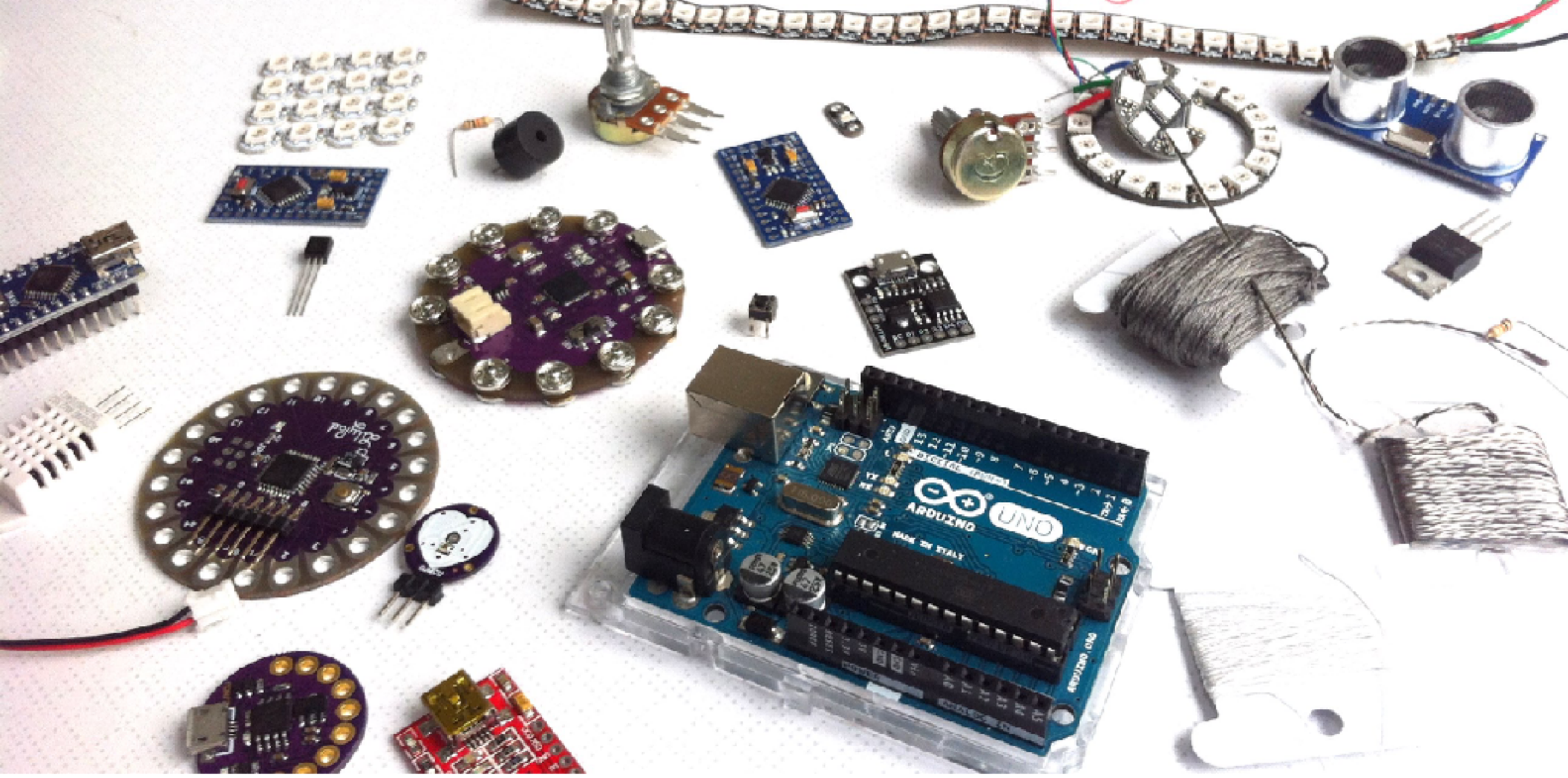
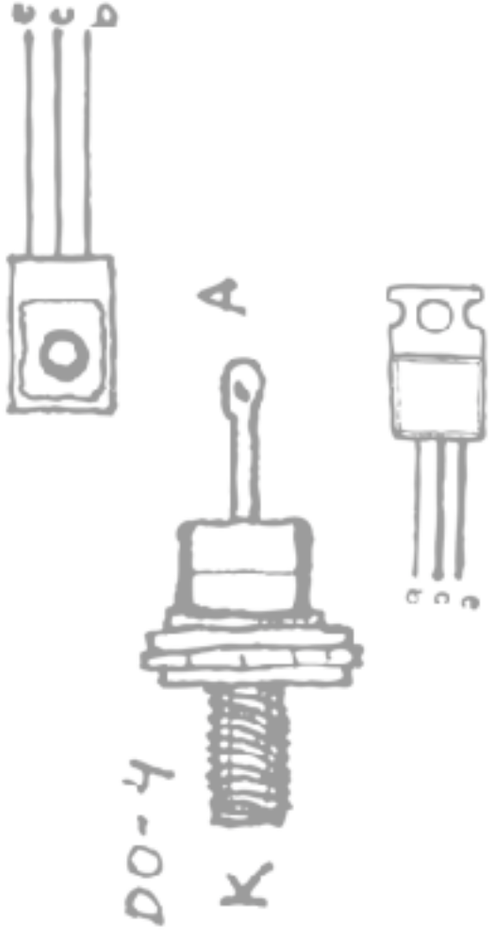


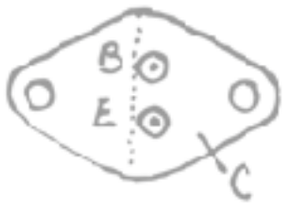
Introdução ao **ARDUINO**

Helder da Rocha
www.eletronicaparaartistas.com.br

argonavis



ARDUINO 1



Introdução
Instalação e configuração

argonavis

Helder da Rocha
www.eletronicaparaartistas.com.br

O que é Arduino?

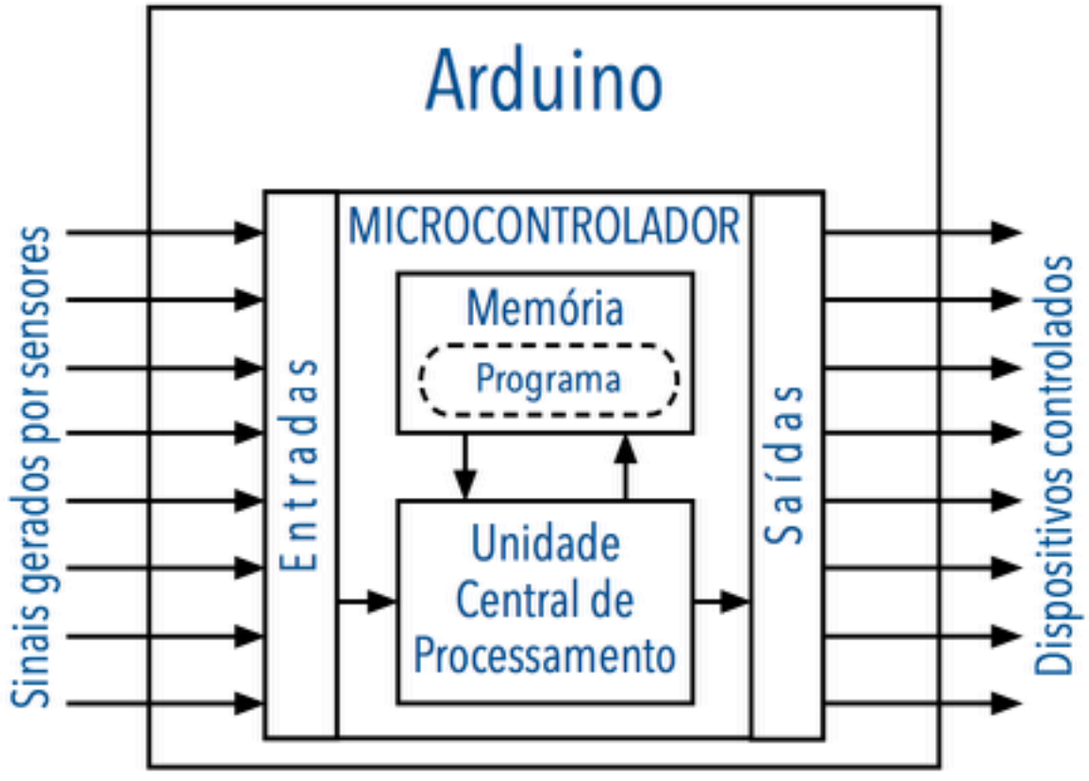
- Uma **placa** para prototipagem de projetos eletrônicos
- Uma plataforma eletrônica que consiste de um **circuito** contendo um **microcontrolador** (da família AVR) configurado para facilitar a **programação** e controle de **entradas e saídas**.
- Criado para **facilitar** o uso da eletrônica por artistas
- **Open hardware** - circuito pode ser montado e vendido sem precisar pagar royalties

Arquitetura do Arduino

Entradas
(sensores, chaves, geradores de sinais)

Processamento
(plataforma de computação baseada em microcontrolador)

Saídas
(dispositivos, mecanismos, processadores de sinais)

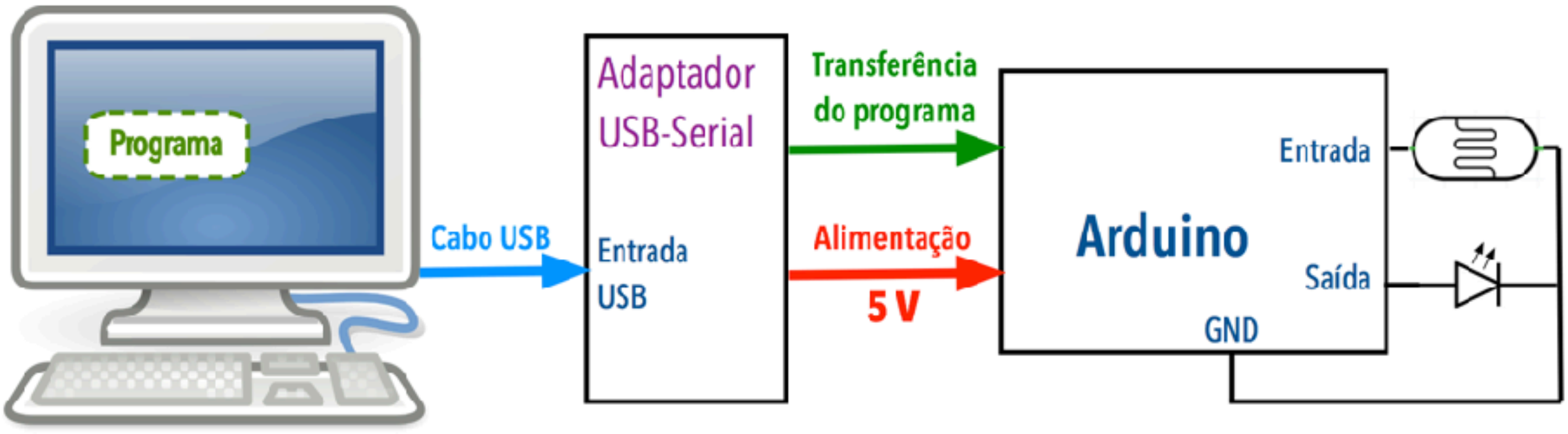


Programação do Arduino

A programação do Arduino é feita em uma **linguagem de alto-nível** (é uma versão simplificada da linguagem C)

A linguagem define abstrações (estruturas, variáveis e funções) que representam pinos e funcionalidades do Arduino.

A função dos pinos do Arduino (se irão funcionar como entradas ou saídas) é definida através de programação

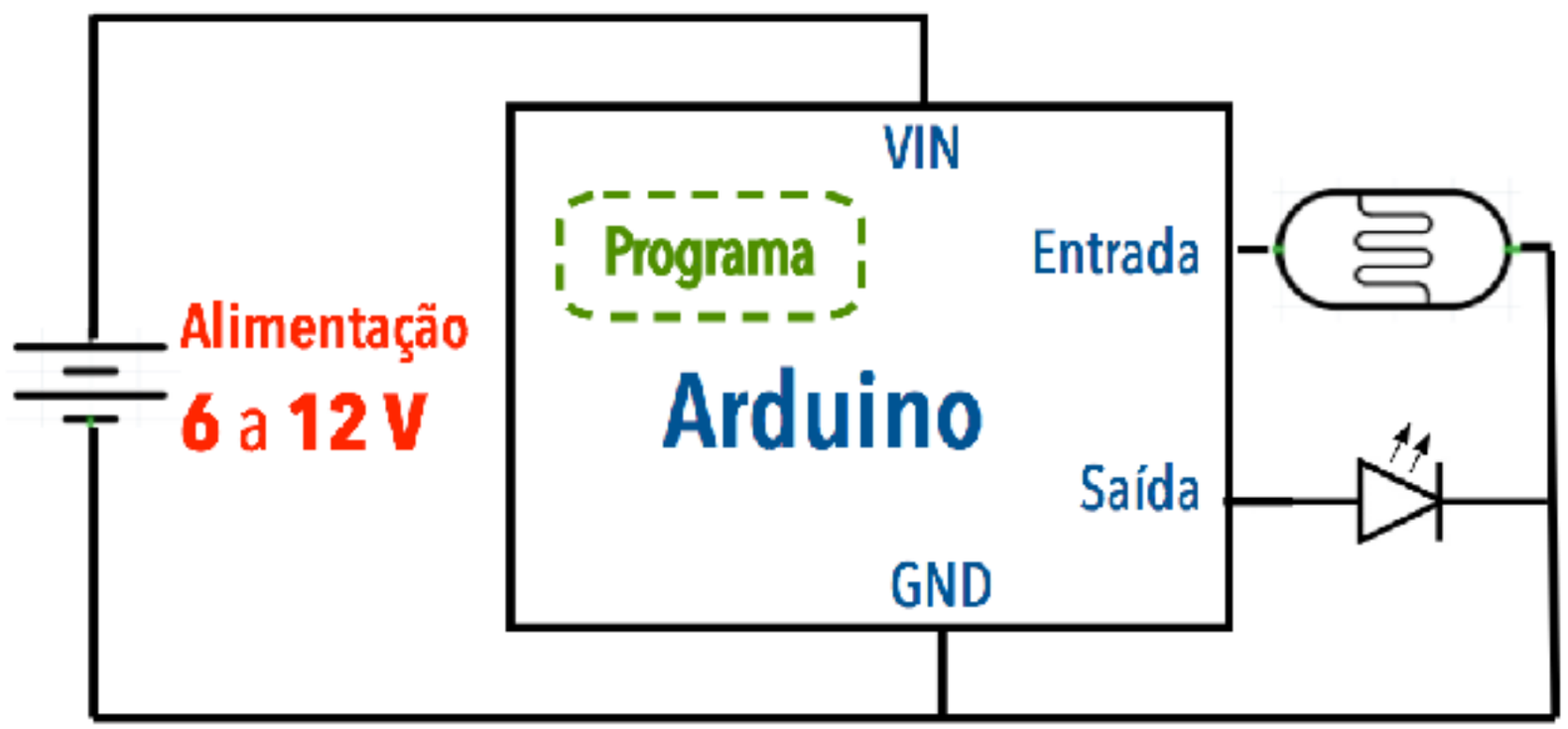


Antes de ser transferido para a memória do Arduino, o programa precisa ser **compilado** (traduzido para a linguagem de máquina do Arduino)

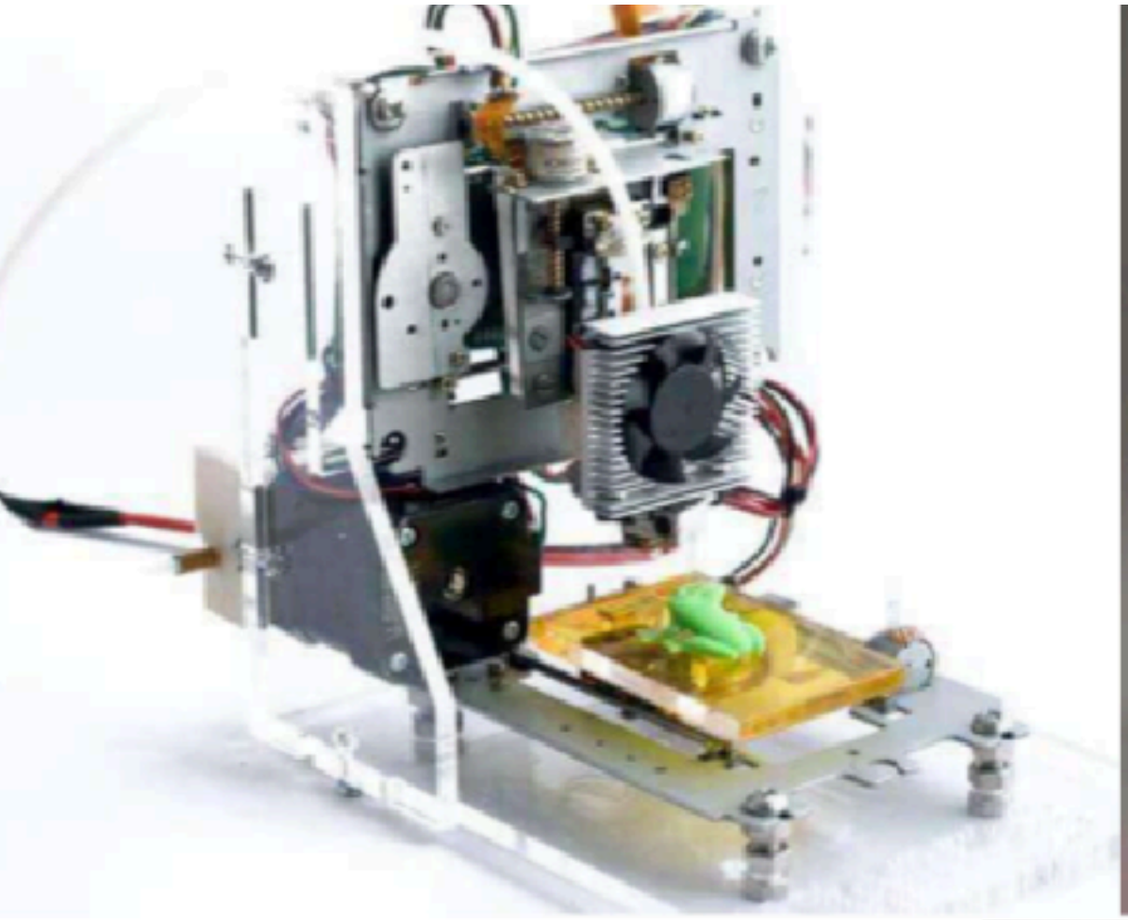
A transferência é feita através da saída USB do computador, que também fornece energia para a placa. Algumas placas possuem entrada USB, outras requerem um adaptador.

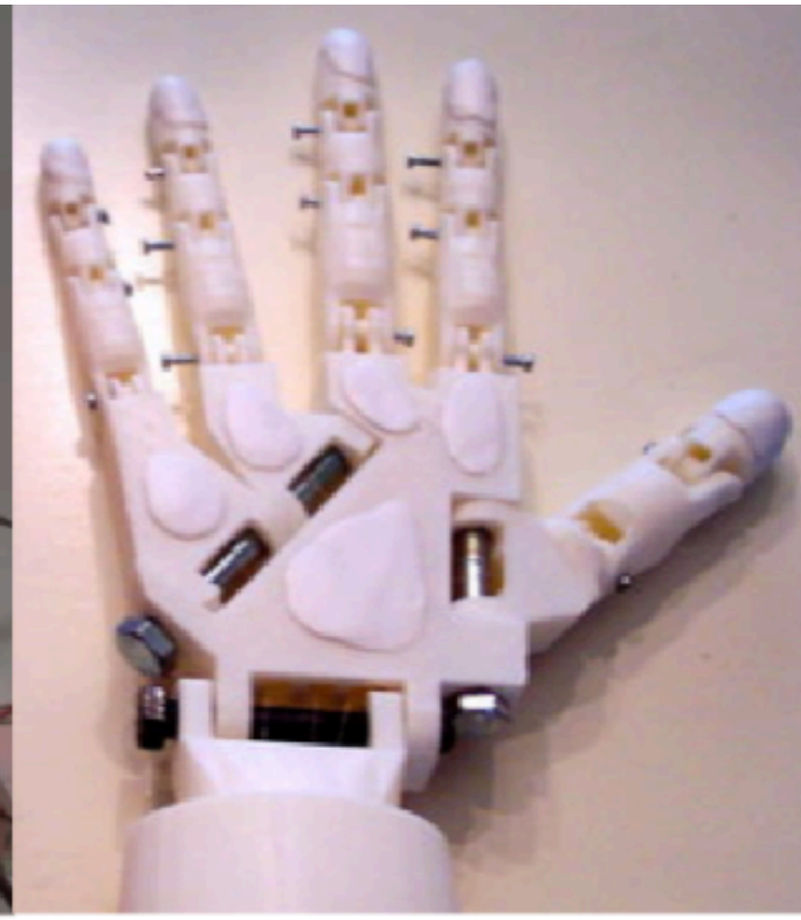
Uso do Arduino

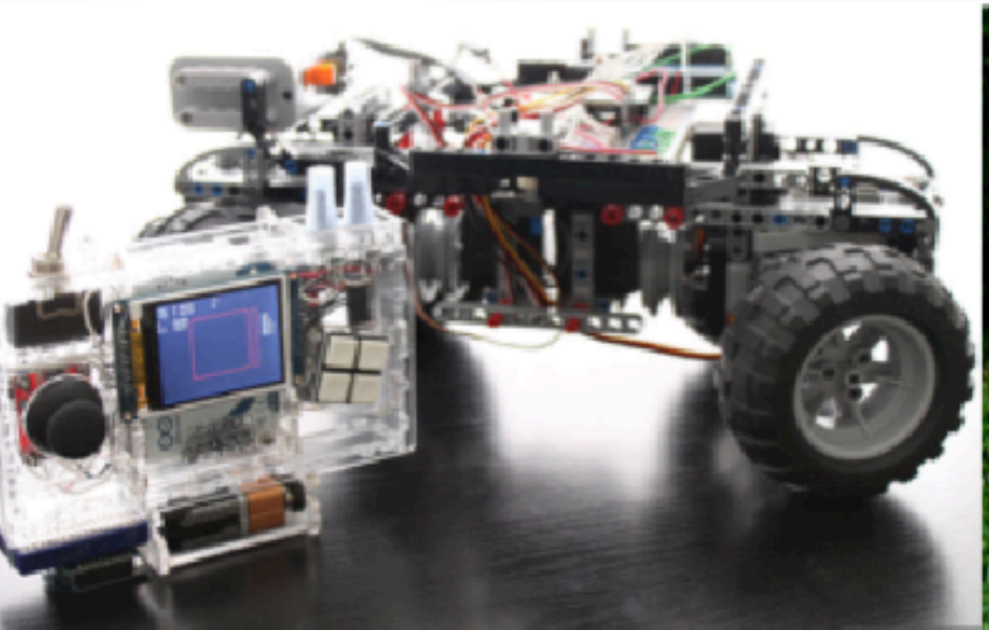
- Depois que um programa é transferido para a **memória** do Arduino, ele executará **automaticamente** sempre que for conectado a uma fonte de energia elétrica
- Um programa geralmente é construído para executar continuamente, lendo os sinais recebidos de sensores e controlando dispositivos
- O botão **RESET** reinicia o Arduino (o programa é interrompido e reiniciado novamente)



Alguns projetos com Arduino



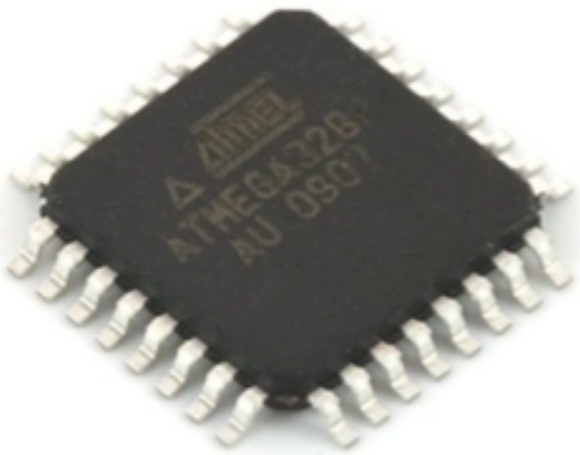




Microcontroladores

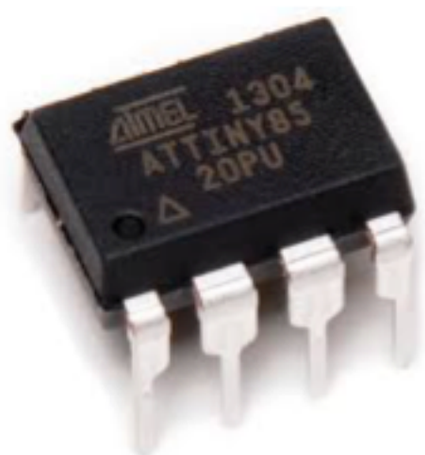
- Um **microcontrolador** é um pequeno computador contido em um chip
- O Arduino é um circuito que existe para dar suporte ao microcontrolador (que é quem faz todo o trabalho)
- Arduinos usam microcontroladores da arquitetura **AVR** fabricados pela ATMel. Os mais usados são da série **ATMega** e **ATTiny**

ATMega328 SMD

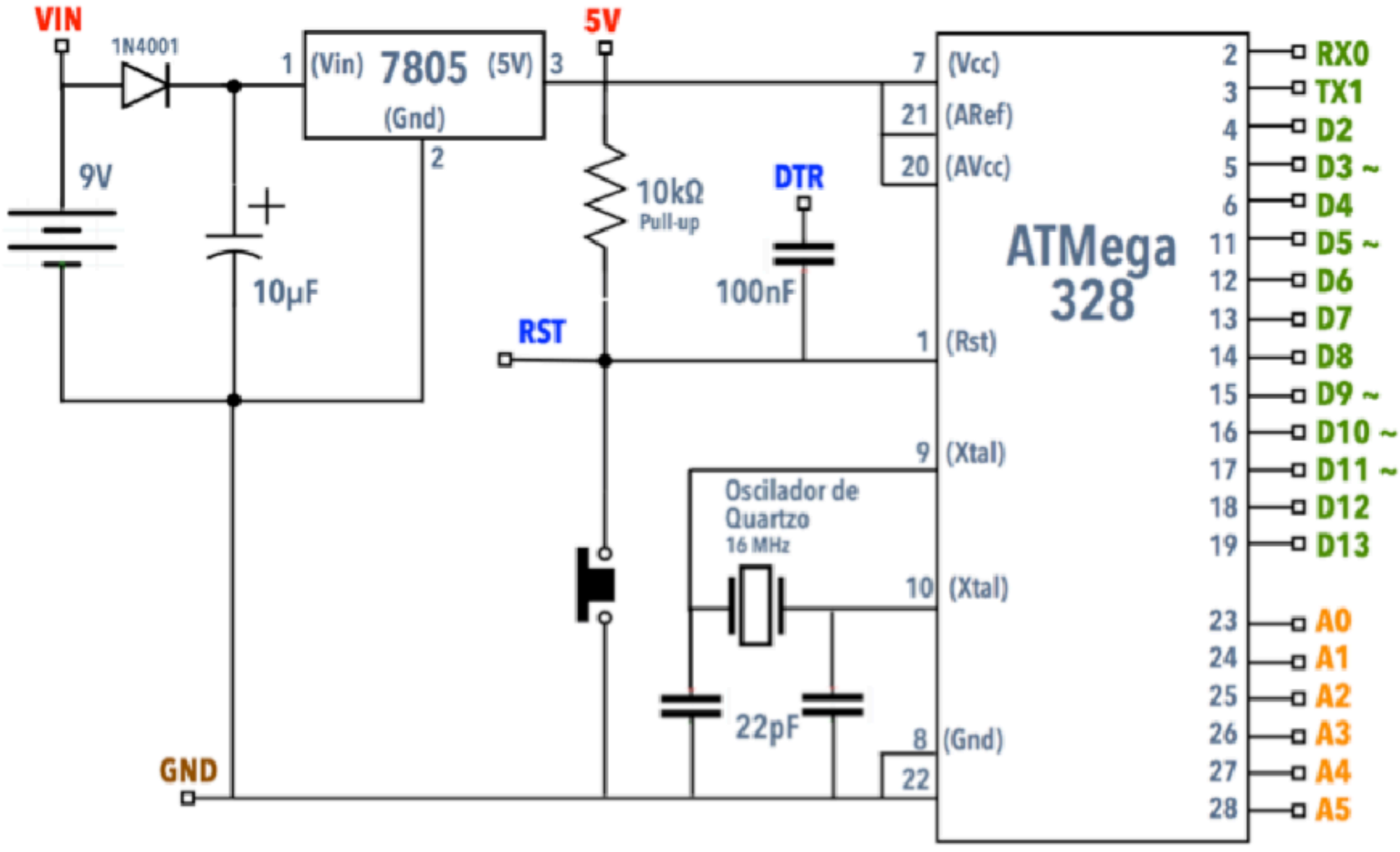


ATMega168 DIL

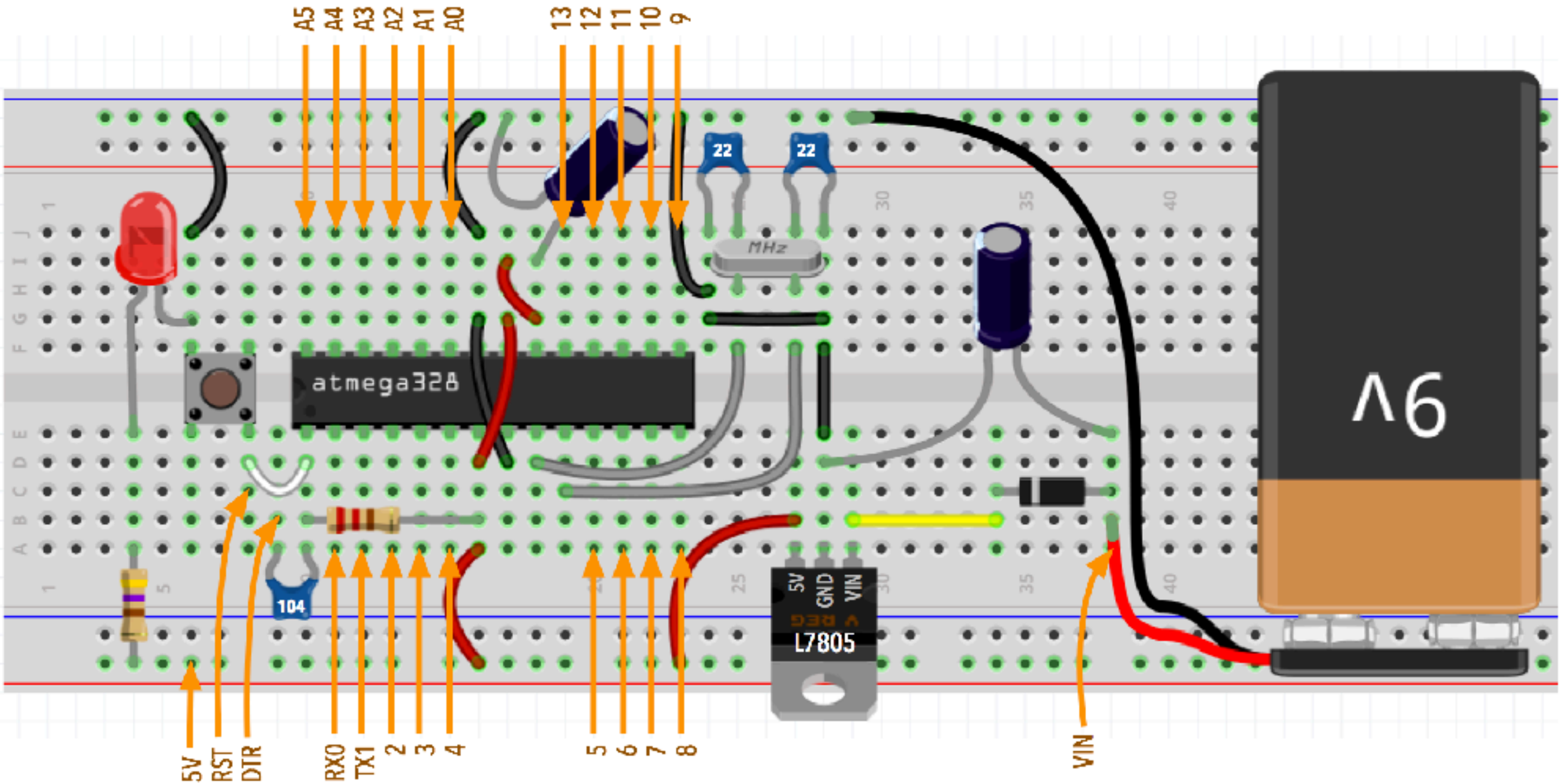
ATTiny85



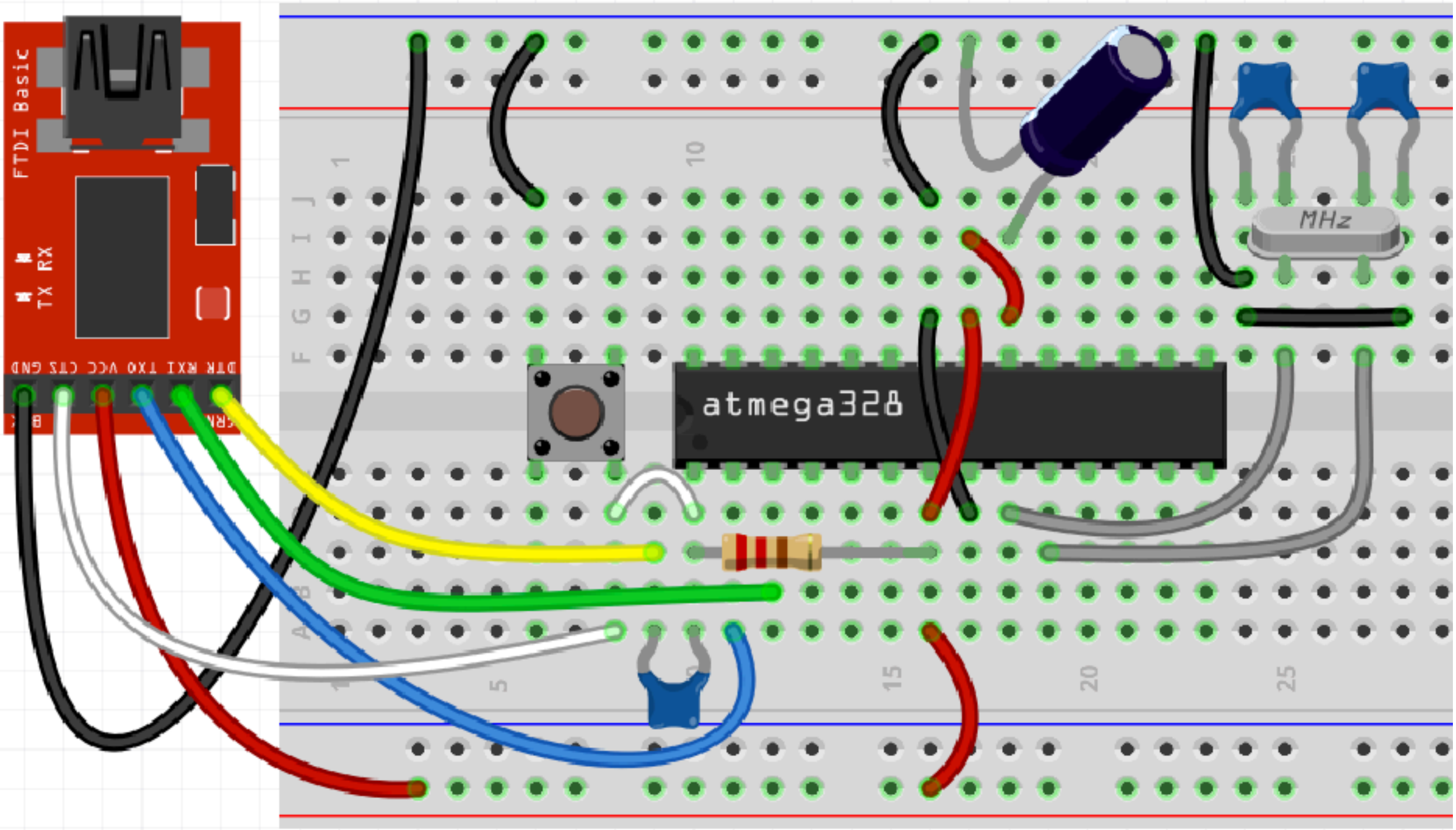
Esquema de um circuito Arduino usando ATMega328



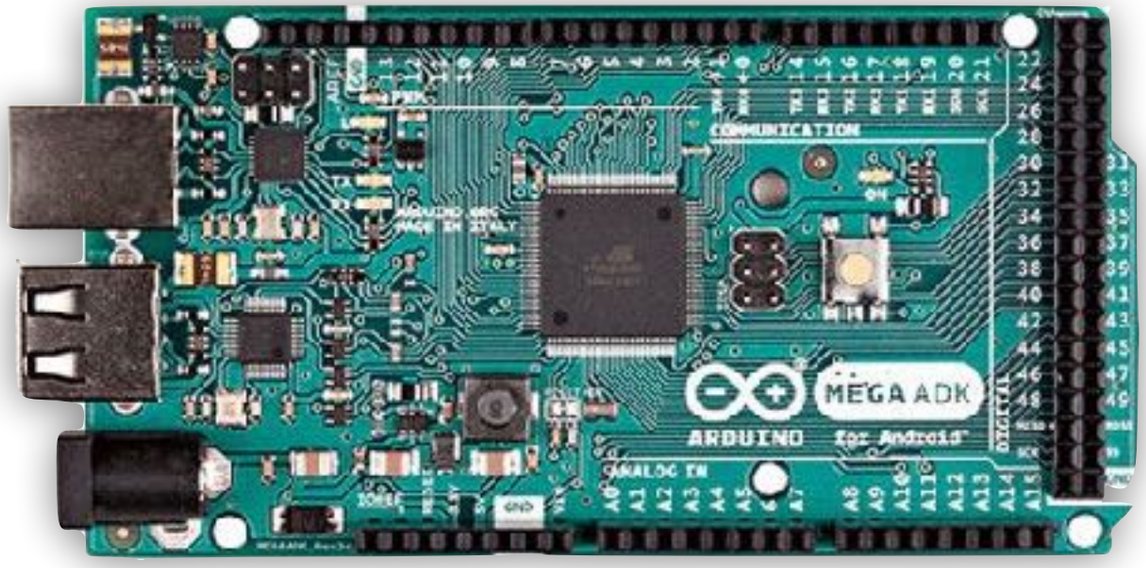
Exemplo de um circuito Arduino usando ATMega328



