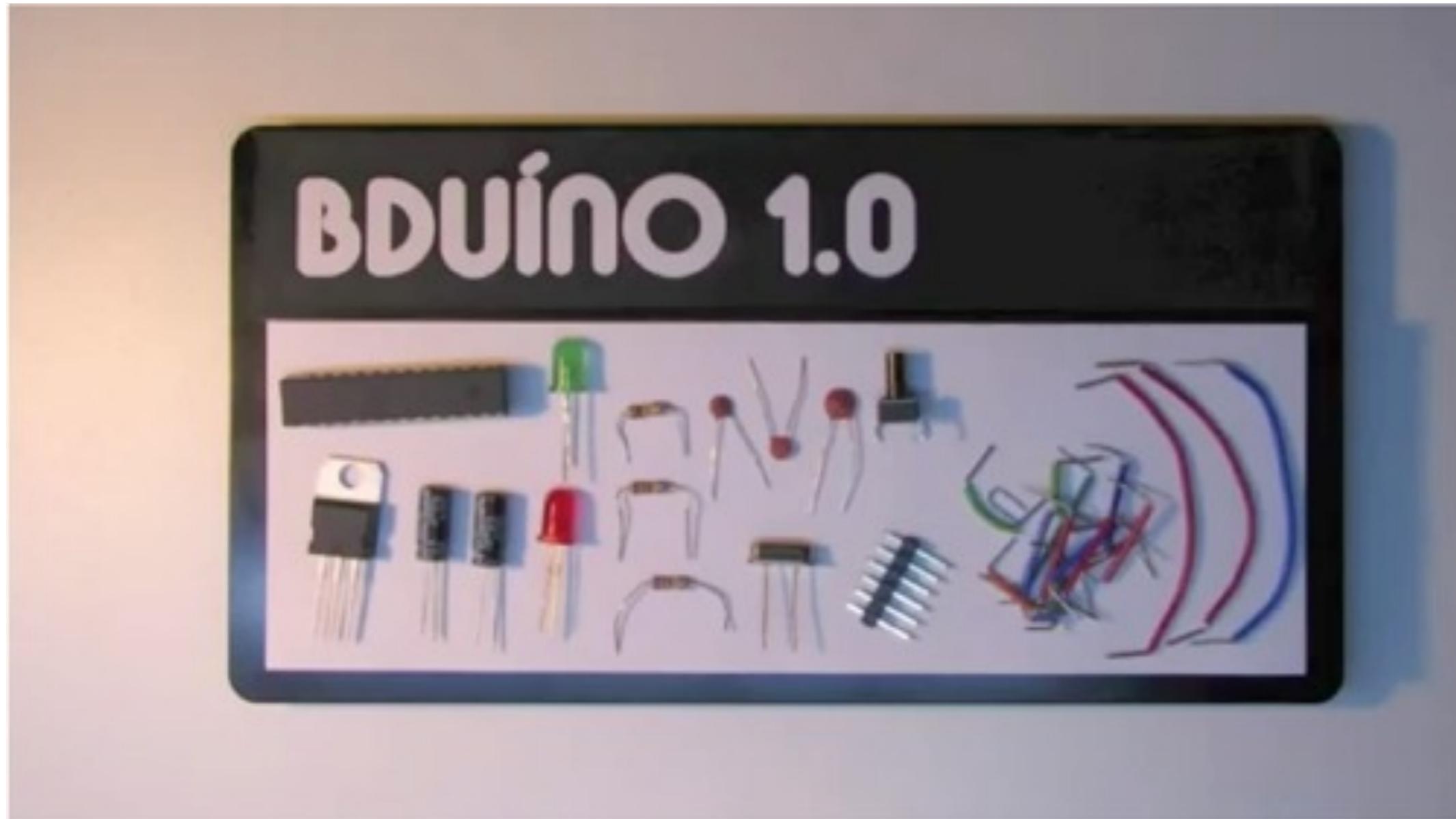
Clones e derivados

- Alguns exemplos:
 - Severino
 - Garagino
 - Sanguino
 - Program-ME

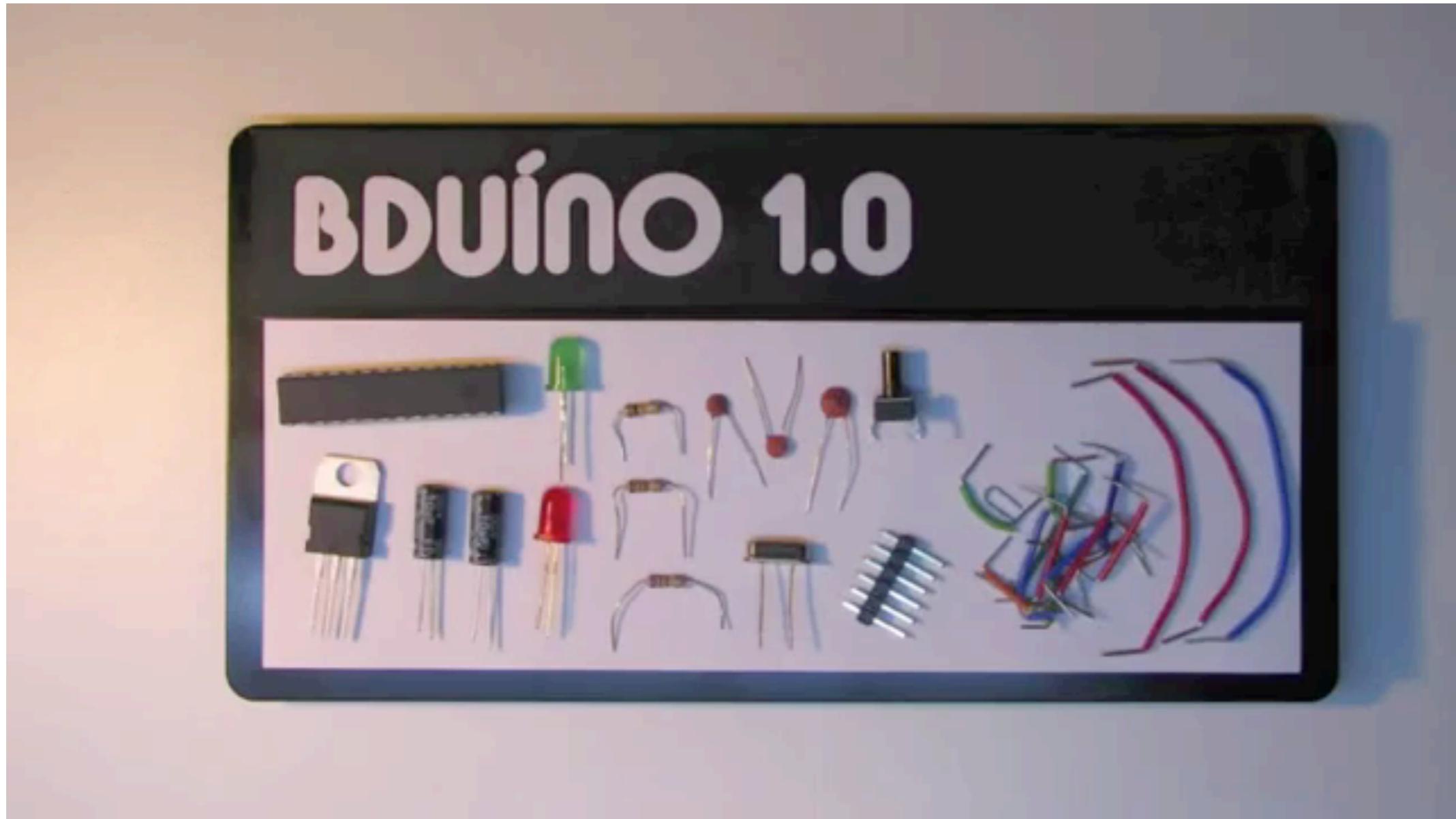


Severino

Clone de Arduino em breadboard



Clone de Arduino em breadboard



Vídeo: <http://www.youtube.com/watch?v=S4nIV99RMtg>

Mais componentes do kit