Uso de um adaptador USB



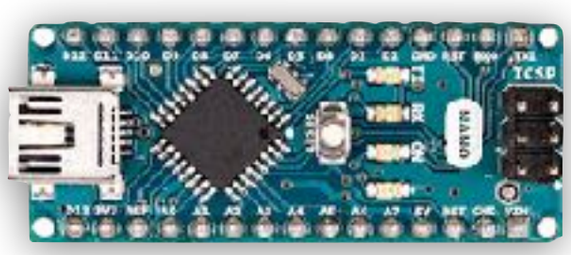
Placas Arduino



Arduino MEGA

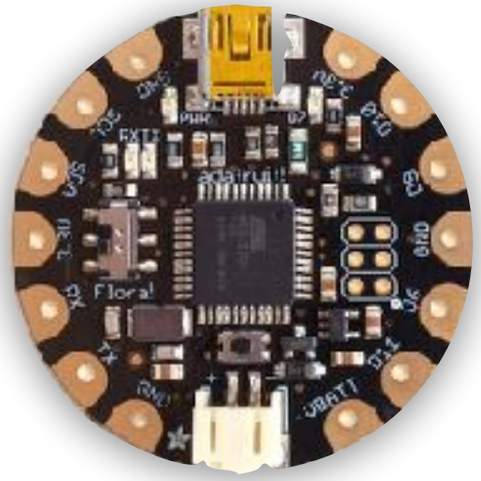
DigiSpark
ATTiny

Arduino Leonardo

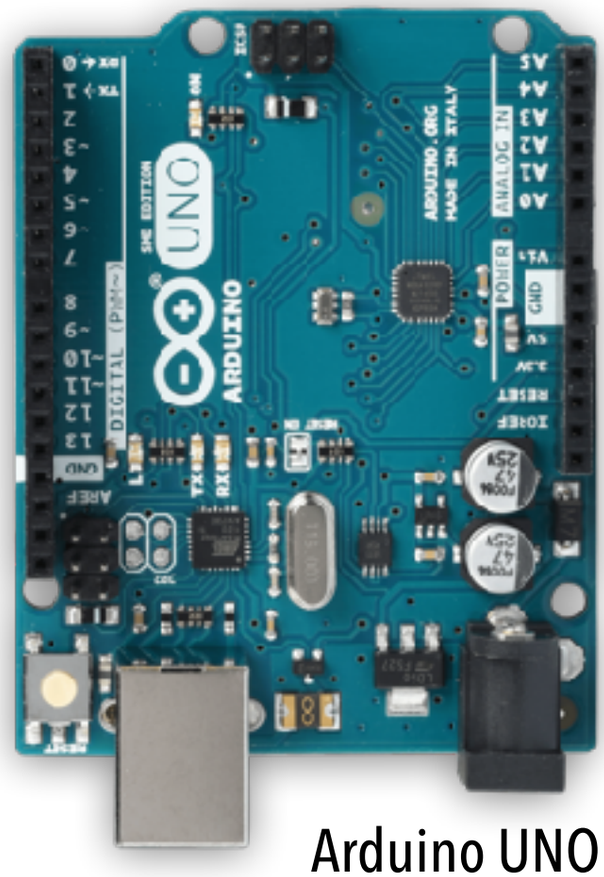


Arduino Nano

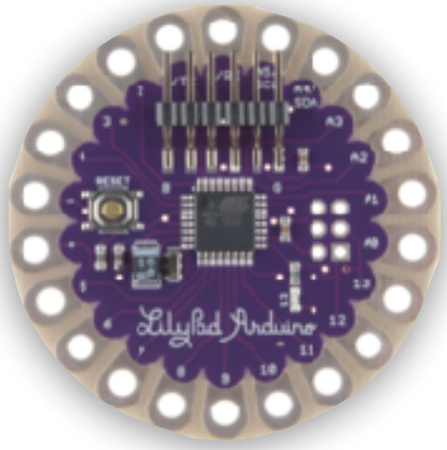
Arduino Pro Mini

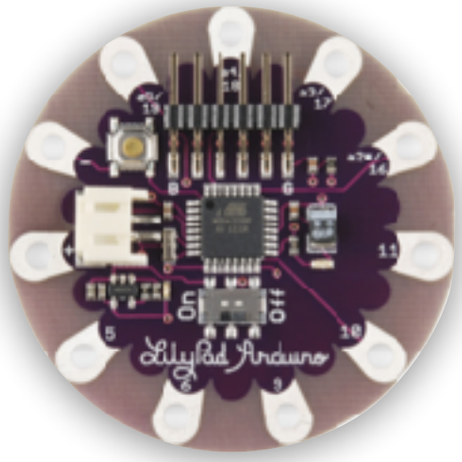


AdaFruit Flora

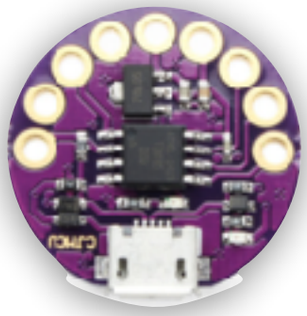


Arduino UNO





LilyPad Arduino

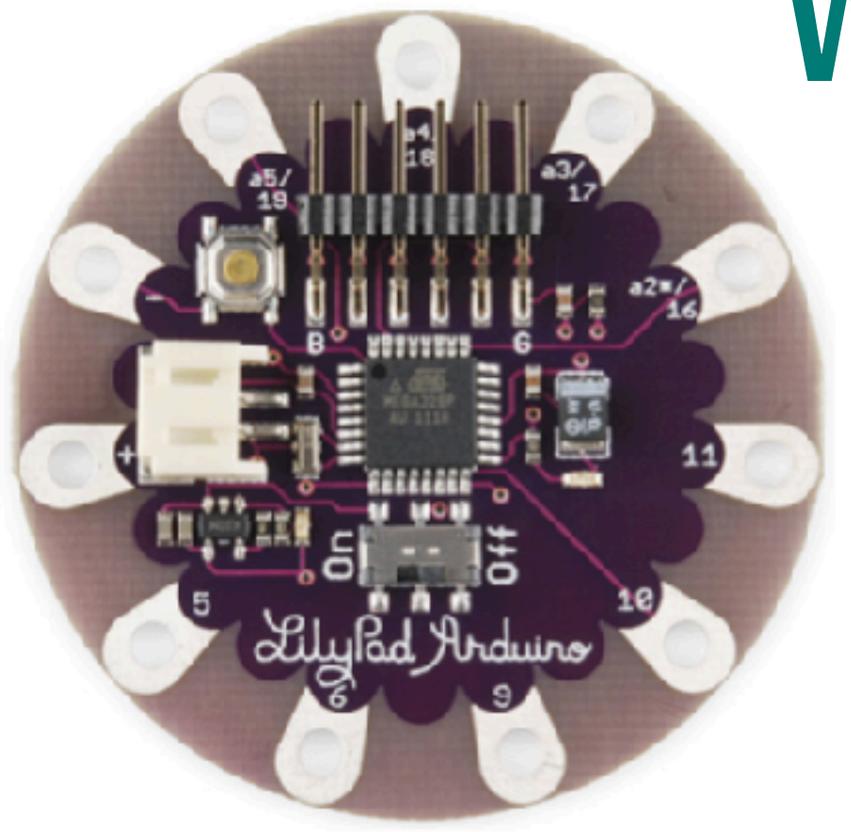


DigiSpark
LilyTiny

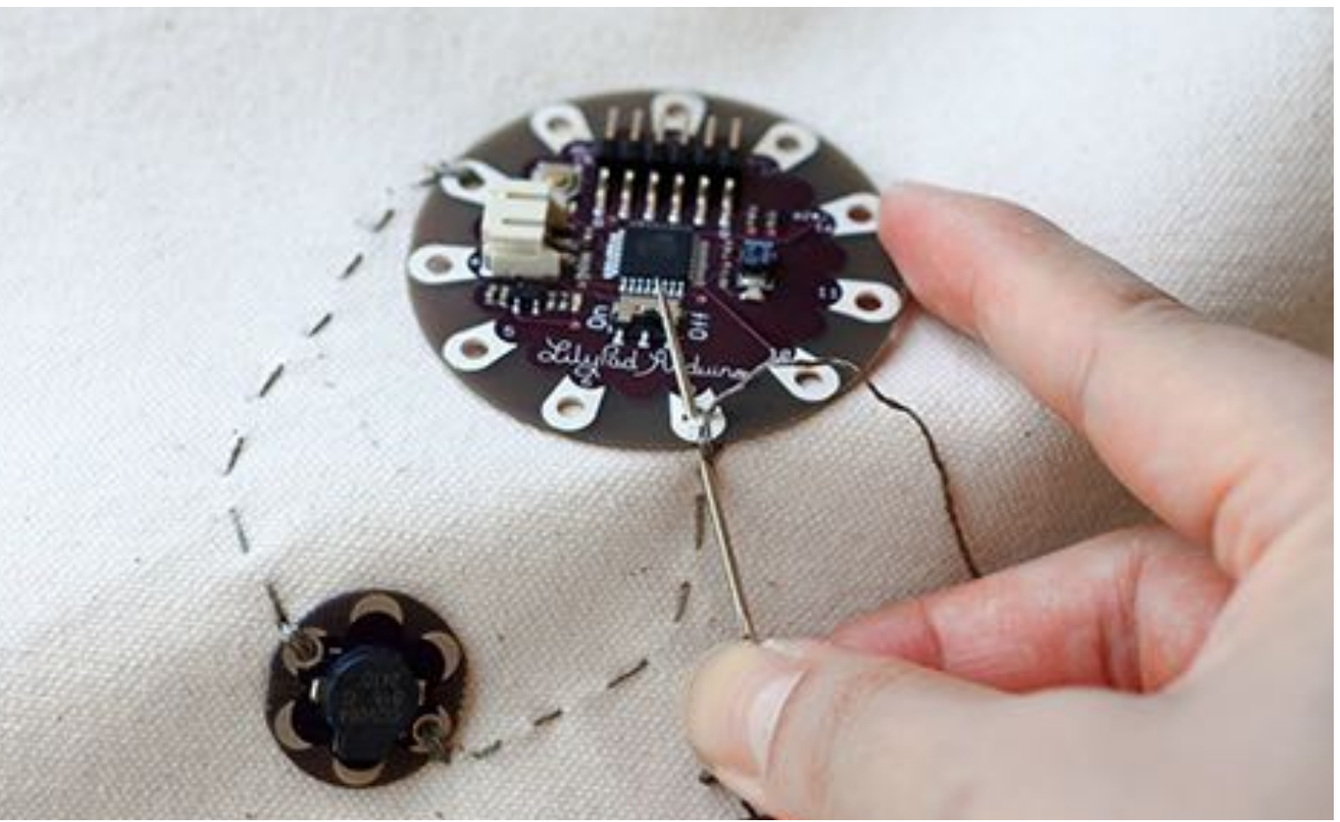
AdaFruit
Gemma

Wearables

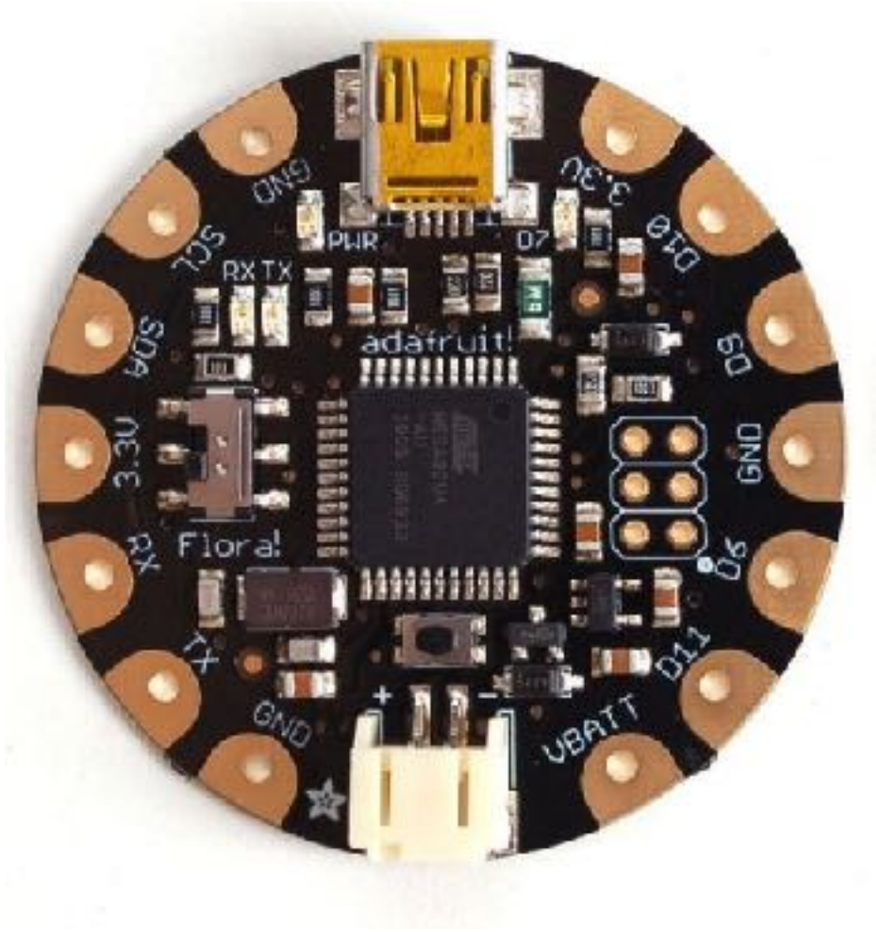
Arduino para vestir



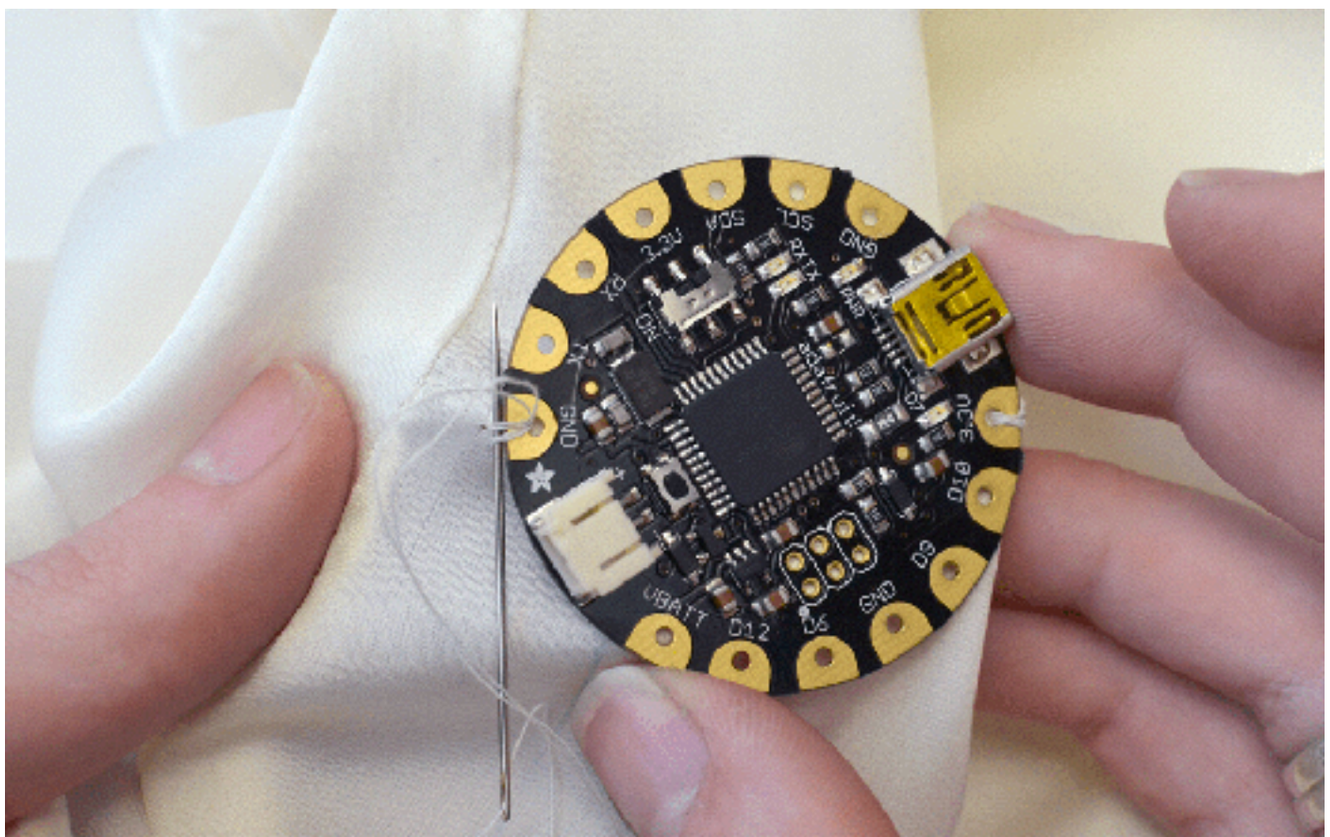
Arduino LilyPad



Wearables



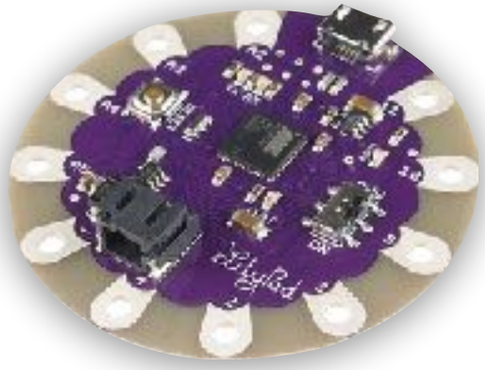
AdaFruit Flora



Placas Arduino com USB

- Podem ser conectadas diretamente ao computador através de um **cabo USB**
- A entrada USB permite a **transferência** de programas para o Arduino (ocupa o pino digital 1) e **fornece energia** (5V)

LilyPad Arduino USB



Arduino Nano

Cabo USB

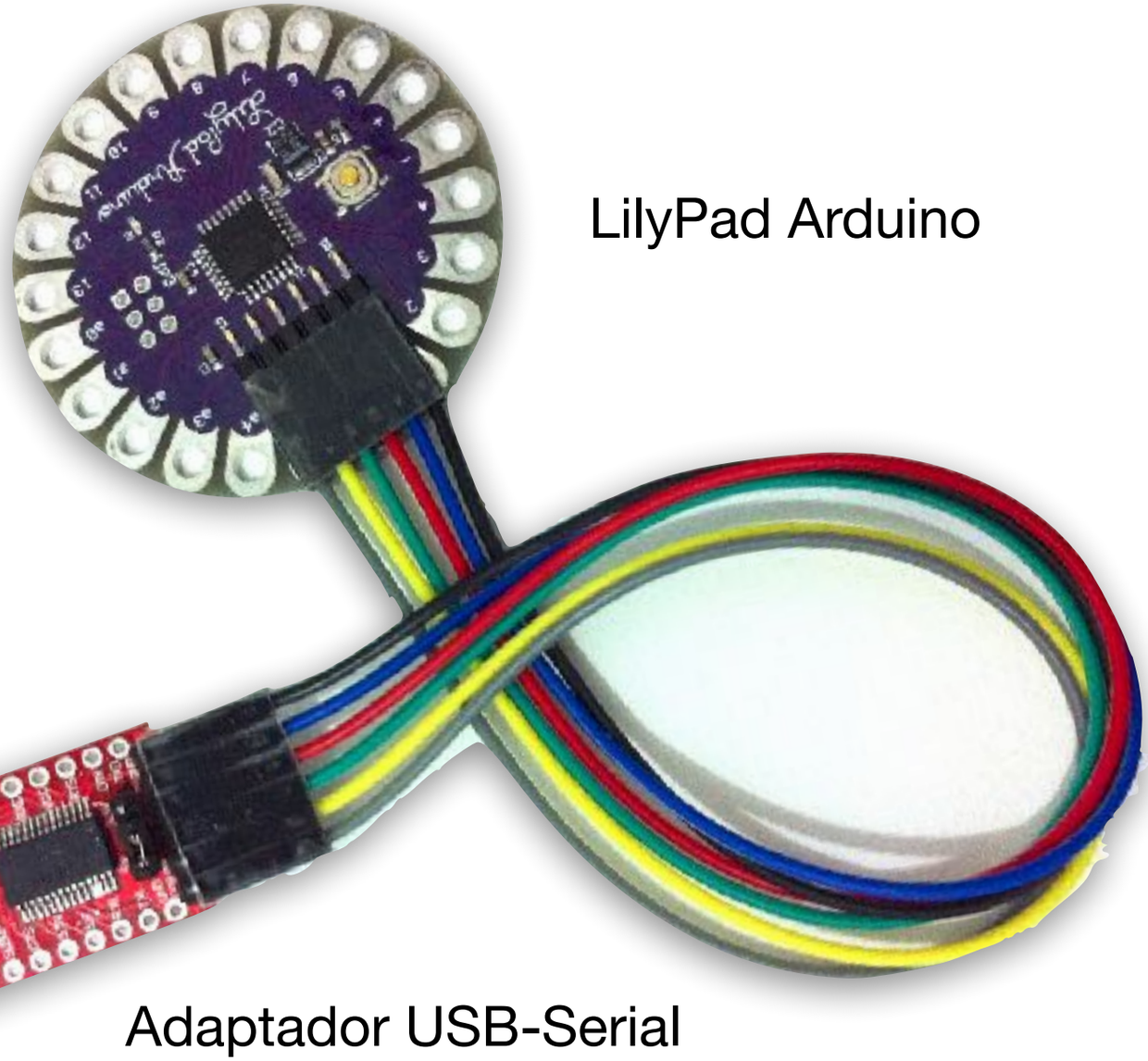
Arduino Mega

Arduino Uno

Placas Arduino sem USB

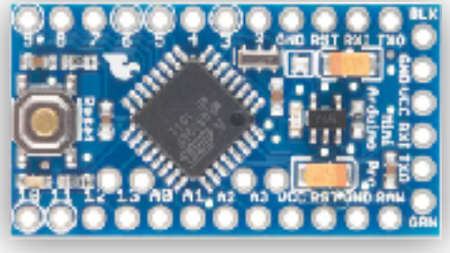
- Para transferir o programa do computador para o Arduino é preciso utilizar um **circuito USB-Serial**

Cabo USB

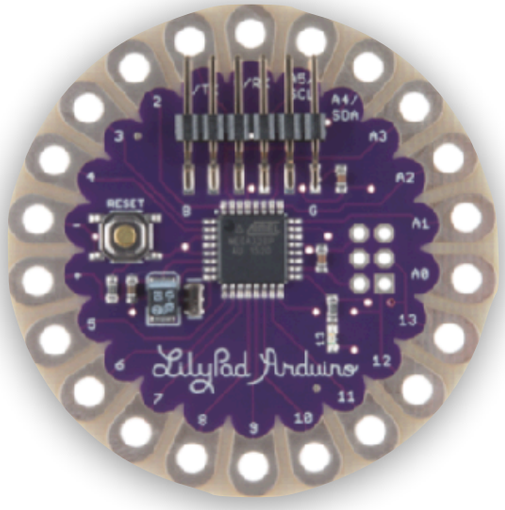


Adaptador USB-Serial

LilyPad Arduino



Arduino Pro Mini



LilyPad Arduino

Adaptador USB-Serial

Instalação

- Para programar o Arduino é necessário usar um **ambiente de desenvolvimento** (IDE) para digitar o código, compilar (traduzir o código para linguagem do microcontrolador) e transferir o programa através de USB

- Para que a comunicação via USB seja possível, é preciso que o computador esteja configurado para **reconhecer a porta**. Isto às vezes requer a instalação de um **driver** (programa que realiza a comunicação via USB)

- O IDE pode ser baixado do **site arduino.cc**. O driver precisa ser baixado do **site do fabricante** (isto depende do Arduino usado, se é original da Itália ou clone).

Driver do adaptador USB

- Descubra se sua placa precisa e qual o fabricante; depois **baixe** e execute o **instalador** do driver).

- Principais fabricantes são ATMel, FTDI e CH341

FTDI Mac, Linux & Windows

https://learn.sparkfun.com/tutorials/how-to-install-ftdi-drivers

CH341

Windows: http://www.wch.cn/download/CH341SER_EXE.html

Mac: http://www.wch.cn/download/CH341SER_MAC_ZIP.html

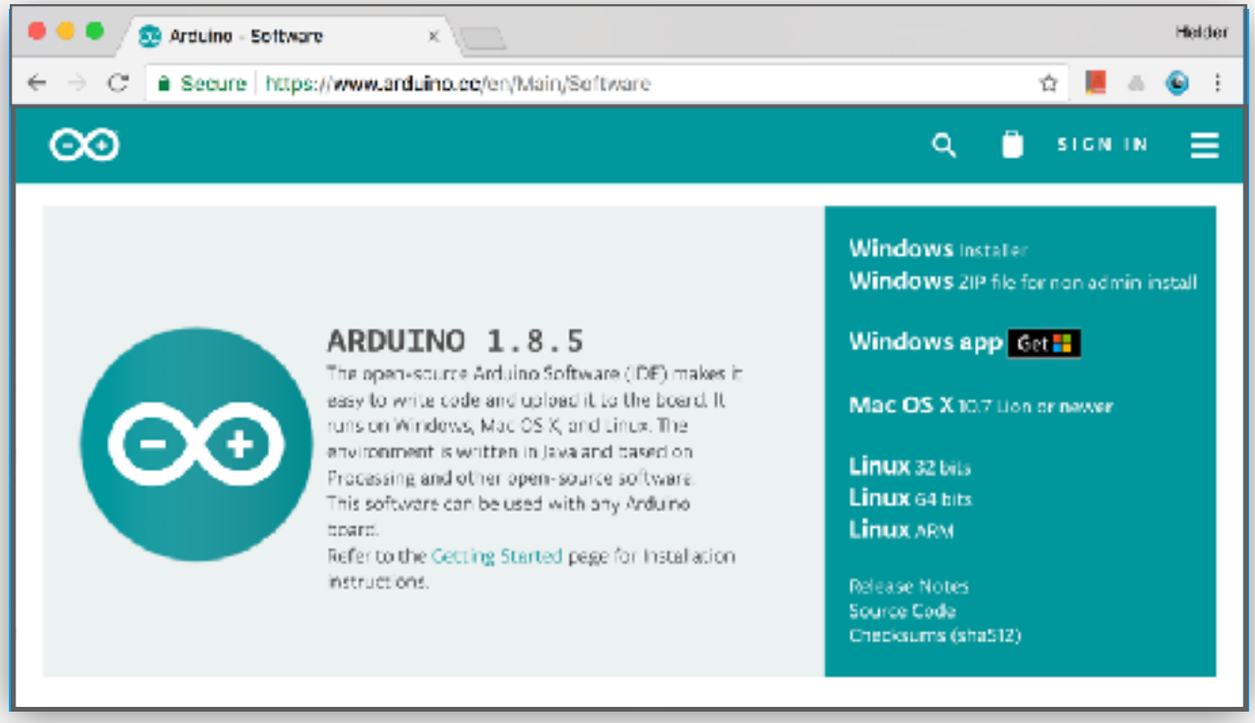
Linux: http://www.wch.cn/download/CH341SER_MAC_LINUX.html

ATMel

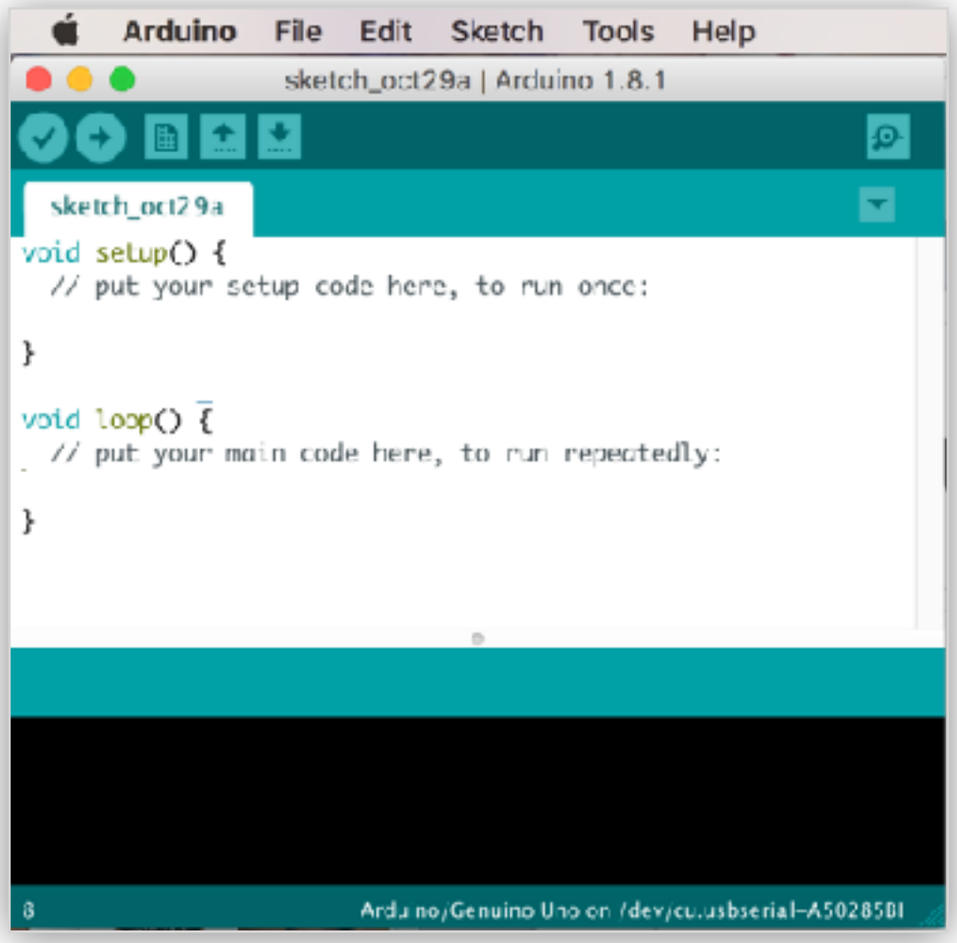
Instalação é feita automaticamente durante a instalação do ambiente de desenvolvimento (IDE)

Aplicação Arduino (IDE)

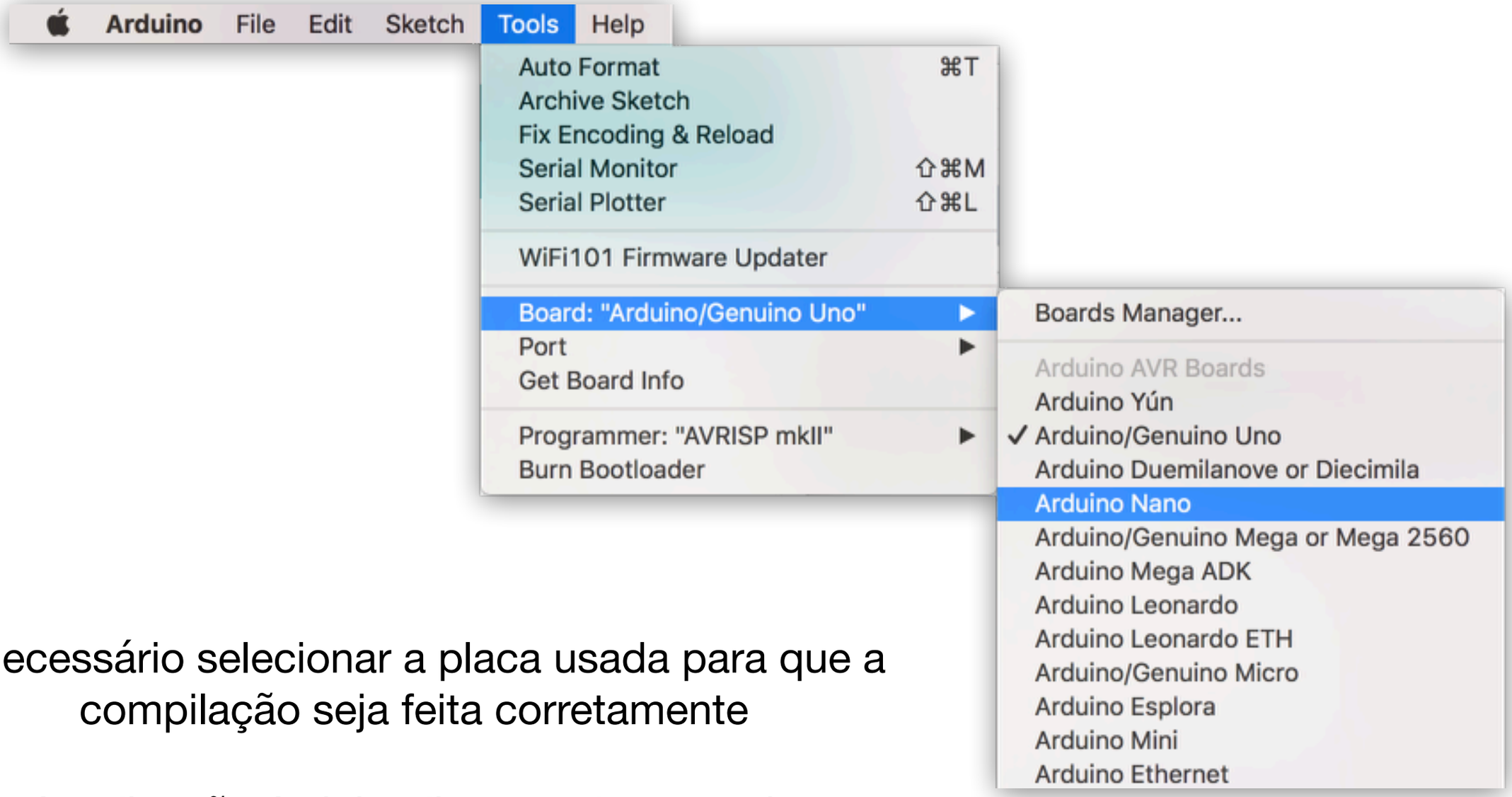
- Baixe o instalador em **https://www.arduino.cc/en/Main/Software**
- Disponível para Windows, Mac e Linux
- Instala automaticamente o driver USB Atmel



Arduino



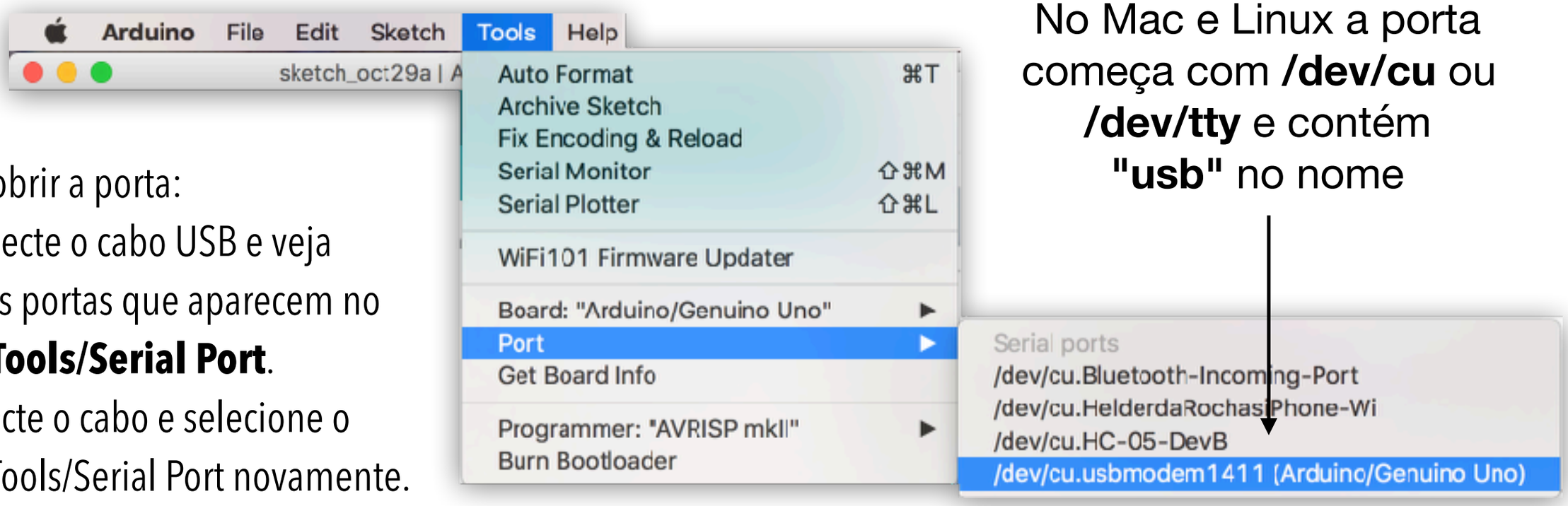
Seleção do tipo de placa



É necessário selecionar a placa usada para que a compilação seja feita corretamente

A aplicação Arduino lista as placas mais conhecidas (se uma placa não estiver na lista, é necessário baixar e instalar suporte para ela)

Seleção da porta de comunicação USB



No Mac e Linux a porta começa com **/dev/cu** ou **/dev/tty** e contém **"usb"** no nome

Como descobrir a porta:

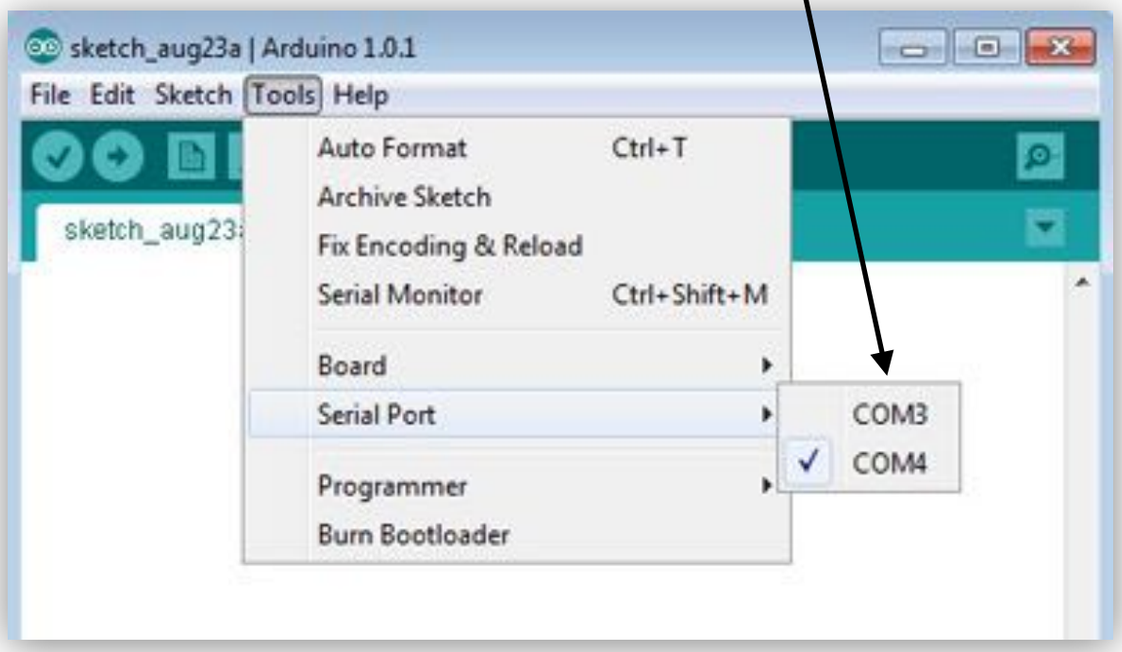
1. Desconecte o cabo USB e veja quais as portas que aparecem no menu **Tools/Serial Port**.
2. Reconecte o cabo e selecione o menu Tools/Serial Port novamente.

   A porta que aparecer na lista é a porta do Arduino

Se não aparecer uma porta nova ao conectar o Arduino em USB, o driver pode não ter sido instalado corretamente

Às vezes é necessário reiniciar o computador (a versão do driver também pode ser incompatível com a versão do sistema operacional usado)

No Windows a porta aparece como **COM2**, **COM3**, **COM4**, etc.



Arduino IDE: barra de menu

Compila o programa
(erros indicam problemas no programa, como erros de sintaxe, falta de bibliotecas, etc.)

Abre uma janela nova
(com um editor em branco ou contendo o programa mínimo)

Monitor serial
Permite imprimir na tela dados do Arduino conectado via USB (usa o pino digital 0)

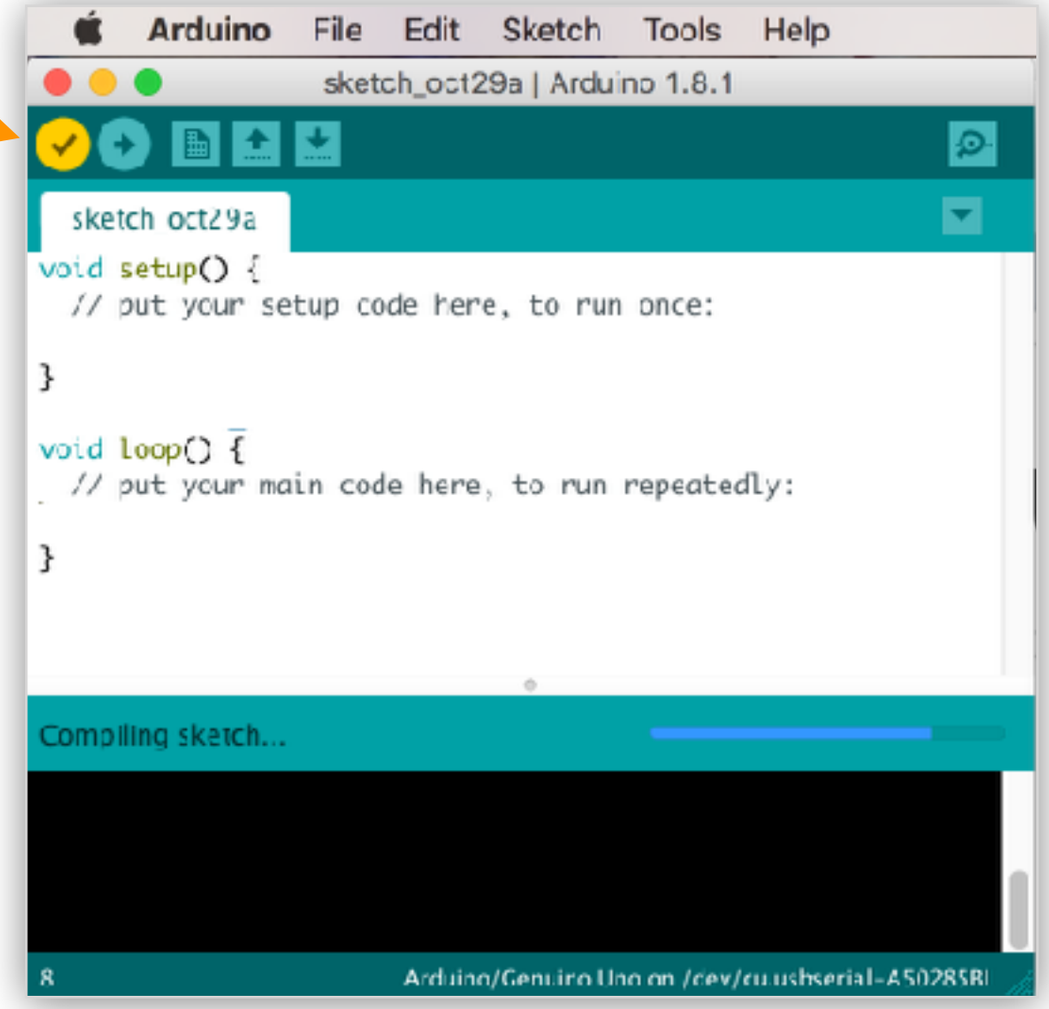
Transfere o programa para o Arduino
(erros indicam problemas na transferência como falta de drivers, uso do pino 1, etc.)

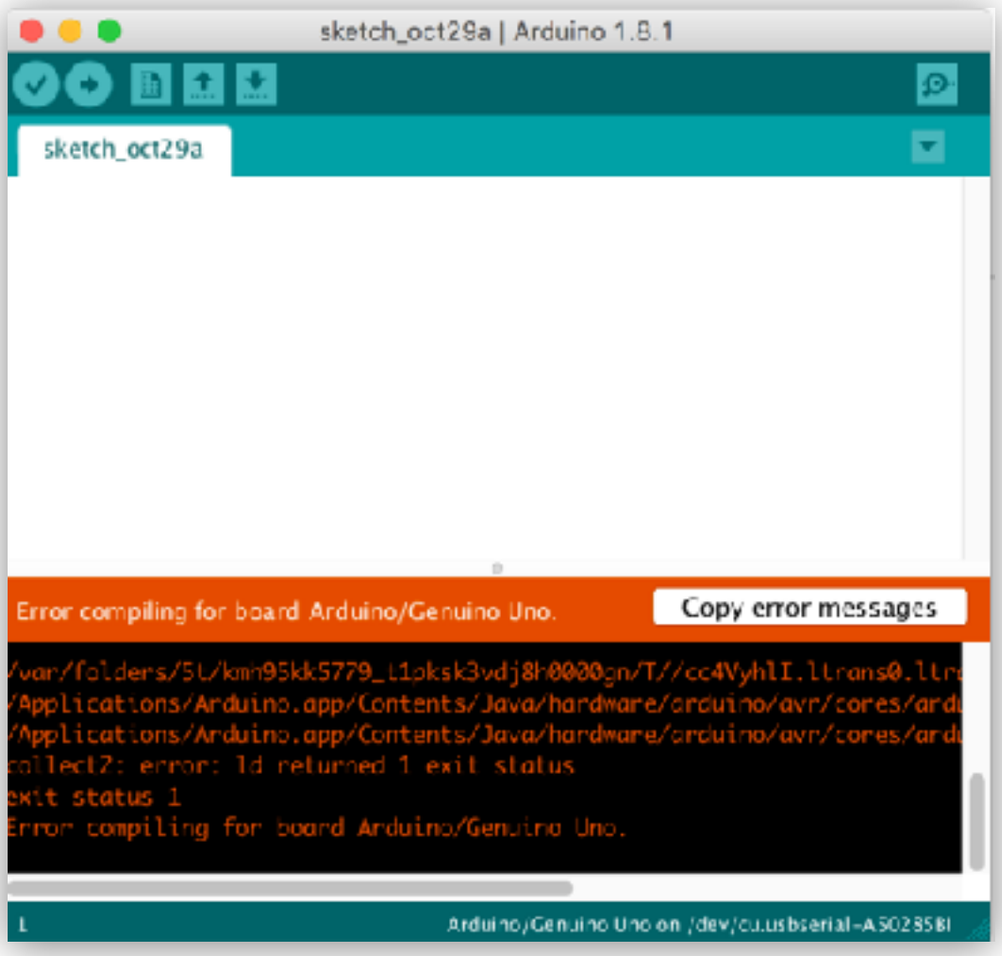
Abre um programa armazenado em disco
(mesmo que File/Open)

Grava um programa em disco
(mesmo que File/Save)

Compilando um programa

Aperte aqui para compilar o programa
(aplicação irá tentar converter o programa em linguagem de máquina)

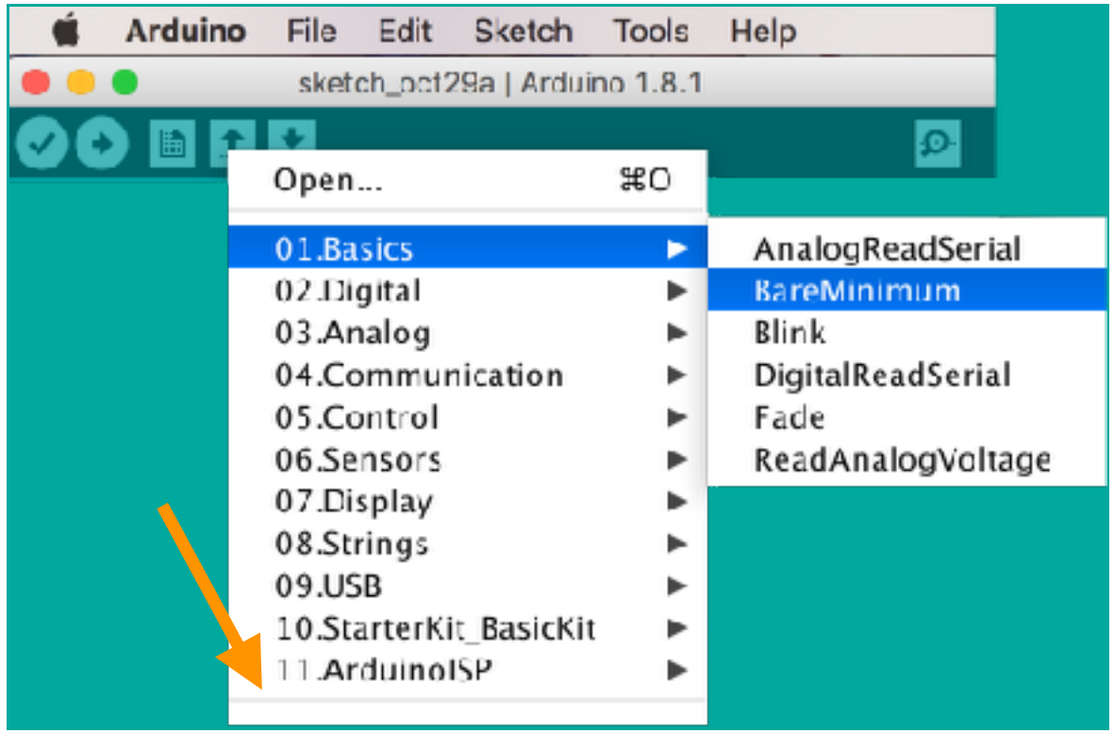


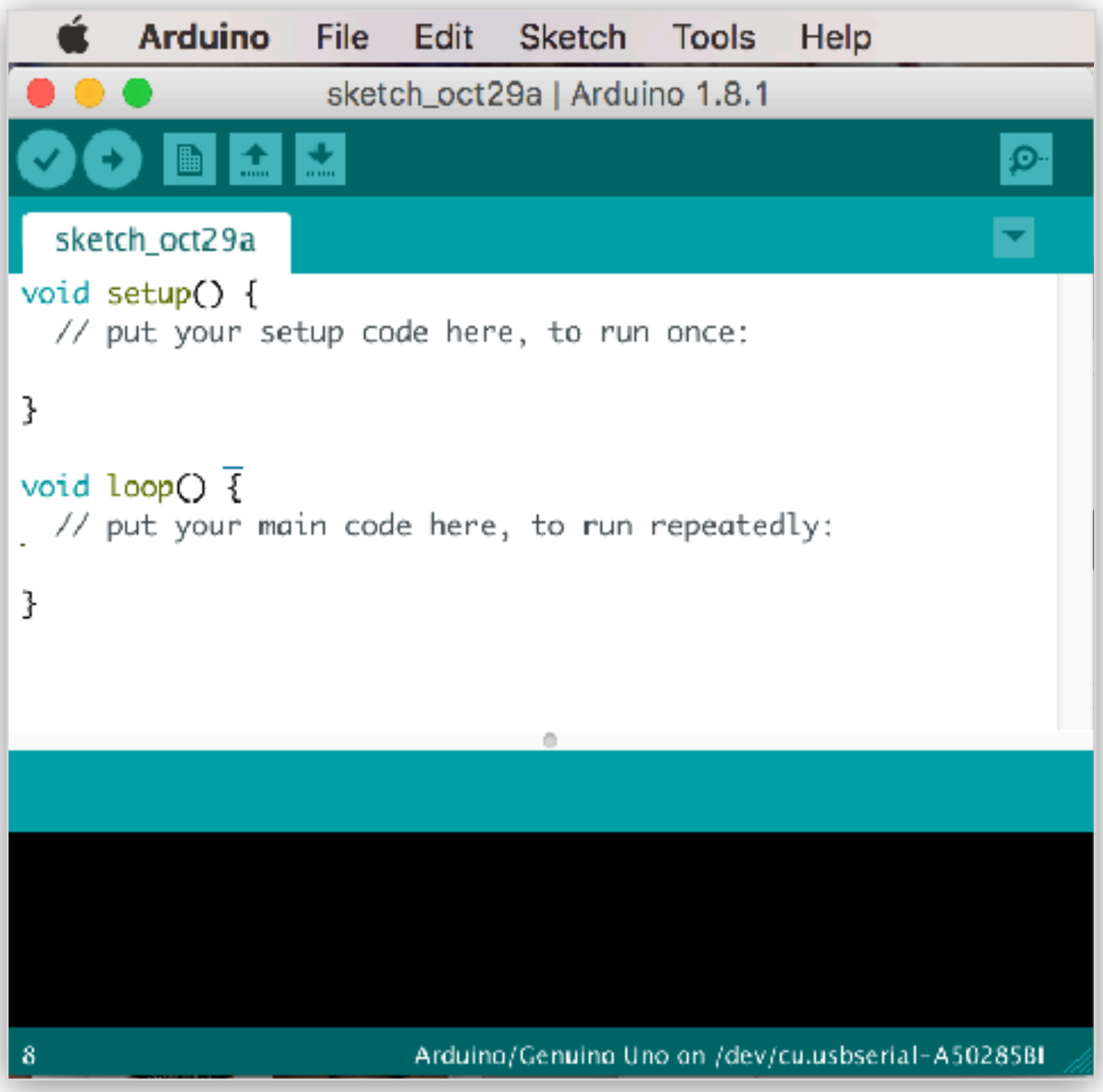


Erros de compilação
impedem a geração do código de máquina (geralmente são causados por erros de sintaxe, falta de bibliotecas, etc.)

Programa mínimo

Carregue programa BareMinimum
é o programa mais simples que compila (mas não faz nada)





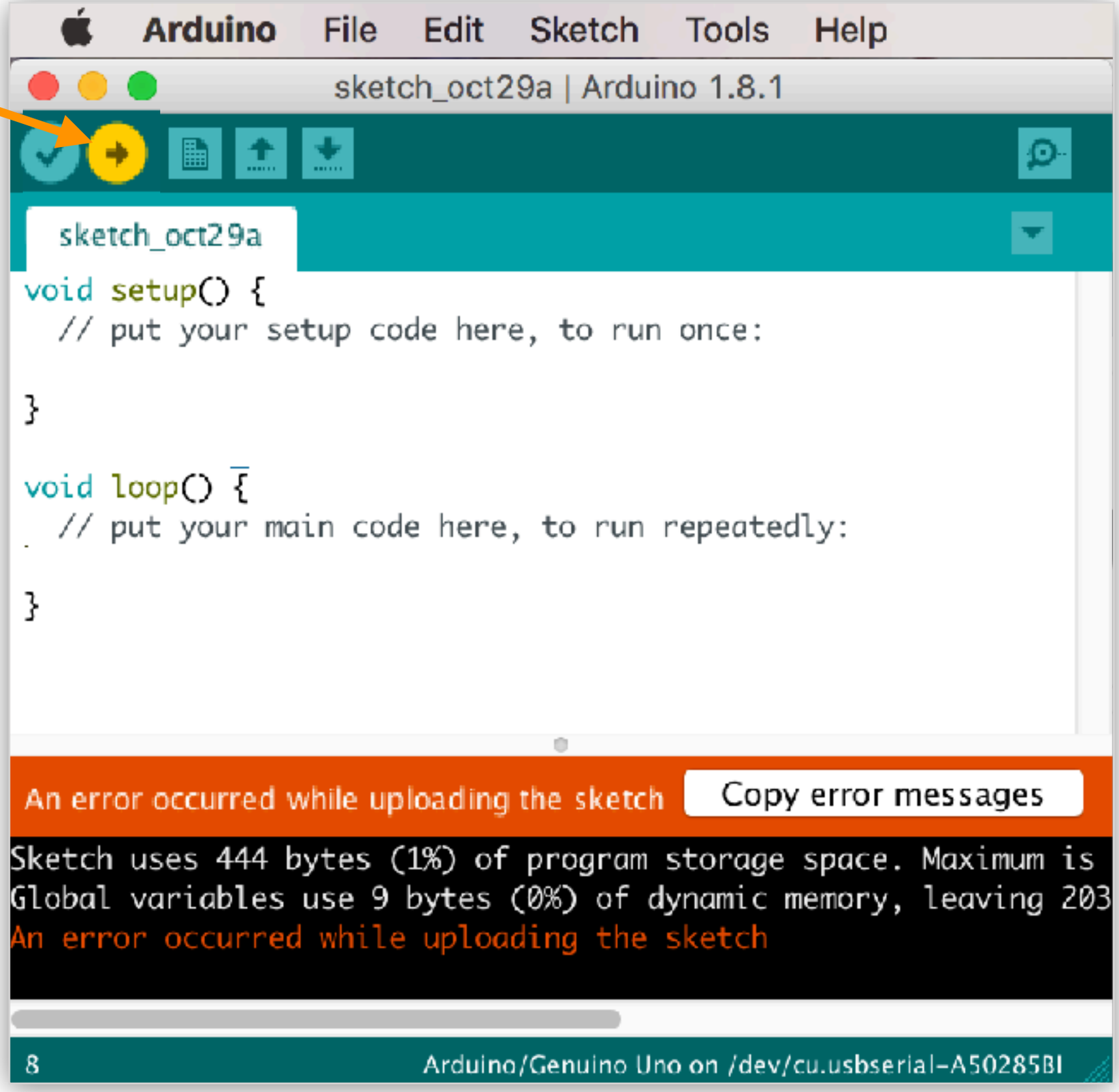
Programa mínimo contém dois blocos
(instruções estarão entre chaves { ... })

void setup():
Contém instruções que irão executar uma única vez (na inicialização do programa)

void loop():
Contém instruções que irão executar continuamente (as instruções se repetem)

Transferindo o programa para a placa

Aperte aqui para transferir o programa



Depois de compilado com sucesso (sem erros) o programa poderá ser **transferido** para o Arduino

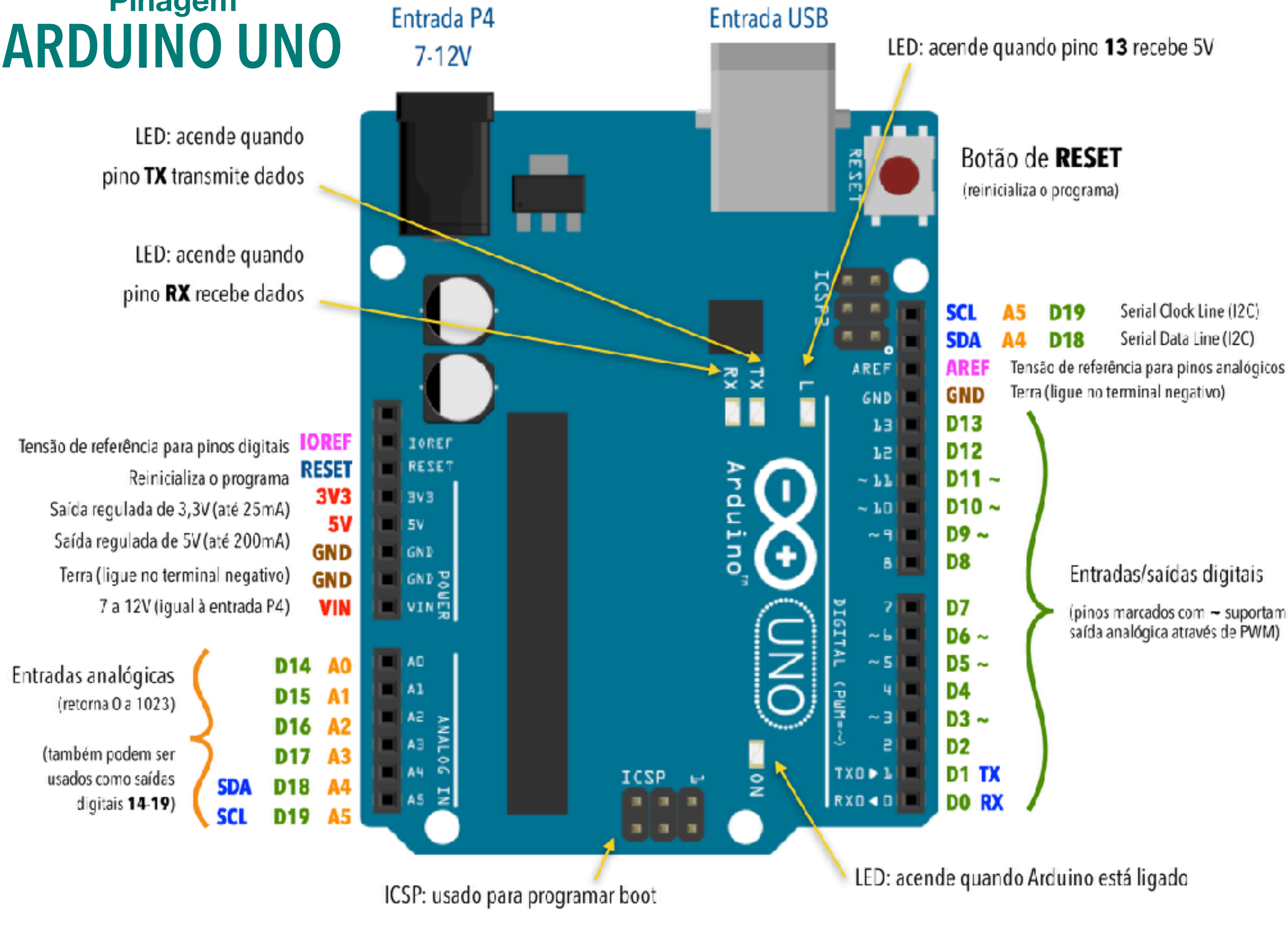
Se a transferência ocorrer com sucesso, o Arduino poderá ser desconectado

Causas mais comuns de erros que acontecem durante a transferência

- **Tipo de Arduino** incorreto (veja menu **Board**)
- **Porta de comunicação** incorreta (veja menu **Port**)
- **Pino 1** está sendo usado pelo circuito (desligue-o durante a transferência)

Pinagem

ARDUINO UNO



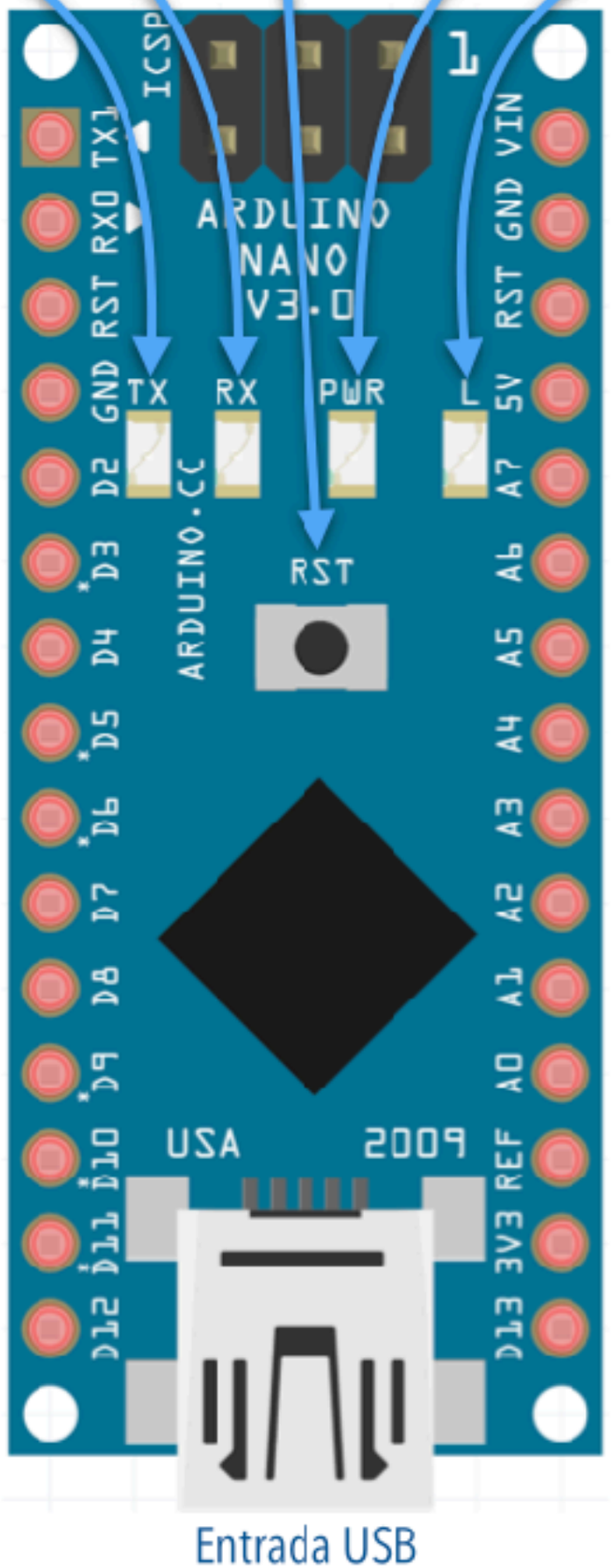
LED: acende quando pino **TX** transmite dados

LED: acende quando pino **RX** recebe dados

Botão de **RESET**
(reinicializa o programa)

LED: acende quando Arduino está ligado

LED: acende quando pino **13** recebe 5V



Entradas/saídas digitais

TX **D1**

RX **D0**

Reinicializa o programa **RESET**

Terra (ligue no terminal negativo) **GND**

Entradas/saídas digitais

(pinos marcados com ~ suportam saída analógica através de PWM)

D2 **~D3** **D4** **~D5** **~D6** **D7** **D8** **~D9** **~D10** **~D11** **D12**

VIN Alimentação externa de 5 a 12V

GND Terra (ligue no terminal negativo)

RESET Reinicializa o programa

5V Saída regulada de 5V (até 200mA)

A7 **A6** **A5** **A4** **A3** **A2** **A1** **A0**

Entradas analógicas (valores 0 a 1023)

(pinos **A0** a **A5** também podem ser usados como saídas digitais **14**-**19** - sem suporte PWM)

AREF Tensão de referência para pinos analógicos

3V3 Saída regulada de 3,3V (até 25mA)

D13 Entrada/saída digital

Entrada USB

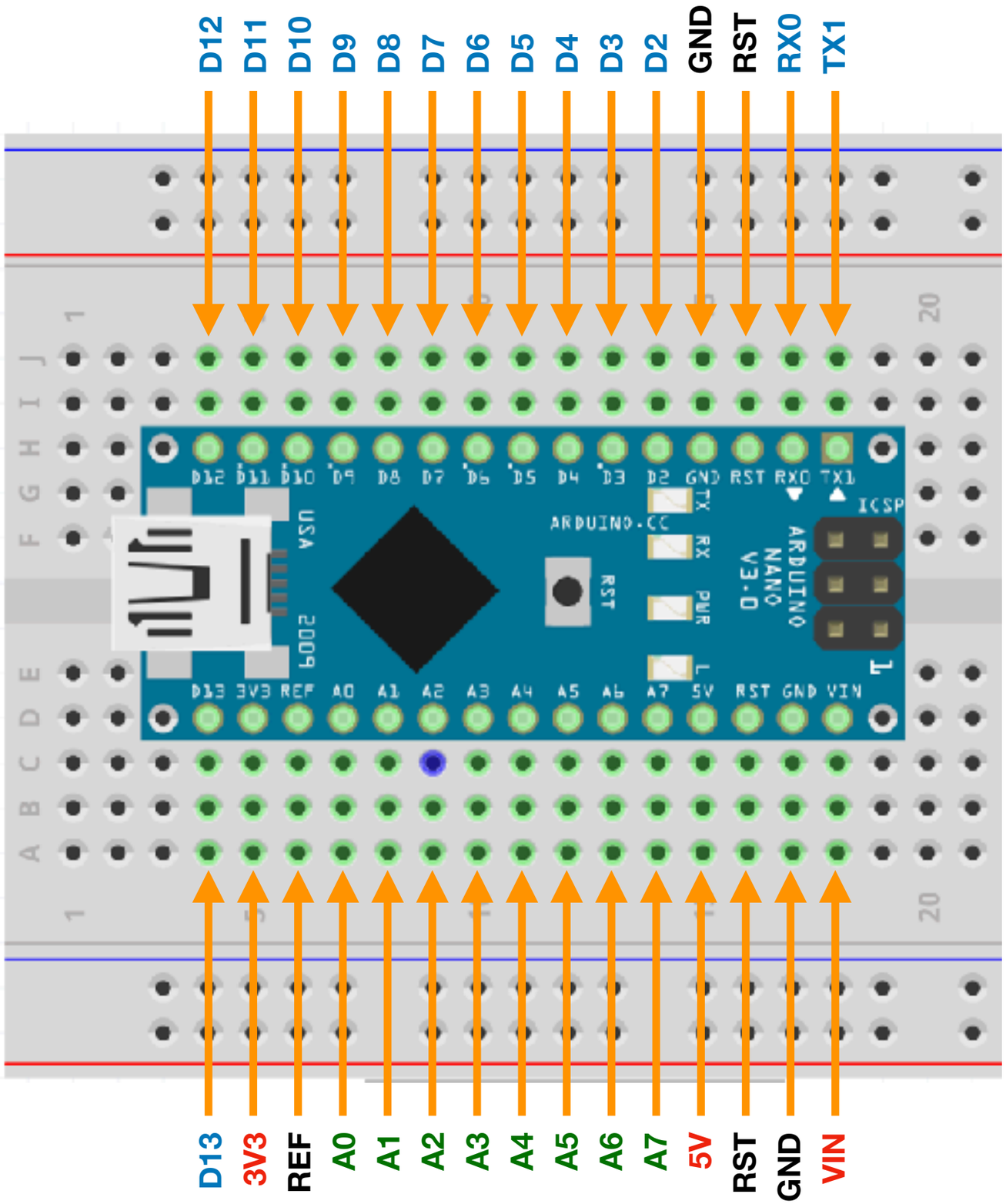
Evite usar pinos D0, D1 e D13

- Programas não podem ser transferidos via USB se o pino **D1** estiver sendo usado.
- O Monitor Serial não irá funcionar se o pino **D0** estiver sendo usado.
- O pino **D13** está permanentemente conectado a um LED

Pinagem

ARDUINO NANO

Arduino Nano no protoboard



Protoboard conecta 2 a 3 soquetes por pino

Encaixe os terminais com cuidado, mas firmemente (não deve haver maus contatos)

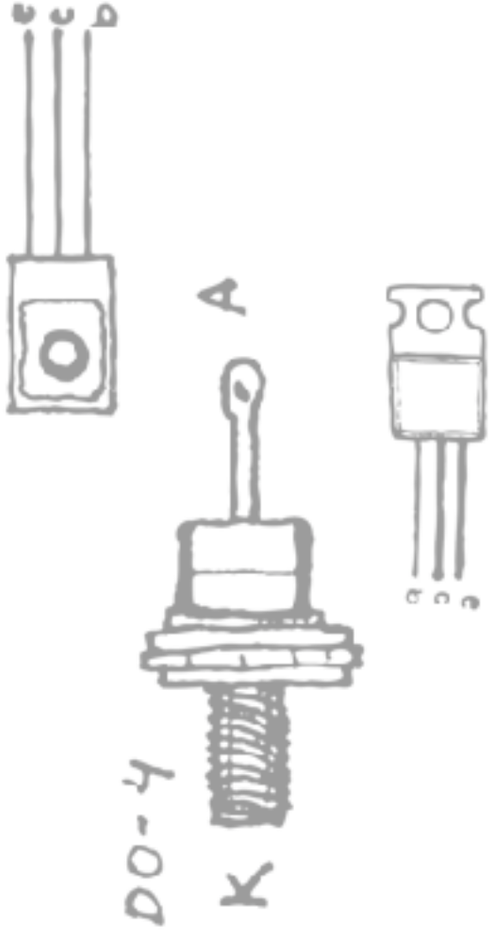
Teste da instalação com o programa "Blink"

1. Conecte o Arduino ao computador via USB

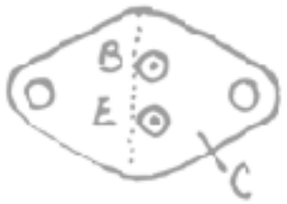
2. Carregue o programa **"01.Basics/Blink"**

3. Veja o LED **"L"** da placa Arduino piscar

4. Desconecte o Arduino do computador e reconecte a uma fonte de energia (9V) usando a entrada **P4** (Arduino UNO) ou **pinos VIN(+) e GND(-)** (Arduino Nano)



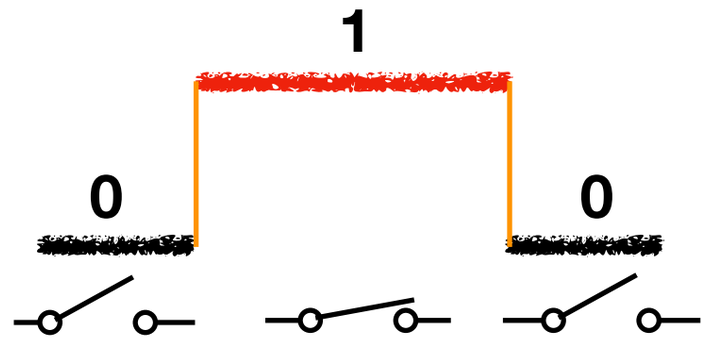
ARDUINO 2



Saída digital

Helder da Rocha
www.eletronicaparaartistas.com.br

Lógica digital



- Associa **valores de tensão** a **níveis lógicos** binários (0 ou 1)

- O valor da tensão é uma **referência**. A referência mais comum para o valor **ALTO** (HIGH) ou **LIGADO** (ON) é **5V**

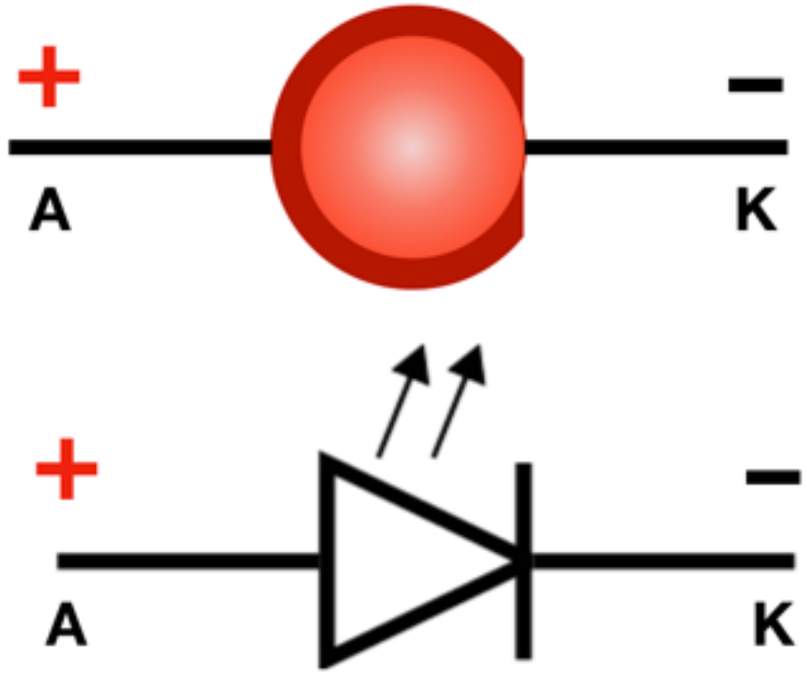
- A referência para o valor **BAIXO** (LOW) ou **DESLIGADO** (OFF) é **0V**

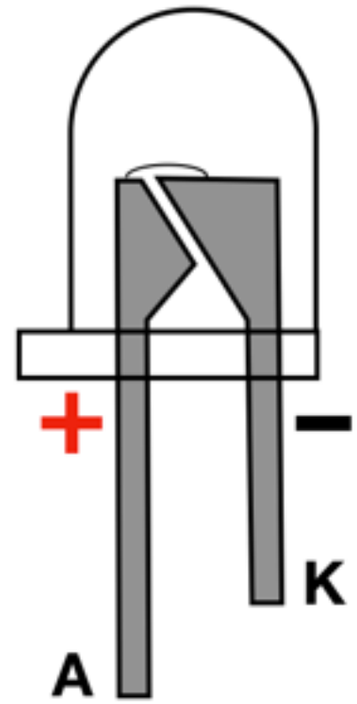
- Em programação, **5V** é representado pelo algarismo **1** e **0V** é representado pelo algarismo **0**

- **Informação** é representada em **programas de computador** através de **sequências de 0s e 1s** (lógica binária). A unidade é o **bit** (que pode ser 0 ou 1) e palavras (sequências) de 8 bits são **bytes**.

- **Bits** são representados em **eletrônica** por **chaves** (geralmente transistores) que podem reter 5V (armazenam a informação 1) ou 0V (armazenam 0). A lógica também pode ser inversa.

LED (Light-Emmiting Diode)

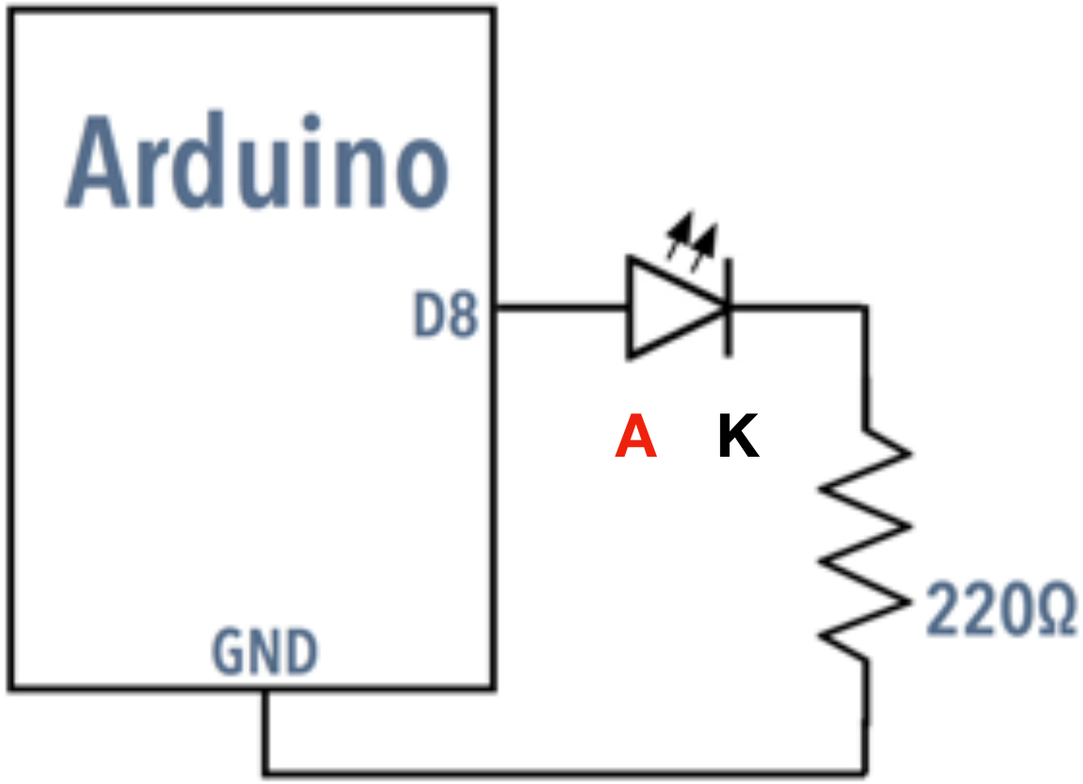




Corrente máxima: **20mA**
Tensão sobre o LED em 20mA é fixa
LEDs vermelhos e amarelos: **~2V**
LEDs verdes, azuis e brancos: **~3V**

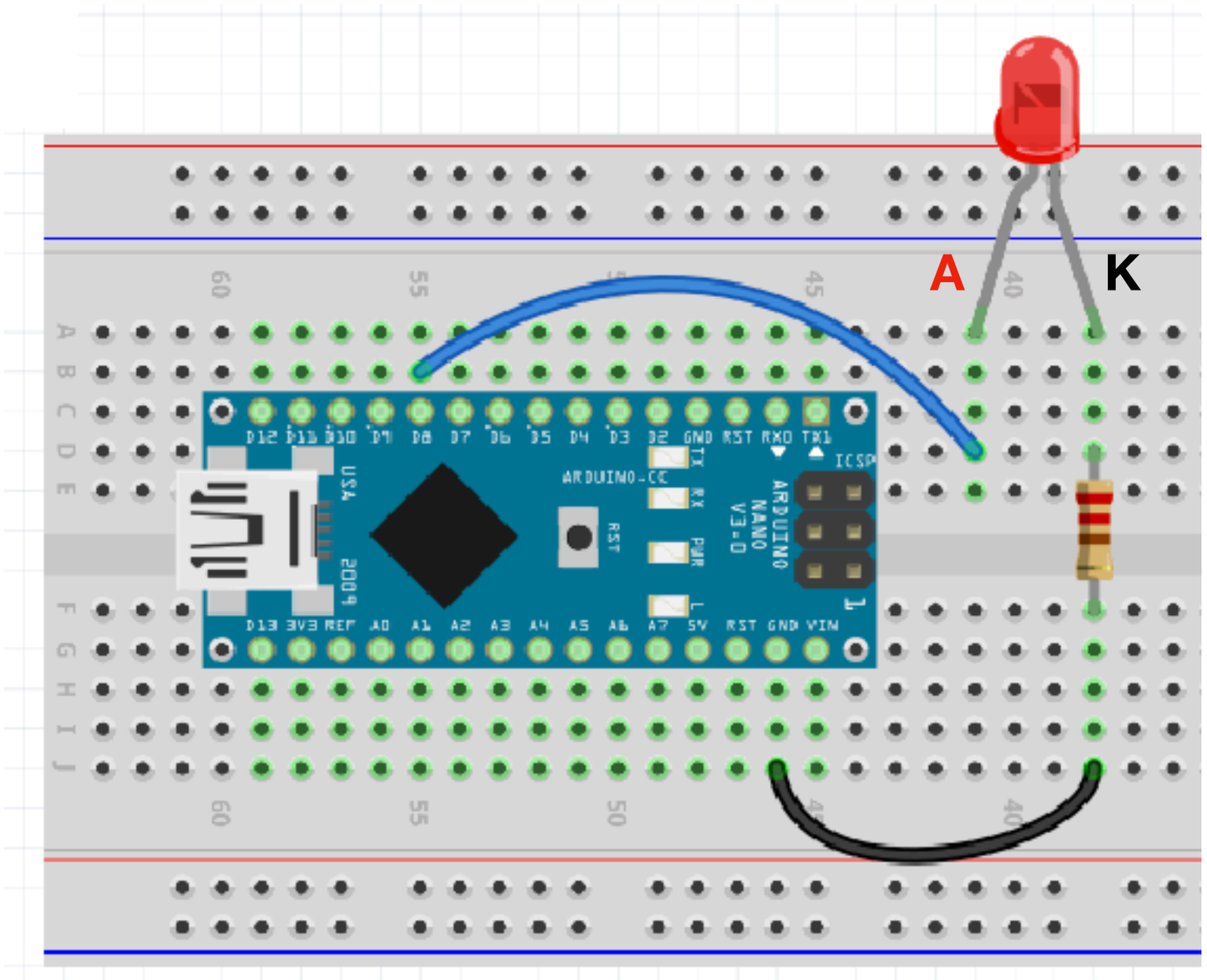
Para limitar a corrente em 20mA em circuitos com Arduino use um resistor de **no mínimo 150 ohms**

digitalWrite



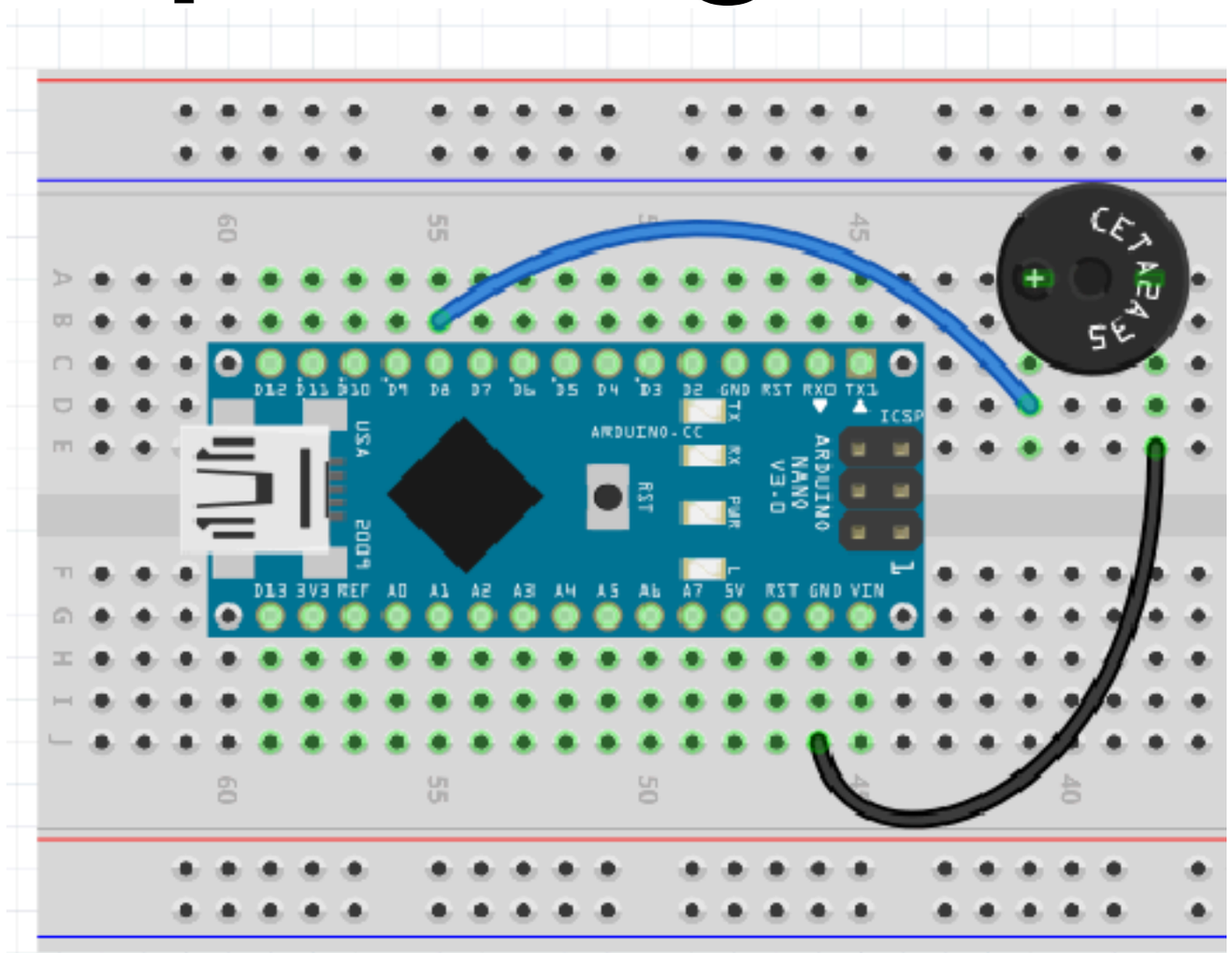
"Blink"
LED acende quando pino 8 for HIGH
LED apaga quando pino 8 for LOW

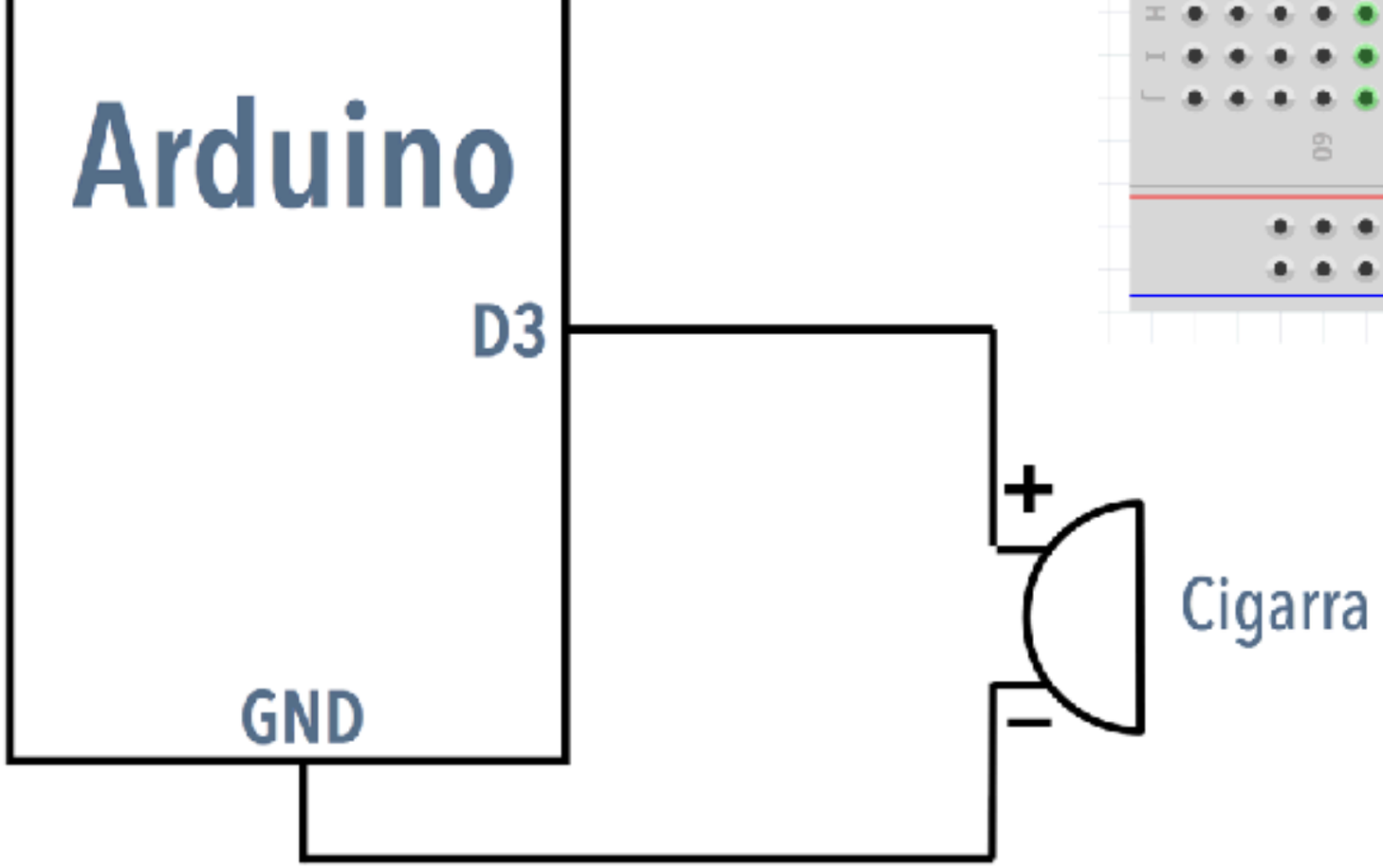
```
void setup() {
    pinMode(8, OUTPUT);
}

void loop() {
    digitalWrite(8, HIGH);
    delay(50);
    digitalWrite(8, LOW);
    delay(50);
}
```

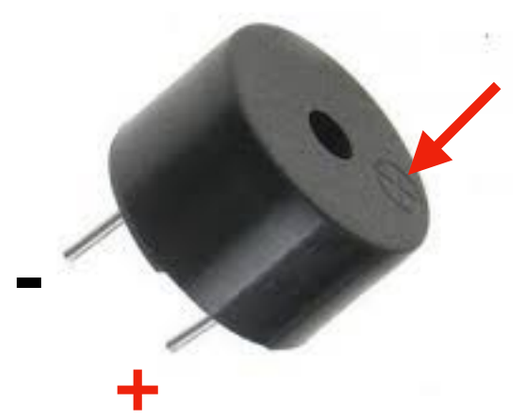


Troque o LED pela cigarra

(use o mesmo programa que pisca o LED)







Positivo e negativo

Nos esquemas e dispositivos (módulos, arduinos) é comum usar os nomes:

- **VCC** (+), ou **V+**

  (**V**oltage = tensão; CC é sufixo usado por convenção)

- **GND** (-)

  (**G**rou**ND** = terra = pólo negativo = 0 volts)

Referência é relativa!

Tensão é DIFERENÇA de potencial

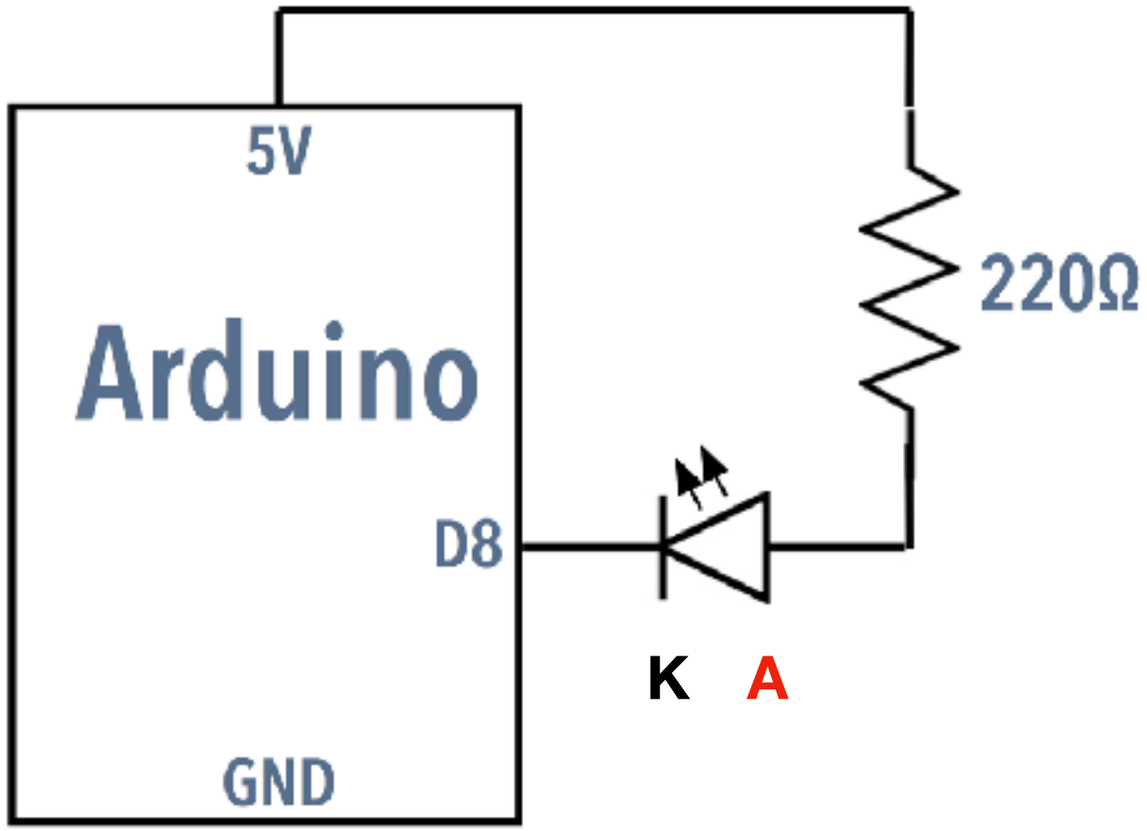
VCC para GND (+)

GND para VCC (-)

GND para GND (0)

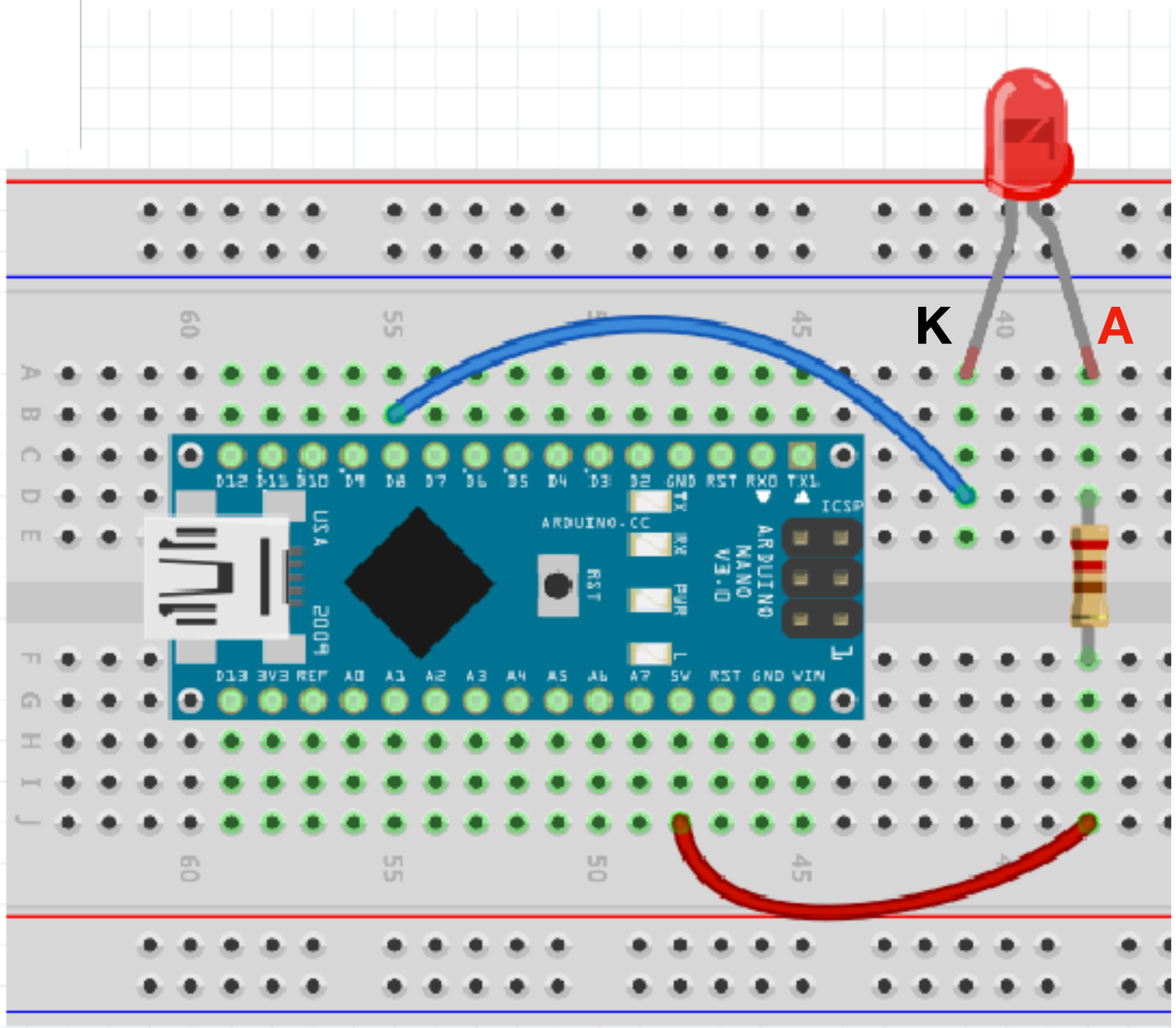
VCC para VCC (0)

digitalWrite

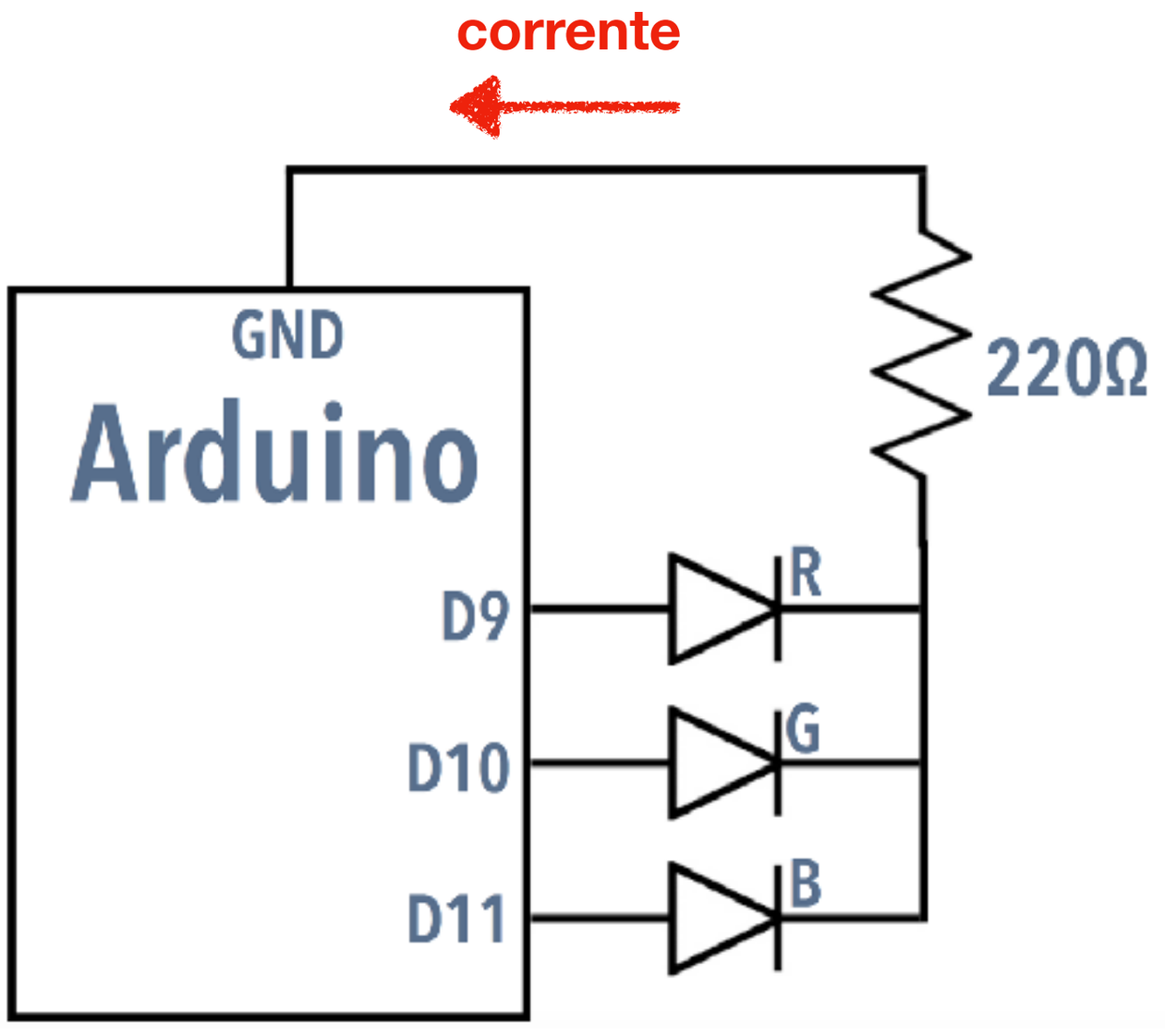


"Blink"
LED acende quando pino 8 for LOW
LED apaga quando pino 8 for HIGH

```
void setup() {
    pinMode(8, OUTPUT);
}

void loop() {
    digitalWrite(8, HIGH);
    delay(50);
    digitalWrite(8, LOW);
    delay(50);
}
```