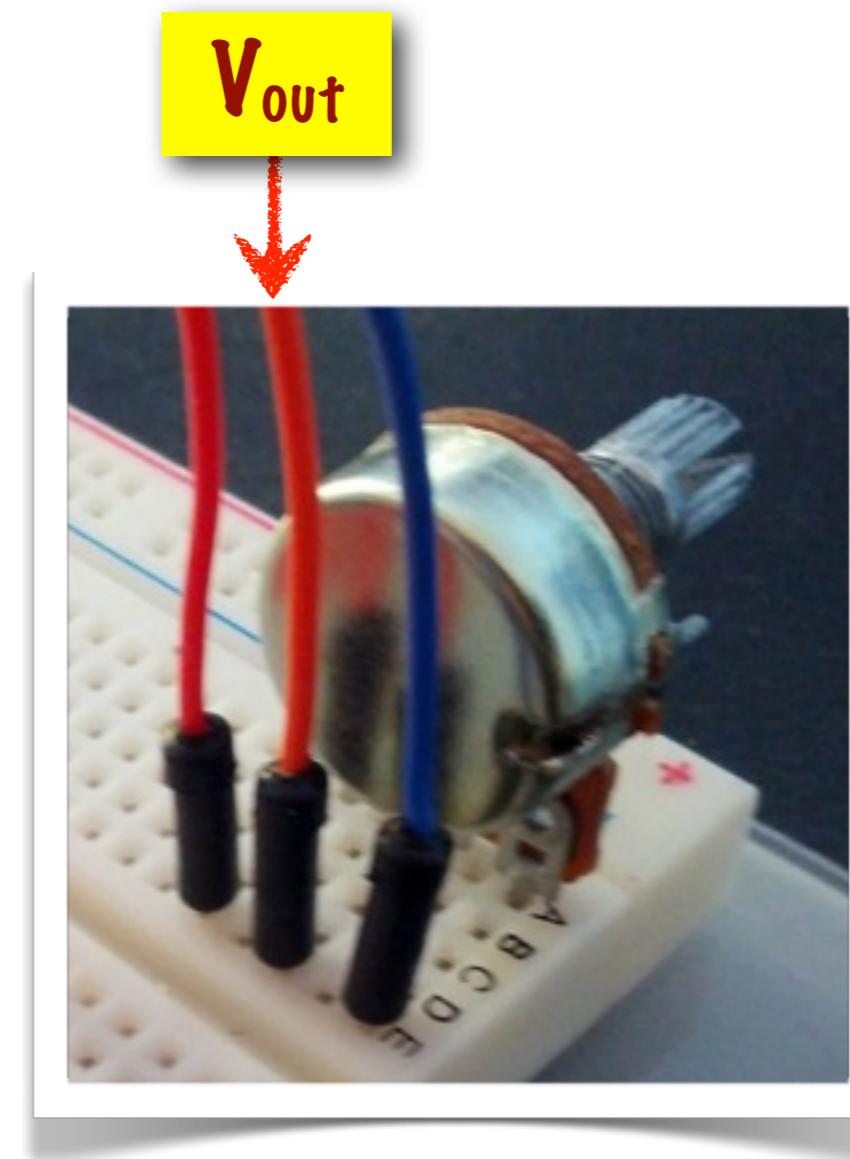
Potenciômetro

- Resistor ajustável
- Kit: $2 \times 100\text{ k}\Omega$
- Símbolo em esquemas:



Potenciômetro: como usar

- Ligar pinos laterais na alimentação
- Ligar pino central V_{out} em um pino de entrada analógico

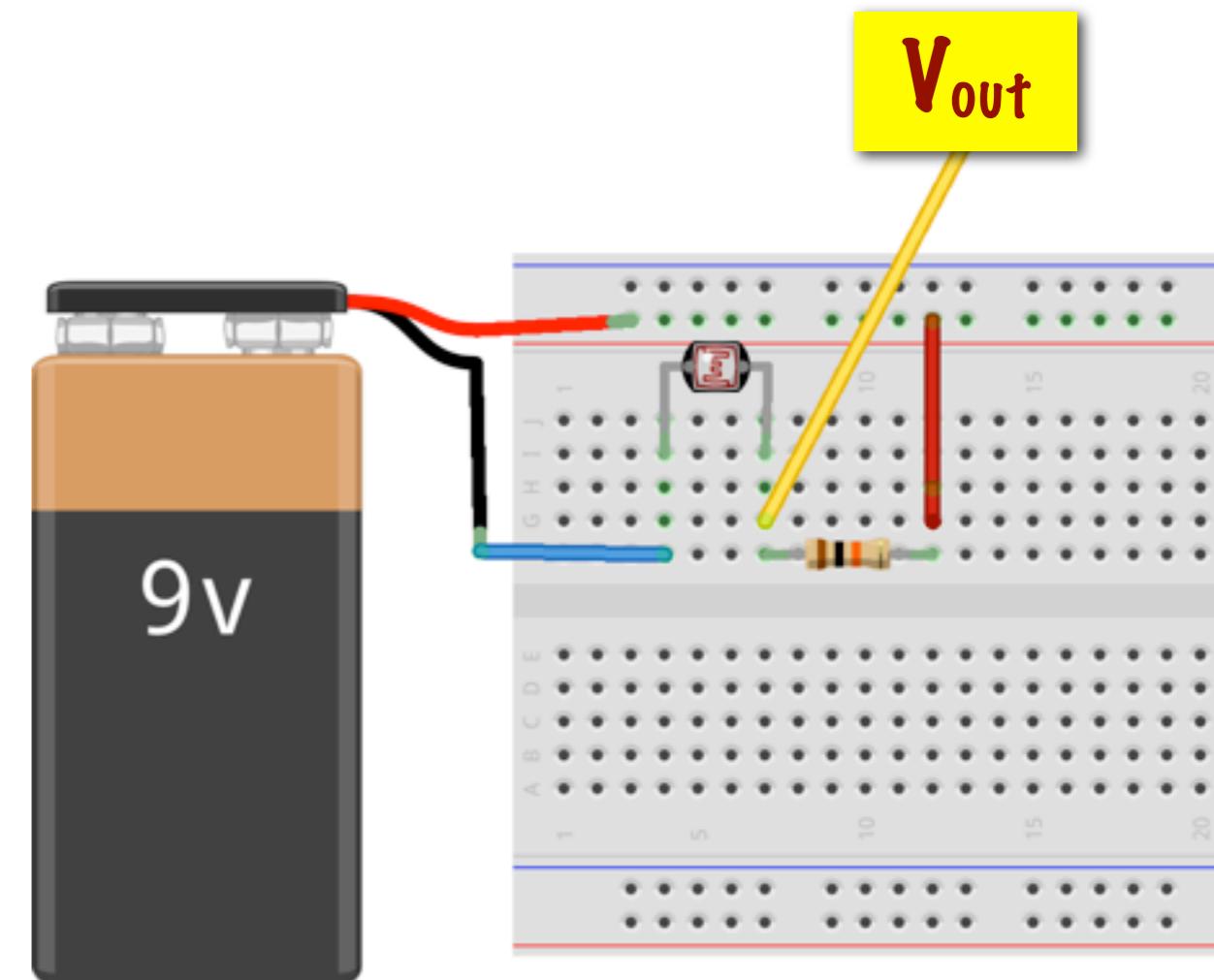
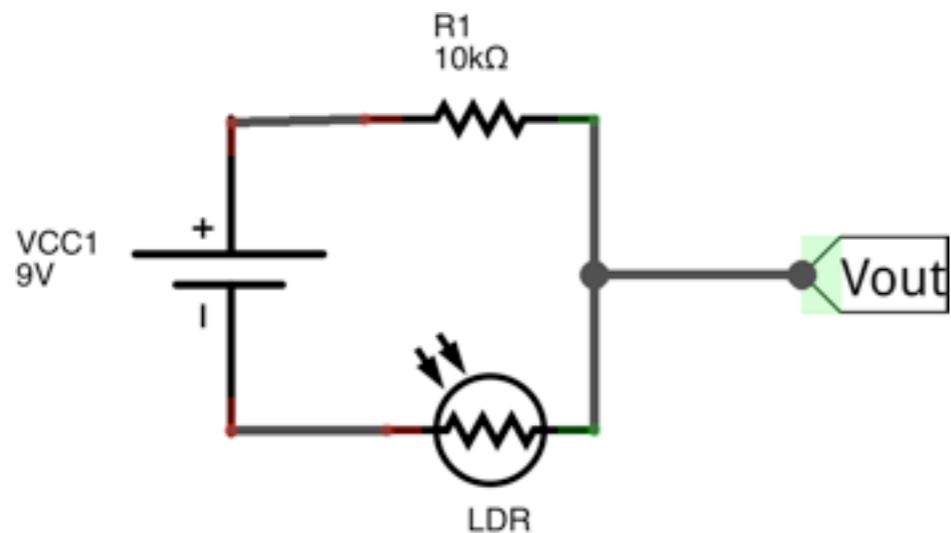