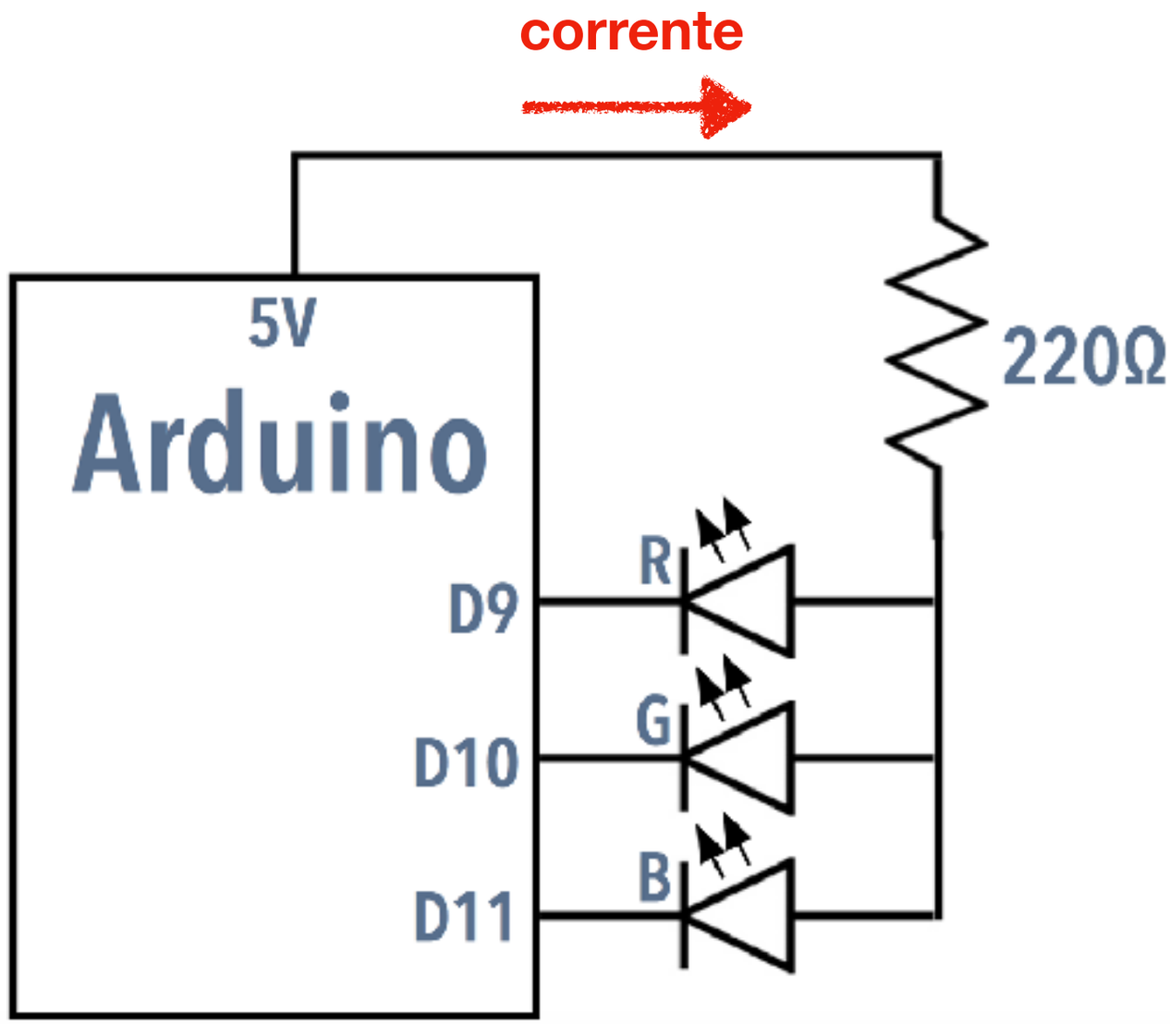


Usando múltiplos pinos



Pino-GND

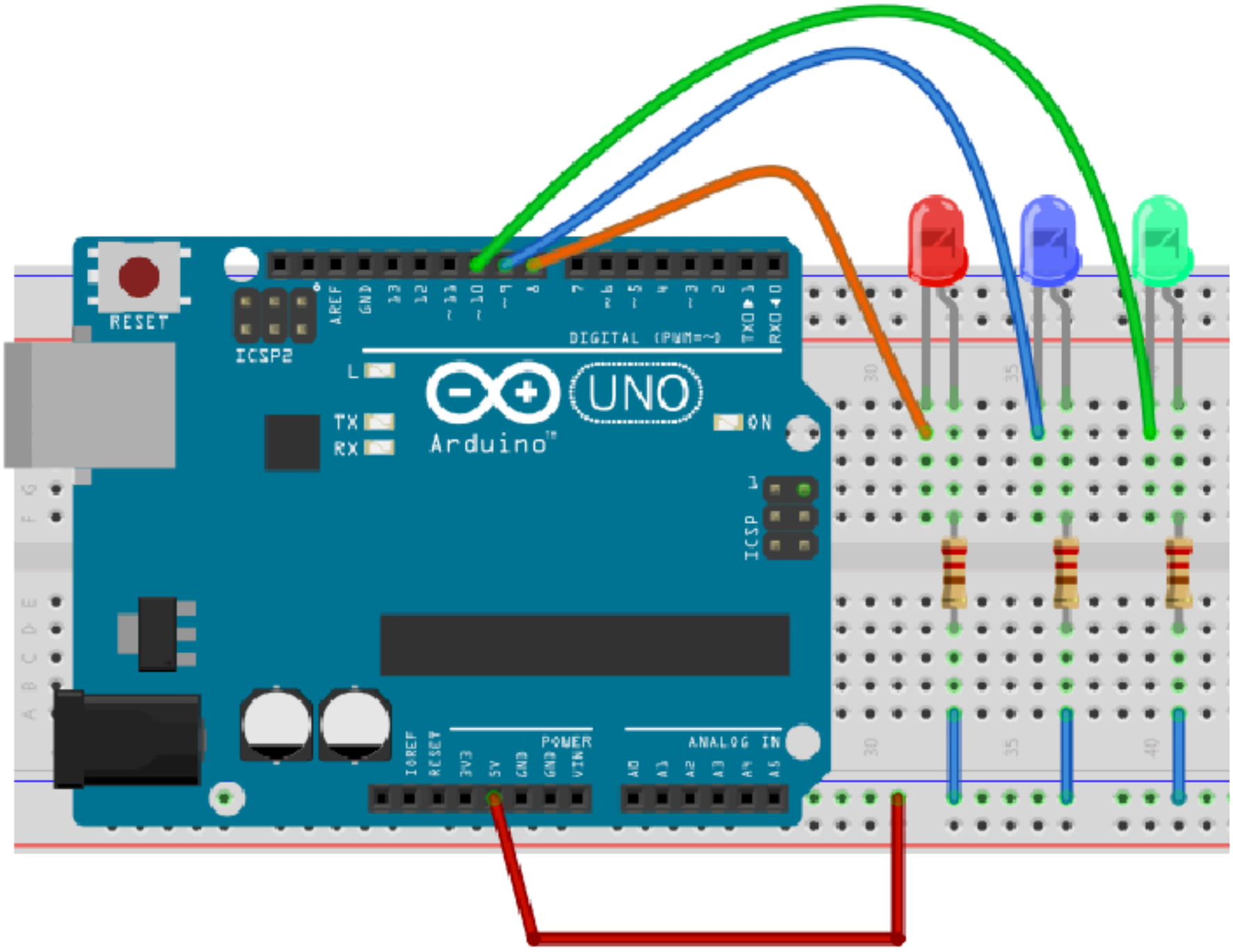
Catodo comum
ligado em GND



Pino-VCC

Anodo comum
ligado em VCC

Vários LEDs (anodo comum)



Anodo comum: LED acende quando pino em LOW (VCC é sempre HIGH)

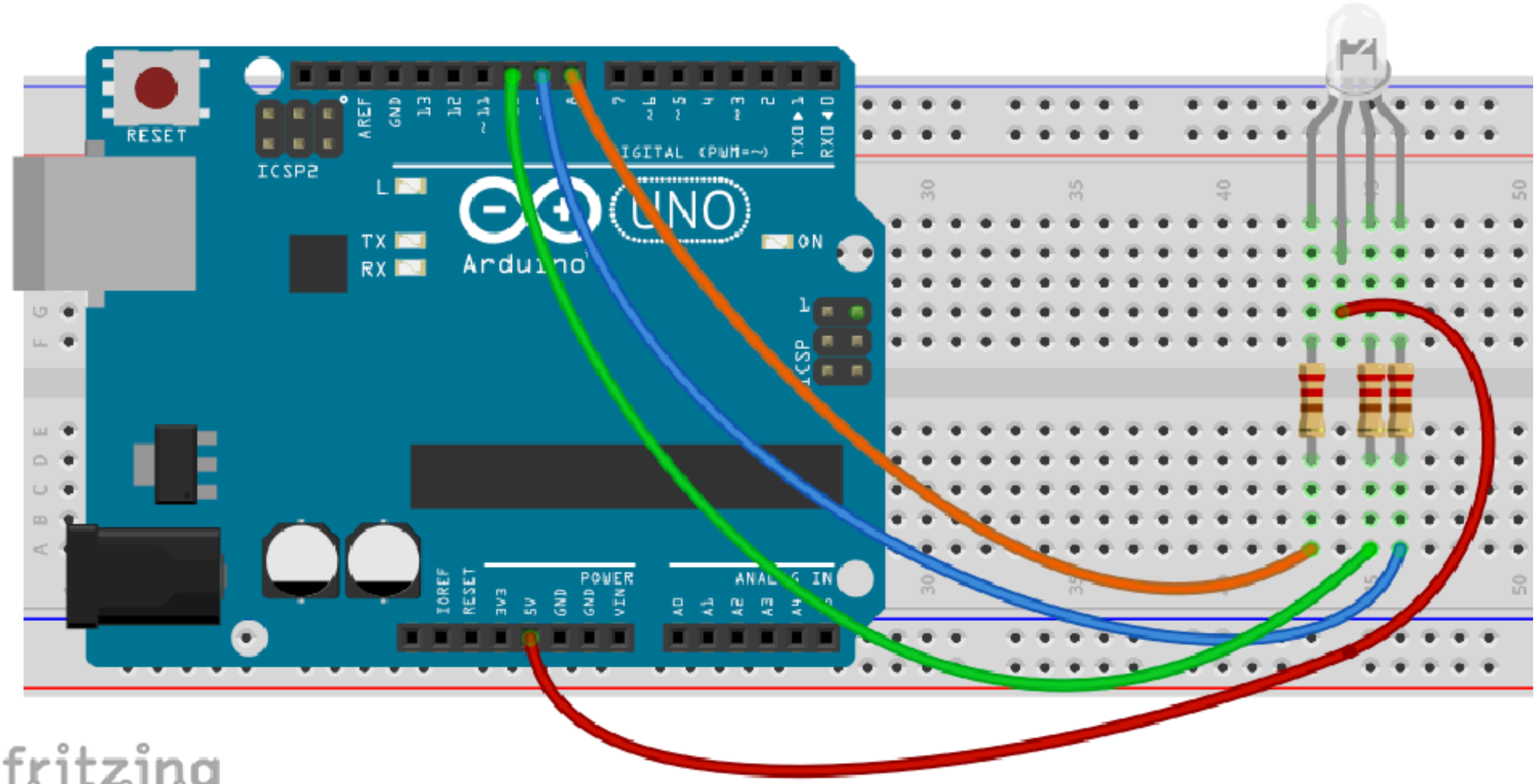
LEDs em anodo comum

```
int R = 8;
int B = 9;
int G = 10;

void setup() {
    pinMode(R, OUTPUT);
    pinMode(B, OUTPUT);
    pinMode(G, OUTPUT);
}

void loop() {
    digitalWrite(R, LOW);  // acende vermelho
    digitalWrite(B, HIGH);
    digitalWrite(G, HIGH);
    delay(1000);
    digitalWrite(R, HIGH);
    digitalWrite(B, LOW);  // acende azul
    digitalWrite(G, HIGH);
    delay(1000);
    digitalWrite(R, HIGH);
    digitalWrite(B, HIGH);  // acende verde
    digitalWrite(G, LOW);
    delay(1000);
}
```

Nesta configuração, LEDs **acendem** quando pino está em **LOW.**

Se LEDs forem montados em **catodo comum**, os catodos são ligados em GND e os LEDs **acendem** quando seu pino estiver em **HIGH**

LED RGB (anodo comum)



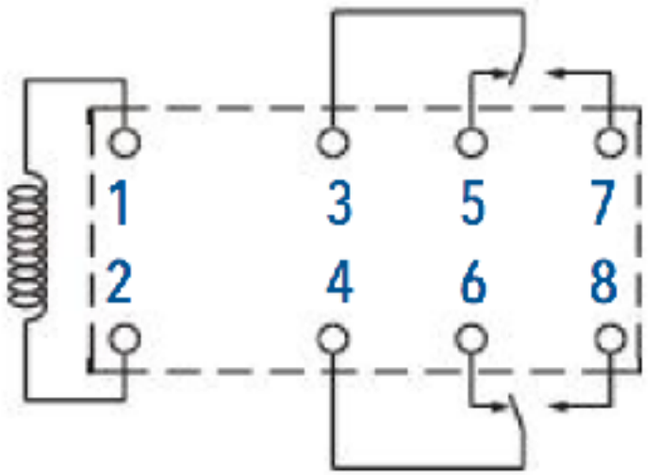
Anodo comum: LED acende quando pino em LOW (VCC é sempre HIGH)

Relé (chave eletromecânica)

8 pinos

2 pinos: eletroimã (circuito do Arduino): 5V

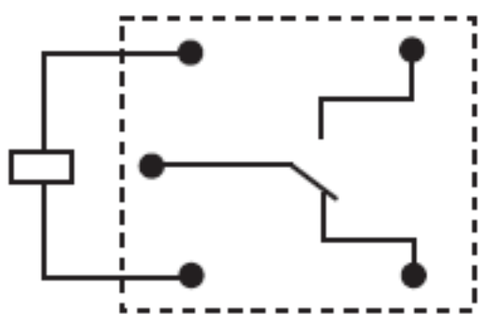
6 pinos: 2 chaves de 2 posições (circuitos independentes): até 120V / 1A



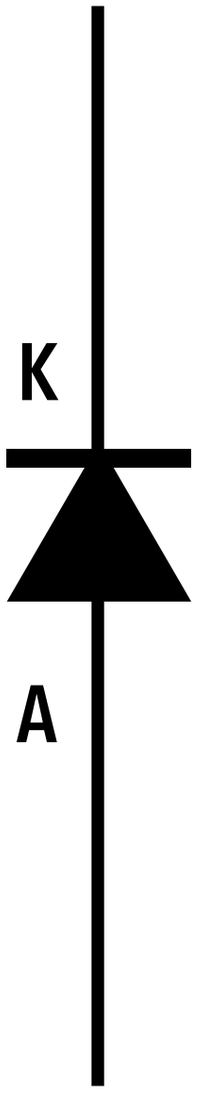
5 pinos

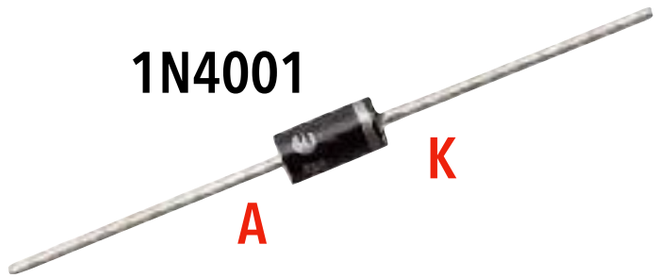
2 pinos: eletroimã (circuito do Arduino): 5V

3 pinos: chave de 2 posições (circuito independente): até 240V / 7A



Diodos





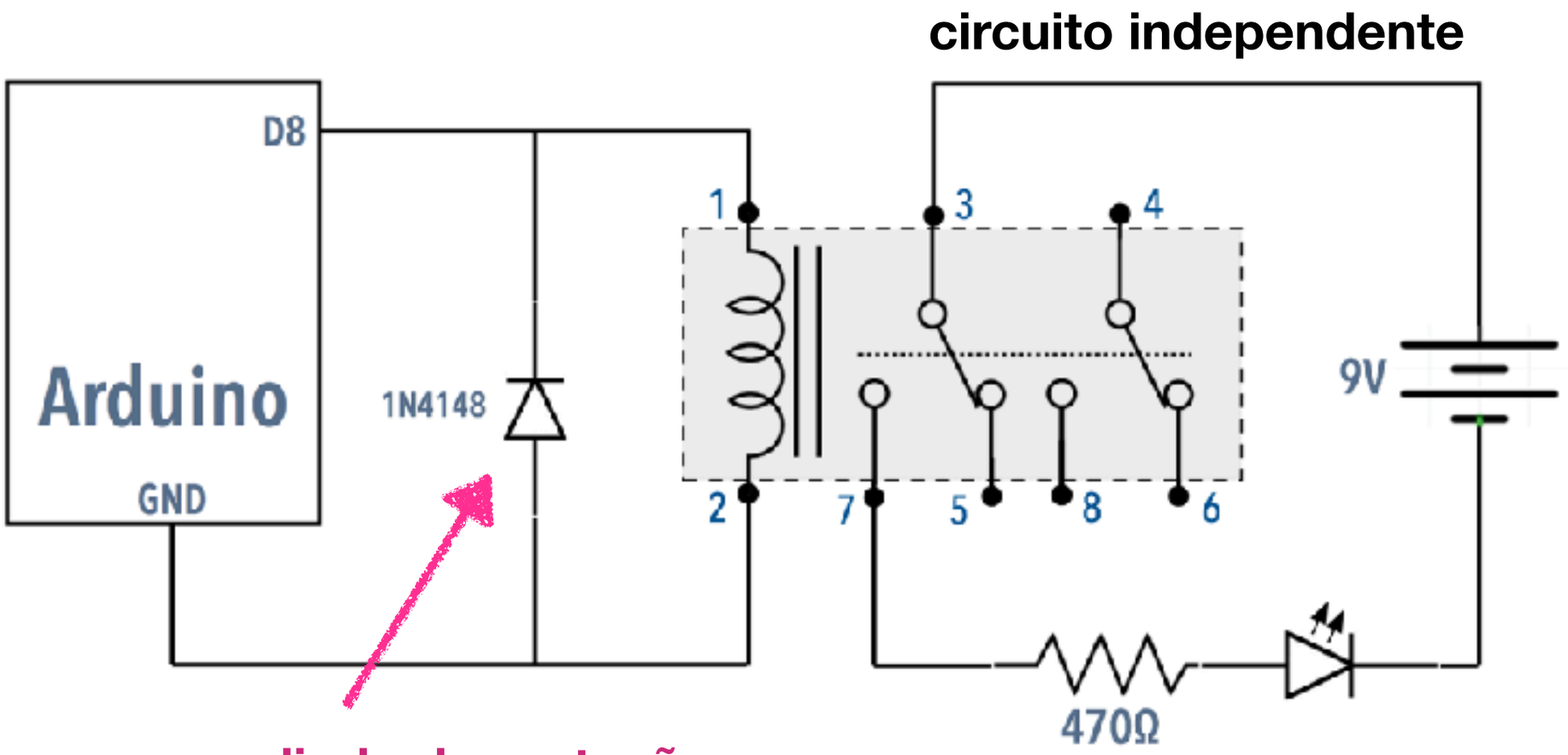
Os diodos do kit podem ser 1N4001 ou 1N4148

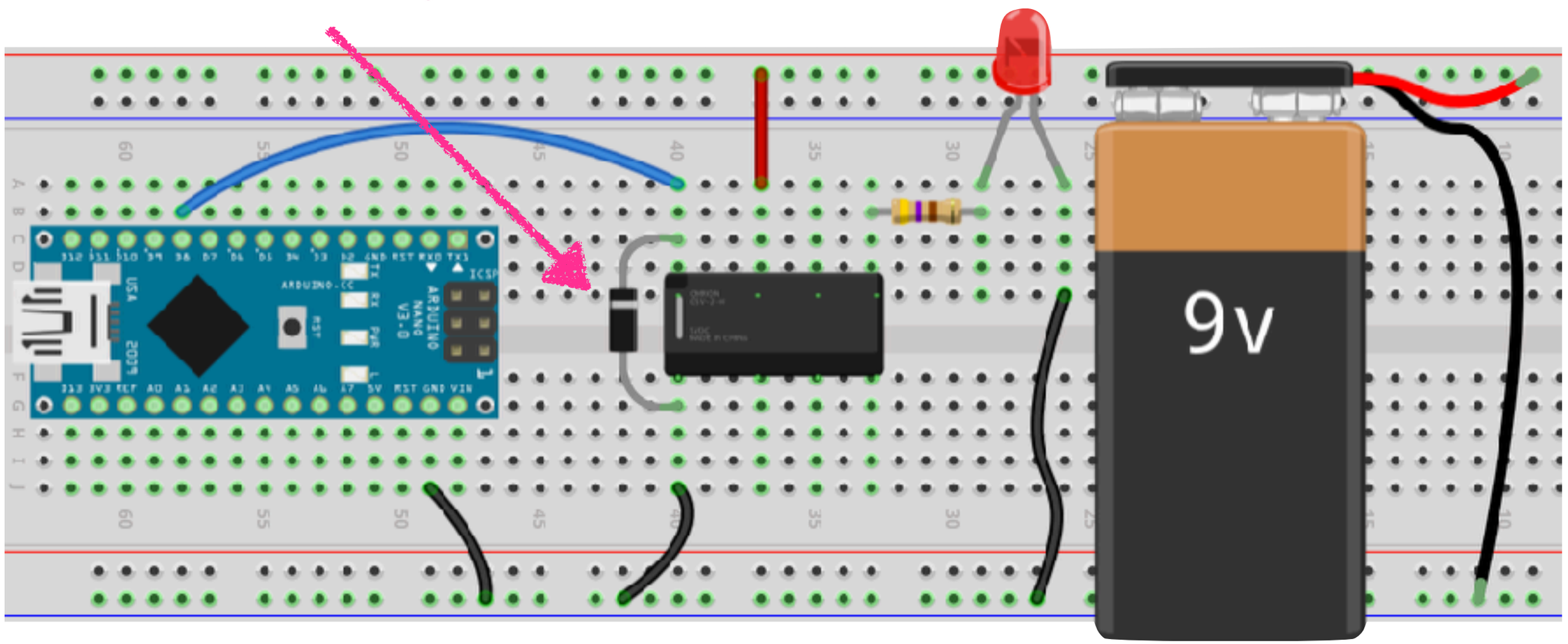
Permite a passagem de corrente apenas em um sentido (não conduz no outro sentido)

Diodos de propósito geral são usados para em retificadores de corrente (corta correntes negativas), circuitos de proteção (contra correntes armazenadas em motores e relés)

Troque o LED por um relé

(use o mesmo programa que pisca o LED)





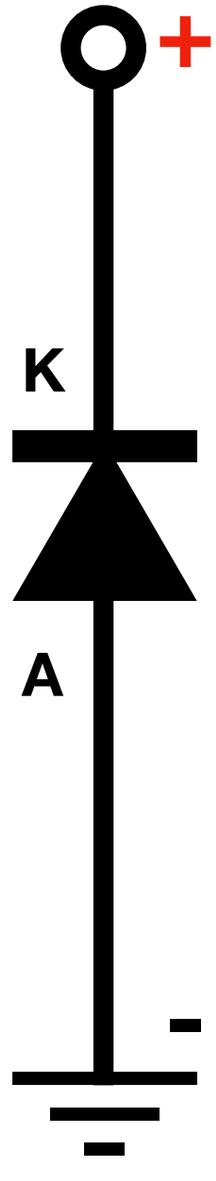
O diodo de proteção

- Relés (e indutores em geral - inclui motores, solenóides) **armazenam corrente** que fluem no sentido oposto quando componente é desligado

- Para evitar que essa corrente oposta volte para o Arduino através do pino GND (podendo causar danos), deve-se colocar um **diodo de proteção** em posição reversa e em paralelo com os terminais do relé.

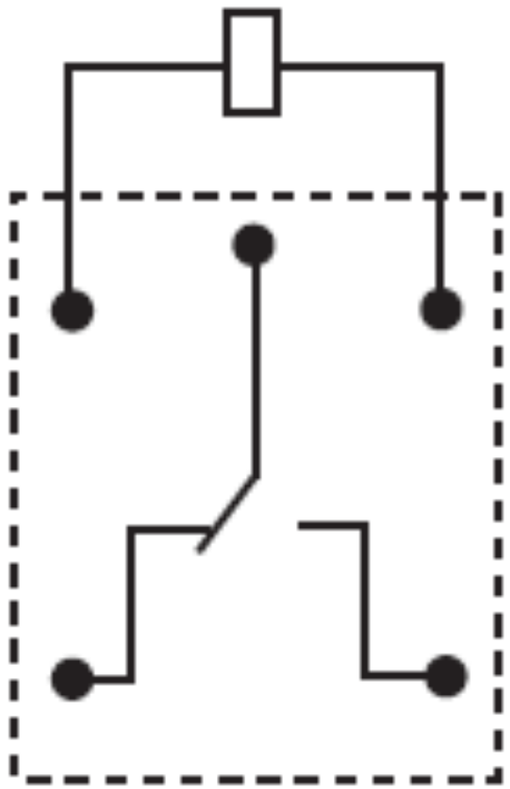
- O diodo só permite a passagem da corrente em um sentido, e anulará correntes reversas com um curto no indutor.

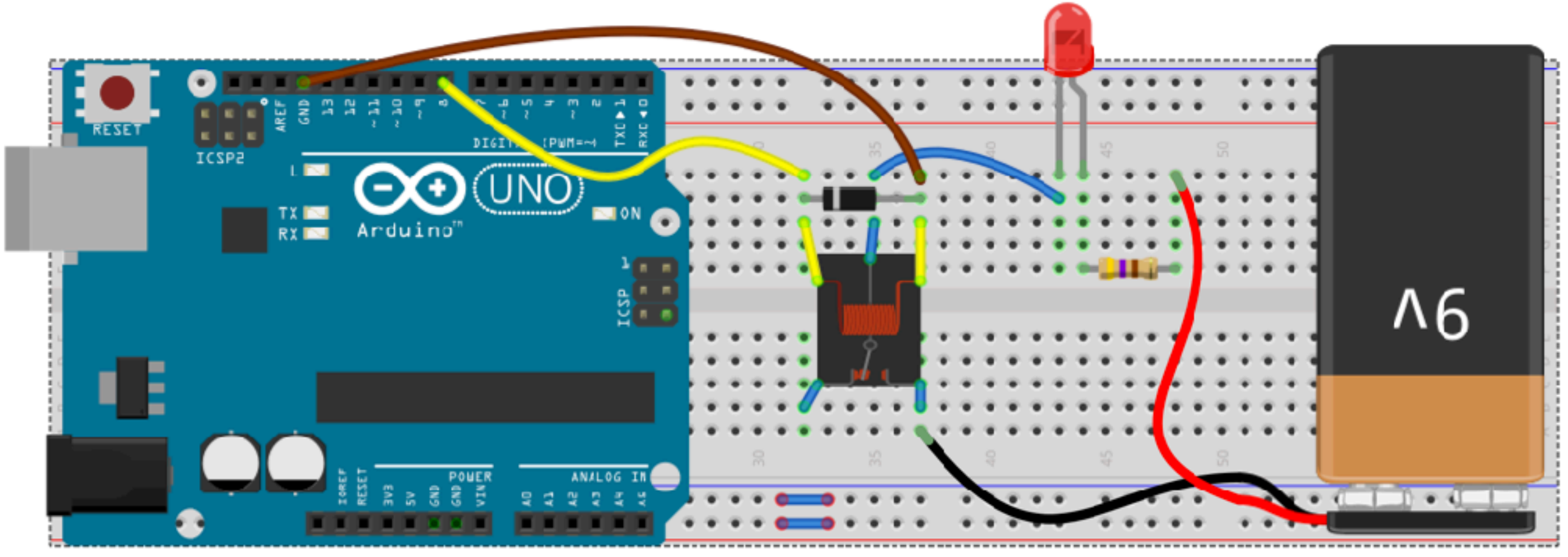
- IMPORTANTE: **o diodo precisa estar em posição REVERSA**, com o catodo (K) do lado positivo e o anodo (A) no negativo. Se a configuração for Pino-GND, ligue A em GND e K no pino. Na configuração Pino-VCC, ligue K em VCC/5V e A no pino. **O diodo em posição direta causará um curto e queimará o pino.**

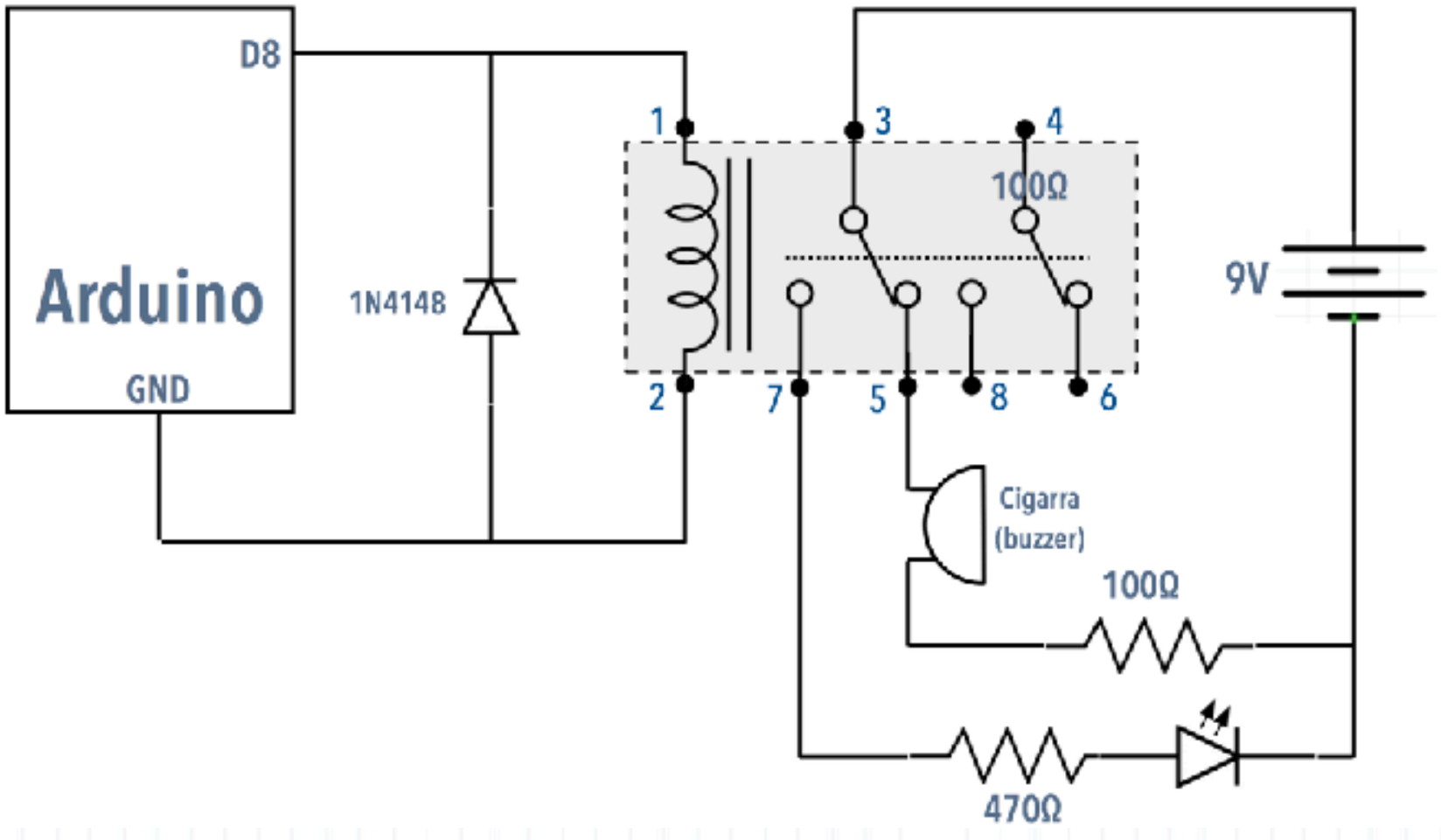


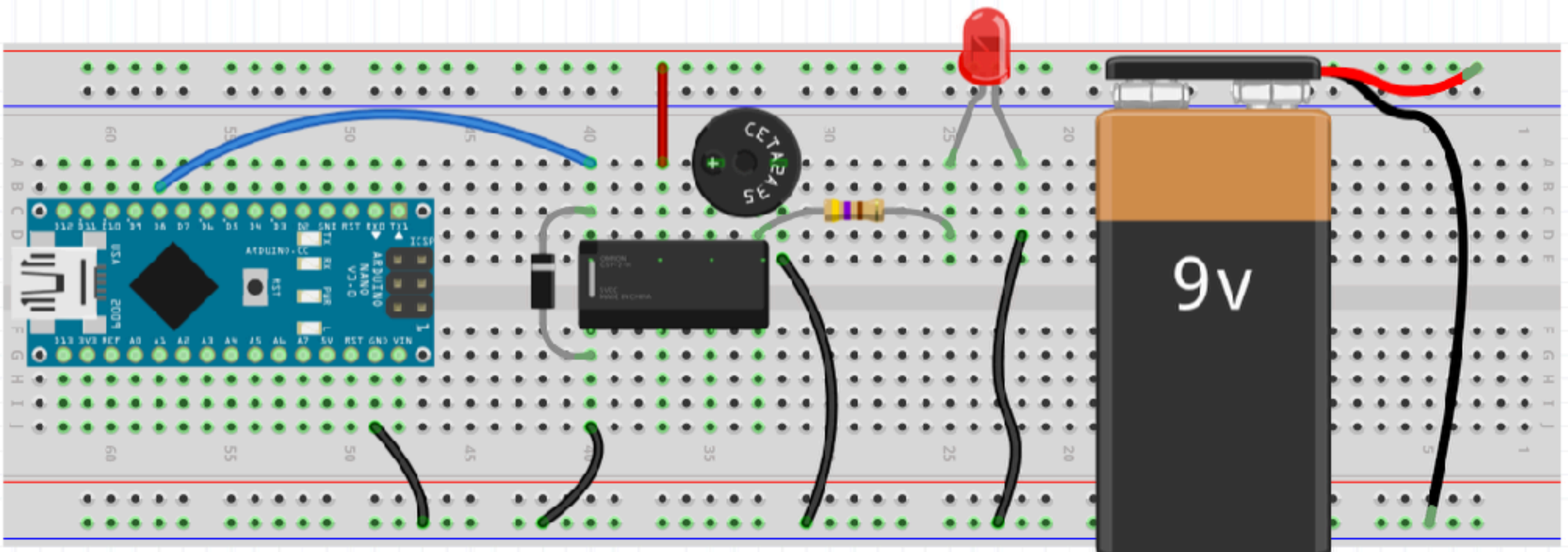
Usando um relé de 5 pinos

(1 circuito, 2 posições)



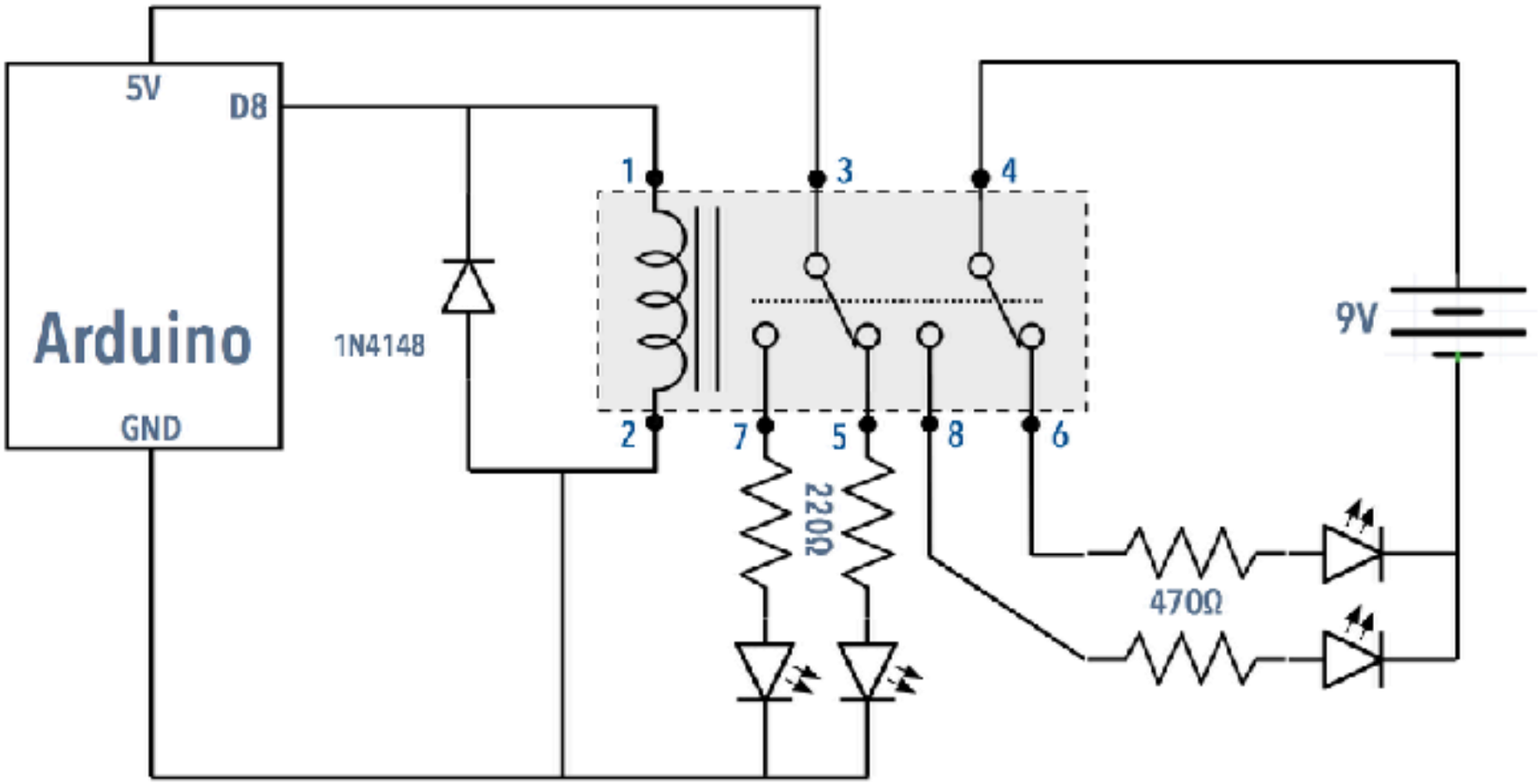


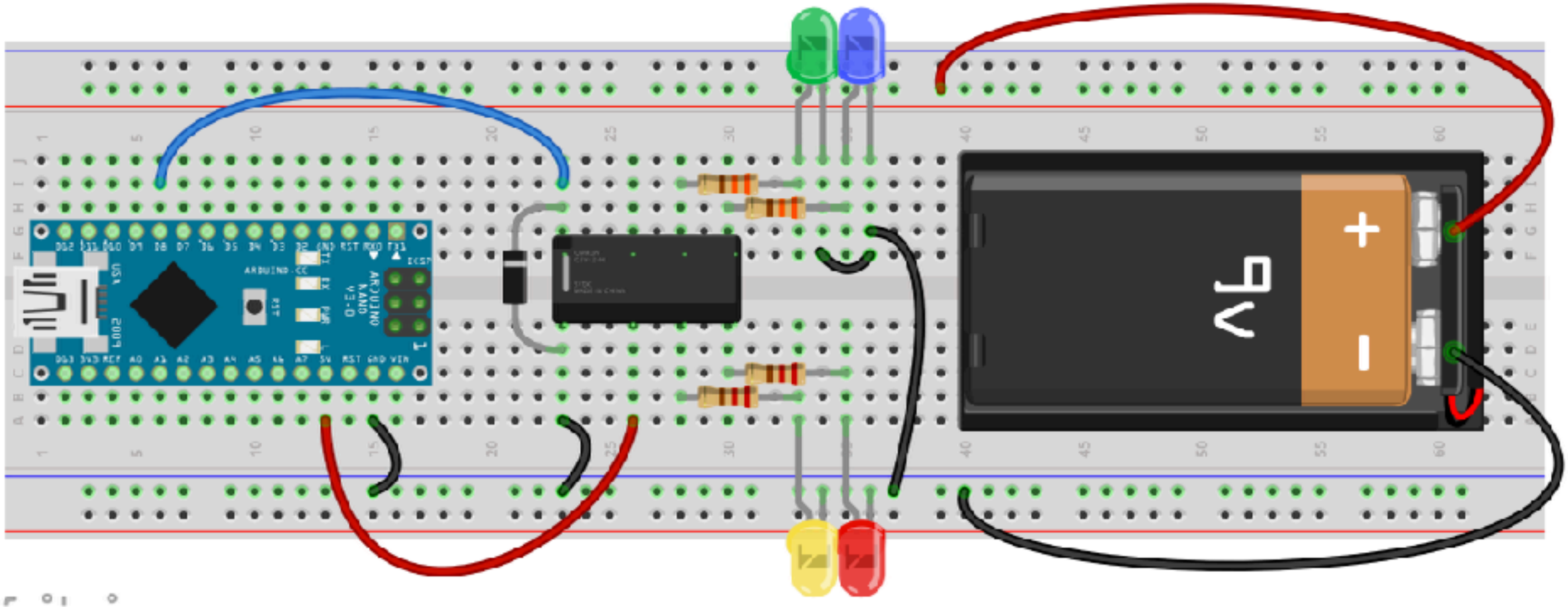




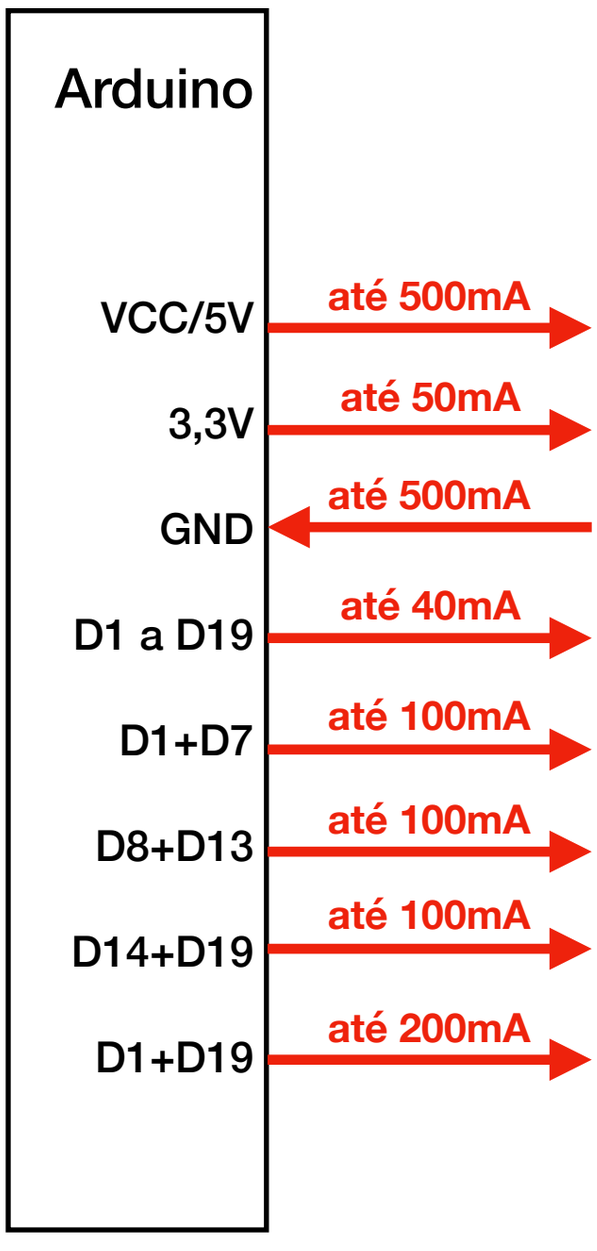
Relé chaveando duas posições

Relé chaveando 2 circuitos em 2 posições





Níveis lógicos (saída)



Nível lógico em pino configurado com pinMode OUTPUT é traduzido para uma tensão de valor VCC (5V ou 3,3V) ou 0V.

Para usar a tensão produzida por um pino OUTPUT em um circuito é preciso observar os limites de corrente*:

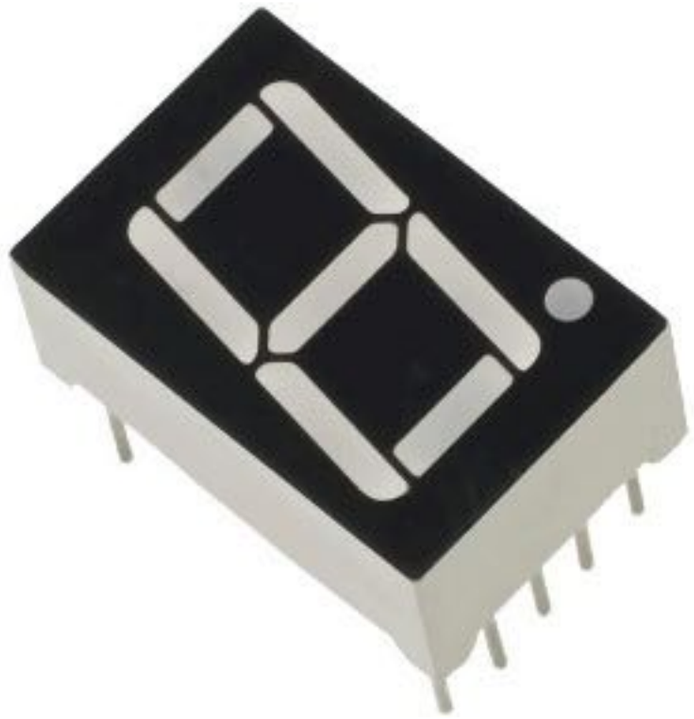
- **~40mA para qualquer pino digital**
- **~100mA para pinos de um grupo (D1-D7, D8-D13, A0-A5)**
- **~200mA para o Arduino inteiro (vários pinos)**
- **~50mA para o pino 3,3V (nos modelos de 5V)**
- **~500mA para o pino VCC/5V ou GND (limite USB)**

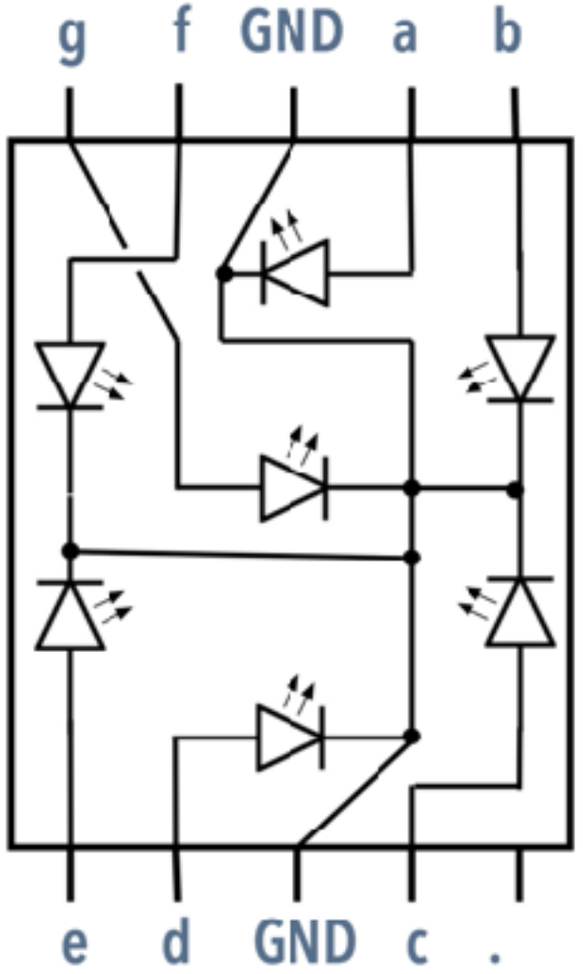
ATENÇÃO: Ligar OUTPUT com nível LOW diretamente a VCC, ou OUTPUT com nível HIGH diretamente a GND sem limitar a corrente produz um curto-circuito que queima o pino.

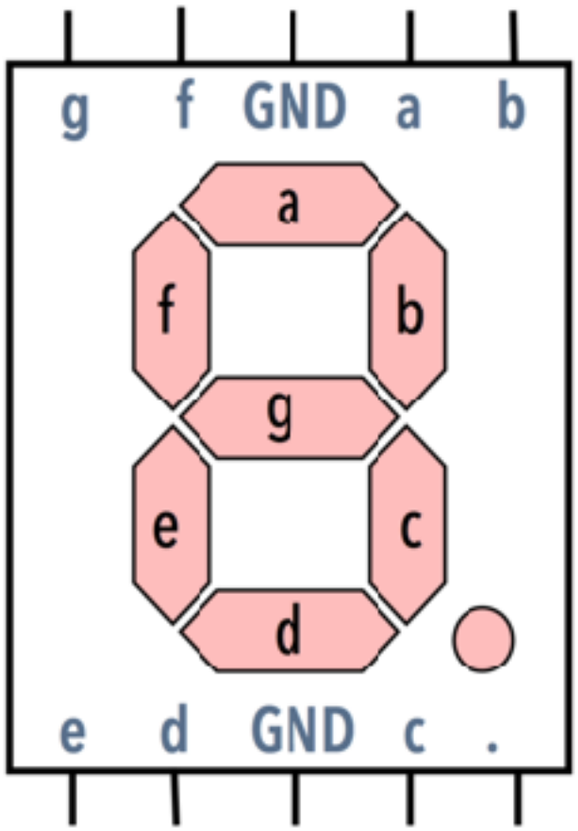
Use um resistor calculado para limitar a corrente demandada pelo circuito de saída quando a diferença de potencial entre o pino e VCC ou GND for máxima (5 volts)

* valores típicos do Uno - valores são diferentes em outros modelos e clones

Display de LEDs de 7 segmentos





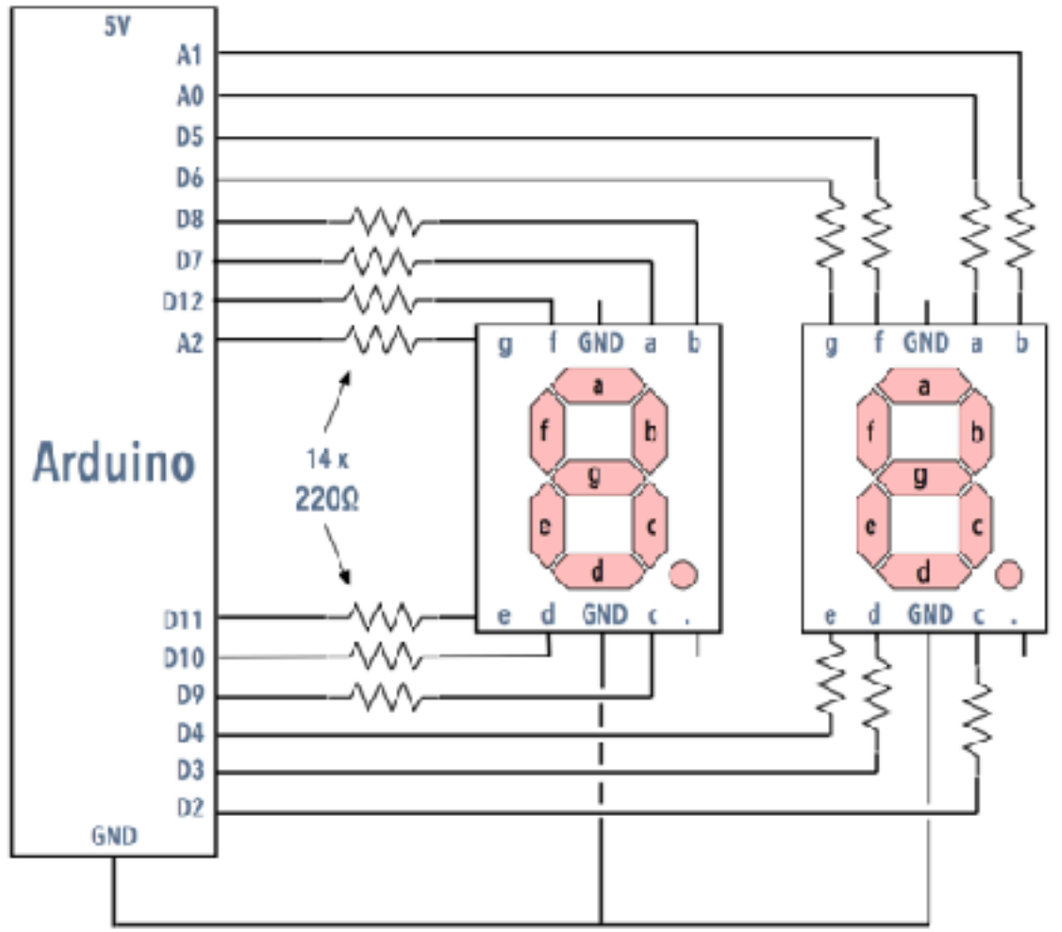


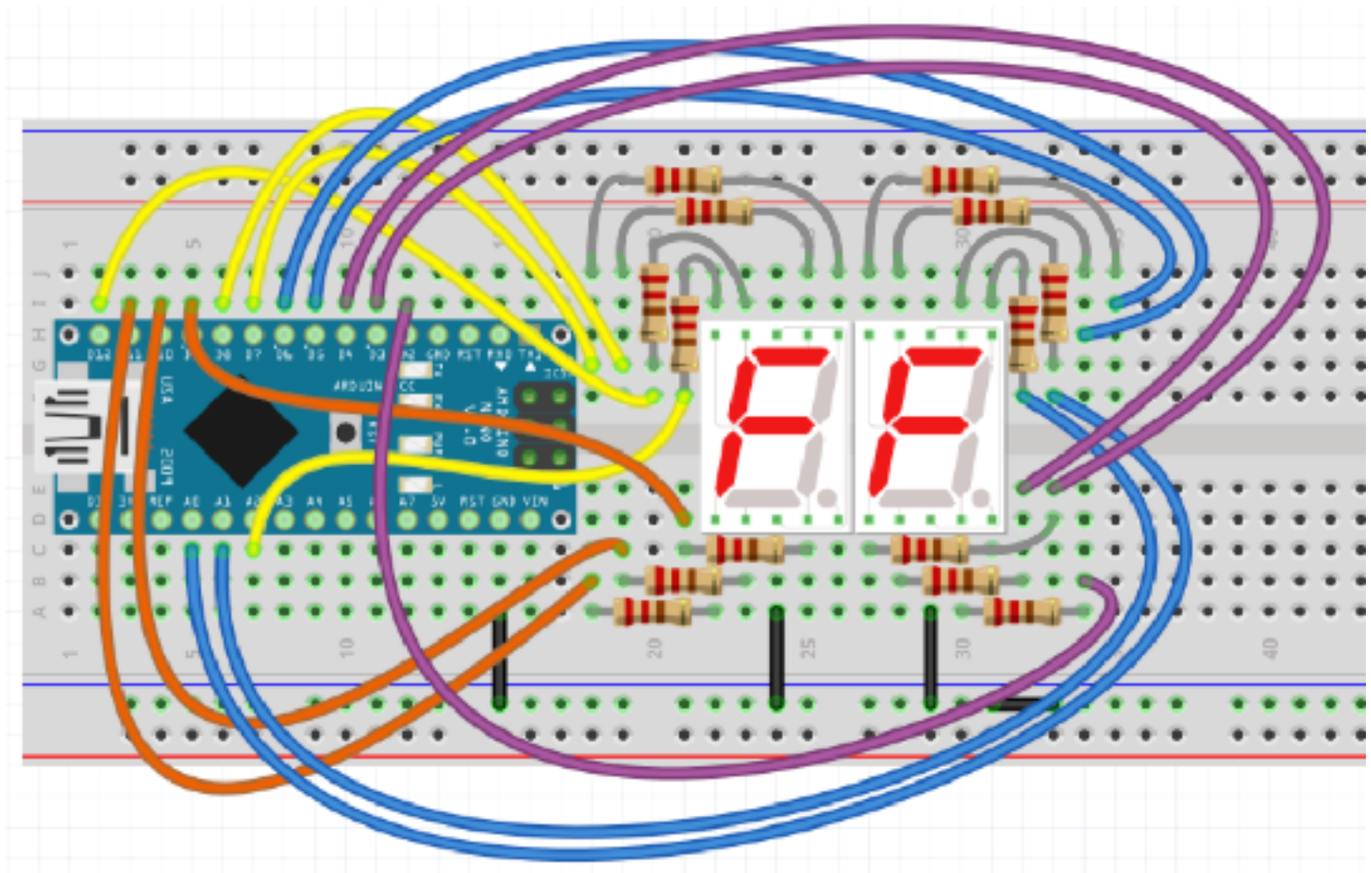
O modelo acima (HS 5101A) possui um **catodo comum**

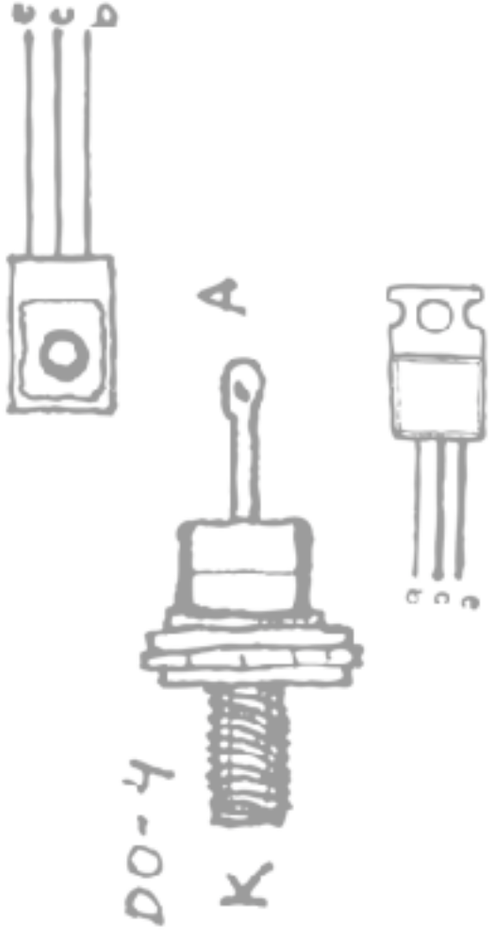
```ino
int unidades[]  = {14, 15, 2, 3, 4, 5, 6};
int dezenas[]   = {7, 8, 9, 10, 11, 12, 16};

static int digito[10][7] = {
    {HIGH,HIGH,HIGH,HIGH,HIGH,HIGH,LOW},
    {LOW,HIGH,HIGH,LOW,LOW,LOW,LOW},
    {HIGH,HIGH,LOW,HIGH,HIGH,LOW,HIGH},
    {HIGH,HIGH,HIGH,HIGH,LOW,LOW,HIGH},
    {LOW,HIGH,HIGH,LOW,LOW,HIGH,HIGH},
    {HIGH,LOW,HIGH,HIGH,LOW,HIGH,HIGH},
    {HIGH,LOW,HIGH,HIGH,HIGH,HIGH,HIGH},
    {HIGH,HIGH,HIGH,LOW,LOW,LOW,LOW},
    {HIGH,HIGH,HIGH,HIGH,HIGH,HIGH,HIGH},
    {HIGH,HIGH,HIGH,HIGH,LOW,HIGH,HIGH}
};

const int DISPLAY_LEDS = 7;
const int DISPLAY_DIGITS = 10;

void setup() {
  for(int i = 0; i < DISPLAY_LEDS; i++) {
    pinMode(dezenas[i], OUTPUT);
    pinMode(unidades[i], OUTPUT);
  }
}

void acende(int* disp, int dig) {
  for(int i = 0; i < DISPLAY_LEDS; i++) {
    digitalWrite(disp[i], digito[dig][i]);
  }
}

void loop() {
  for(int j = 0; j < DISPLAY_DIGITS; j++) {
    acende(dezenas, j);
    for(int i = 0; i < DISPLAY_DIGITS; i++) {
        acende(unidades, i);
        delay(500);
    }
  }
}
```

Display de LEDs de 7 segmentos

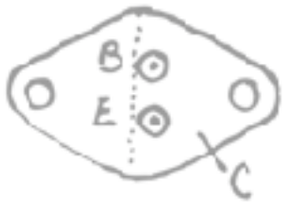
Produz uma contagem de 00 a 99







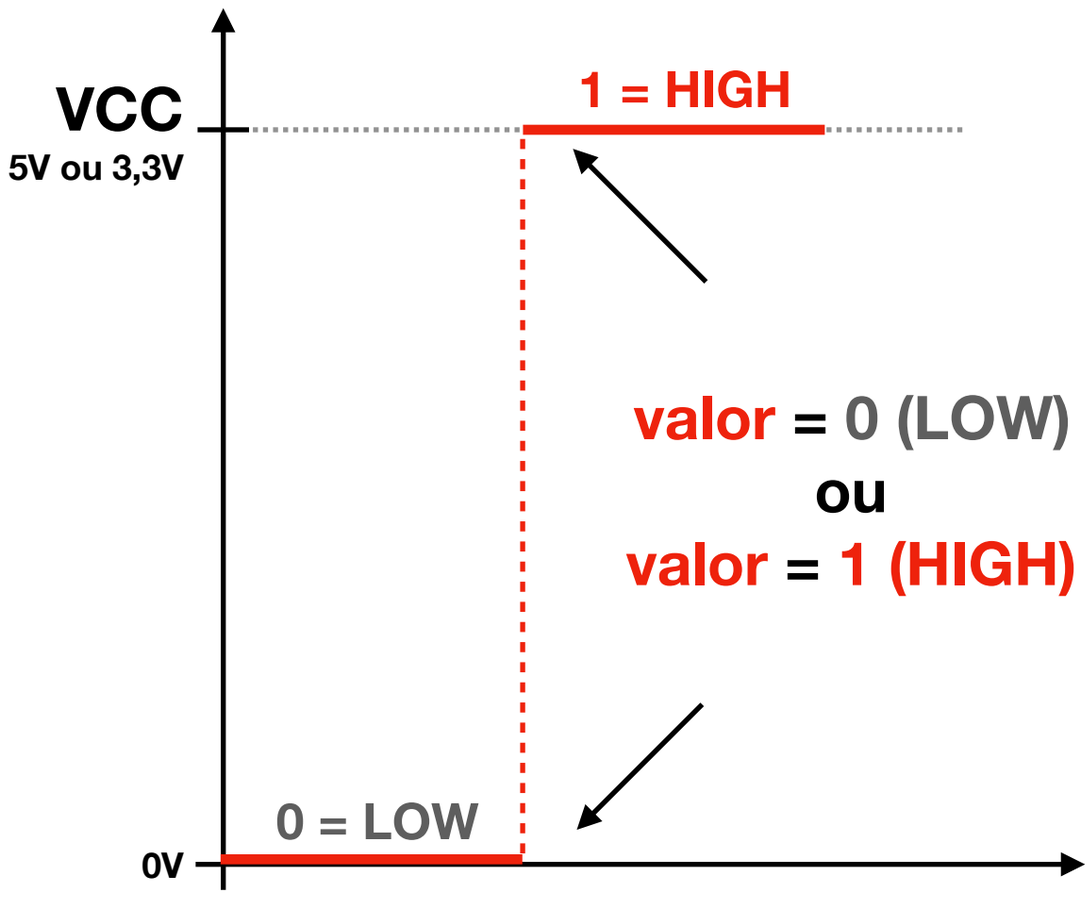
ARDUINO 3



Entrada digital

Helder da Rocha
www.eletronicaparaartistas.com.br

int valor = digitalRead(pino)



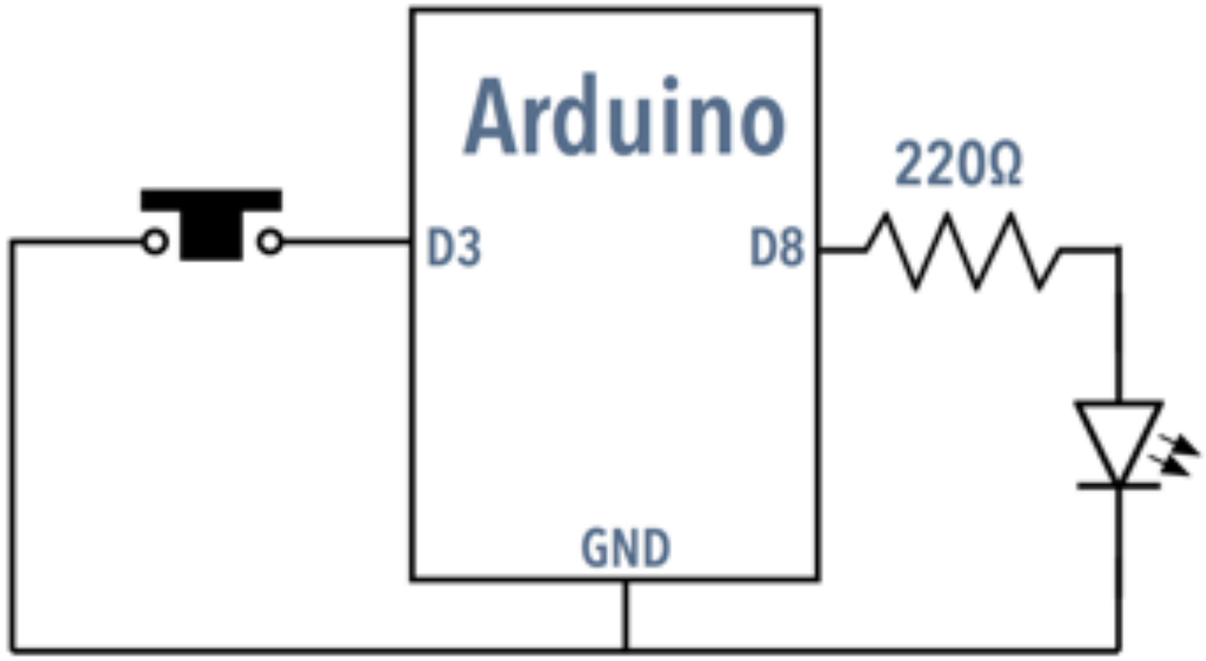
VCC representa o nível lógico HIGH do Arduino (5V ou 3,3V dependendo do tipo)

Se VCC for 5V (Arduino Uno)
LOW = 0V, HIGH = 5V

Se VCC for 3,3V (Due, LilyPad, ProMini)
LOW = 0V, HIGH = 3,3V

Não há como mudar o nível VCC do Arduino. Ele é fixo. Cada pino em nível HIGH terá esse valor.

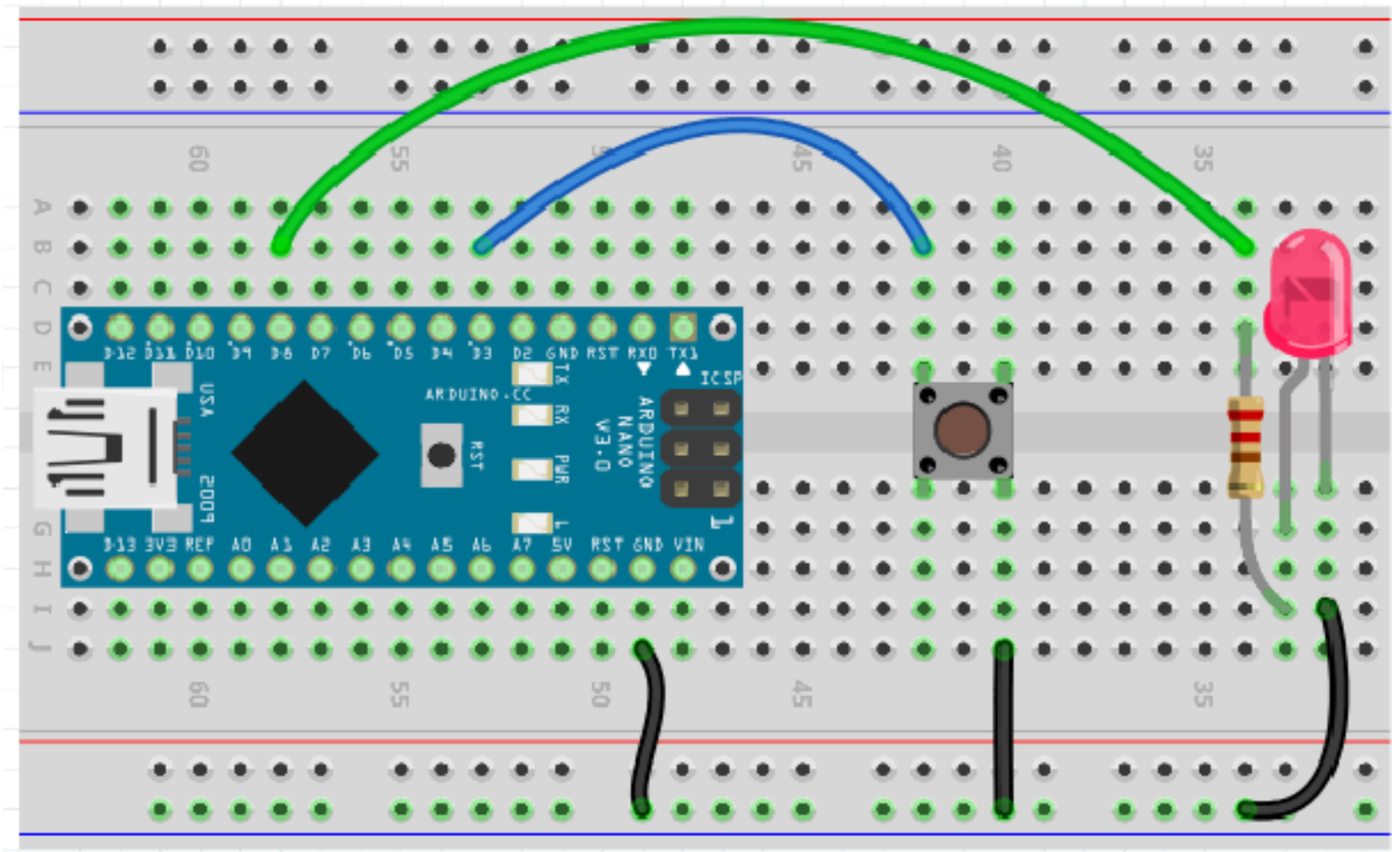
Para conectar componentes com níveis lógicos incompatíveis (3,3V x 5V) é preciso usar um conversor de nível lógico (pode ser um divisor de tensão)



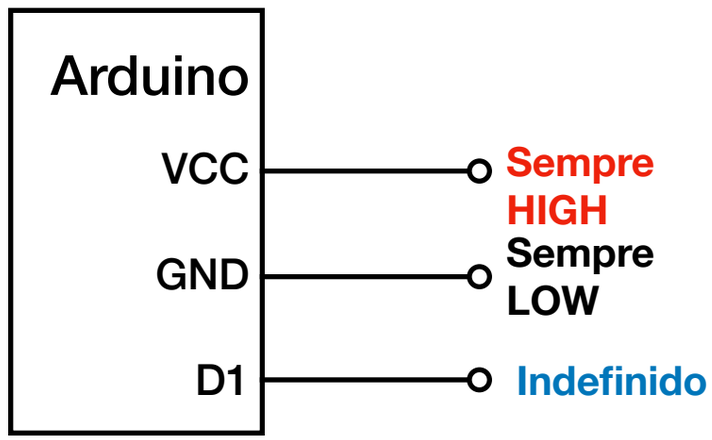
digitalRead

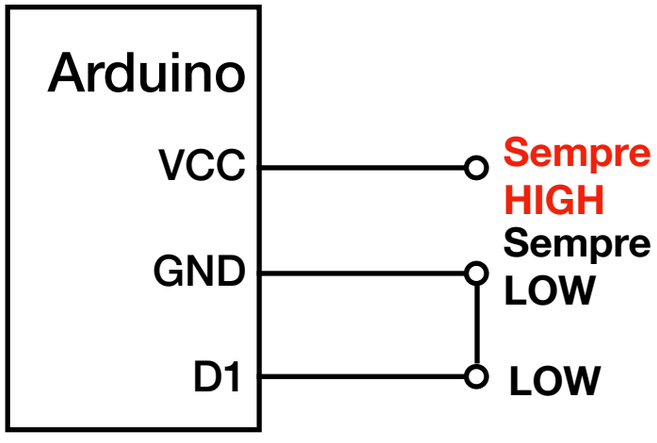
Estado do pino 3 quando chave estiver aberta é INDEFINIDO (pode se traduzido como HIGH ou LOW, ou ficar mudando)

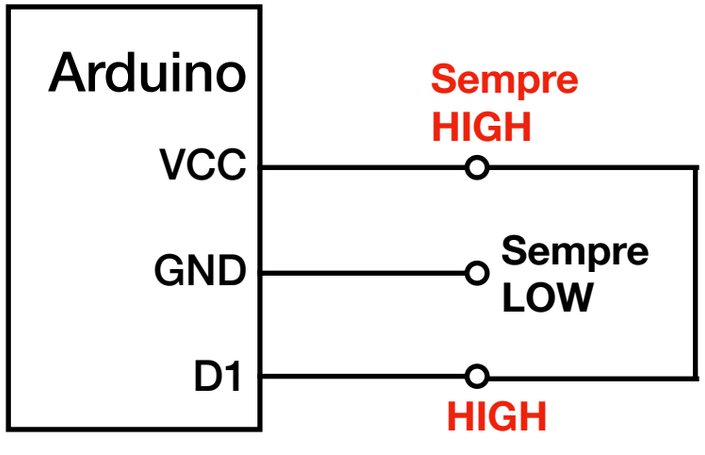
```
void setup() {
  pinMode(8, OUTPUT);
}

void loop() {
  int estado = digitalRead(3);
  delay(10);
  if(estado == LOW) {
    digitalWrite(8, HIGH);
  } else {
    digitalWrite(8, LOW);
  }
}
```



Níveis lógicos (entrada)







Pino VCC (Pino 5V em Arduino Uno/Nano) sempre fornece nível lógico HIGH

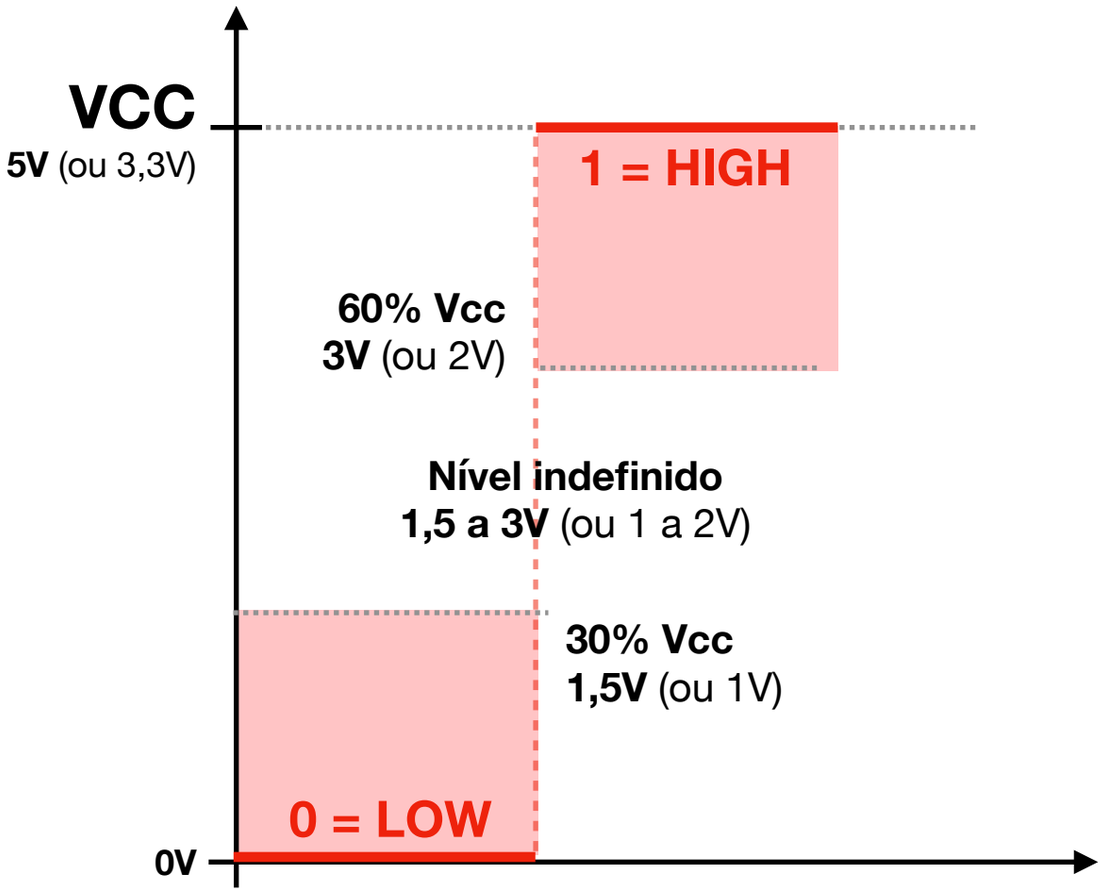
Pino GND sempre fornece nível lógico LOW

Pinos digitais D1 a D19 têm níveis lógicos indefinidos (podem ser interpretados como HIGH ou LOW)

Pino digital configurado como INPUT (default) pode ser ligado a VCC (HIGH) ou GND (LOW) para ter valor definido

Ligação pode ser direta (fio) ou via resistor de pull-up (para VCC) ou pull-down (para GND)

Níveis lógicos reais (entrada)



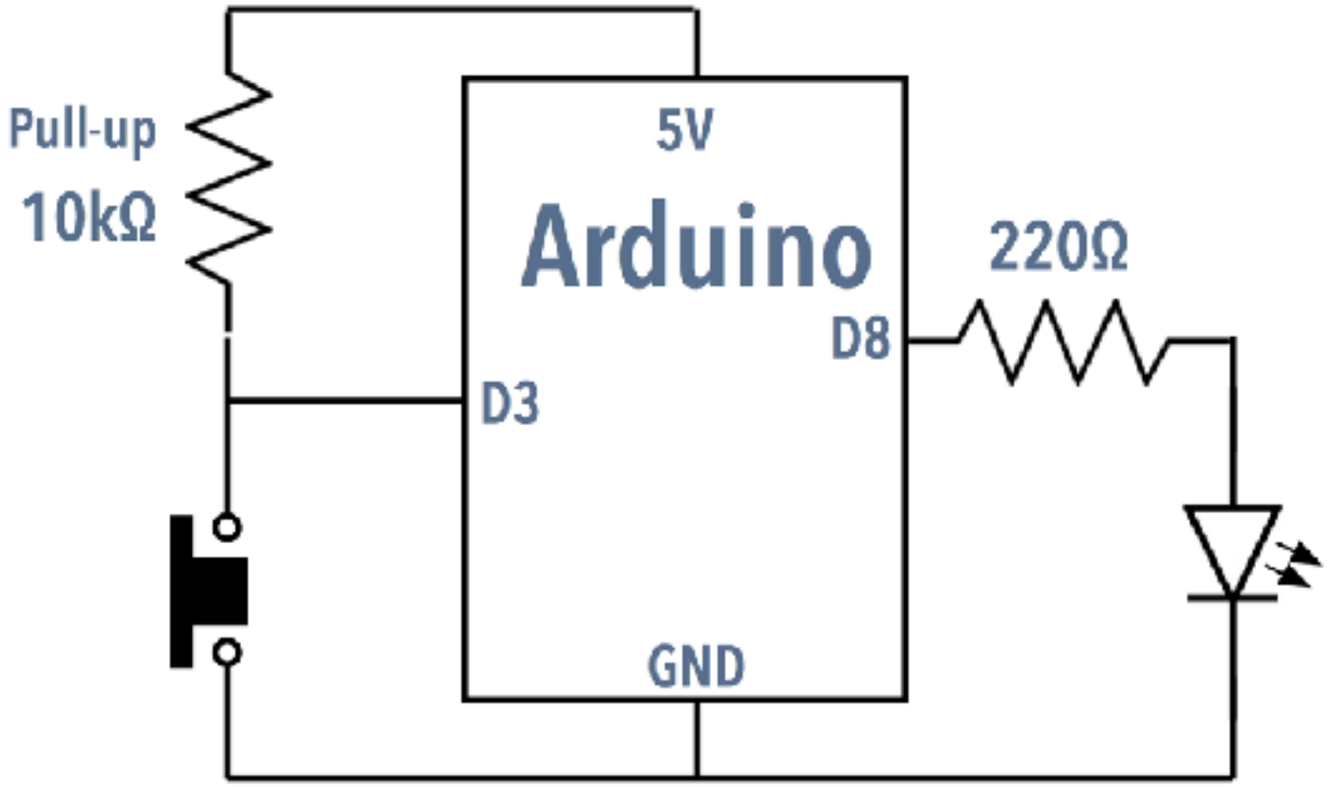
Um valor de tensão é traduzido para o nível lógico 1 (HIGH) se for maior que 60% da tensão VCC do Arduino

Um valor de tensão é traduzido para o nível lógico 0 (LOW) se for menor que 30% da tensão VCC do Arduino

Valores intermediários são indefinidos e serão traduzidos para qualquer um dos dois níveis lógicos (não há como prever)

Pull-up e Pull-down

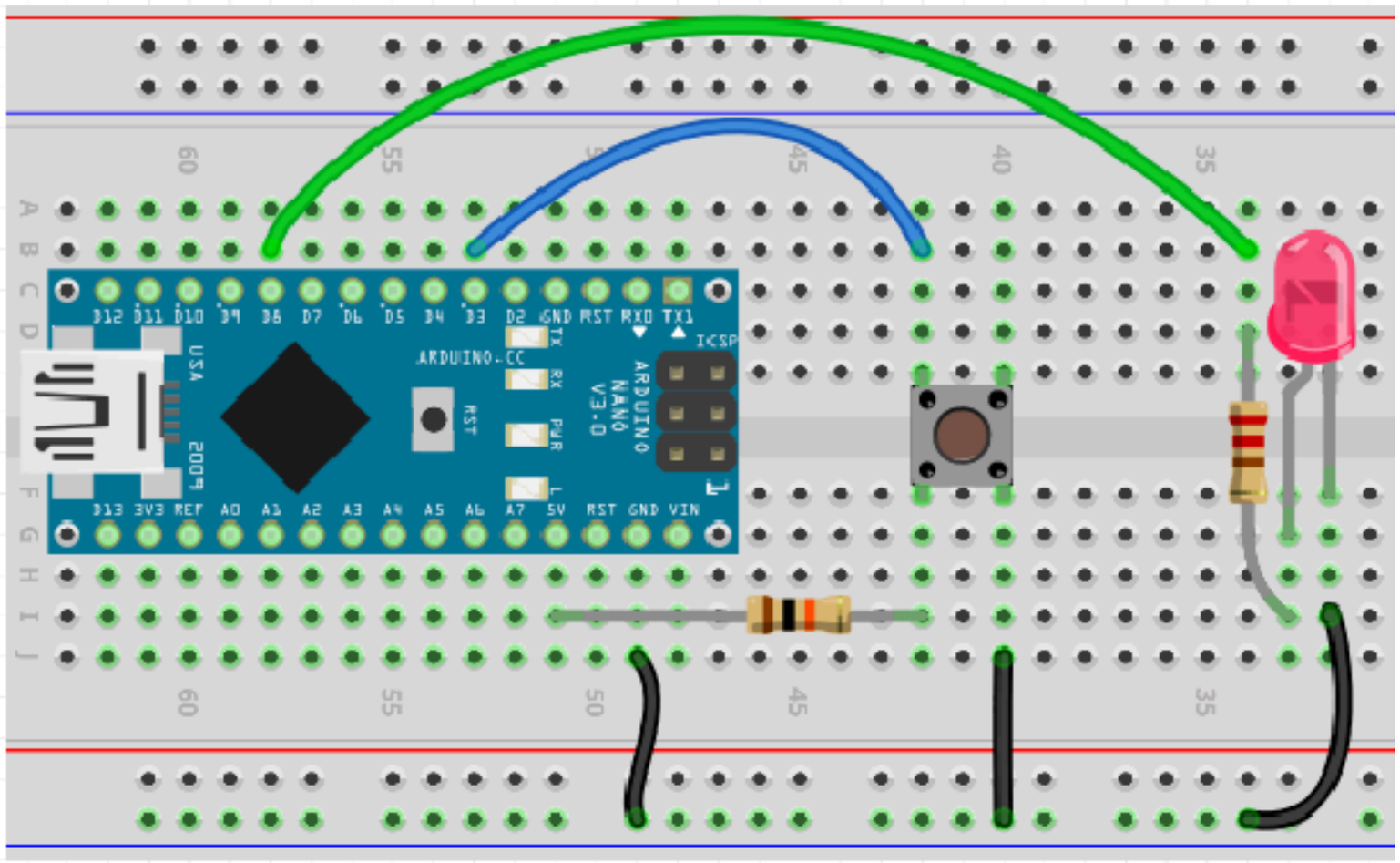
Resistores de pull-down (ou pull-up) garantem estado inicial definido quando a chave estiver aberta



Pino 3 com um resistor de **pull-up**

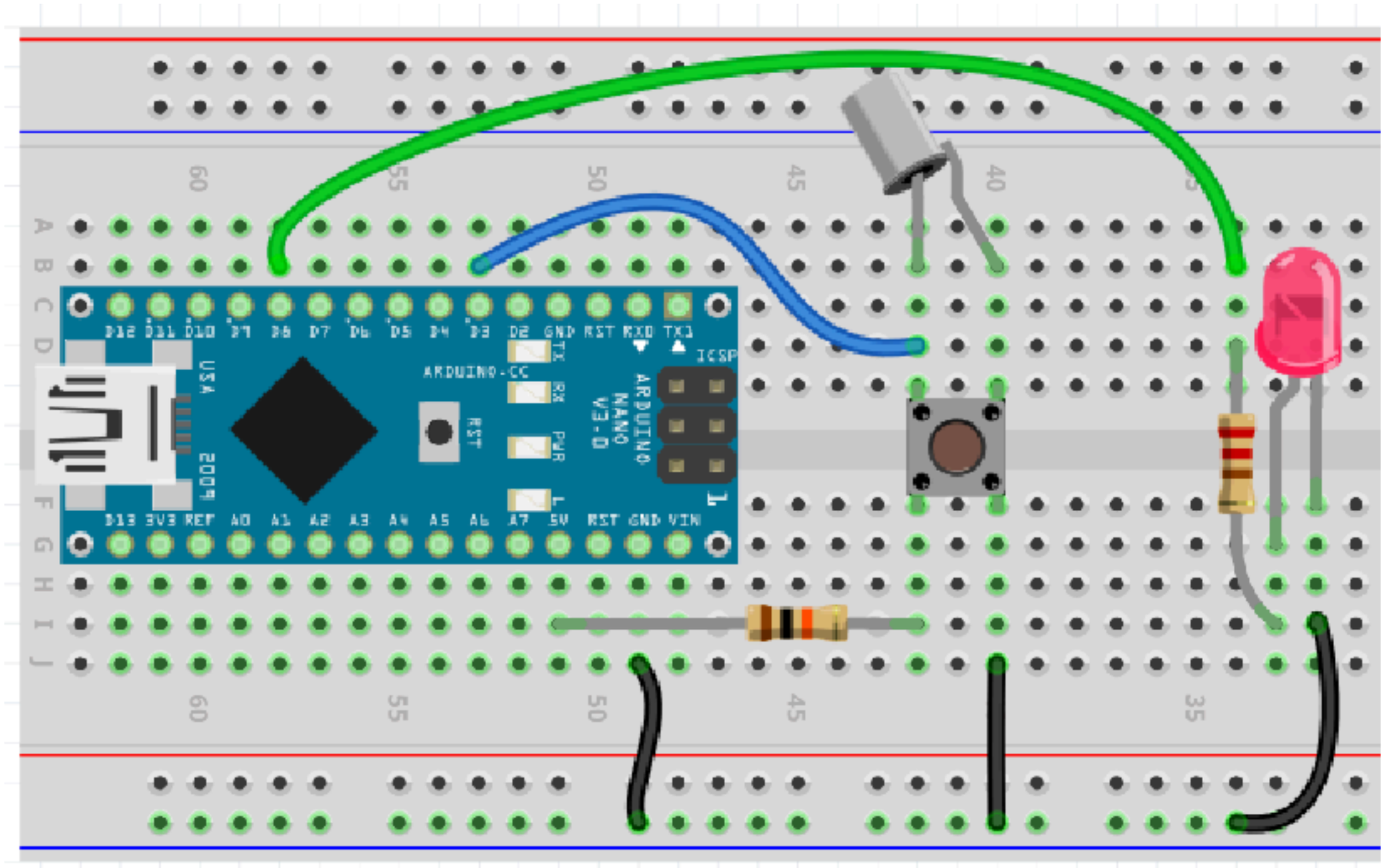
Agora o valor do pino 3 quando o botão não estiver apertado é HIGH e muda para LOW apenas quando estiver sendo pressionado

```
void setup() {
  pinMode(8, OUTPUT);
}

void loop() {
  int estado = digitalRead(3);
  delay(10);
  if(estado == LOW) {
    digitalWrite(8, HIGH);
  } else {
    digitalWrite(8, LOW);
  }
}
```