LDR ou fotoresistor

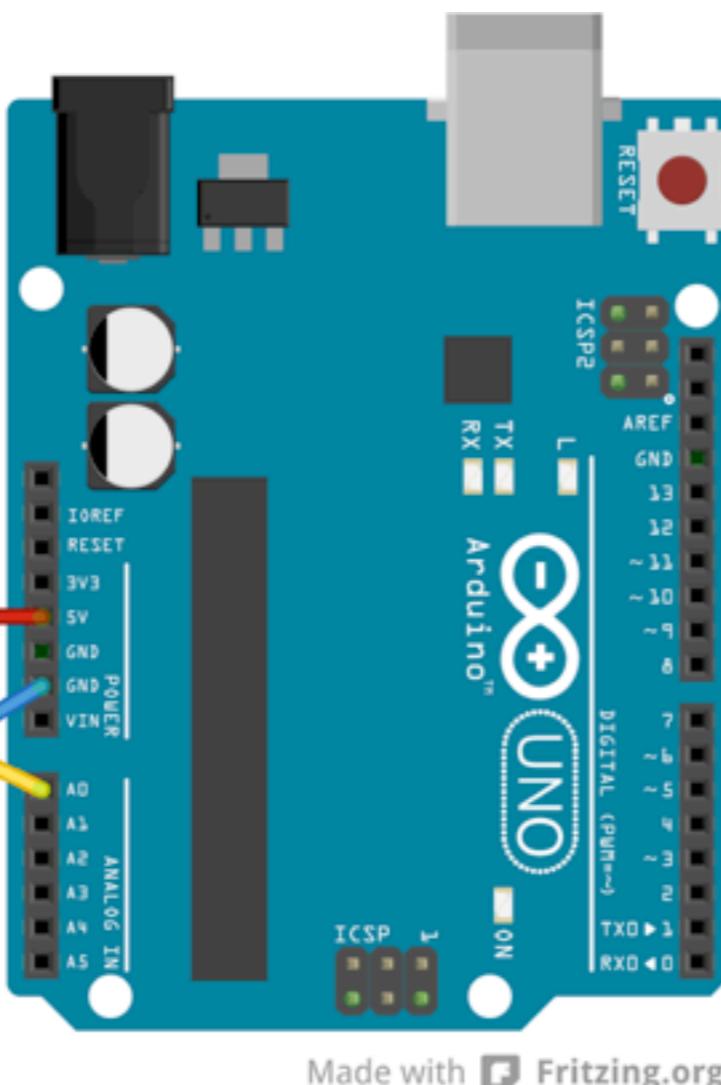
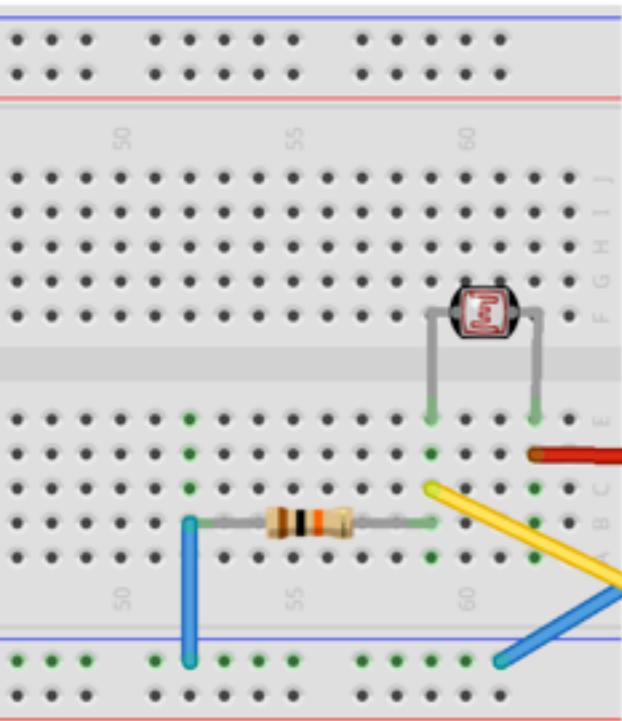
- Light Dependent Resistor
- Kit: 1 × 3mm Ø
- Usar com resistor para fazer um divisor de tensão



Circuito divisor de tensão



Ler LDR no Arduino



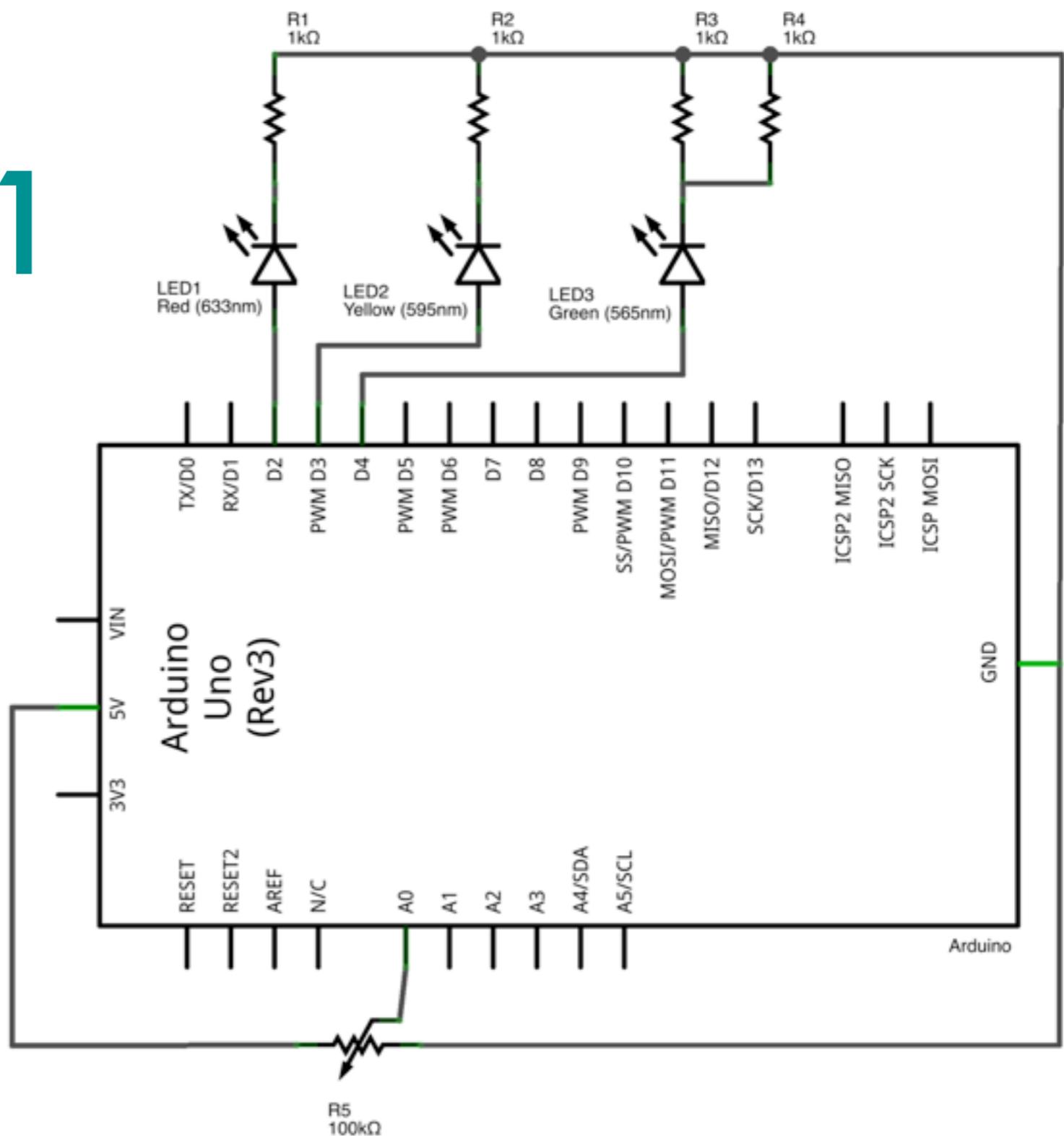
- 5V no LDR
- resistor 10k Ω em série
- entre eles:
 V_{out} ligado a uma entrada analógica (ex. A0)