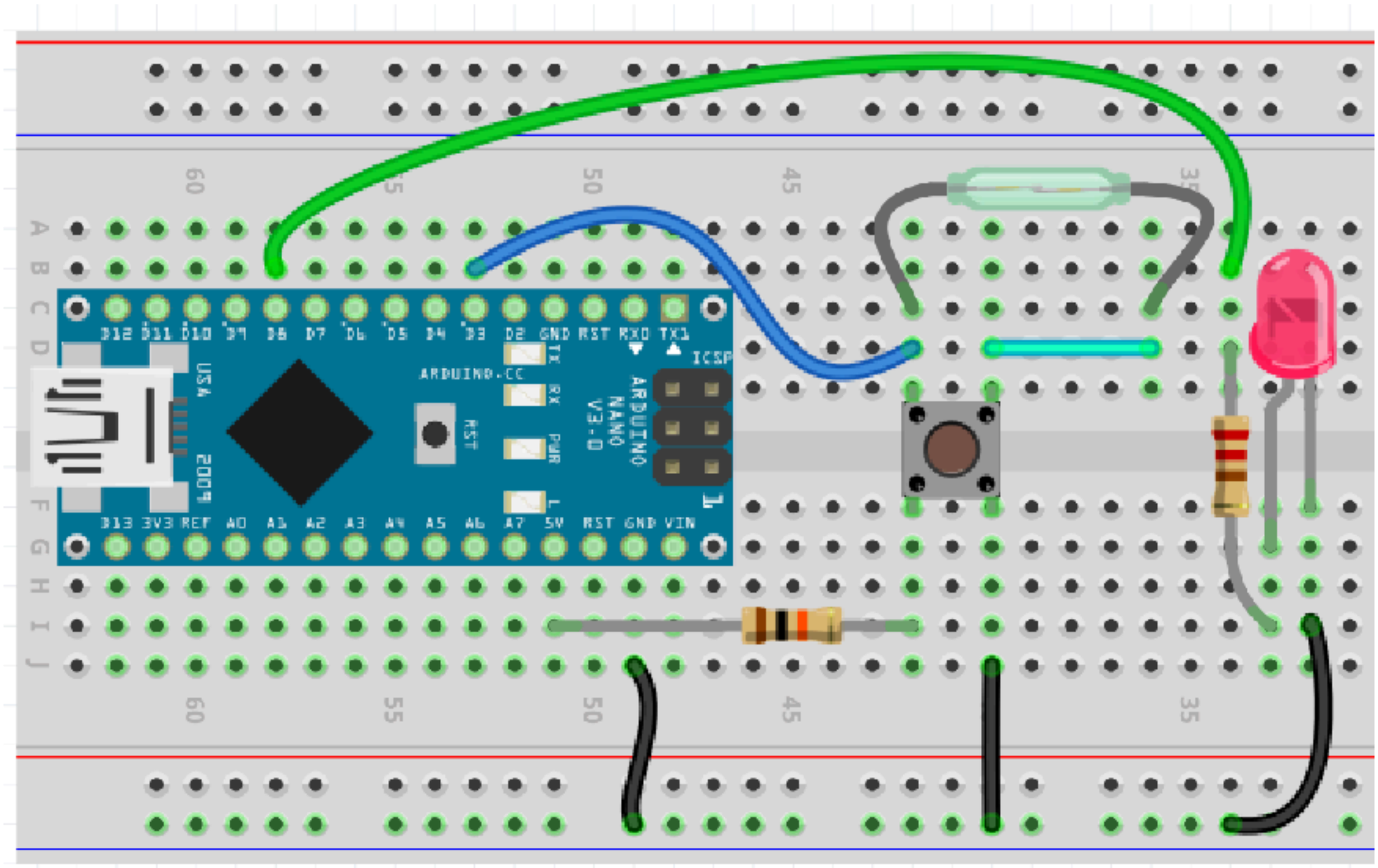
Adicione uma chave de inclinação (tilt)

Incline o protoboard para ver o LED acender ou apagar

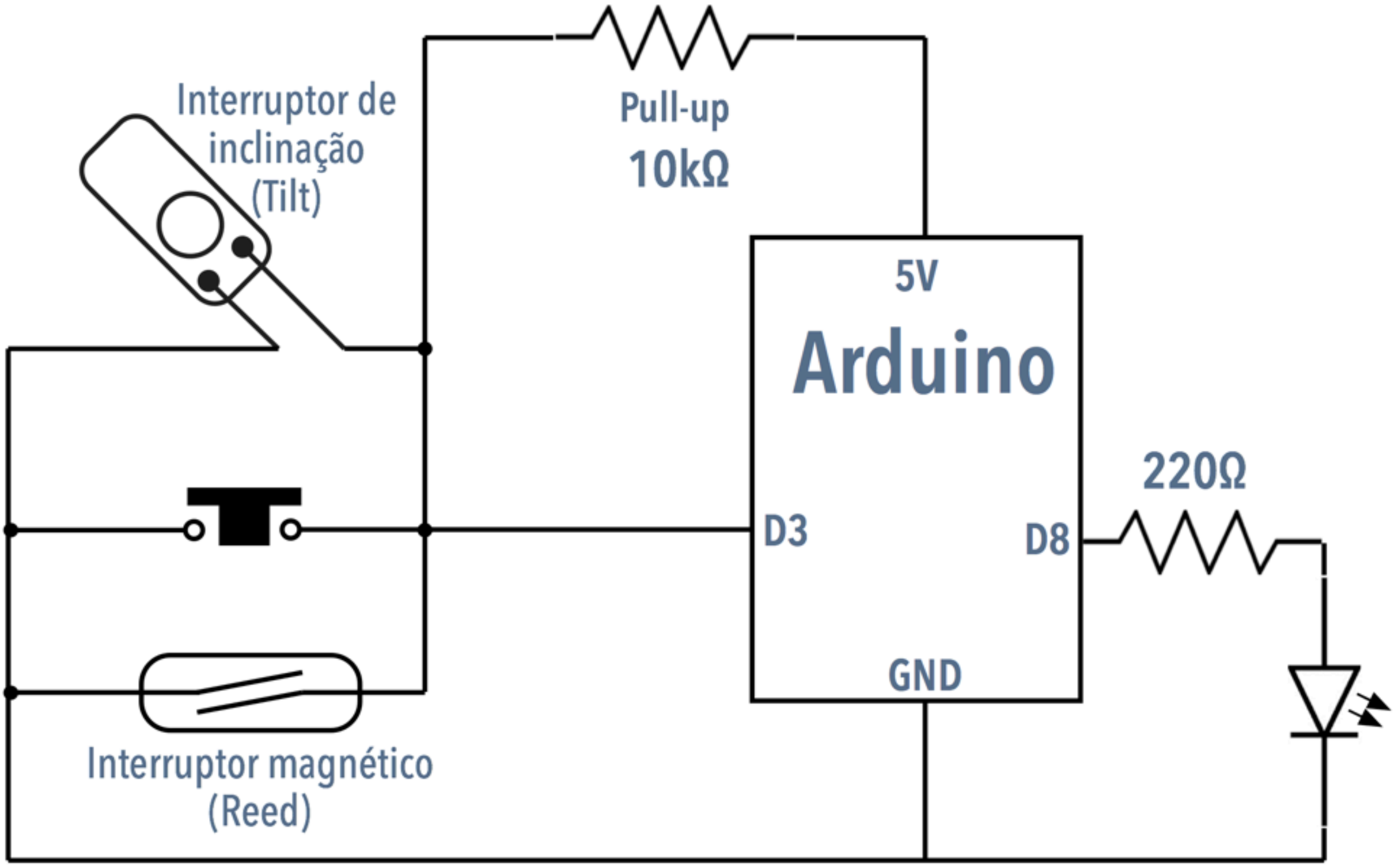


Adicione uma chave magnética (reed)

Aproxime um imã para ver o LED acender ou apagar

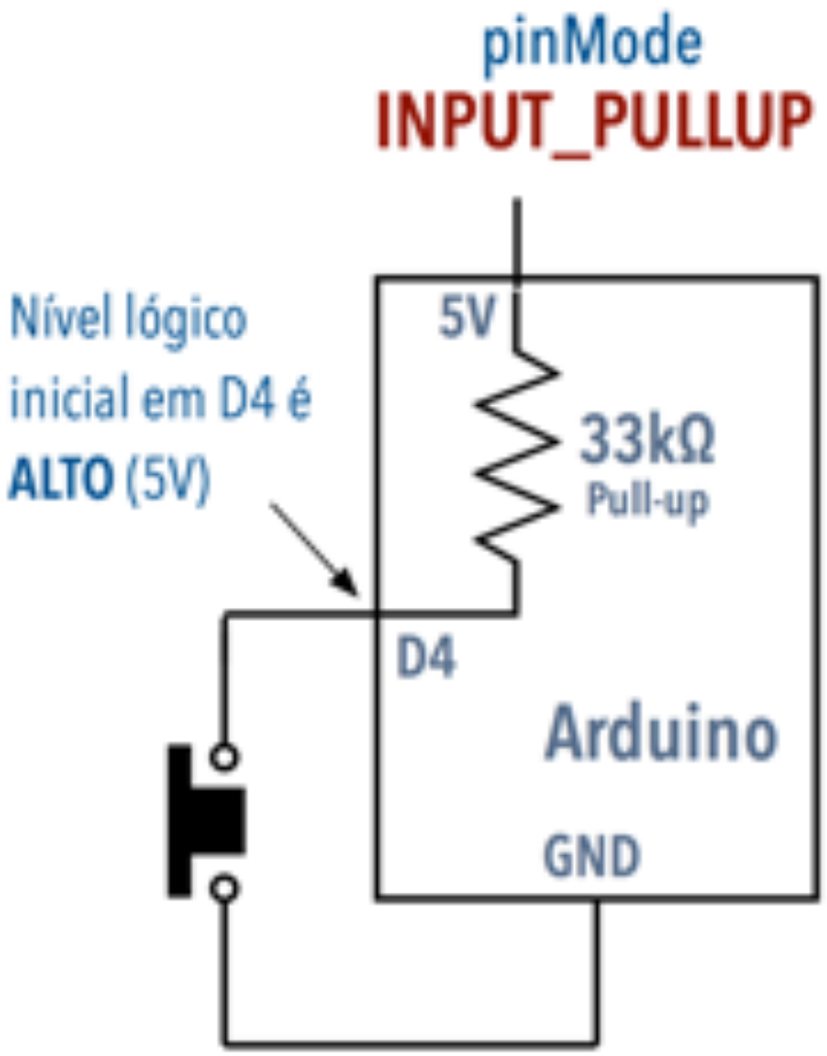


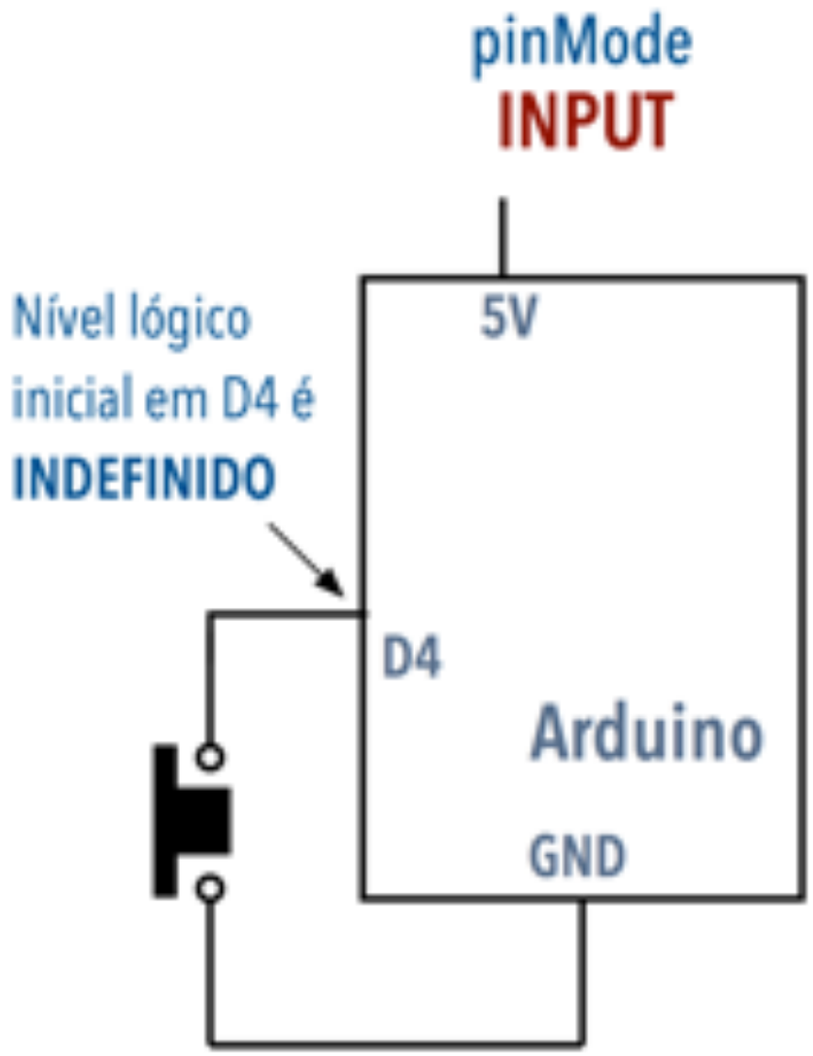
Esquema: Chave + Tilt + Reed



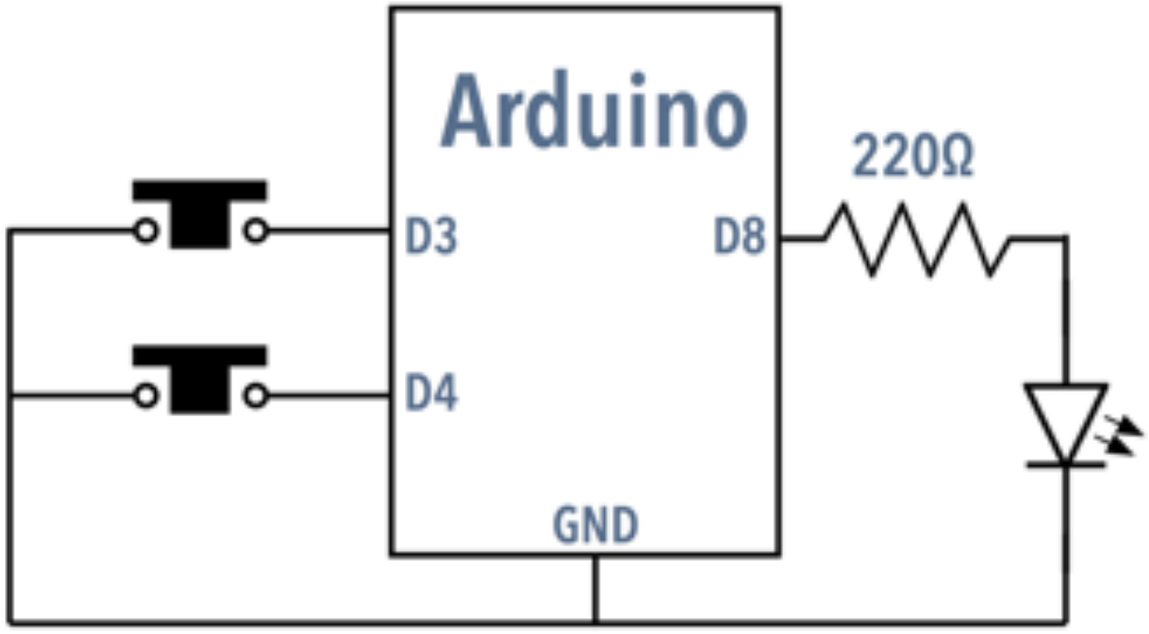
INPUT_PULLUP

pinMode(pino, INPUT_PULLUP);





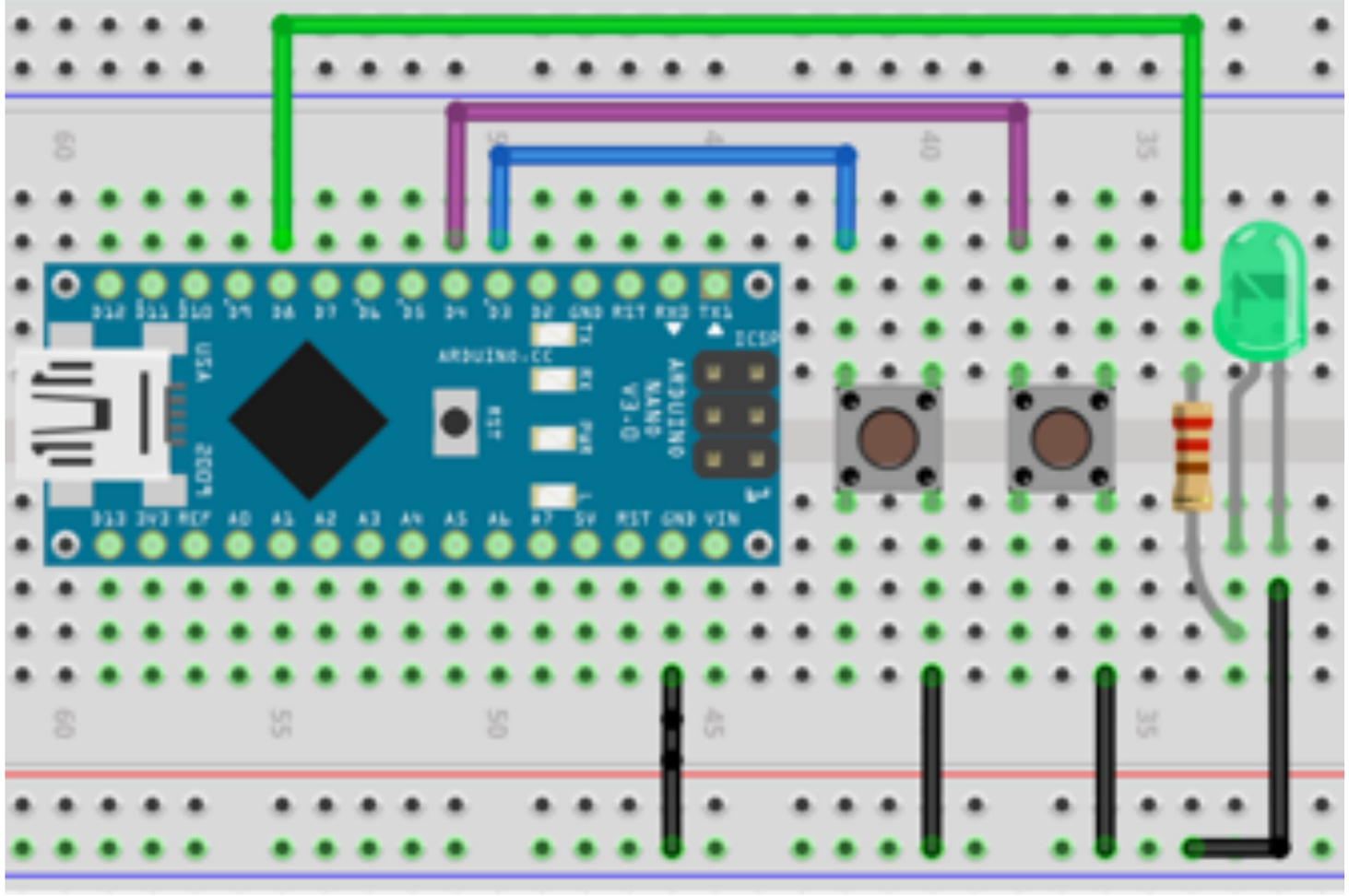
Circuito bi-estável (set/reset)



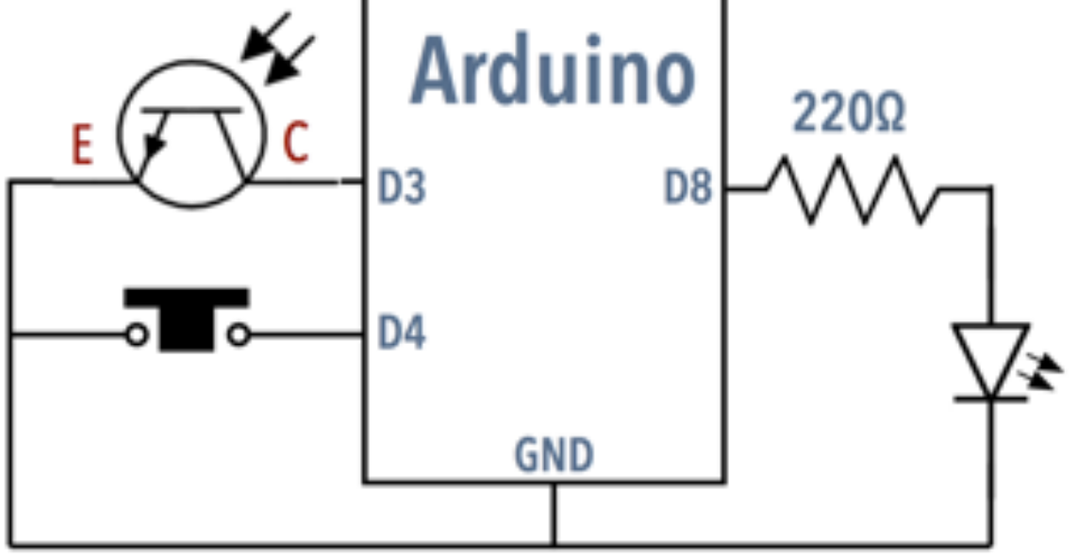
Usando o pinMode INPUT_PULLUP
não é necessário usar resistores de pull-up
(o Arduino já fornece um pull-up interno)

Botão D3: set (acende e mantém aceso)
Botão D4: reset (apaga o LED)

```
void setup() {
  pinMode(8, OUTPUT);
  pinMode(3, INPUT_PULLUP);
  pinMode(4, INPUT_PULLUP);
}

void loop() {
  int acender = digitalRead(3);
  int apagar  = digitalRead(4);

  if(acender == LOW) {
    digitalWrite(8, HIGH);
  }
  if(apagar  == LOW) {
    digitalWrite(8, LOW);
  }
}
```

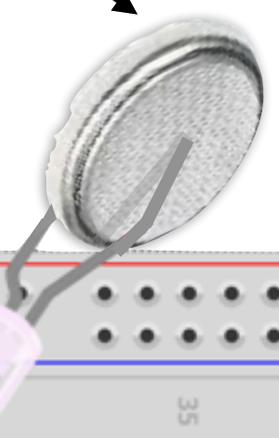


Troque o **set** por um fototransistor



Use um LED infravermelho (ou luz direta) para acionar o sensor

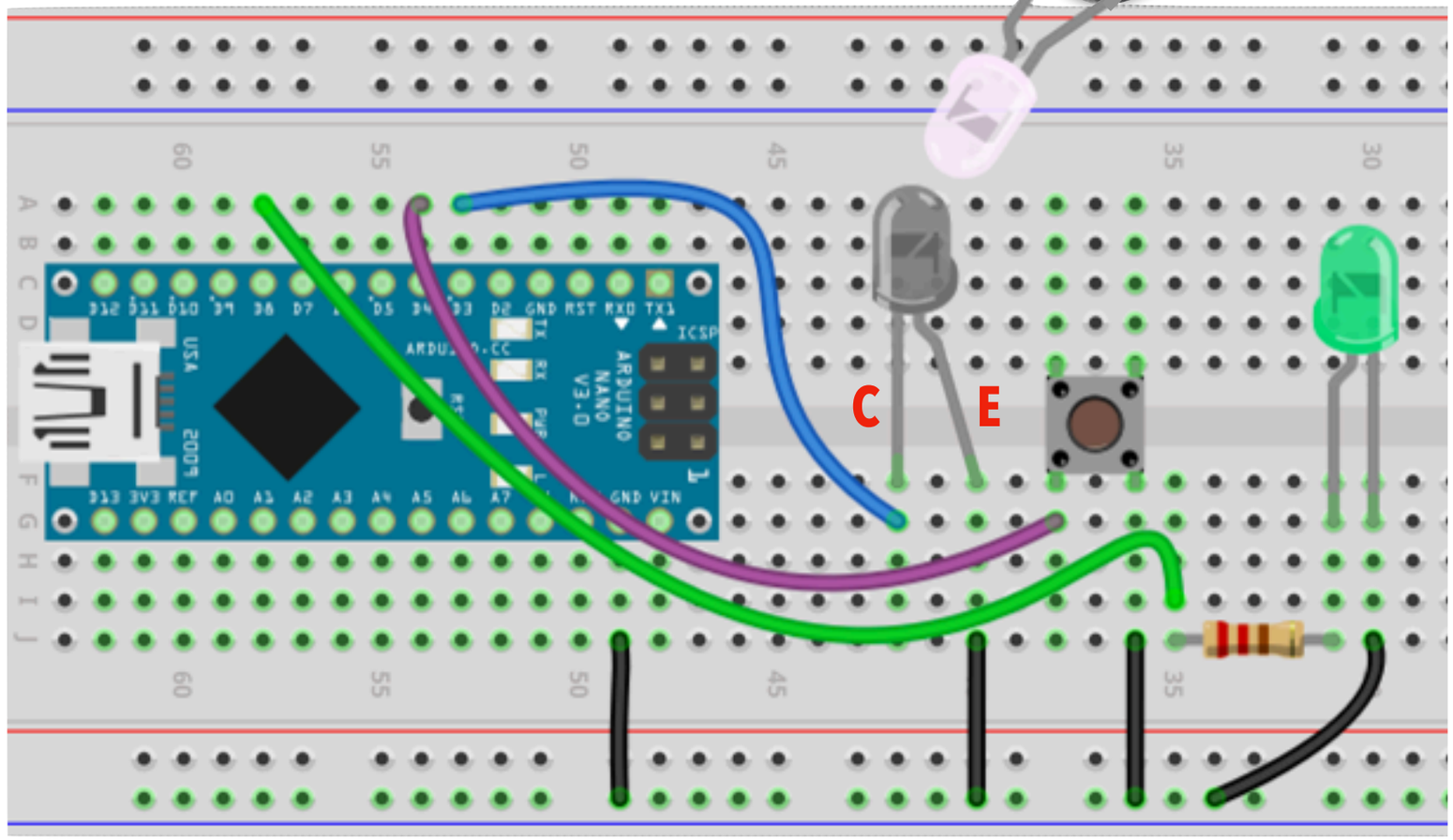
(O LED infravermelho tem uma tensão direta a 20mA de 1,6V; você pode usar com esta pilha de 3V por alguns segundos; conecte um resistor de 100 ohms se precisar manter ligado mais tempo)



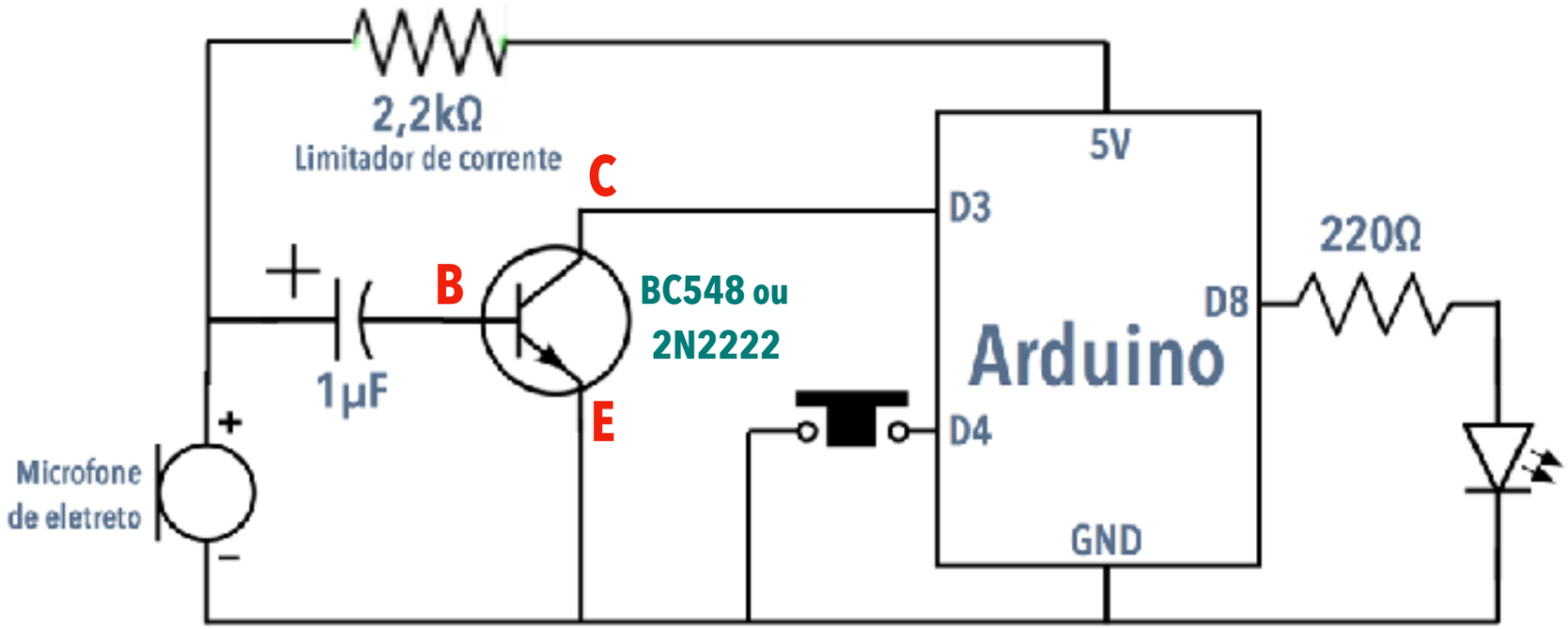
Experimente também usando outros sensores:

Chave tilt
Chave reed

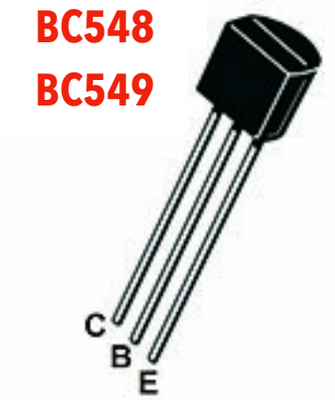
Quando o sensor fechar o circuito o LED acende (**set**) e só apaga quando o botão (**reset**) é apertado



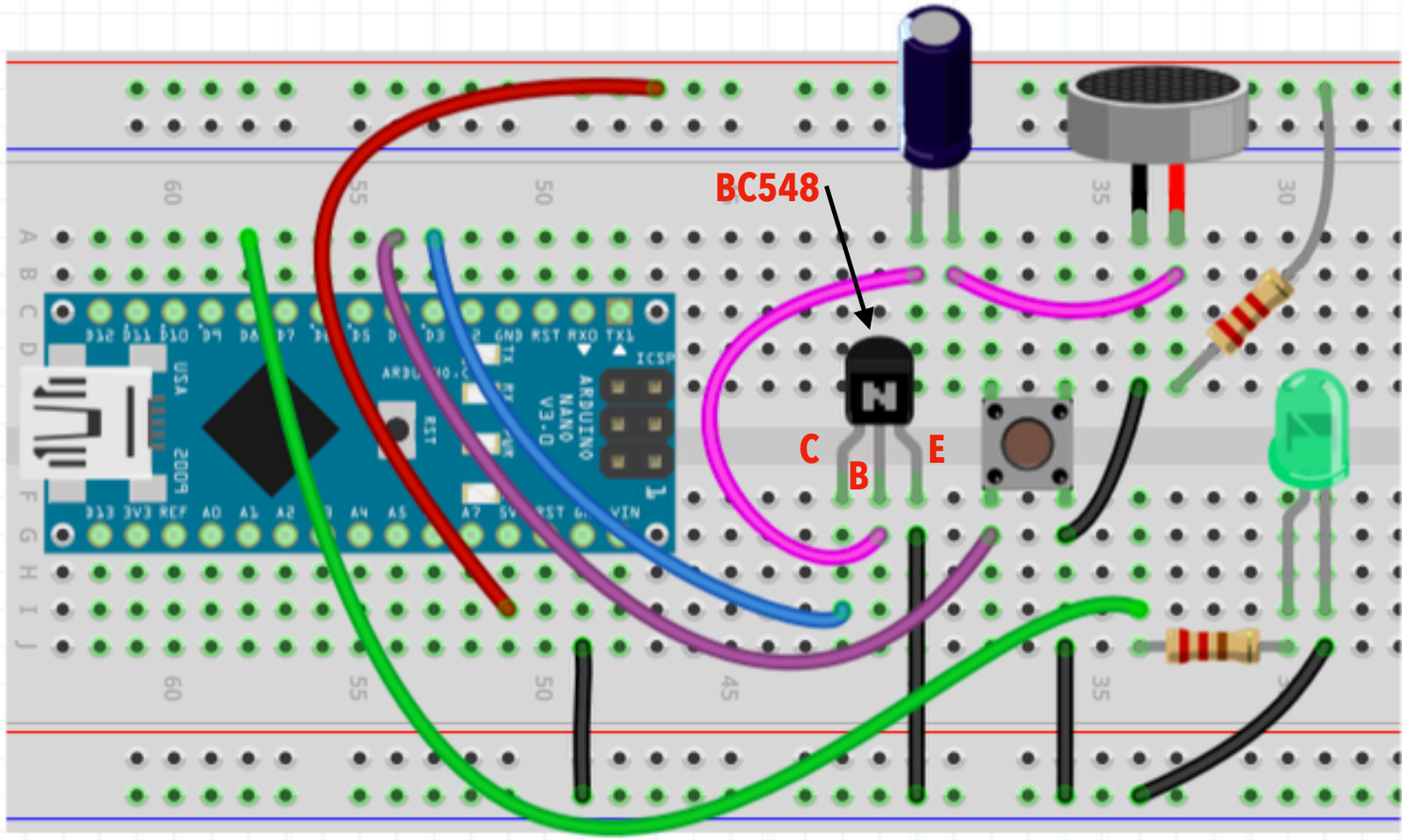
Troque o set por um sensor de som



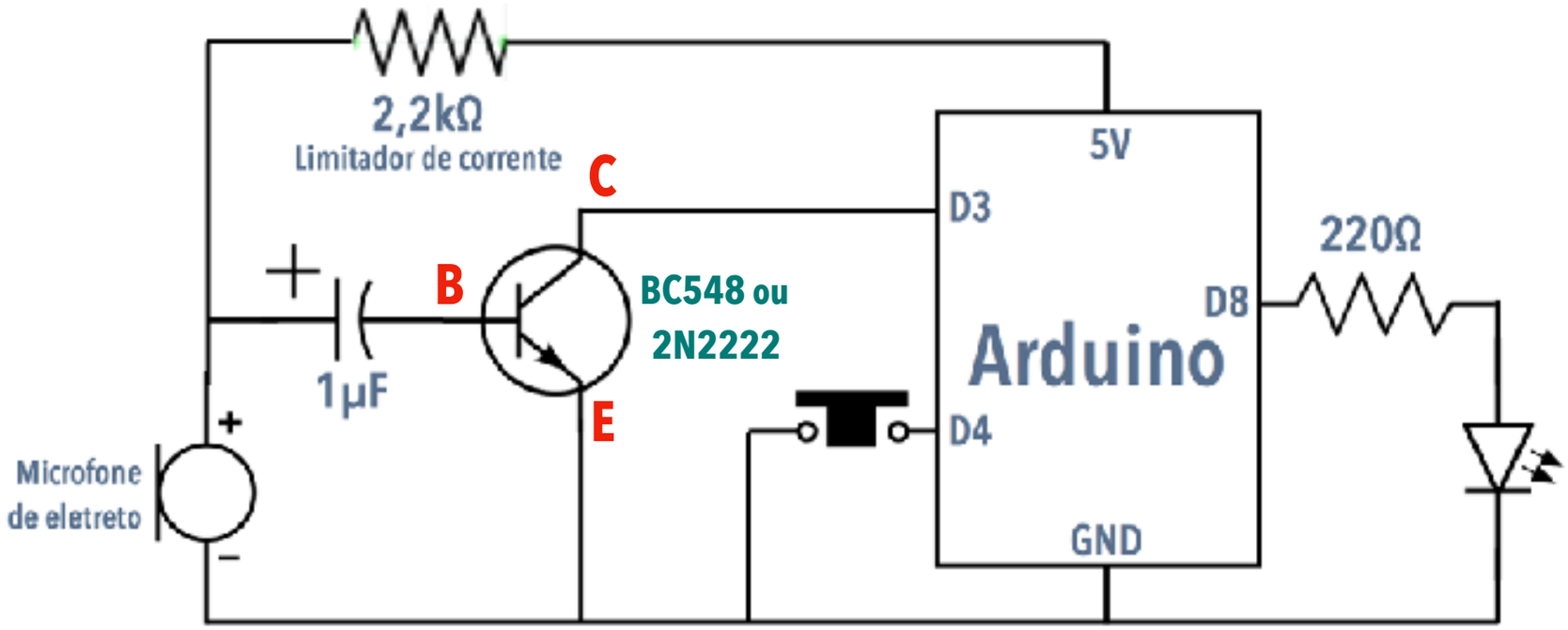
Transistor amplifica o sinal e gera pulso no pino D3



Pequeno pulso no microfone é suficiente para disparar o transistor e mudar o estado do circuito

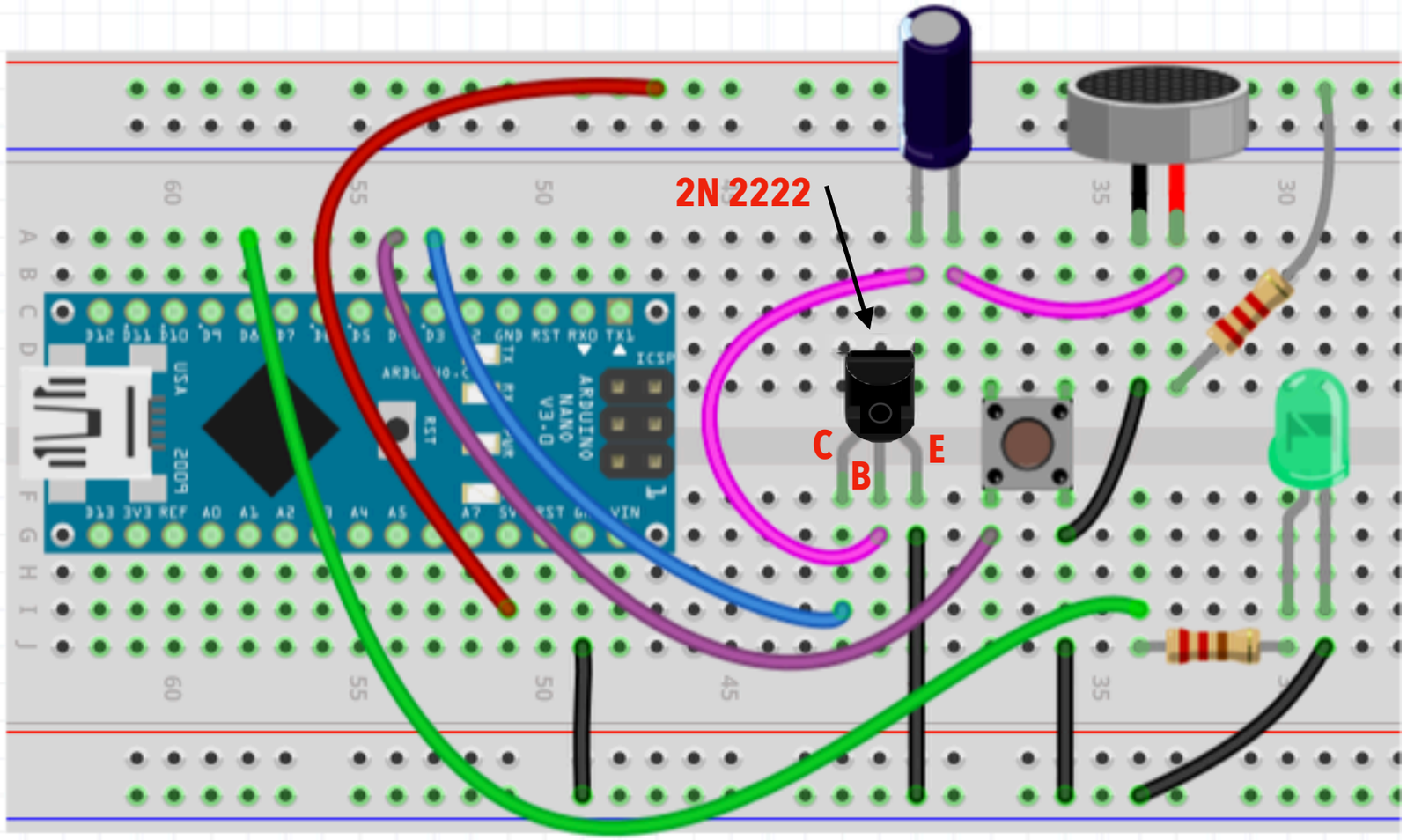


Troque o set por um sensor de som

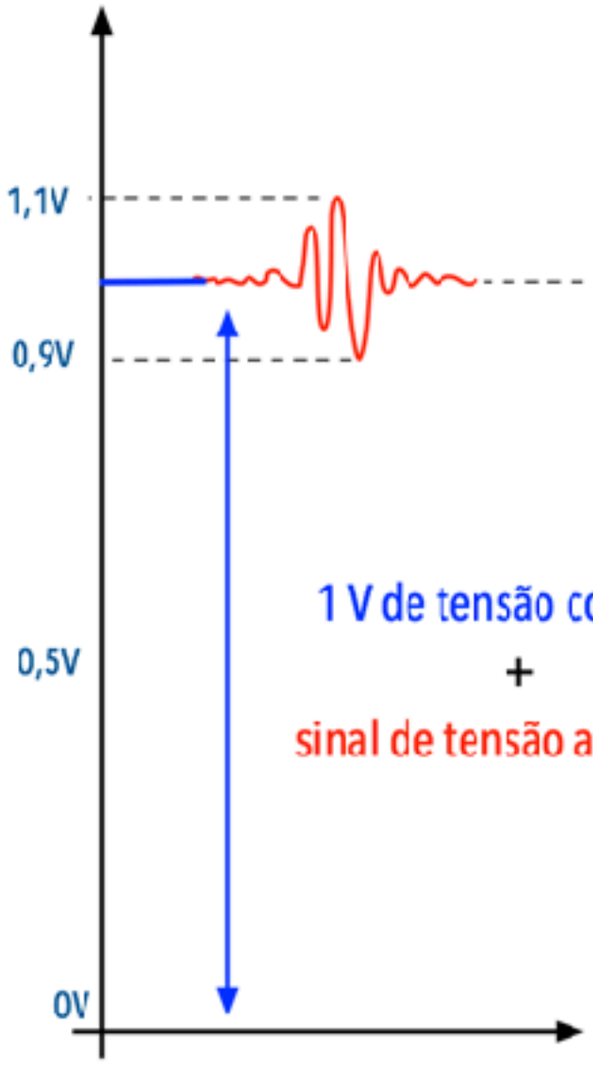


Transistor amplifica o sinal e gera pulso no pino D3

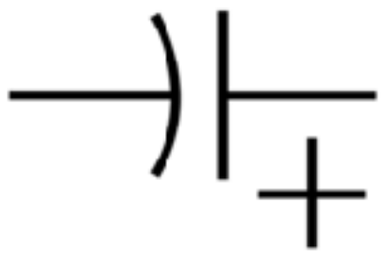
Pequeno pulso no microfone é suficiente para disparar o transistor e mudar o estado do circuito



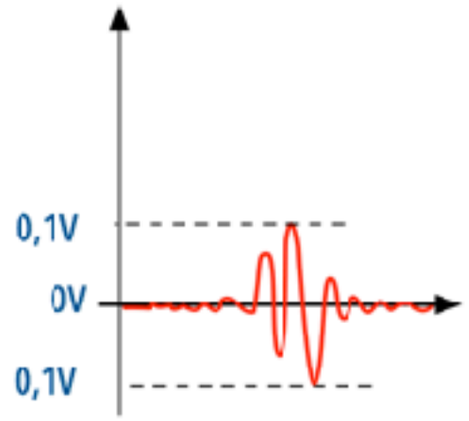
Amplificação do sinal do microfone



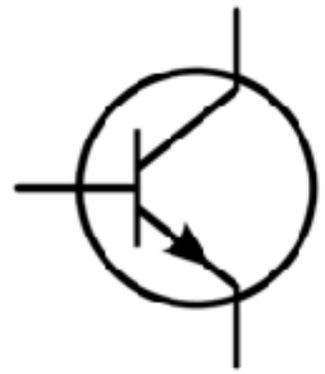
1 V de tensão contínua
+
sinal de tensão alternada



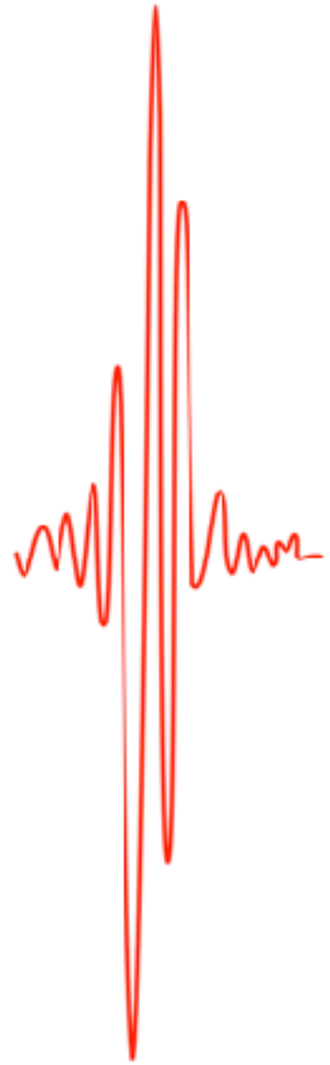
Capacitor não deixa passar a parte contínua