Circuito Semáforo 1

Circuito semáforo 1

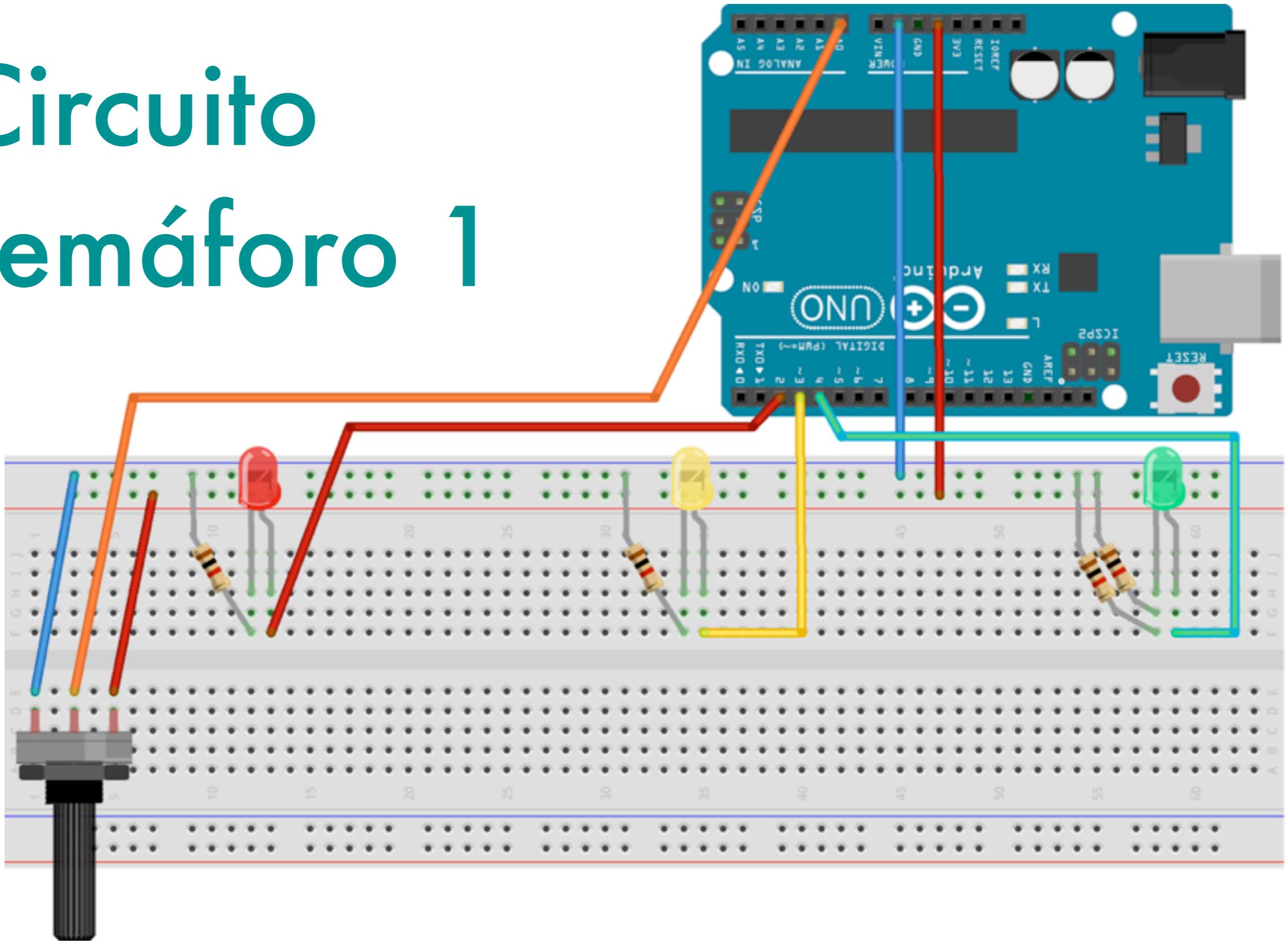
- Componentes:
 - LEDs: verde, amarelo, vermelho
 - 4 resistores de $1\text{k }\Omega$
 - 1 potenciômetro de $100\text{k }\Omega$
- Código:
gist.github.com/ramalho/6202074

Circuito semáforo 1



Made with Fritzing.org

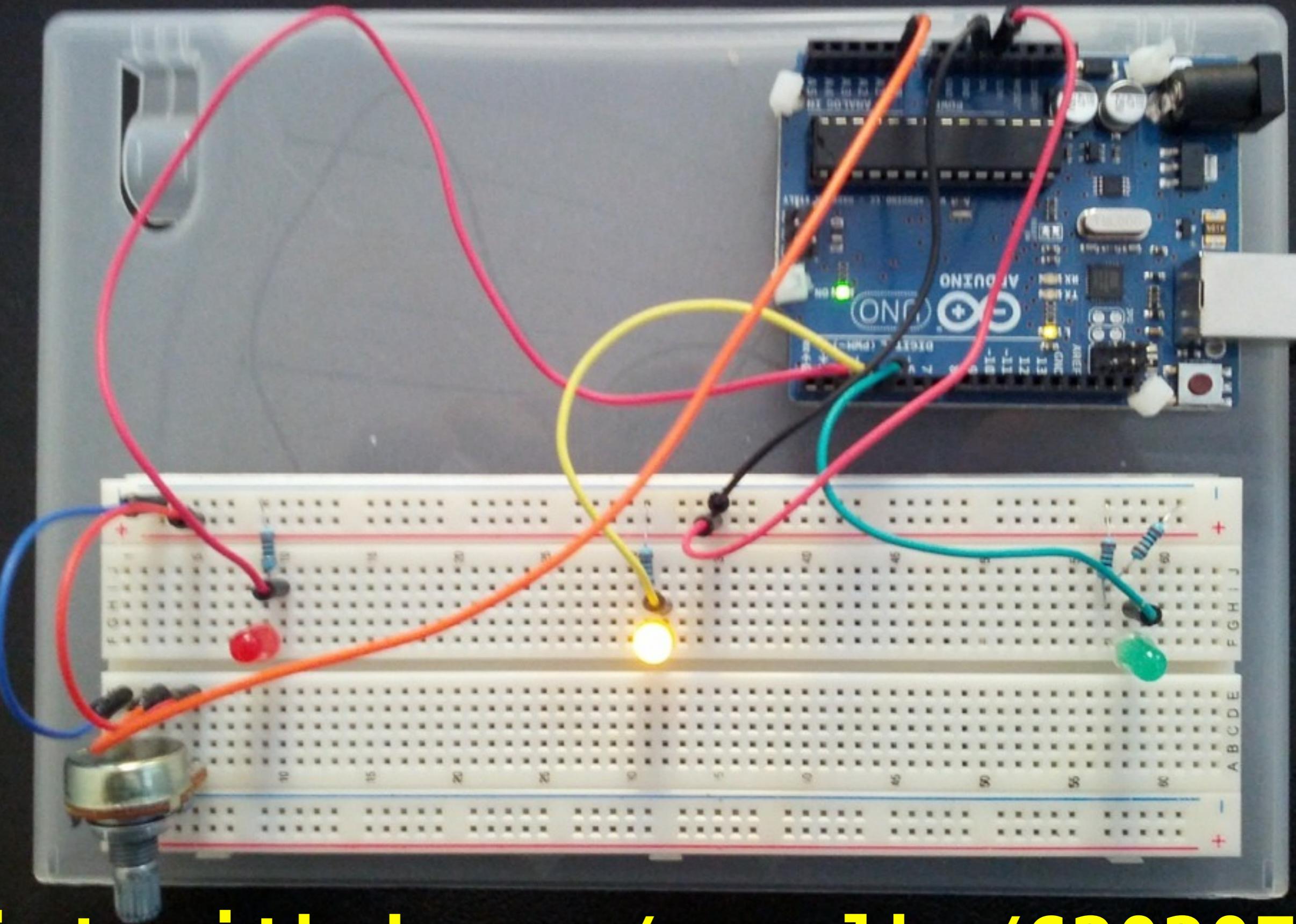
Circuito semáforo 1



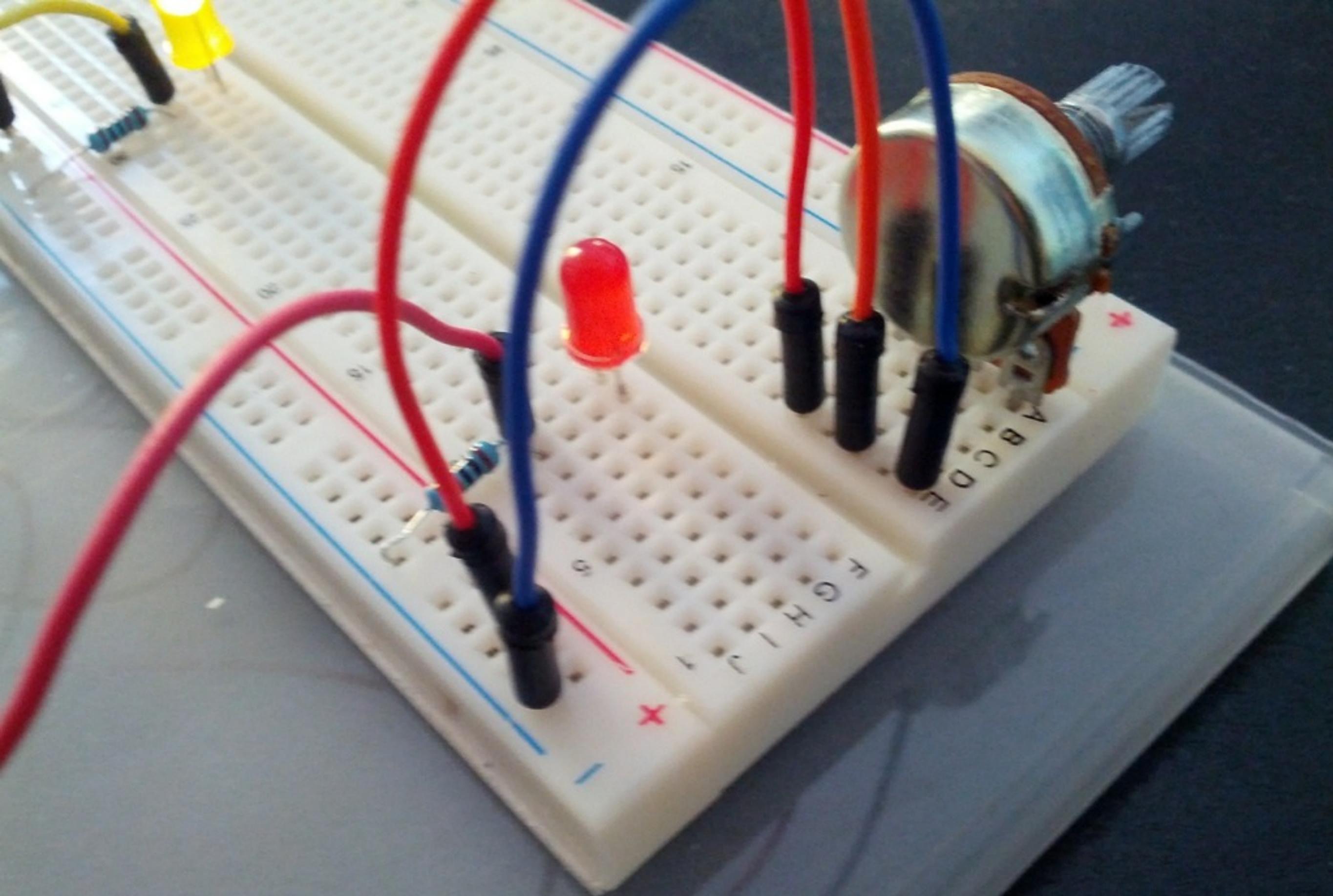
Made with Fritzing.org

código:

gist.github.com/ramalho/6202074



gist.github.com/ramalho/6202074



gist.github.com/ramalho/6202074

Circuito Semáforo 2

Circuito semáforo 2

- Usar um LDR (sensor de luminosidade) em vez do potenciômetro
- Para fazer funcionar, o potenciômetro tem que ser trocado por um divisor de tensão com LDR e resistor de $1k\ \Omega$



LDR

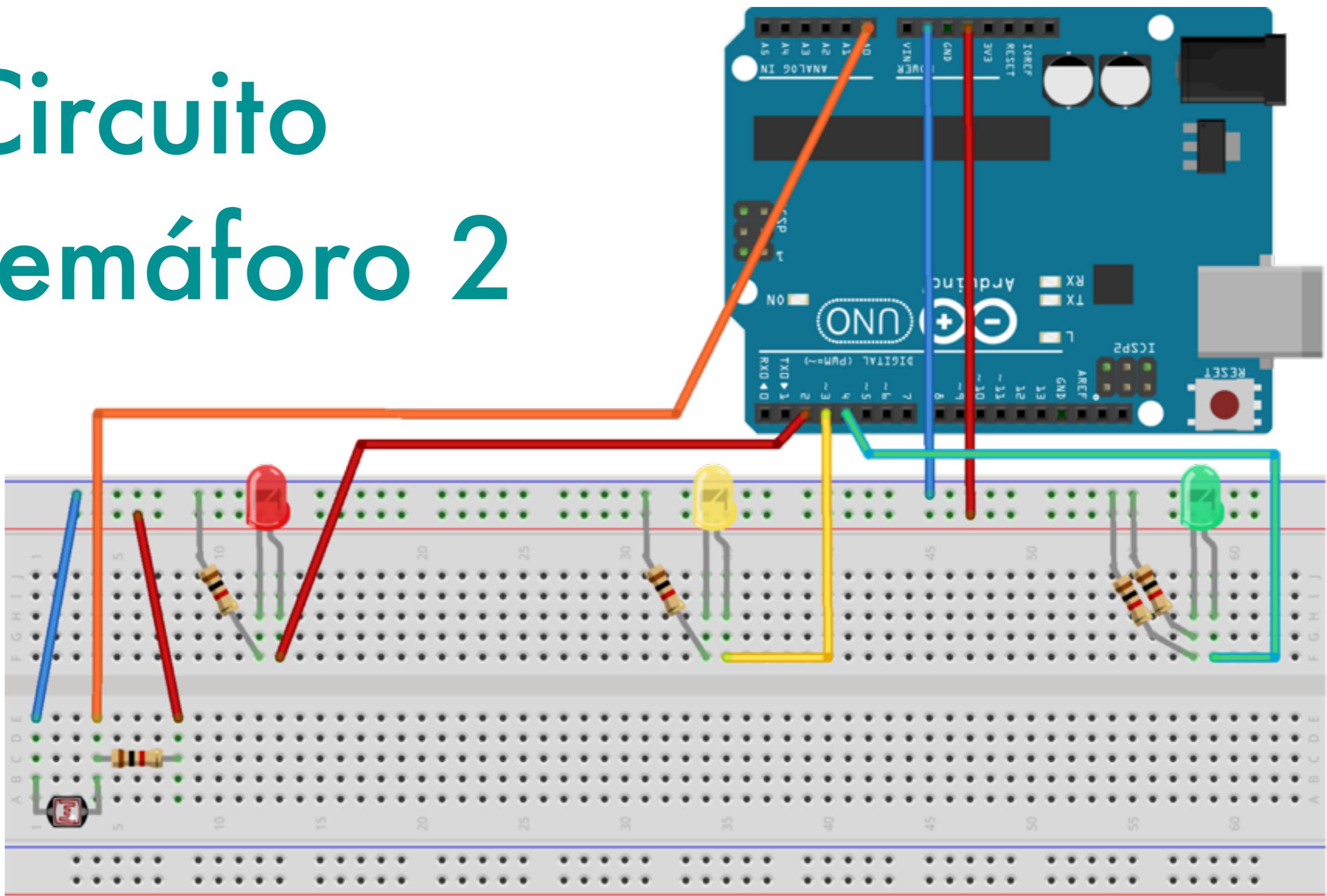
Circuito semáforo 2

- Componentes:
 - LEDs: verde, amarelo, vermelho
 - 4 resistores de $1\text{k }\Omega$
 - ~~1 potenciômetro de $100\text{k }\Omega$~~
 - 1 LDR
 - 1 resistor de $1\text{k }\Omega$



LDR

Circuito semáforo 2



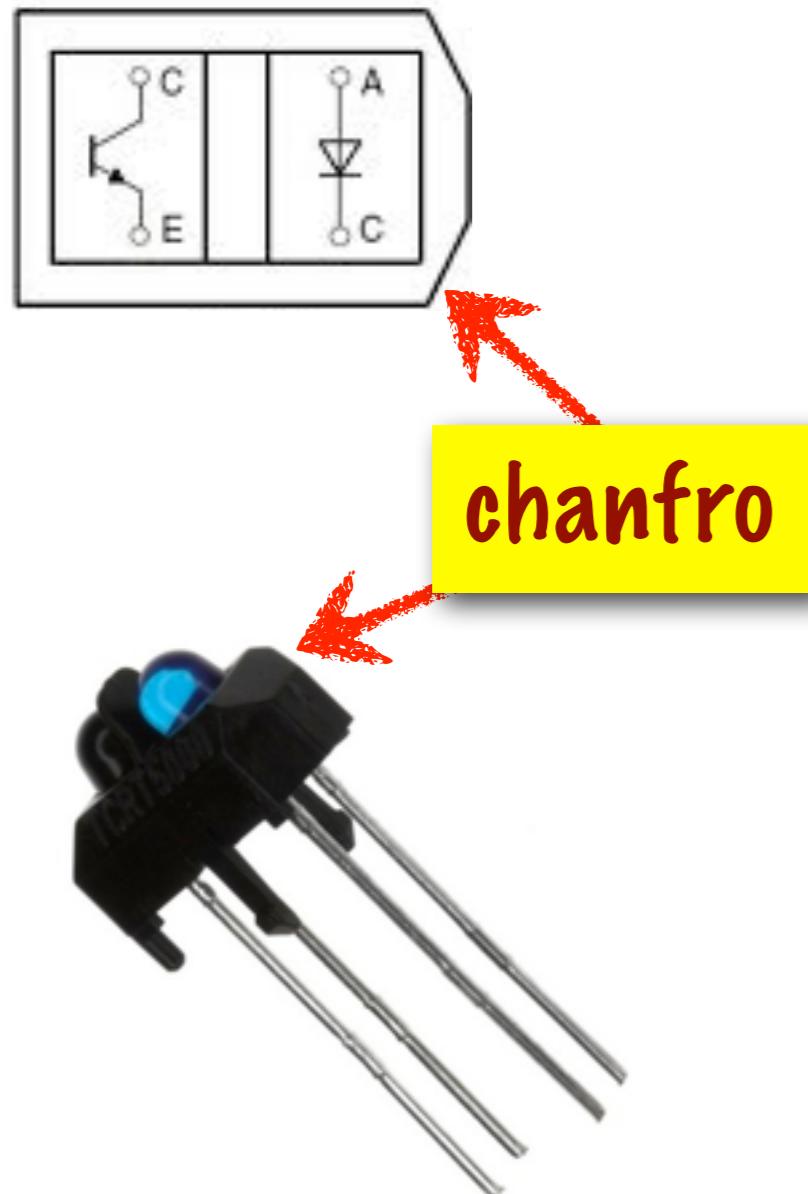
Made with Fritzing.org

código:

gist.github.com/ramalho/6202074

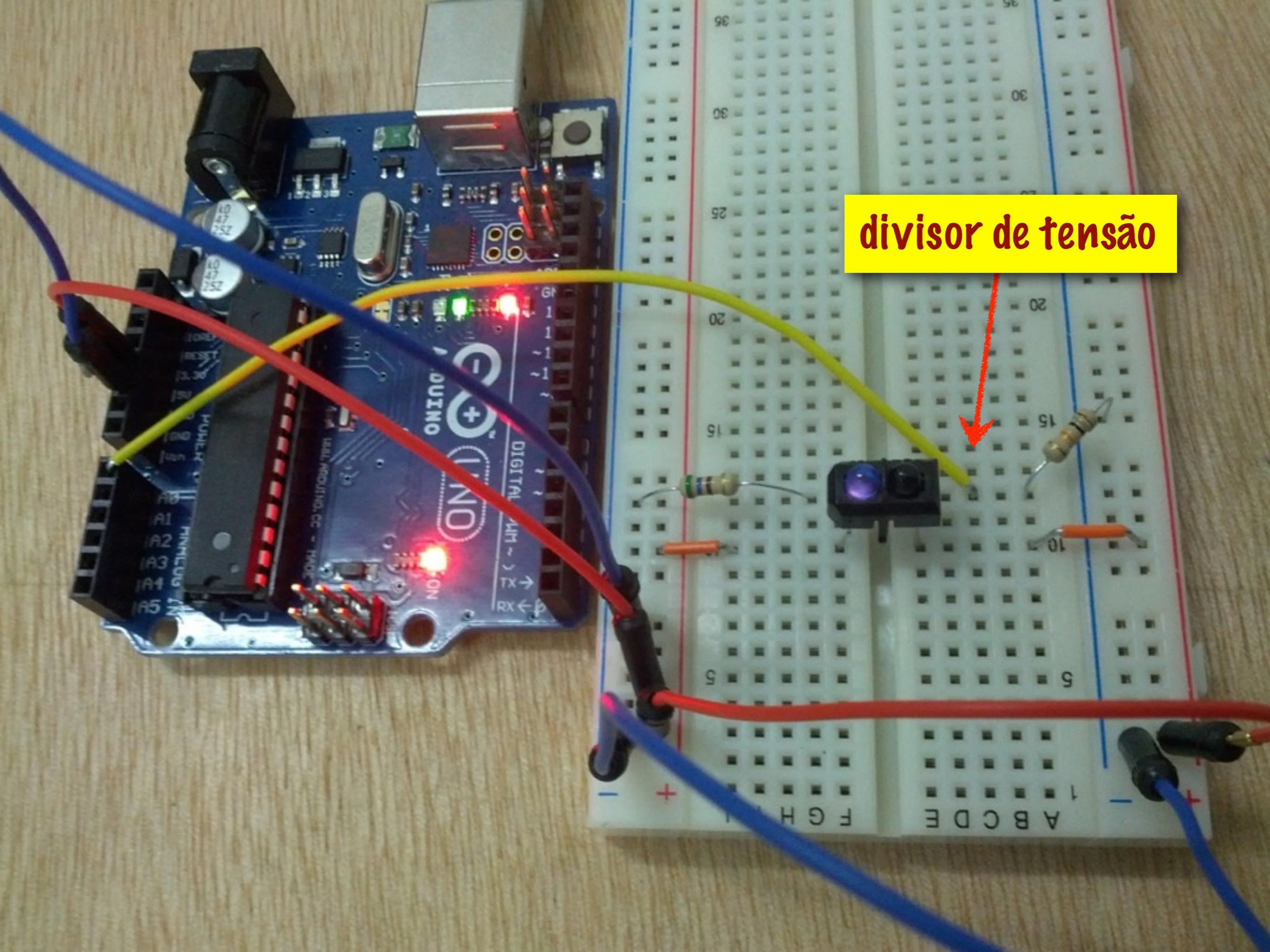
Ainda mais componentes do kit

Sensor óptico reflexivo



- Detecta objetos até 25mm de distância
- Emissor: LED infravermelho
- Receptor: fototransistor protegido contra luz visível
 - Use divisor de tensão para ler

divisor de tensão



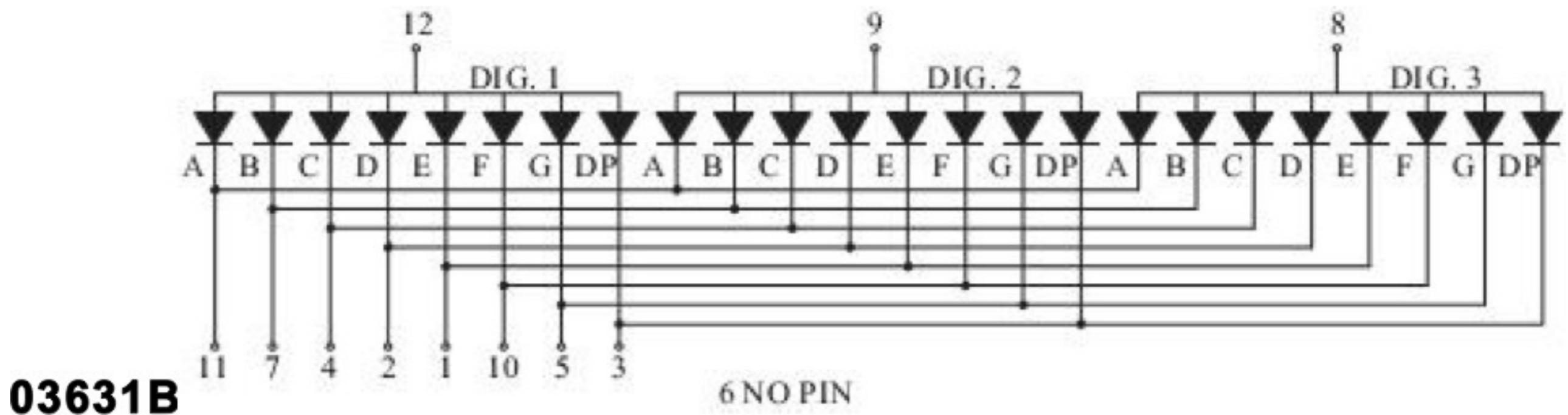
Display 7 segmentos

- 3 dígitos
- modelo CPS3631BR



foto:
CPS3631AR

Display 7 segmentos



Esquema no datasheet CPS3631BR

Display 7 segmentos

- anodos: pinos 8, 9, 12
selecionam o dígito
- demais pinos acionam
os segmentos A...G e o
ponto decimal (dp)