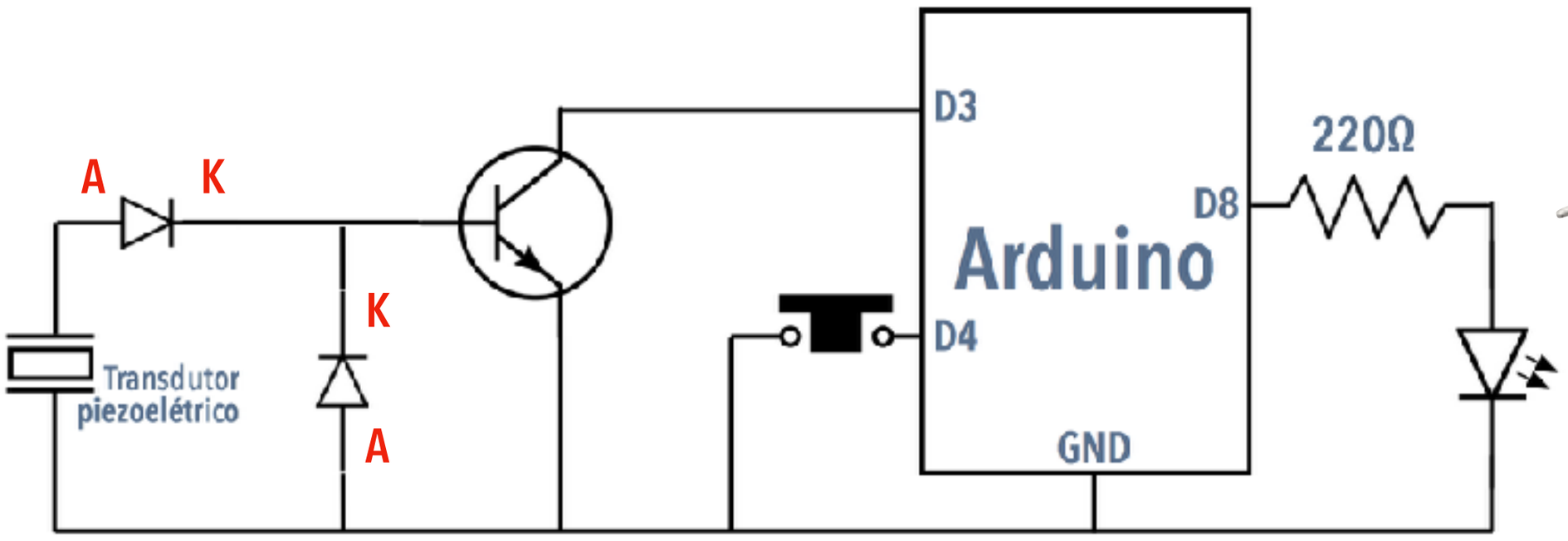
Passa apenas o sinal

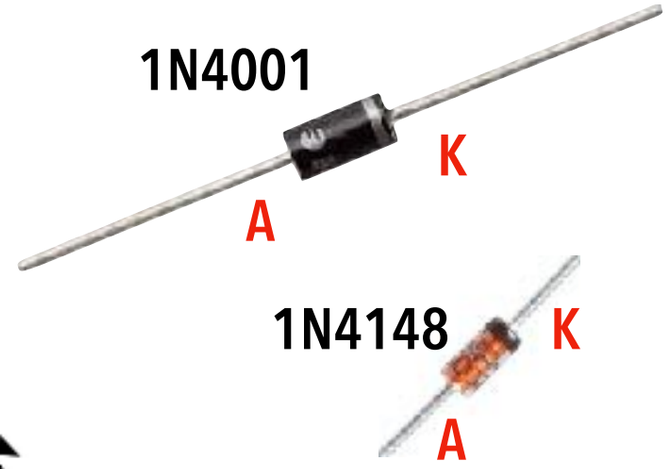


Transistor amplifica (e inverte) o sinal

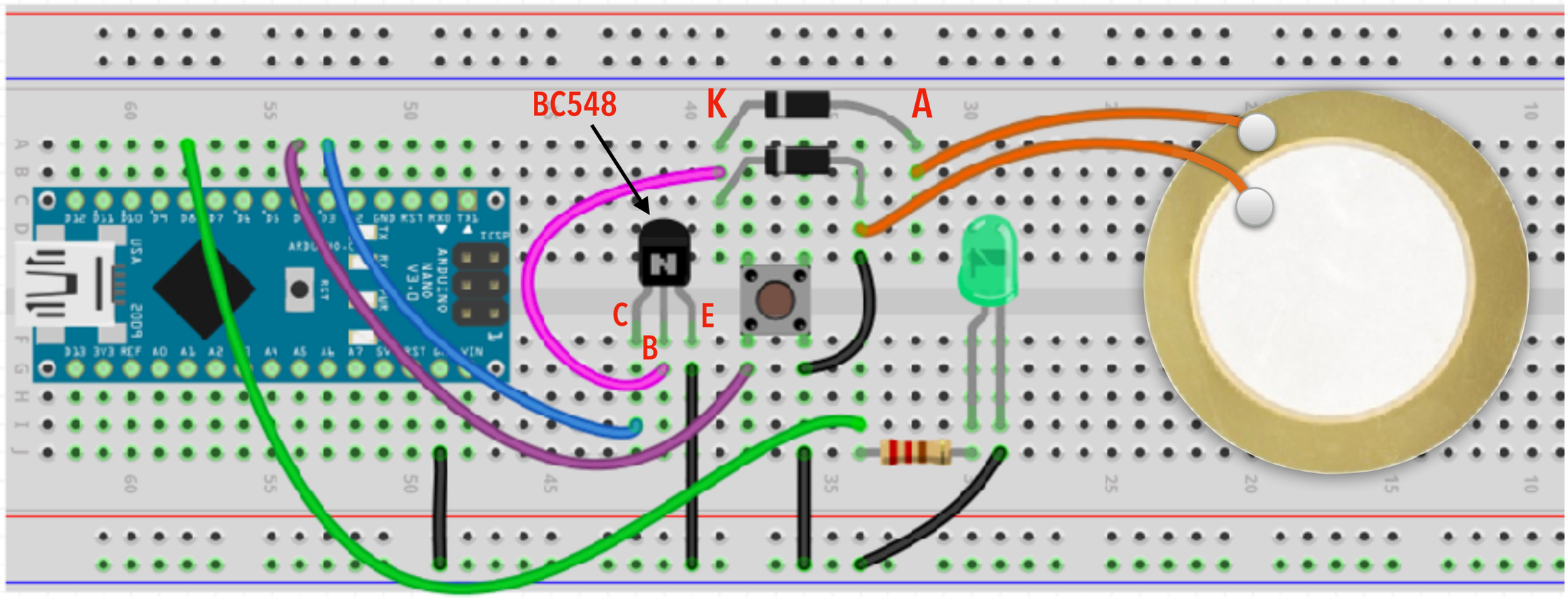


Troque o set por um sensor de deformação

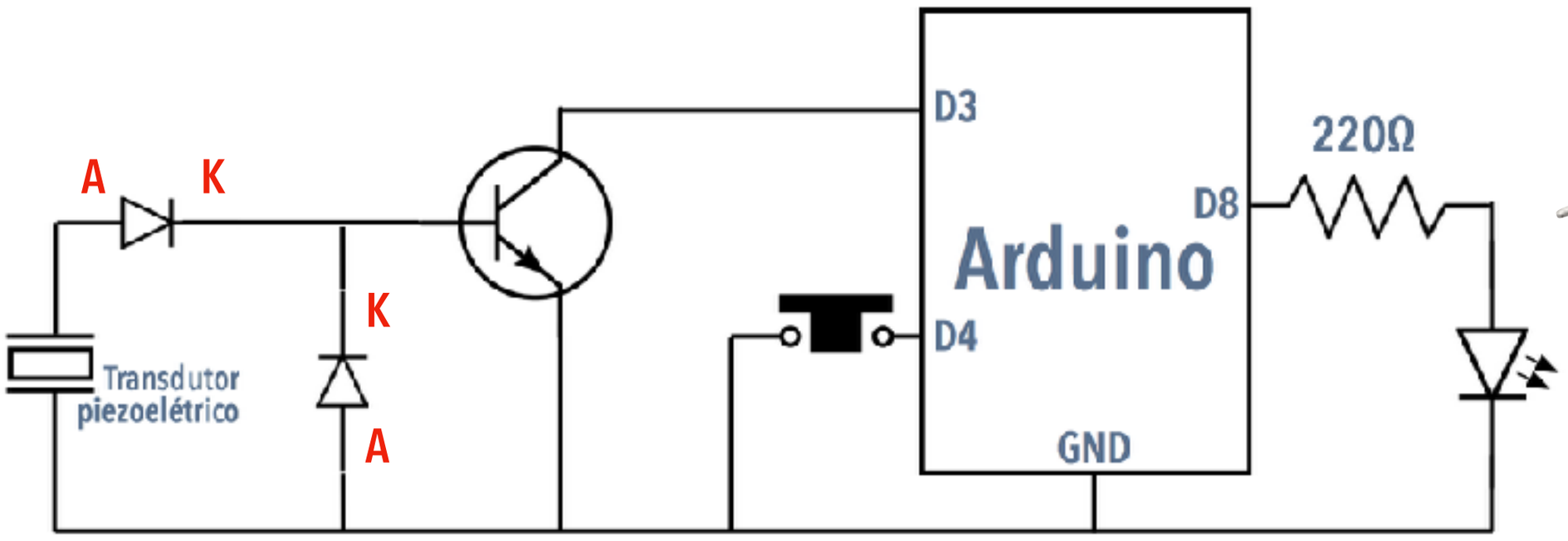


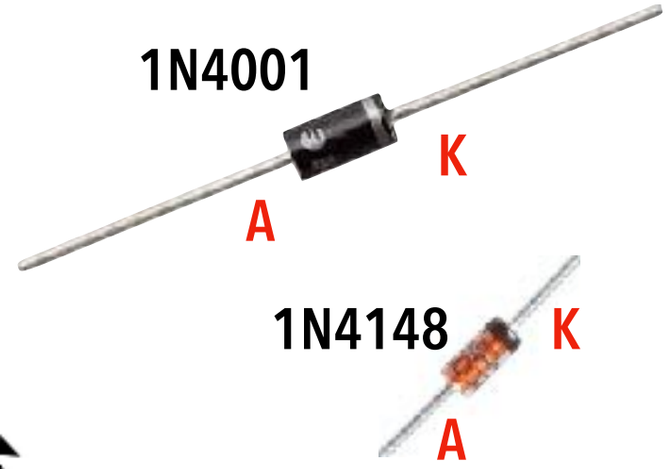


(diodos de propósito geral)

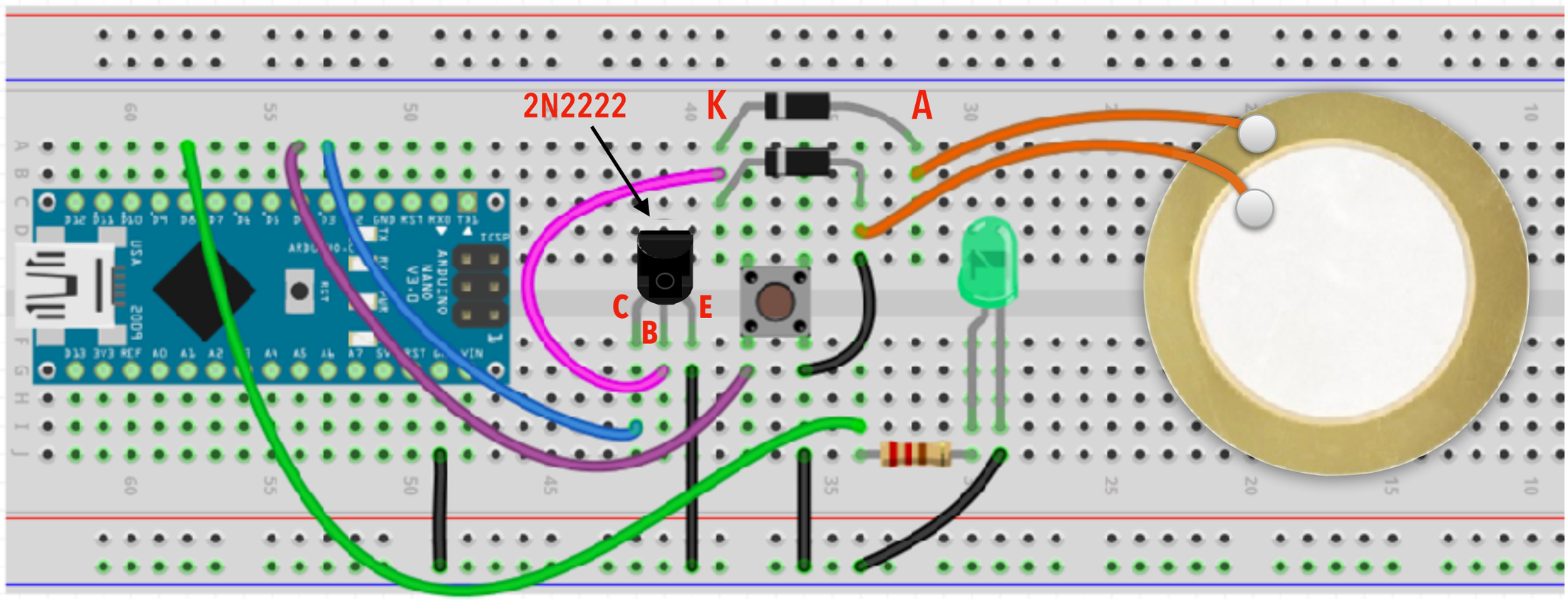


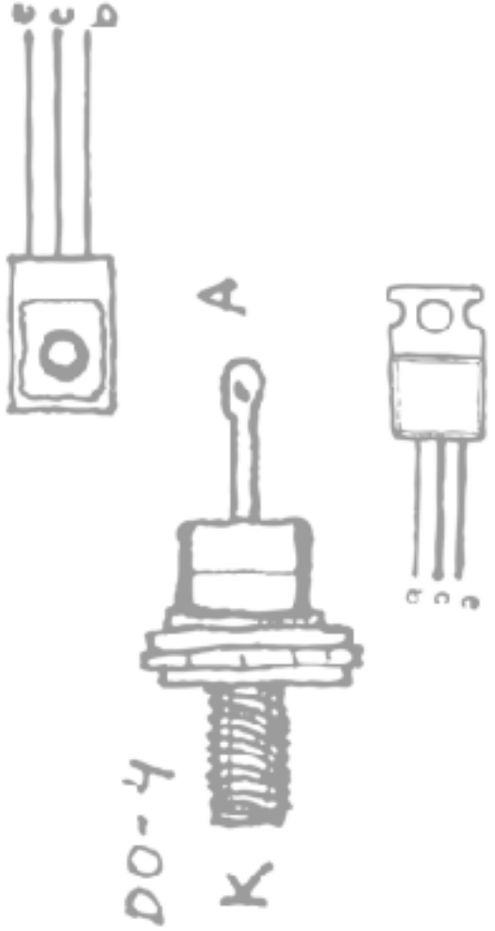
Troque o set por um sensor de deformação



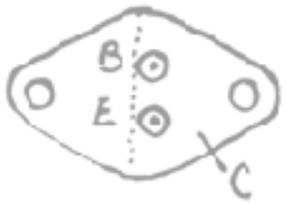


(diodos de propósito geral)





ARDUINO 4



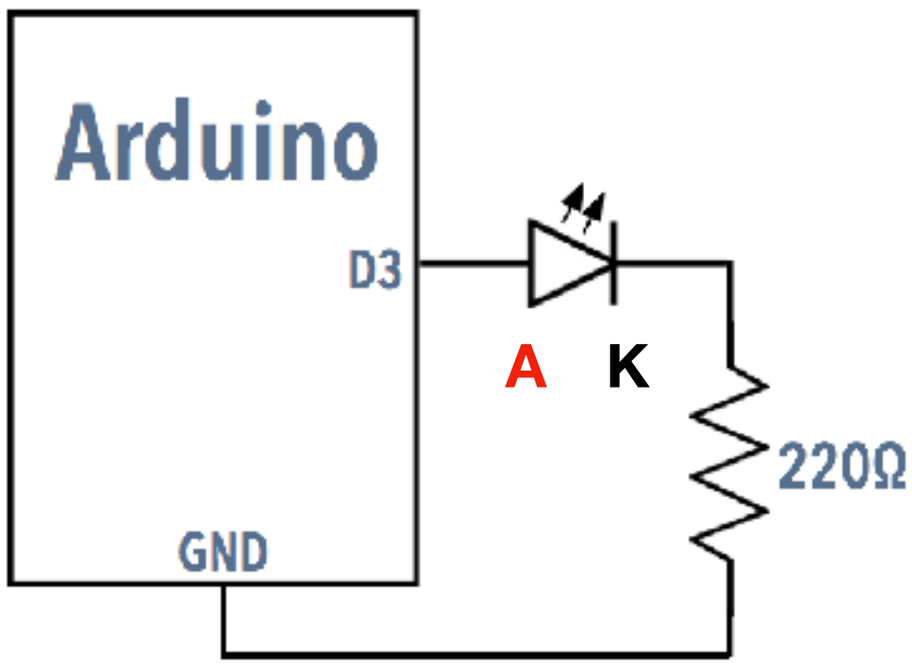
Saída analógica

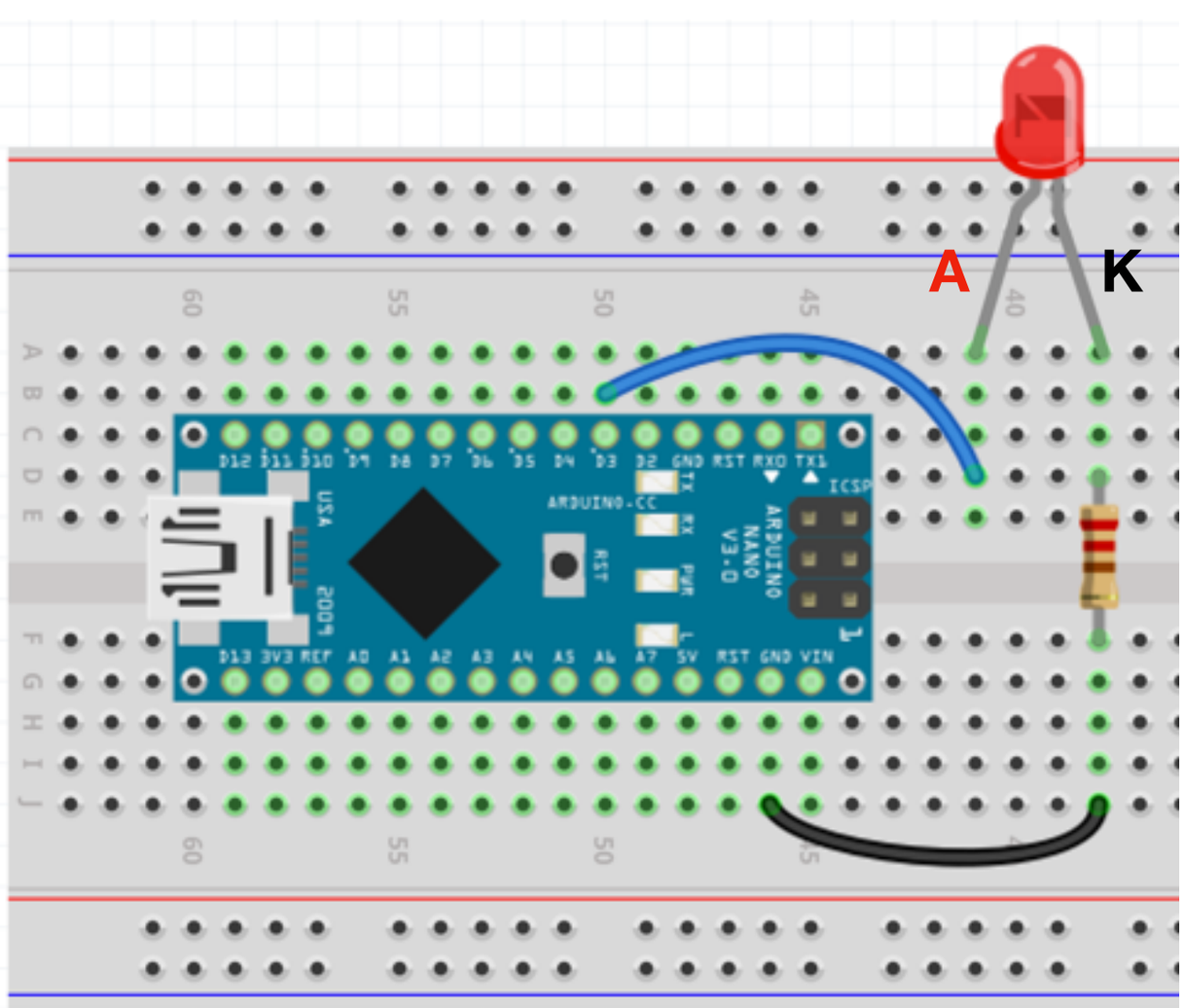
Helder da Rocha
www.eletronicaparaartistas.com.br

analogWrite

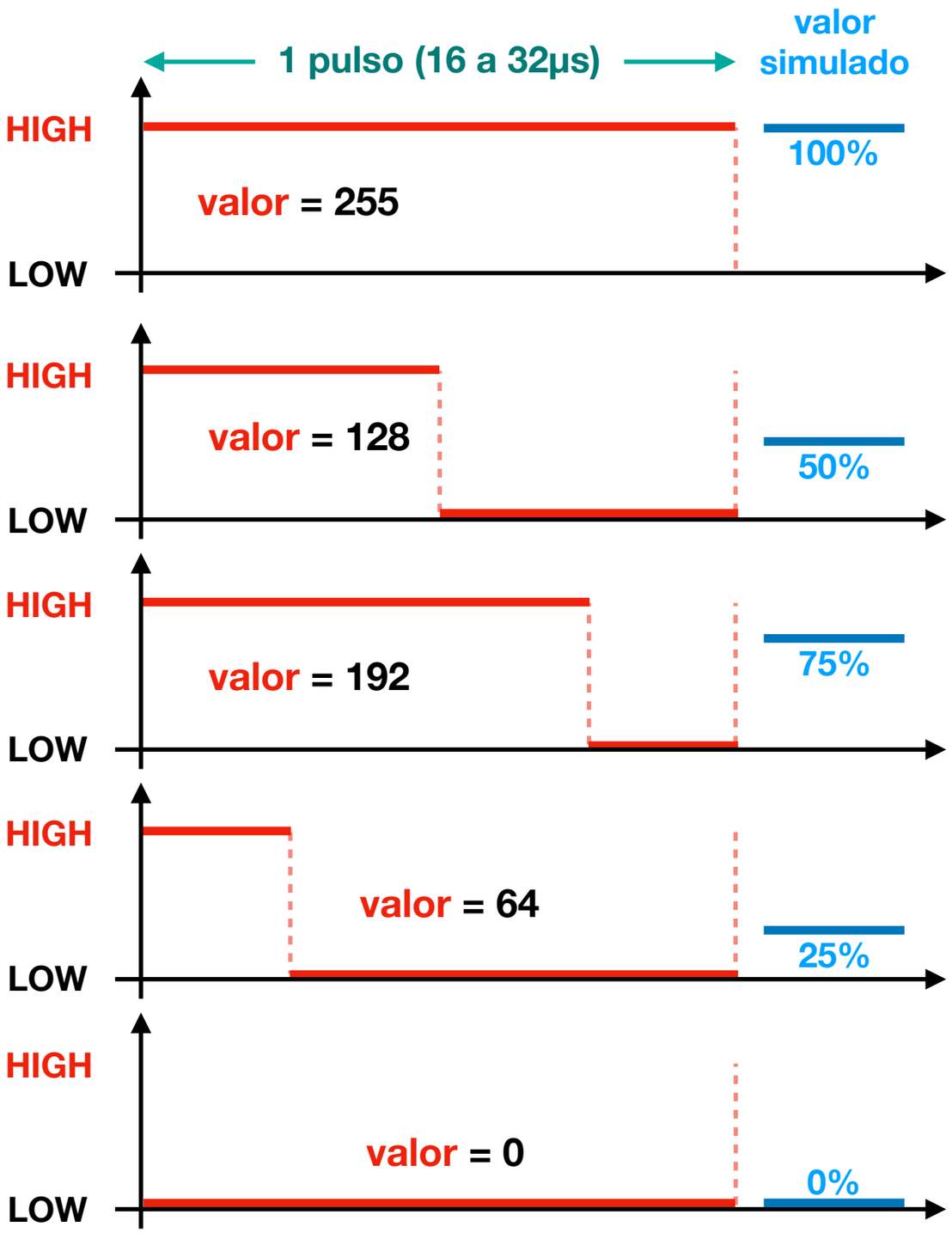
```ino
int LED = 3;        // Qualquer pino PWM: 3, 5, 6, 9, 10, 11
int brilho = 255;   // Inicia com brilho máximo
int direcao = -1;   // 1 = aumentando, -1 = diminuindo

void setup() {
    pinMode(LED, OUTPUT);  // É opcional com analogWrite
}

void loop() {
    analogWrite(LED, brilho);
    delay(5);
    brilho = brilho + direcao;
    if(brilho <= 0 || brilho >= 255) {
        direcao = -direcao;
    }
}
```

Experimente também trocar o LED por cigarra (buzzer)





analogWrite(pino, valor)



PWM (Pulse-Width Modulation) simula saída analógica (não é saída analógica verdadeira)

Acende (HIGH) e apaga (LOW) rapidamente; simula efeito analógico variando tempo em HIGH e LOW

Únicos valores produzidos são VCC (5V ou 3,3V) e 0V

Saída analógica verdadeira requer circuito retificador (com capacitores)

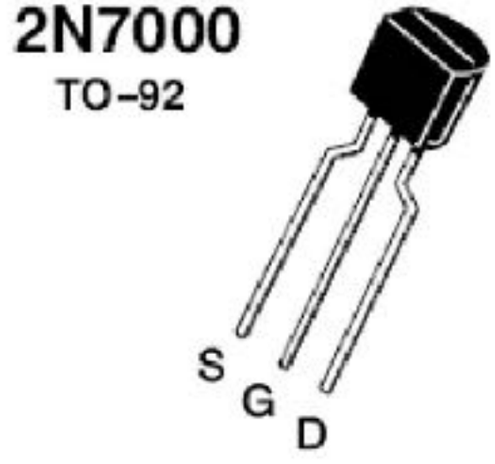
Frequência do PWM em Arduino Uno/Nano

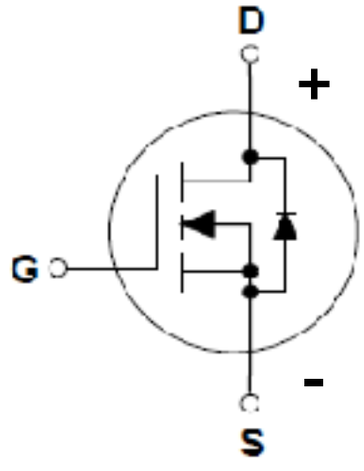
- **Pinos 3, 9, 10 e 11: 31250Hz**
- **Pinos 5 e 6: 62500Hz**

MOSFET

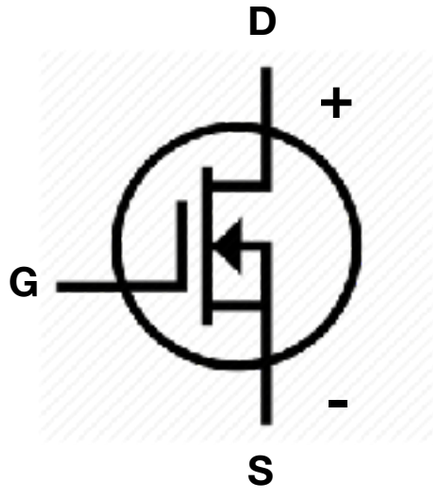
Metal-**O**xide-**S**emiconductor **F**ield-**E**ffect **T**ransistor

"Ligue" o MOSFET aplicando tensão positiva (> +2V) no seu terminal G





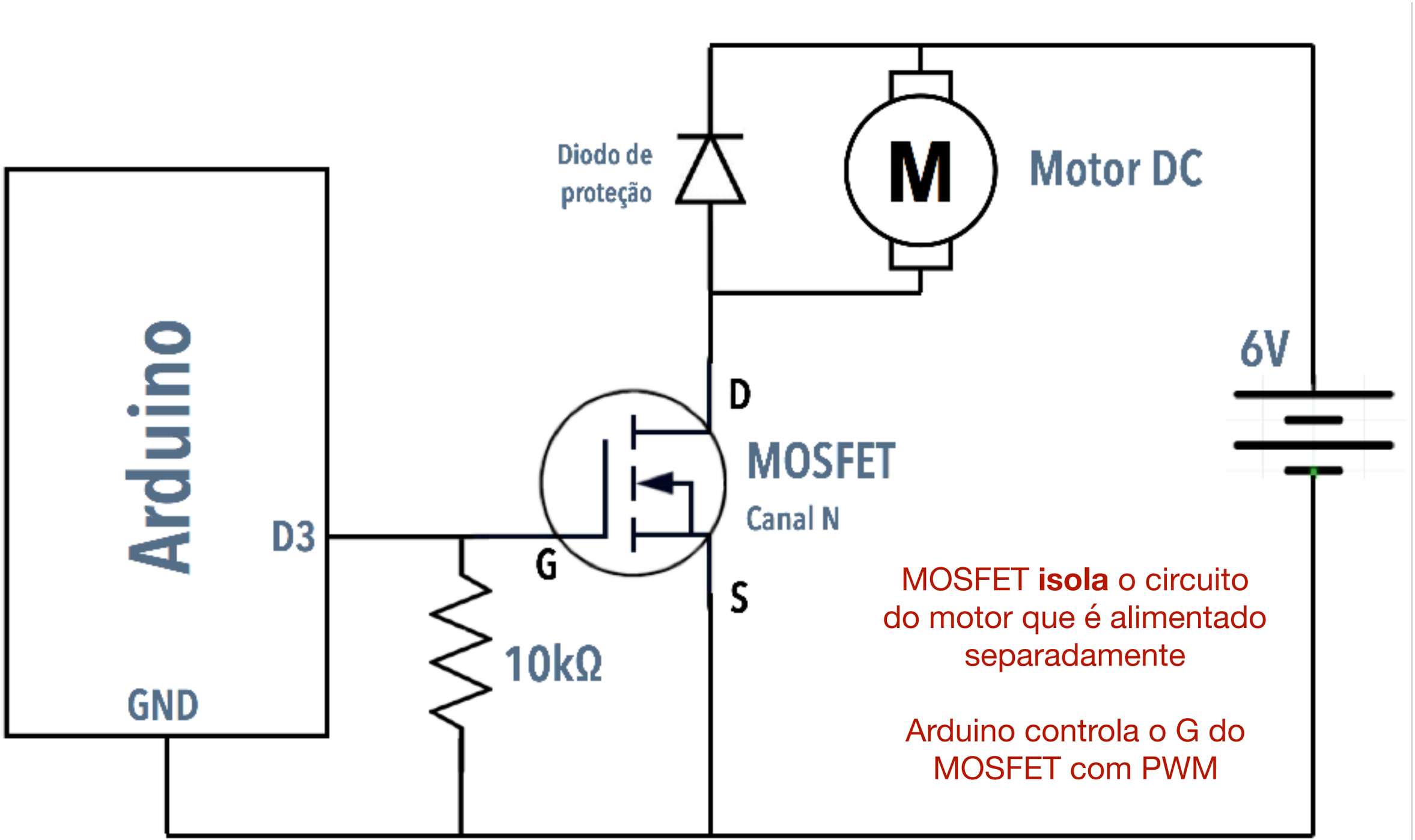
MOSFET Canal-N



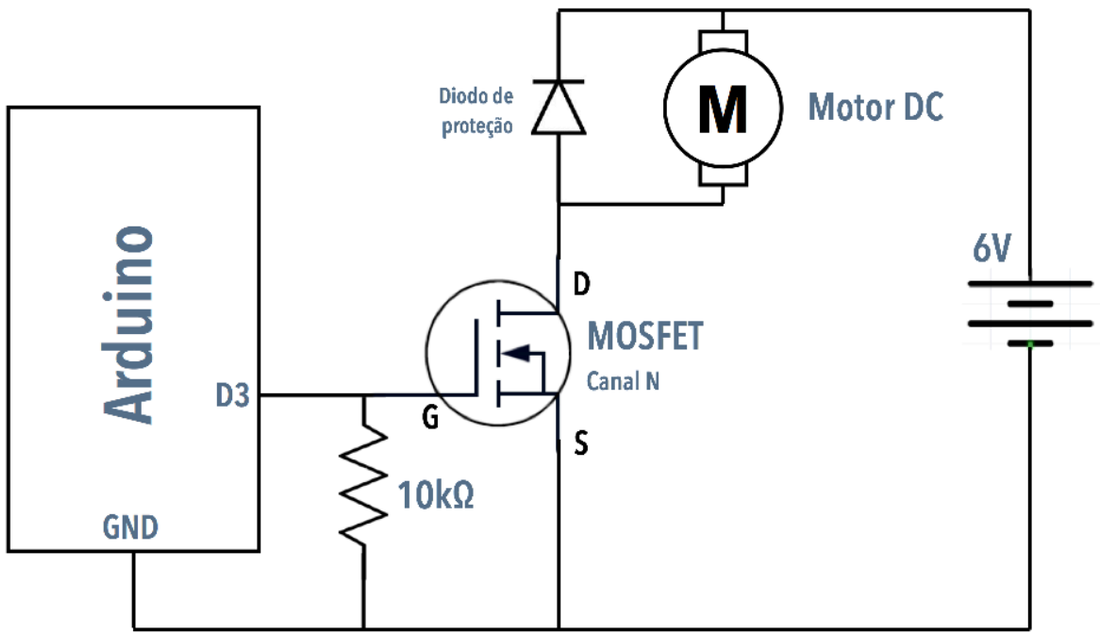
Símbolo simplificado

- MOSFETs são transistores muito usados na eletrônica digital (principalmente em circuitos que utilizam grandes quantidades de transistores, como chips)
- São similares aos transistores bipolares. O terminal **G** (gate, comporta) tem função similar à base (B). **D** (drain, dreno) é similar ao coletor (C), e **S** (source, fonte) ao emissor.
- A principal diferença é que o transistor não requer corrente alguma em G. Ele é ligado aplicando uma tensão em G que carrega o capacitor interno e abre a comporta entre D e G.
- O consumo de energia é baixo, podem ser usados em cascata facilmente, mas são vulneráveis a cargas eletrostáticas.

Controle de motor DC com PWM

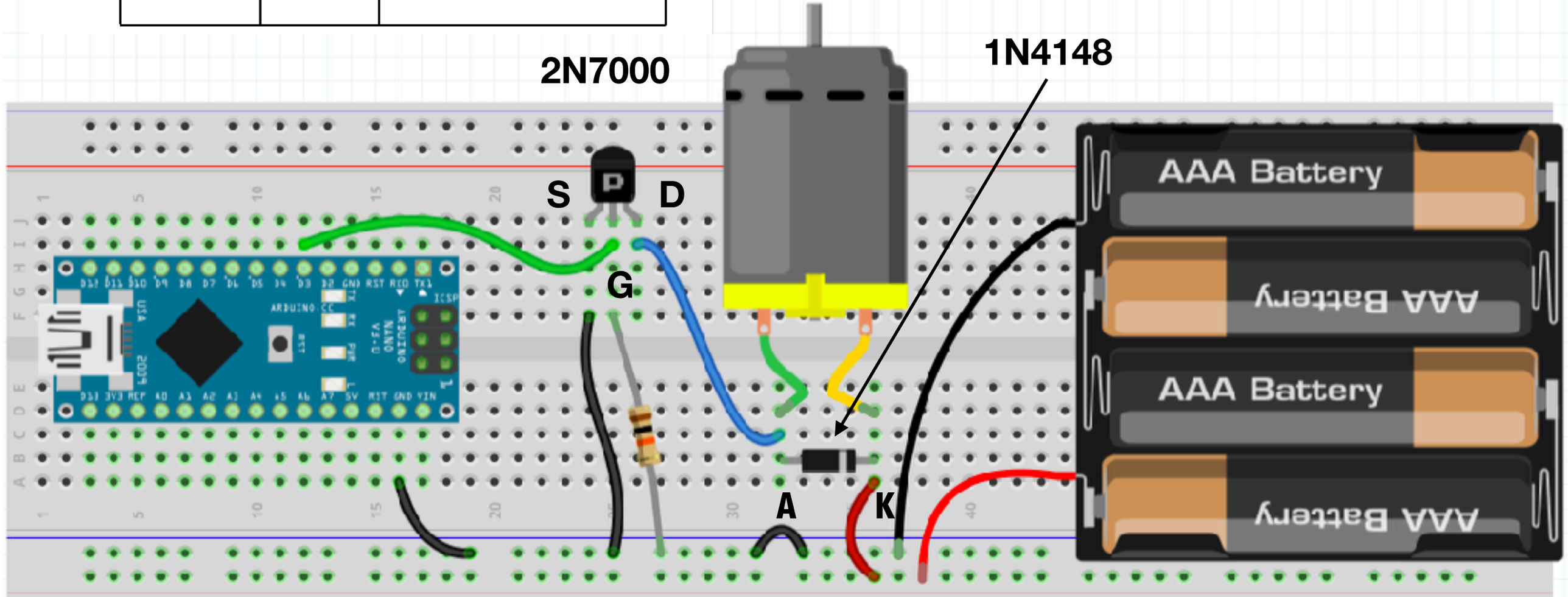


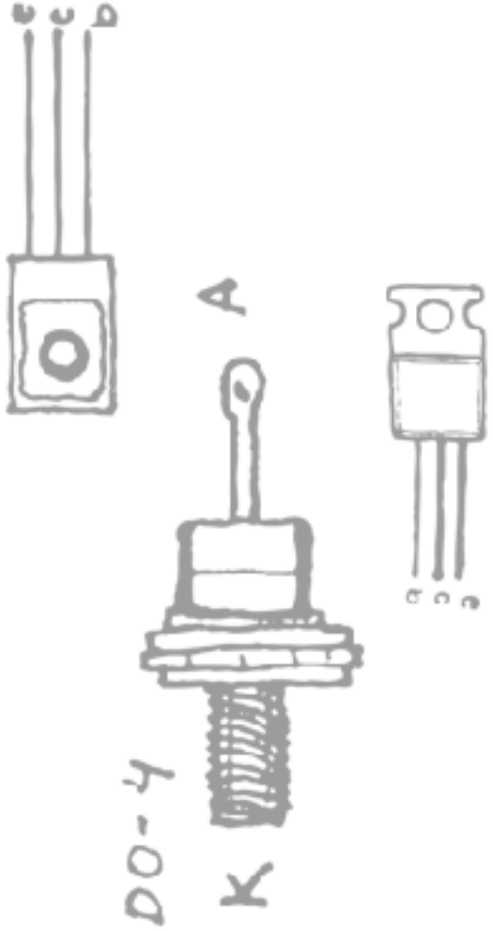
Controle de motor DC com PWM



Use mesmo programa do LED e motor irá acelerar e desacelerar continuamente

O motor de DVD incluído no kit *pode* ser alimentado pelo pino 5V do Arduino pois sua demanda de corrente é baixa (35mA) e os picos não ultrapassam 100mA (mas não faça isto com outros motores DC)





ARDUINO 5

Entrada analógica

Helder da Rocha
www.eletronicaparaartistas.com.br

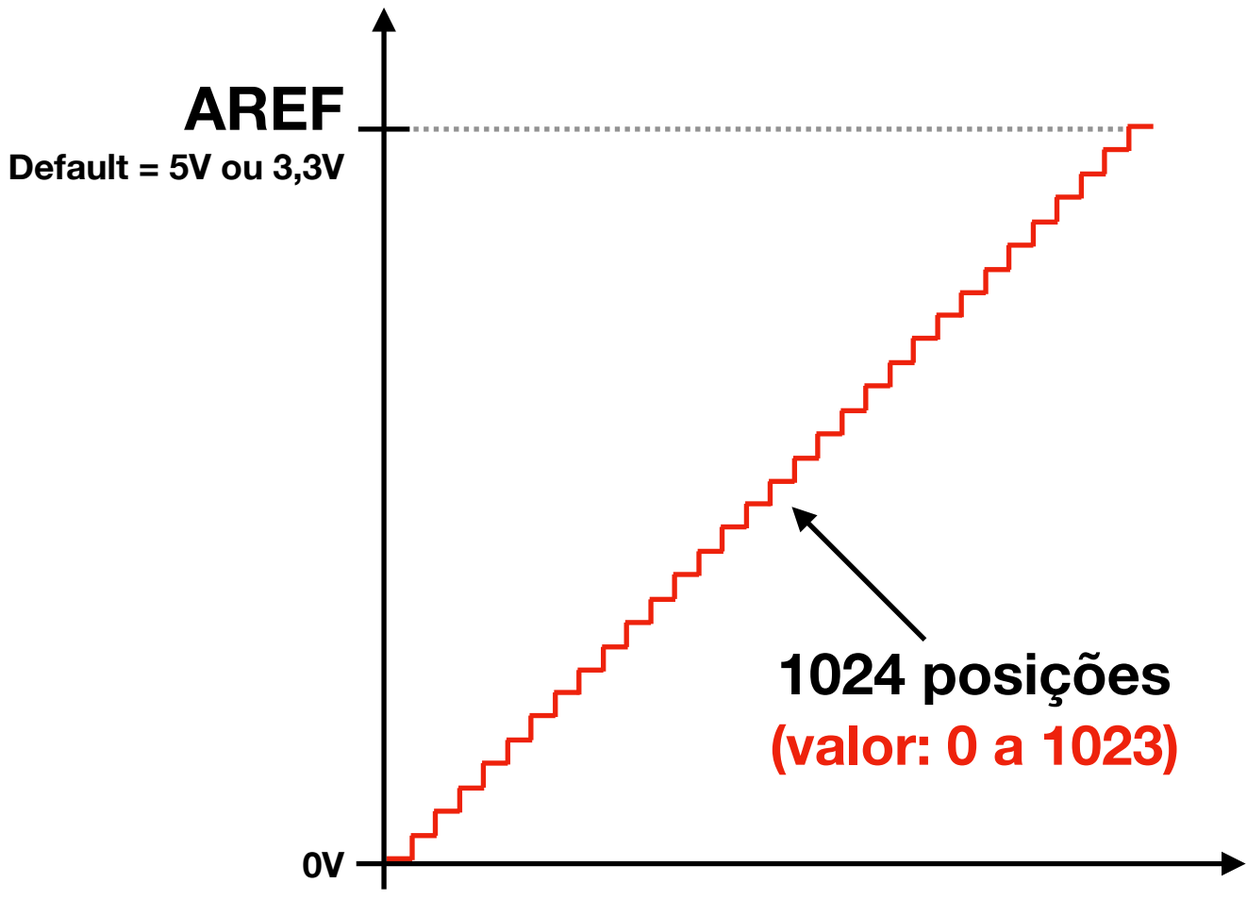
int valor = analogRead(pino)

Valor lido é inteiro com resolução de 1/1024 da tensão de referência AREF

AREF é a referência de tensão máxima para leitura analógica.