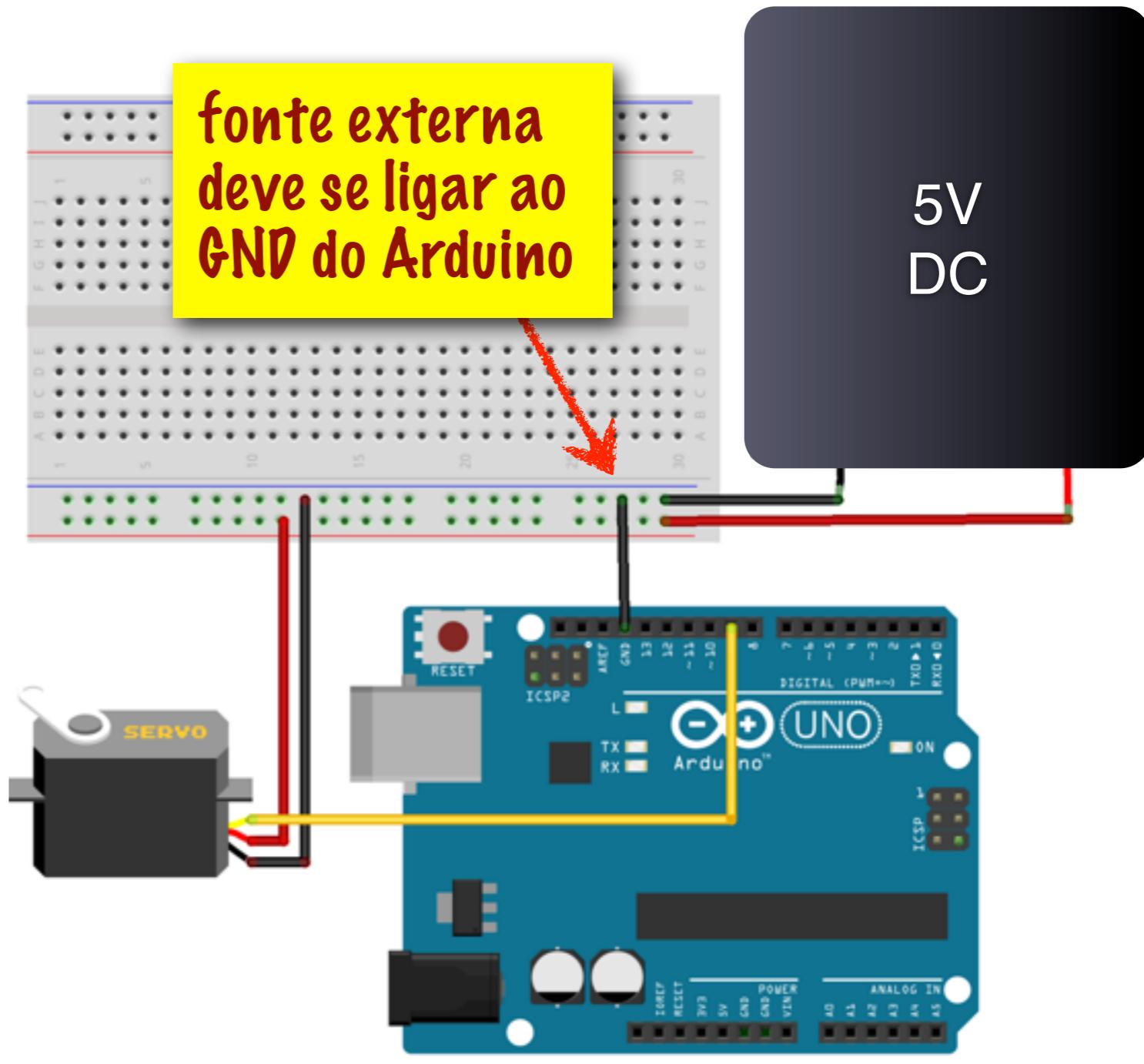
Servomotor

- Movimentos controlados num arco de 180°
- Programação fácil via biblioteca **Servo** no Arduino
 - exemplos incluídos na IDE



Servomotor

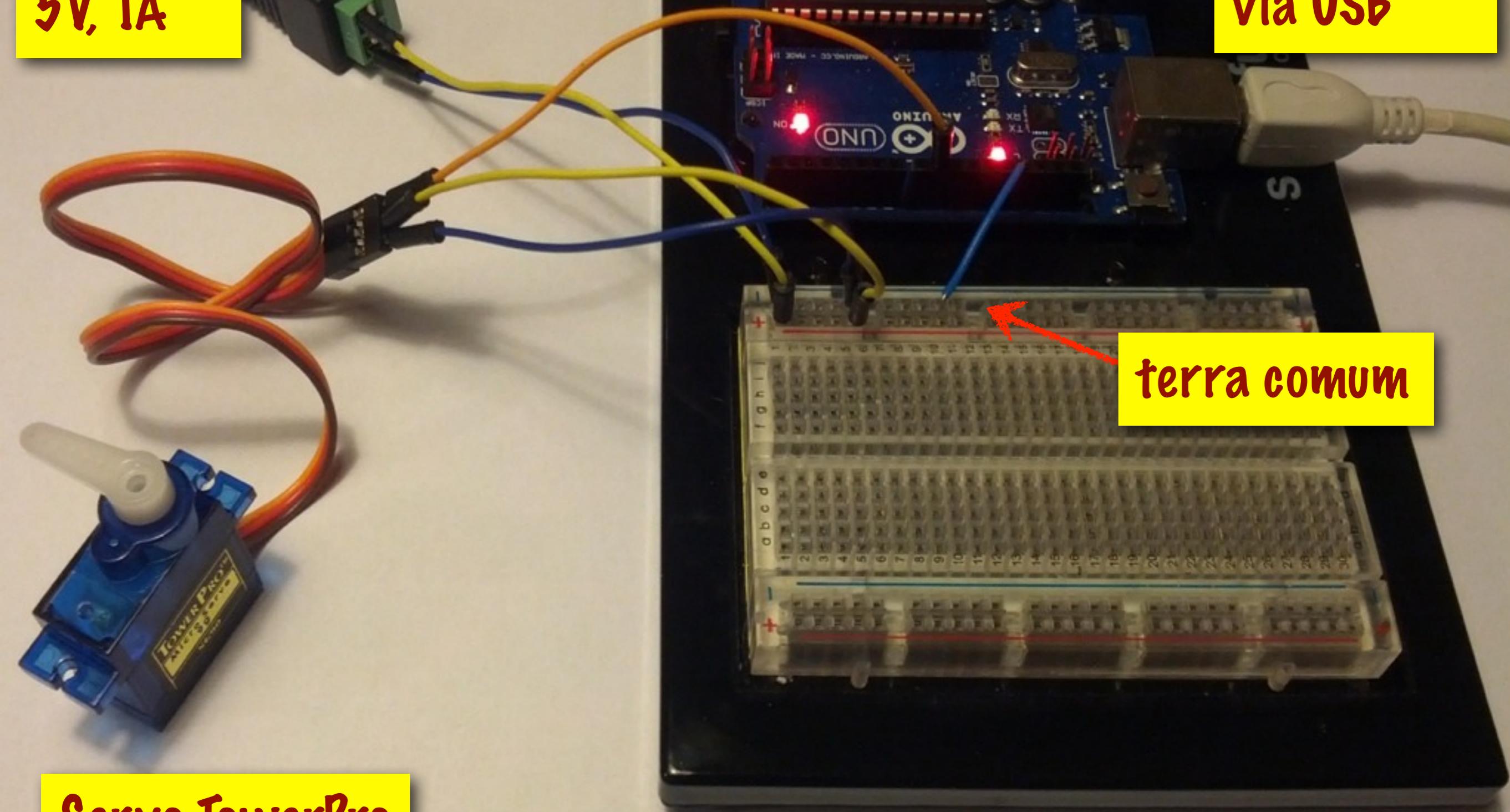
- Alguns servos só funcionam alimentados por fonte externa



Made with Fritzing.org

fonte DC
externa
5V, 1A

Arduino
alimentado
via USB



Servo TowerPro
SG90 (9g)

Fechamento

Referências

- Site oficial:
arduino.cc
- Arduino Experimentation Kit:
oomlout.com/a/products/ardx
- Laboratório de garagem
labdegaragem.com
- Web: blogs, vídeos, wikis, diagramas...

arduino.cc

Arduino - HomePage

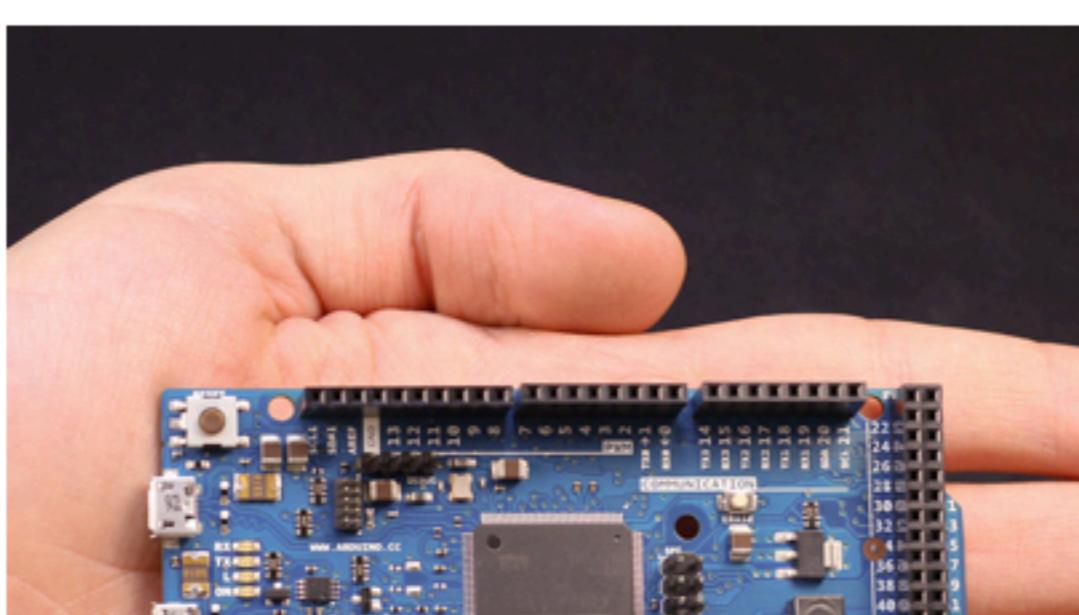
Arduino - HomePage +

arduino.cc/en/

Main Site Blog Playground Forum Labs Store Sign in or Register

ARDUINO

Buy Download Getting Started Learning Reference Products FAQ Contact Us

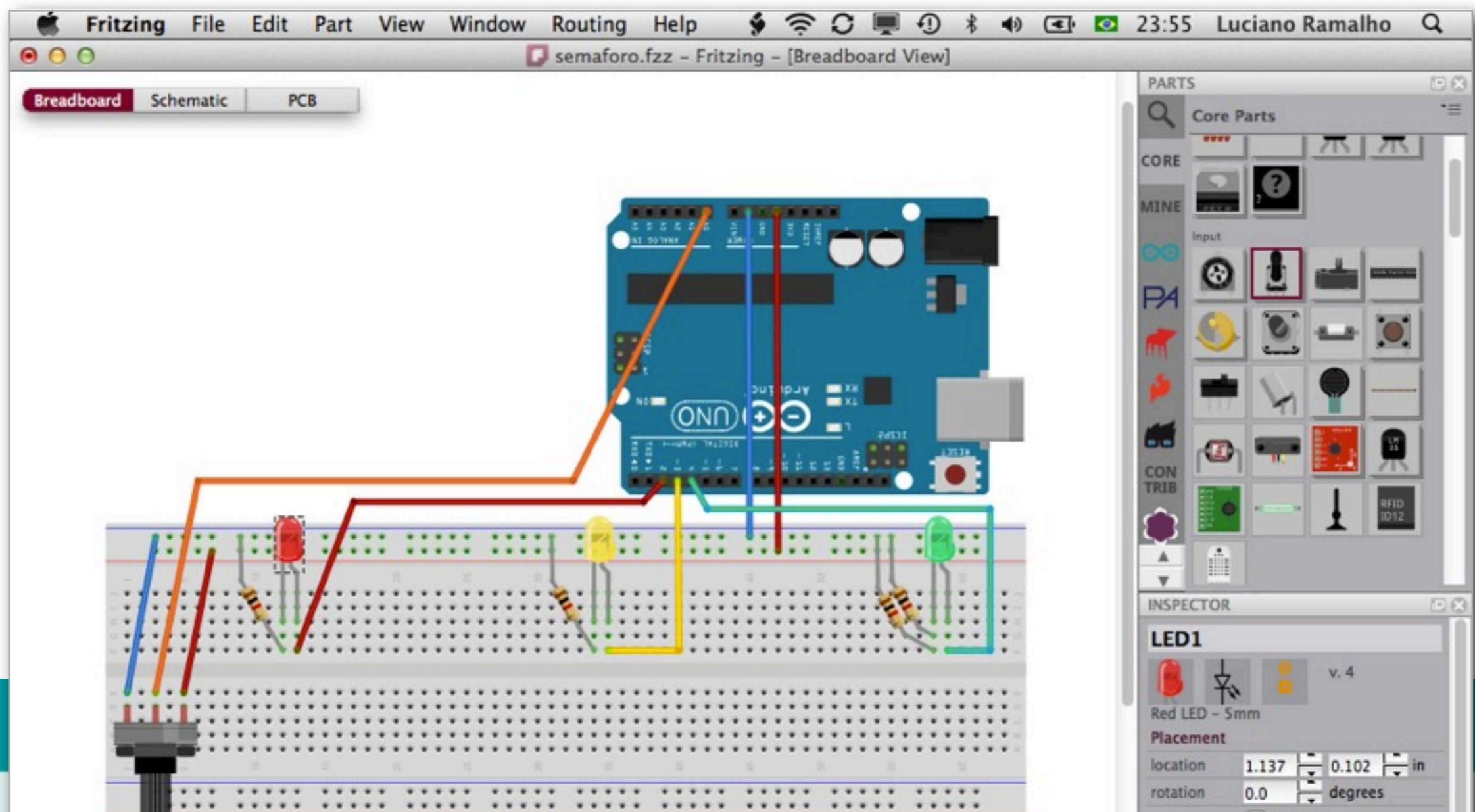


Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists and anyone interested in creating interactive objects or environments.

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and

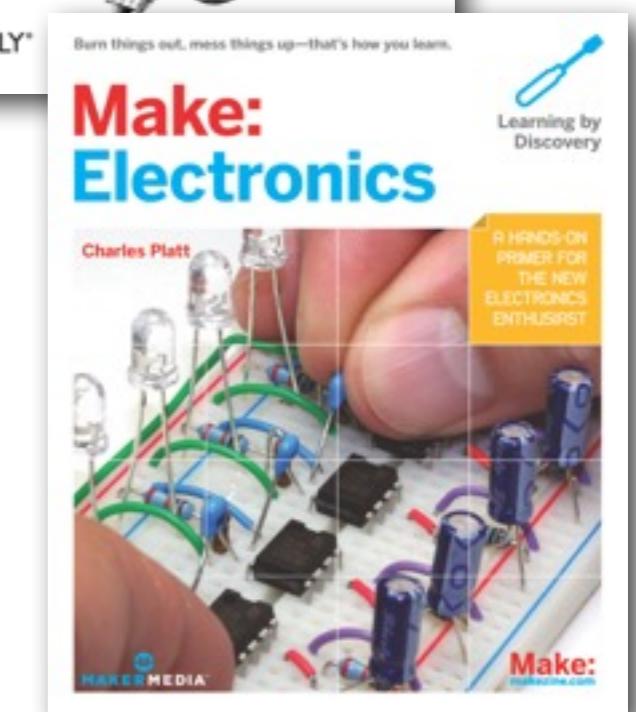
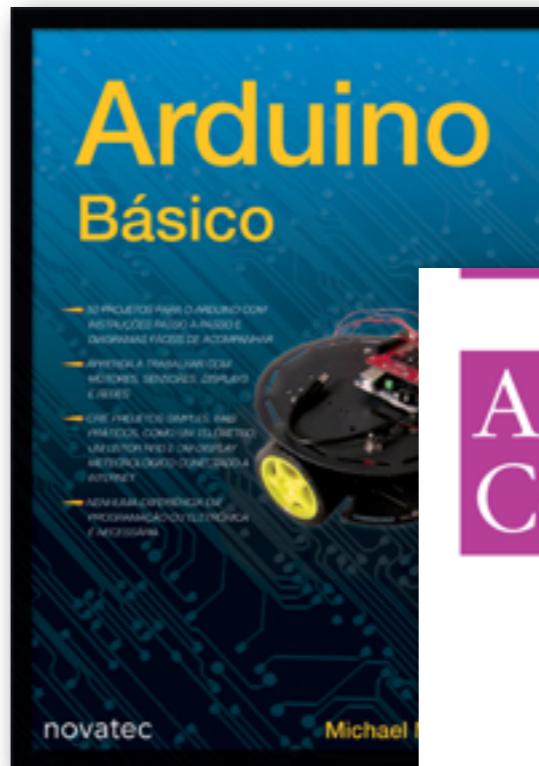
Fritzing.org

- Site com software para download e repositório de projetos



Alguns Livros

- Arduino Básico, Novatec
- Arduino Cookbook, O'Reilly
- Make: Electronics, O'Reilly



Lojas físicas em Sampa

- Multcomercial, R. dos Timbiras, 257
- Lojas nas travessas da Santa Efigênia, inclusive as lojas de sucata
- Laboratório de Garagem, Rua Berta, 60 (metrô Vila Mariana)
- Lojas de bairro, ex: Rua Butantã, 133

Lojas online

- Laboratório de Garagem: labdegaragem.com
- Farnell Newark: farnellnewark.com.br
- Adafruit: adafruit.com
- Sparkfun: sparkfun.com
- Seeedstudio: seeedstudio.com
- Deal extreme: dx.com (muitos fornecedores)

**seeedstudio
com 3 "e"!**

Seeedstudio Shield Bot

- Robô seguidor de linha
- Vem montado
- Basta encaixar o Arduino Uno



US \$ 69.90

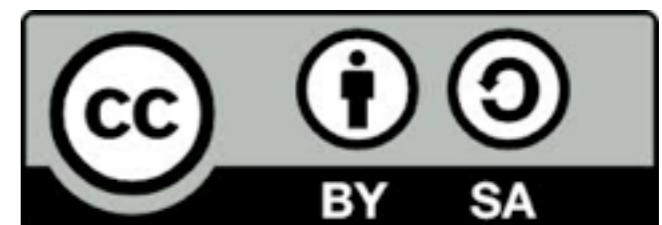
Visite um hackerspace!

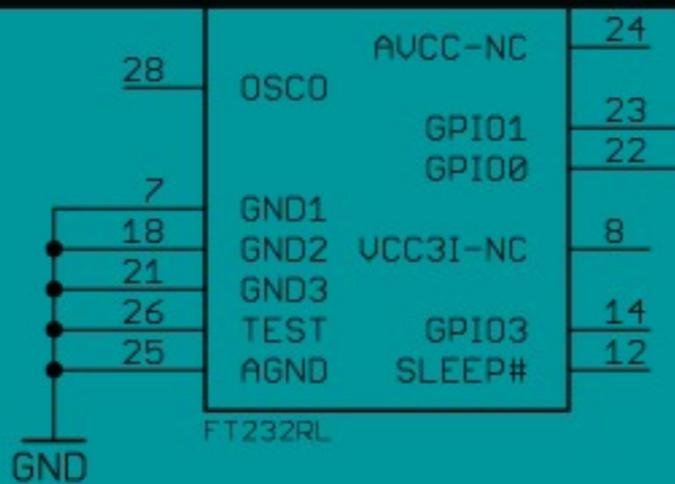


garoa.net.br

Créditos

- Conteúdo compilado, organizado e parcialmente criado por Luciano Ramalho (Oficinas Turing)
- Licença de uso e reprodução: Creative Commons BY-SA





WORKSHOPS