Normalmente é usada a tensão VCC do Arduino (5V ou 3,3V)

Mas AREF pode ser alterada:
- **Via software, usando instrução analogReference(): 1,1V**
- **Via hardware, aplicando tensão no pino AREF: tensão < 5V**

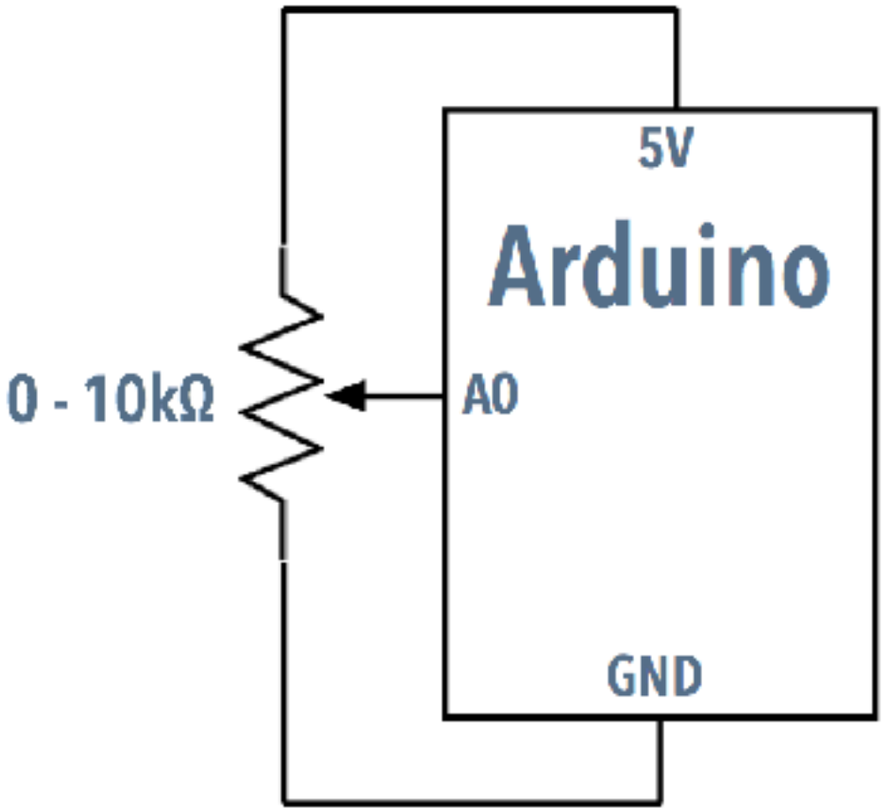


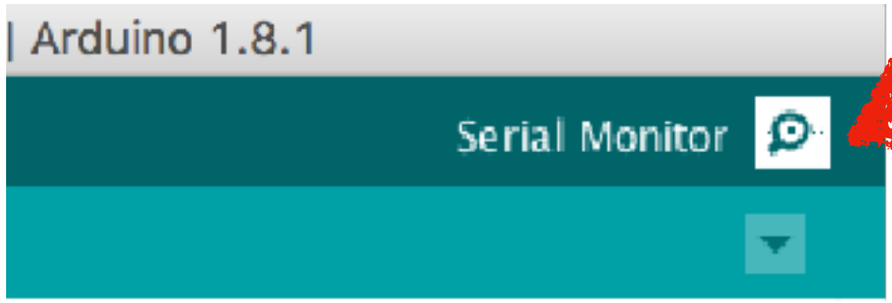
Se AREF for 5V (Arduino Uno, Nano)
V = (valor / 1024) * 5V

Se AREF for 3,3V (Due, LilyPad, ProMini)
V = (valor / 1024) * 3,3V

Se AREF for 1,1V (Internal)
V = (valor / 1024) * 1,1V

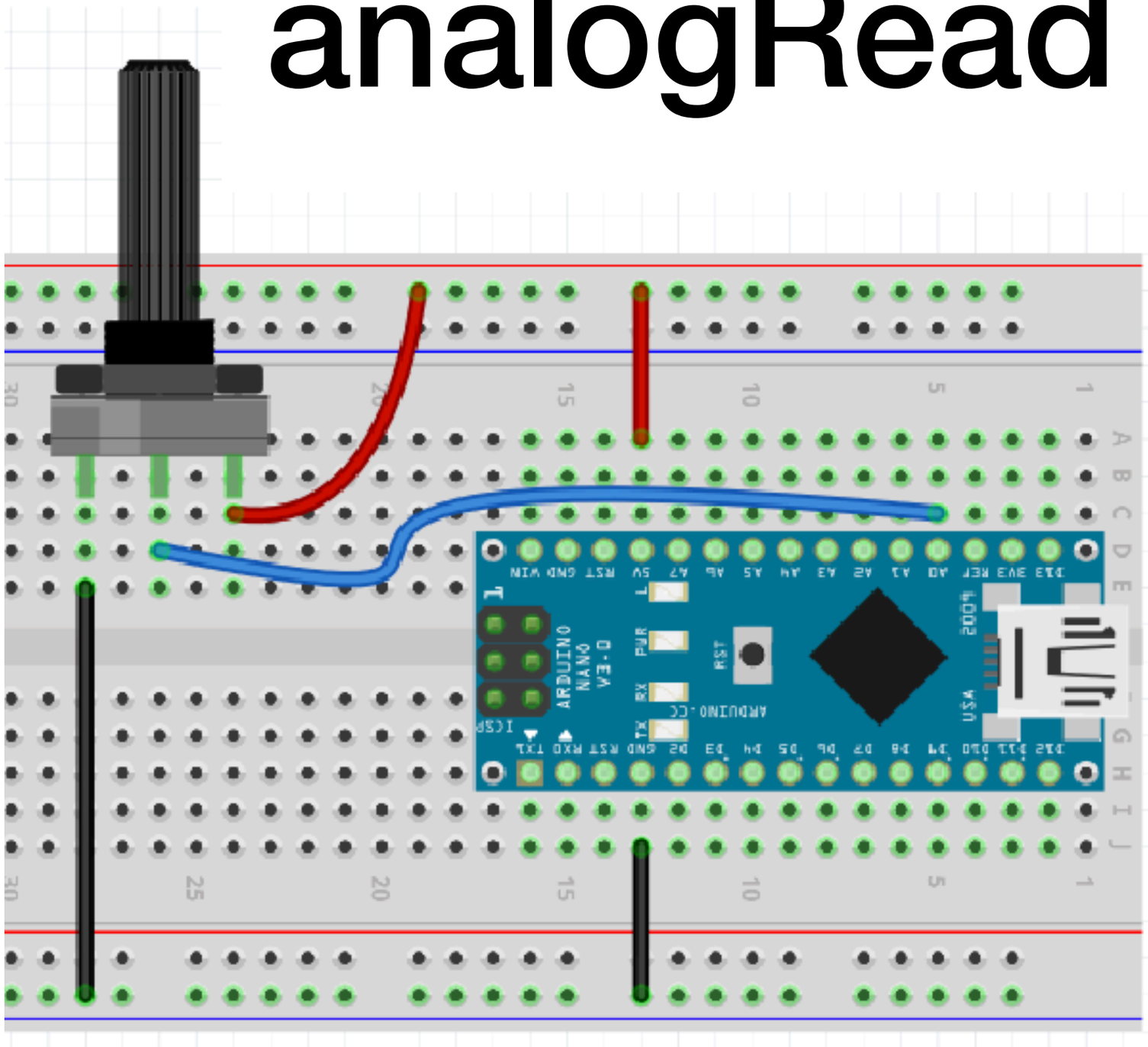
AREF=VREF (<5V no pino AREF)
V = (valor / 1024) * VREF



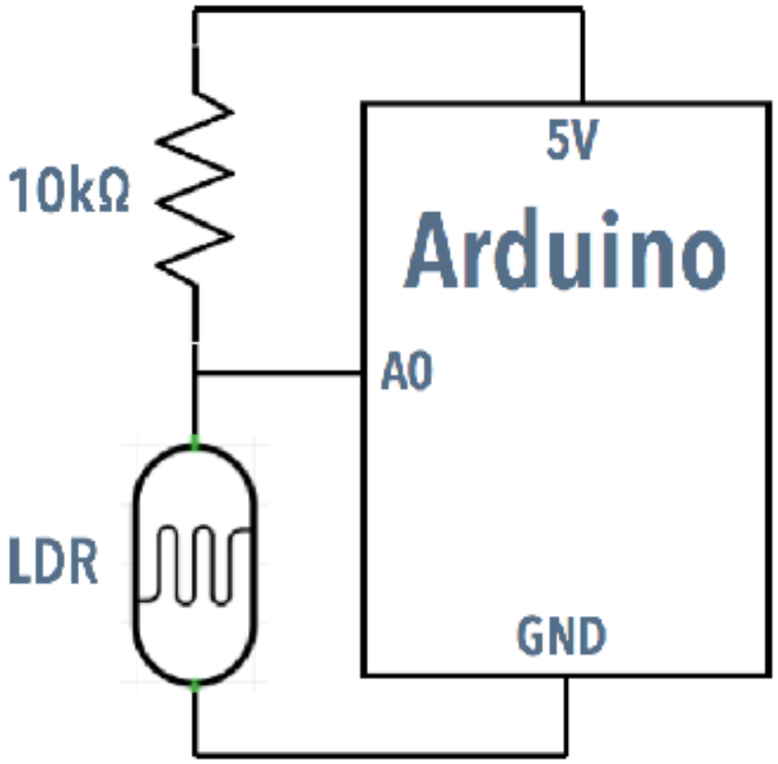


Abra o monitor serial para ver as mensagens com o valor lido no potenciômetro

analogRead



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

void loop() {
   int pot = analogRead(A0);
   Serial.println(pot);
}
```

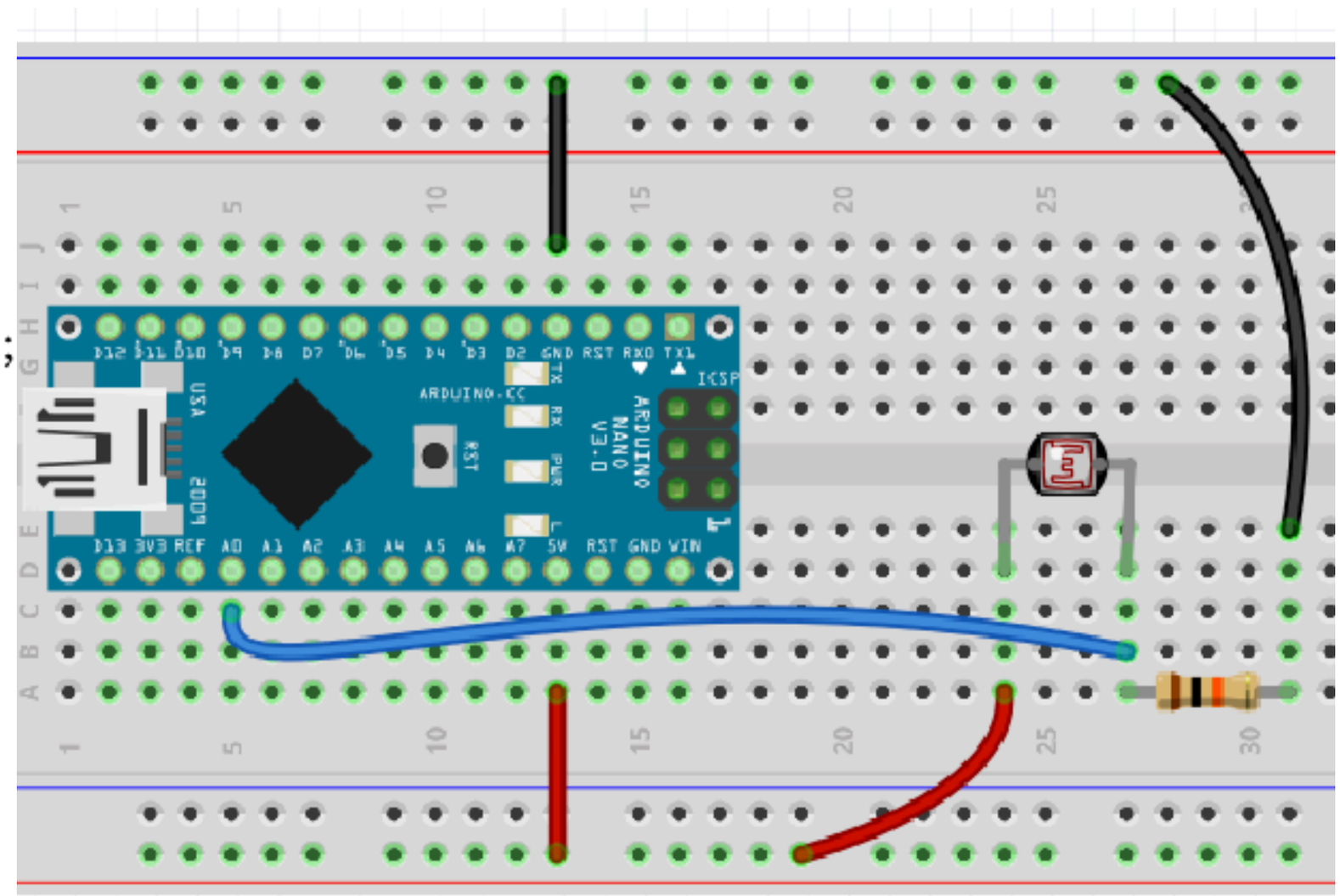


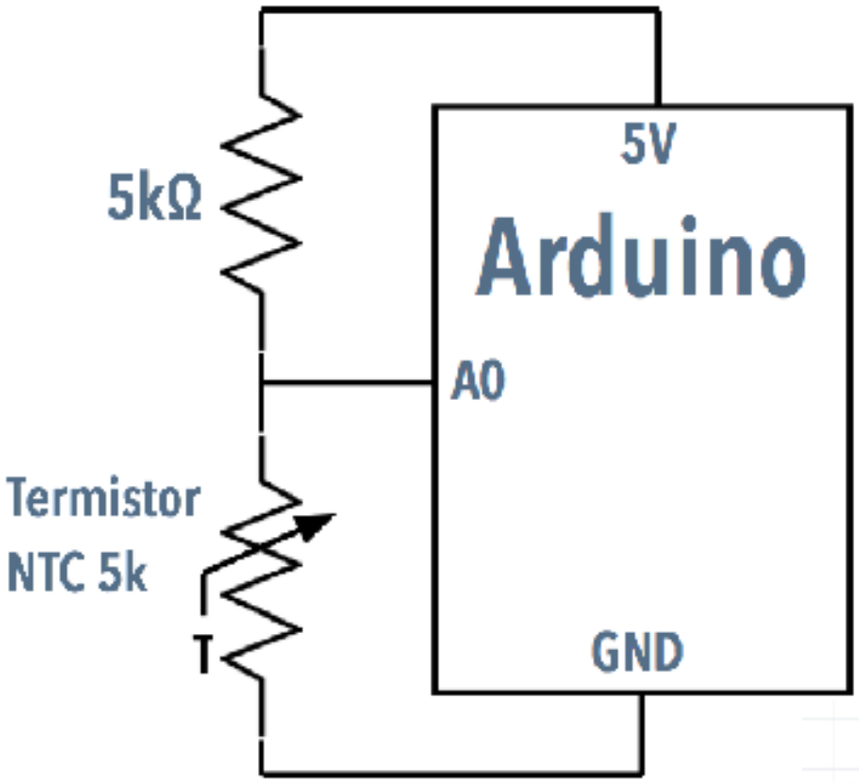
Troque o potenciômetro por um LDR

O **LDR** pode ser conectado a qualquer pino analógico, com um **resistor de pull-up** (se conectado a GND) ou de **pull-down** (se conectado a VCC)

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

void loop() {
  int pot = analogRead(A0);
  float volts = (pot / 1023.0) * 5.0;
  Serial.println(volts);
}
```

A **tensão de referência analógica** é 5V (default) portanto pra calcular a tensão no pino divide-se por 1024 e multiplica-se por 5

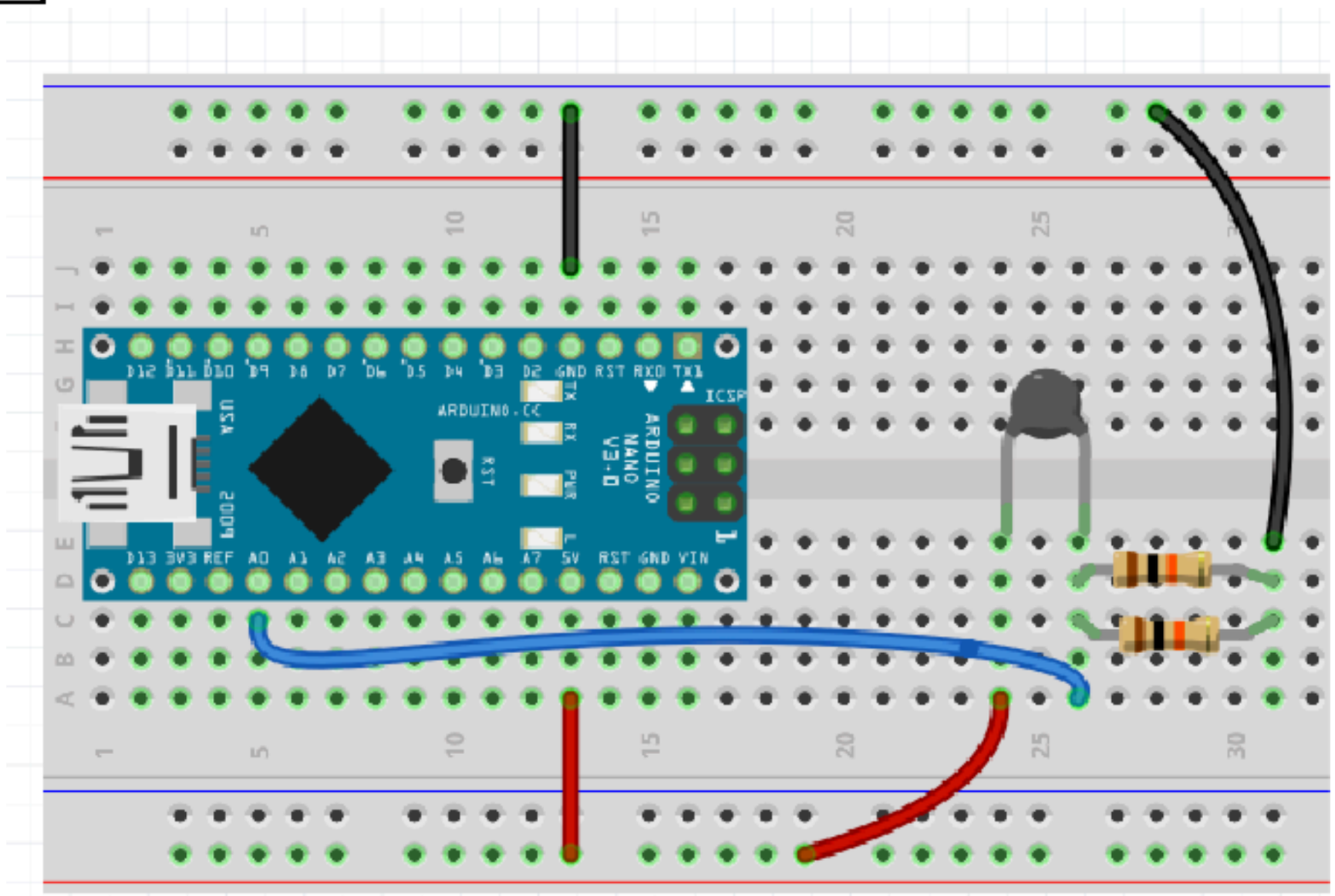




Troque potenciômetro por termistor

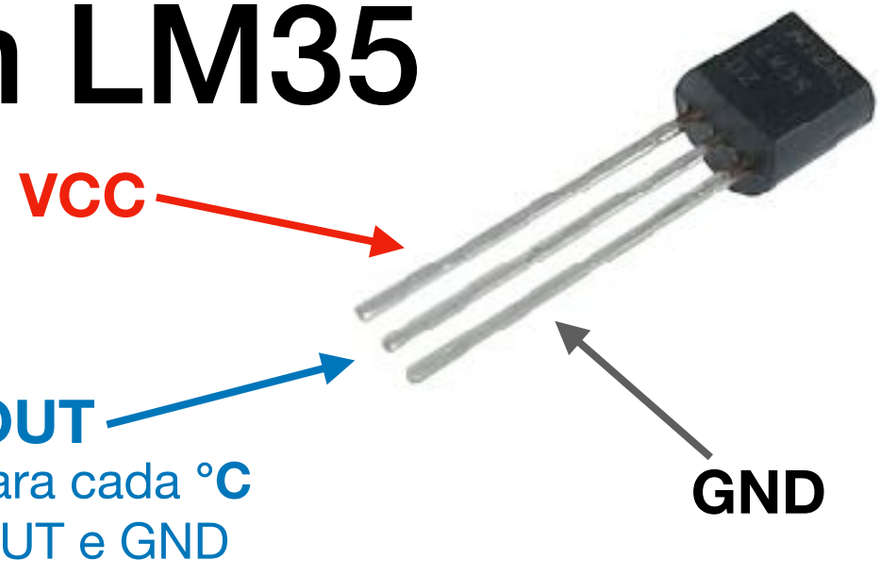
O termistor do kit é NTC 5k (significa que sua temperatura cai com o calor, e que a resistência mede 5k ohms aos 25 graus Celsius)

É preciso calibrar o termistor para determinar a temperatura.

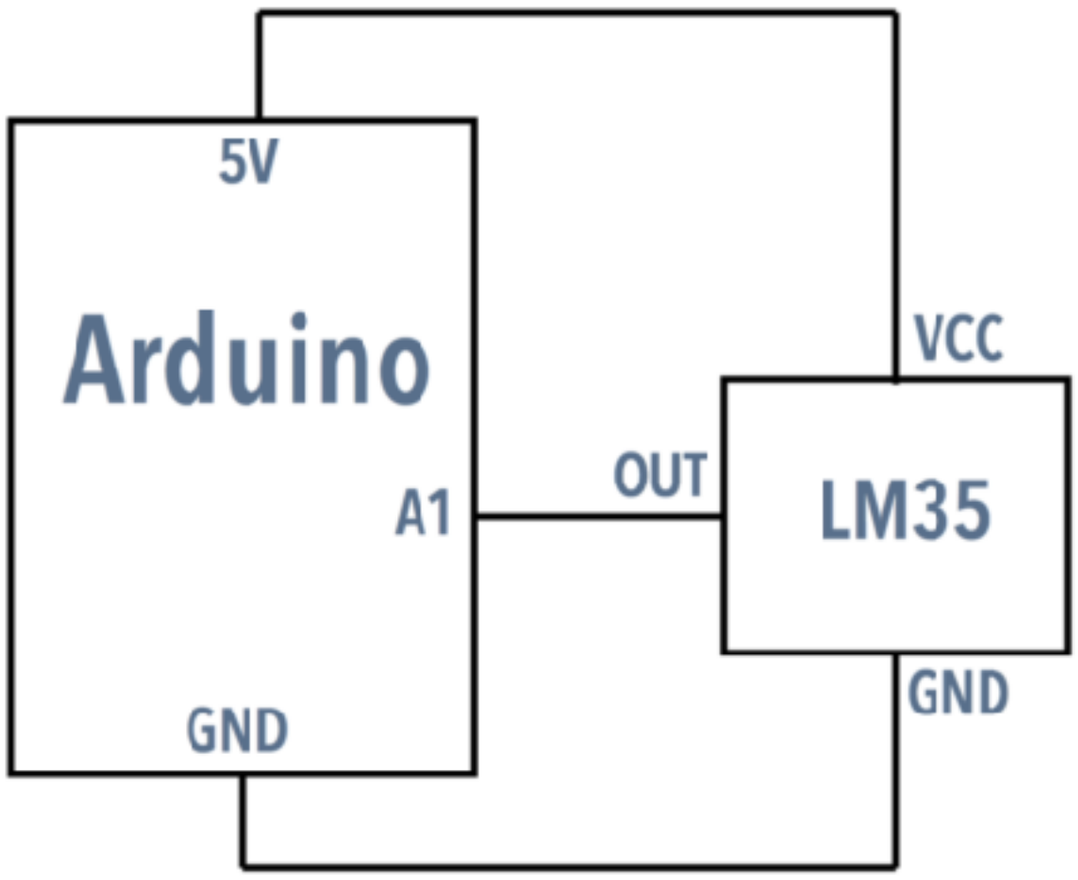


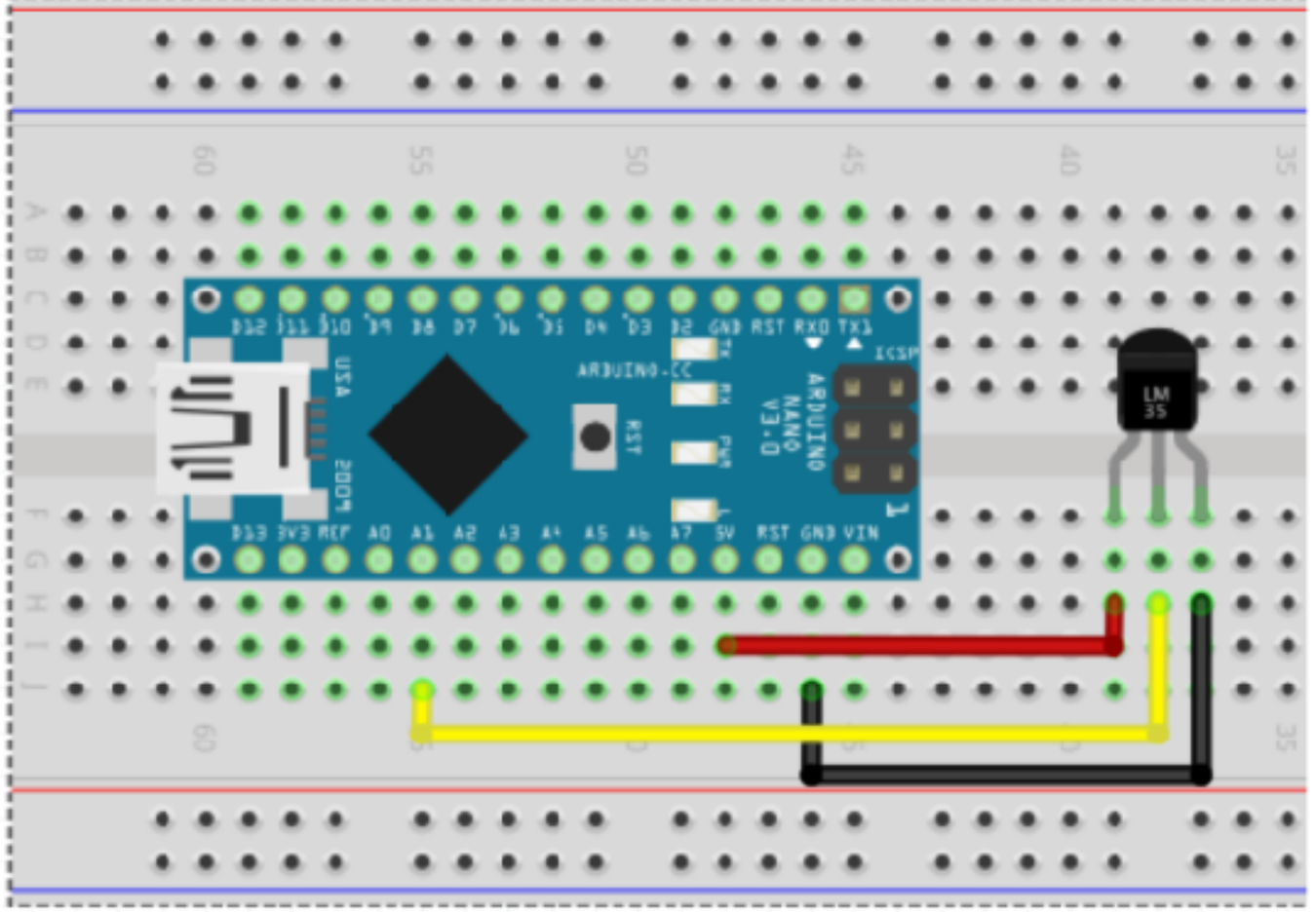
Termômetro analógico com LM35

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

void loop() {
  int leitura = analogRead(A1);
  float volts = (leitura / 1024.0) * 5.0;
  float celsius = (volts) * 100.0;

  Serial.print("Temperatura: ");
  Serial.println(celsius);
  delay(2000);
}
```

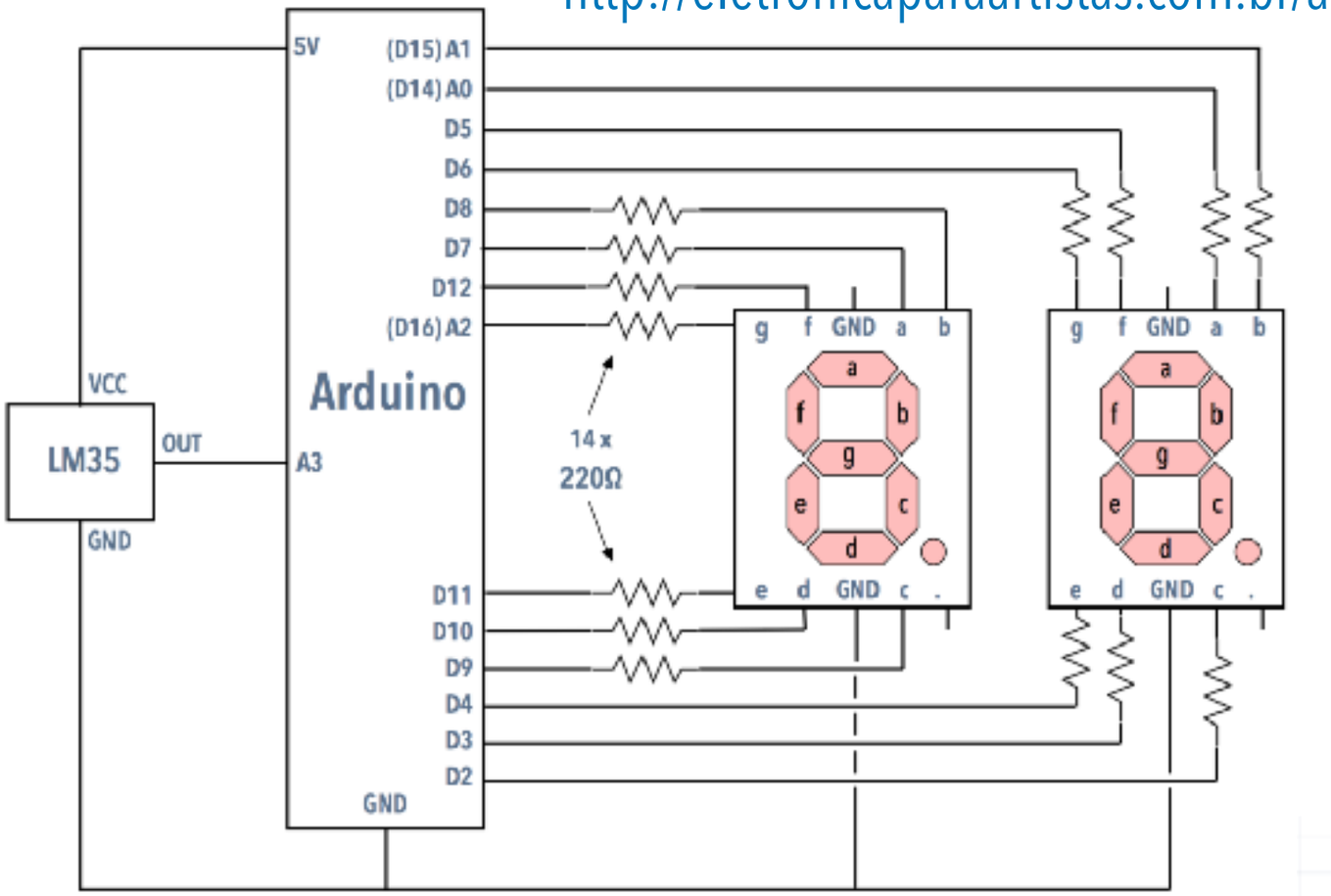


Veja resultados no Monitor Serial



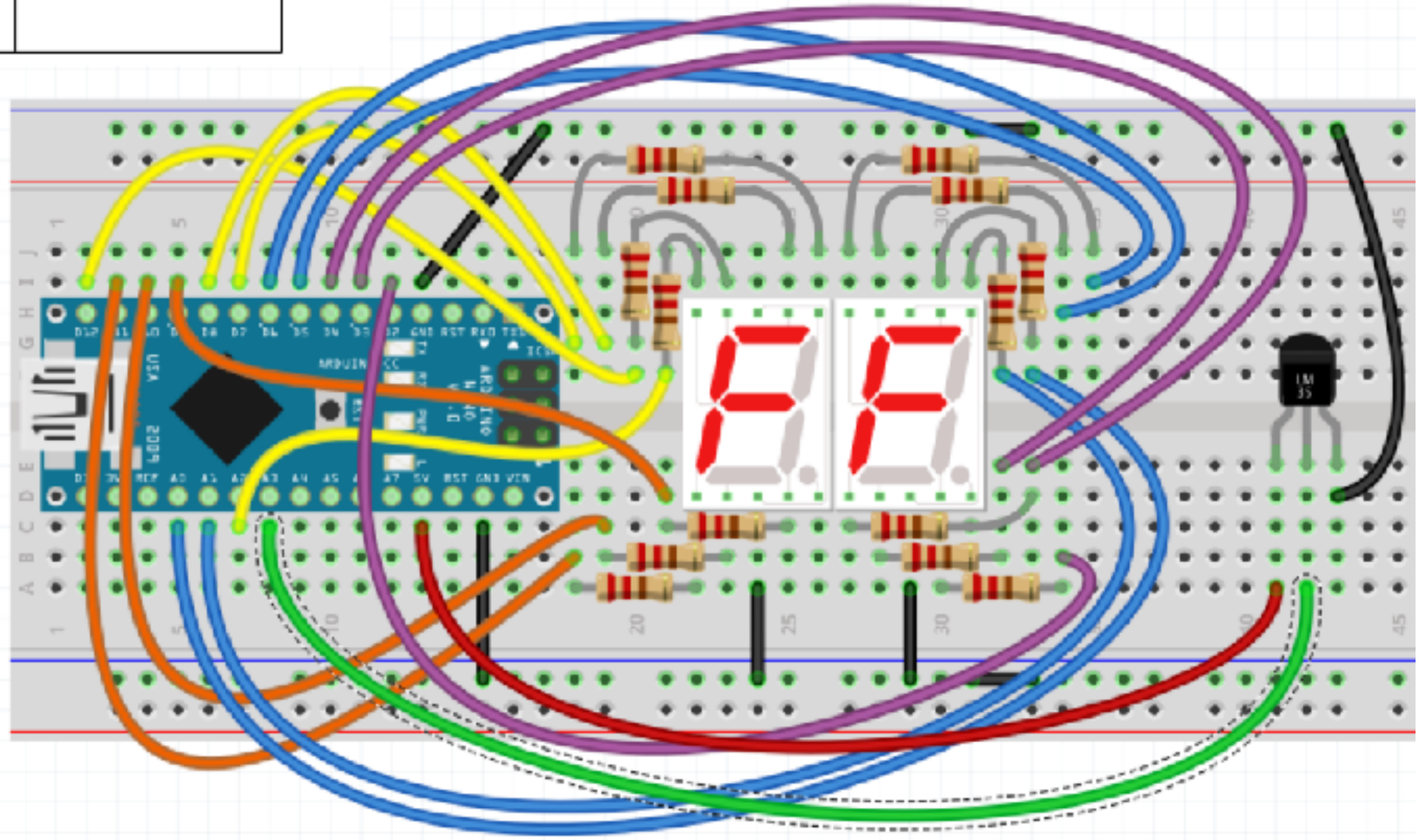


http://eletronicaparaartistas.com.br/alteracao-35-1-termometro-com-display-de-7-segmentos/

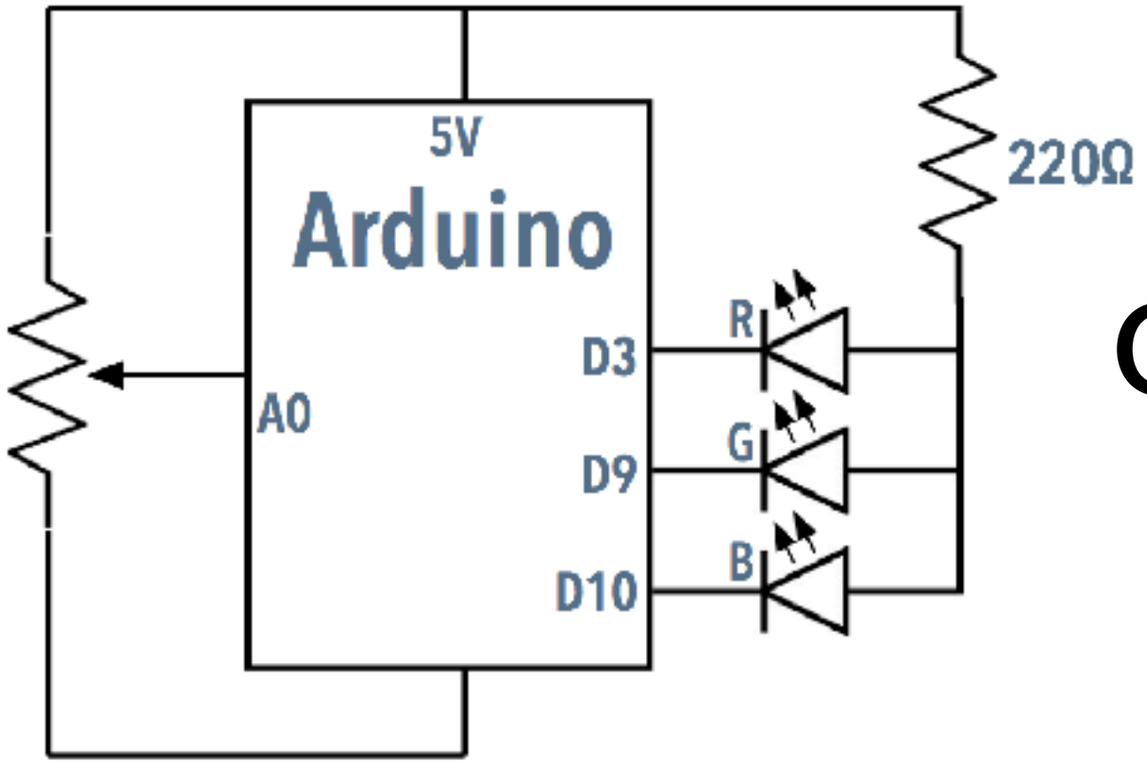


Termômetro usando display de LEDs de 7 segmentos

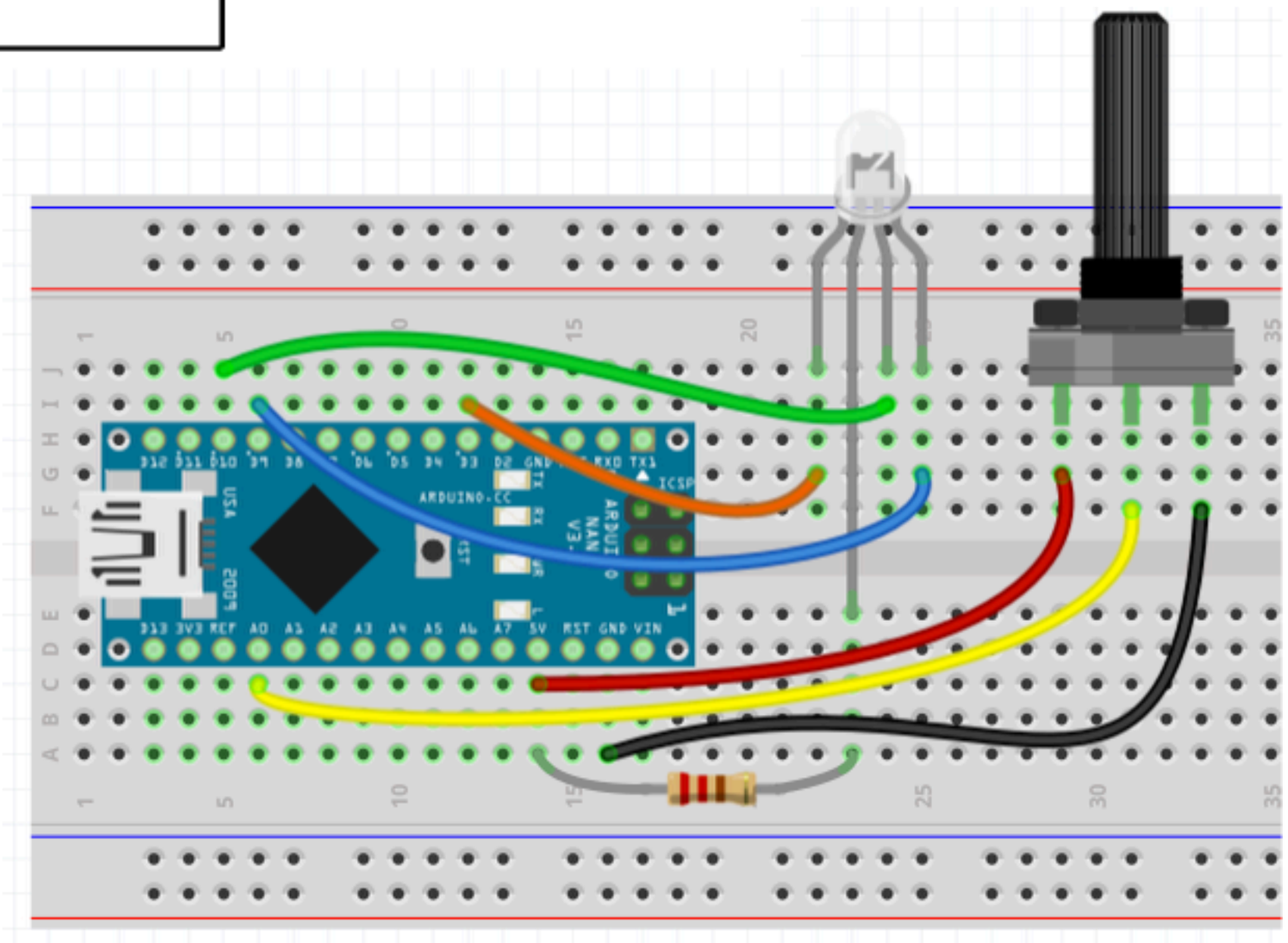
Veja código no link acima ou baixe do GitHub

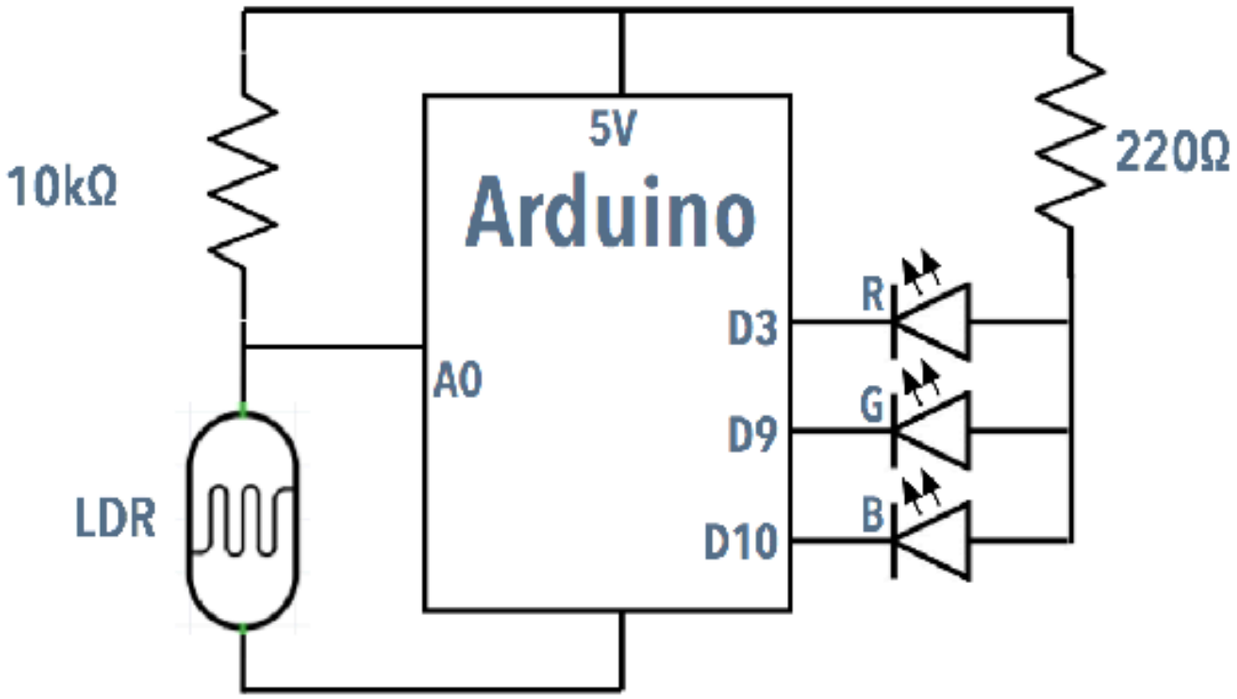


```ino
int R = 3;
int G = 10;
int B = 9;
int POT = A0;

void setup() {
  pinMode(R, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(B, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  delay(100);
  int pot = analogRead(POT);

  if(pot <= 511) {
    analogWrite(R, pot/2);
  } else {
    analogWrite(R, 255);
  }

  if(pot > 256 && pot < 768) {
    analogWrite(G, abs(512 - pot));
  } else {
    analogWrite(G, 255);
  }

  if(pot >= 512) {
    analogWrite(B, (1023 - pot)/2);
  } else {
    analogWrite(B, 255);
  }
}
```



Controle de LED RGB

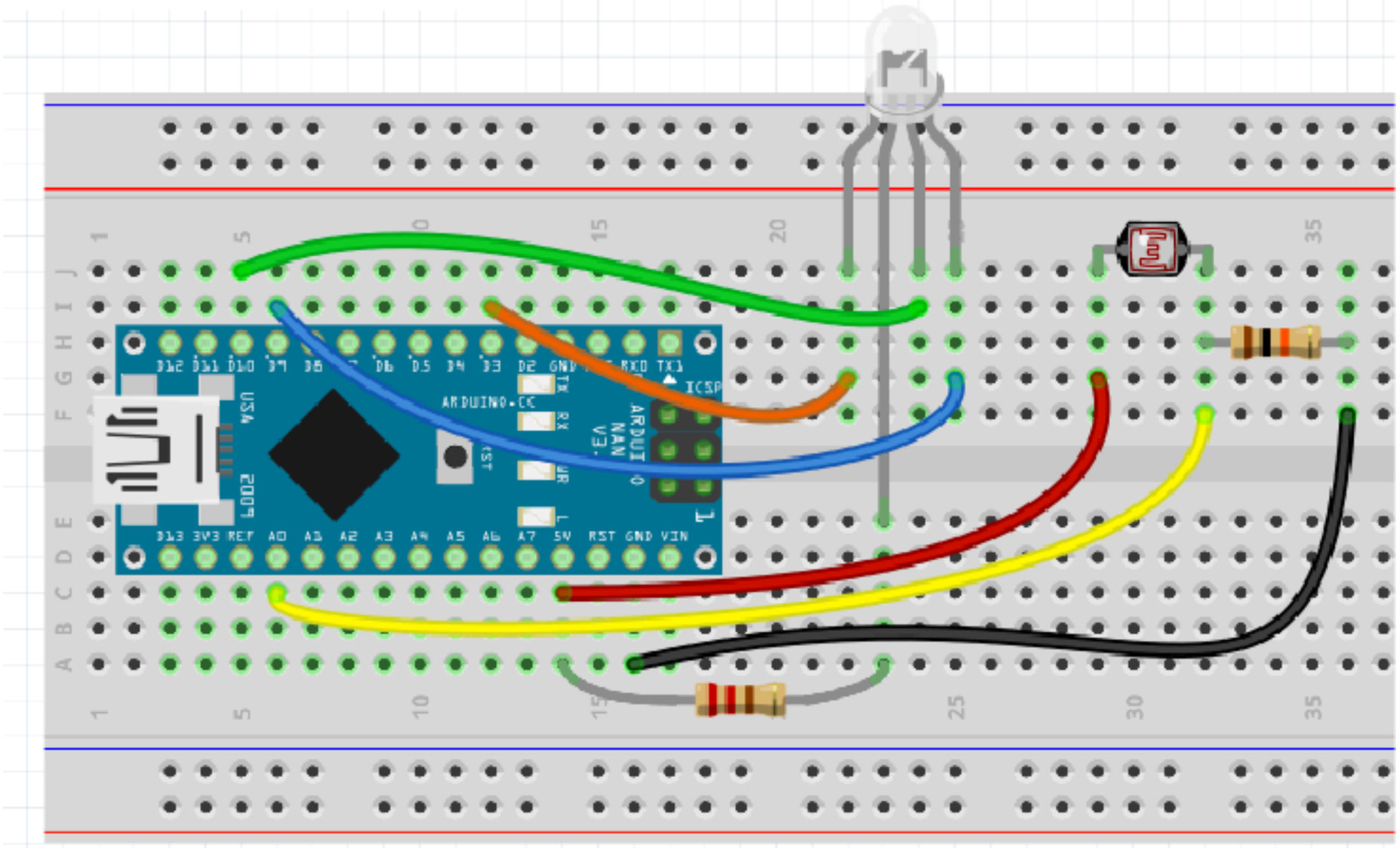




Sensor de luminosidade com LED RGB

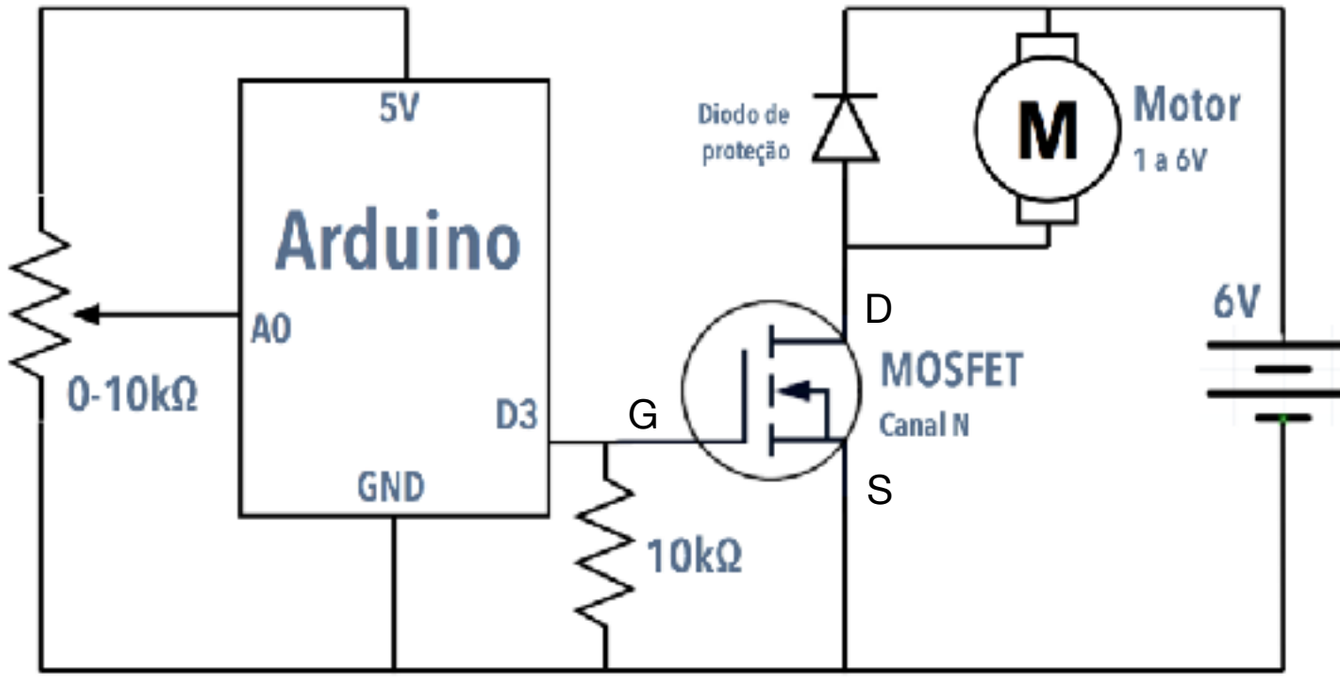
Troque o potenciômetro por um divisor de tensão com LDR

LED ficará vermelho com pouca luz e azul com muita luz. Ficará verde com luz intermediária

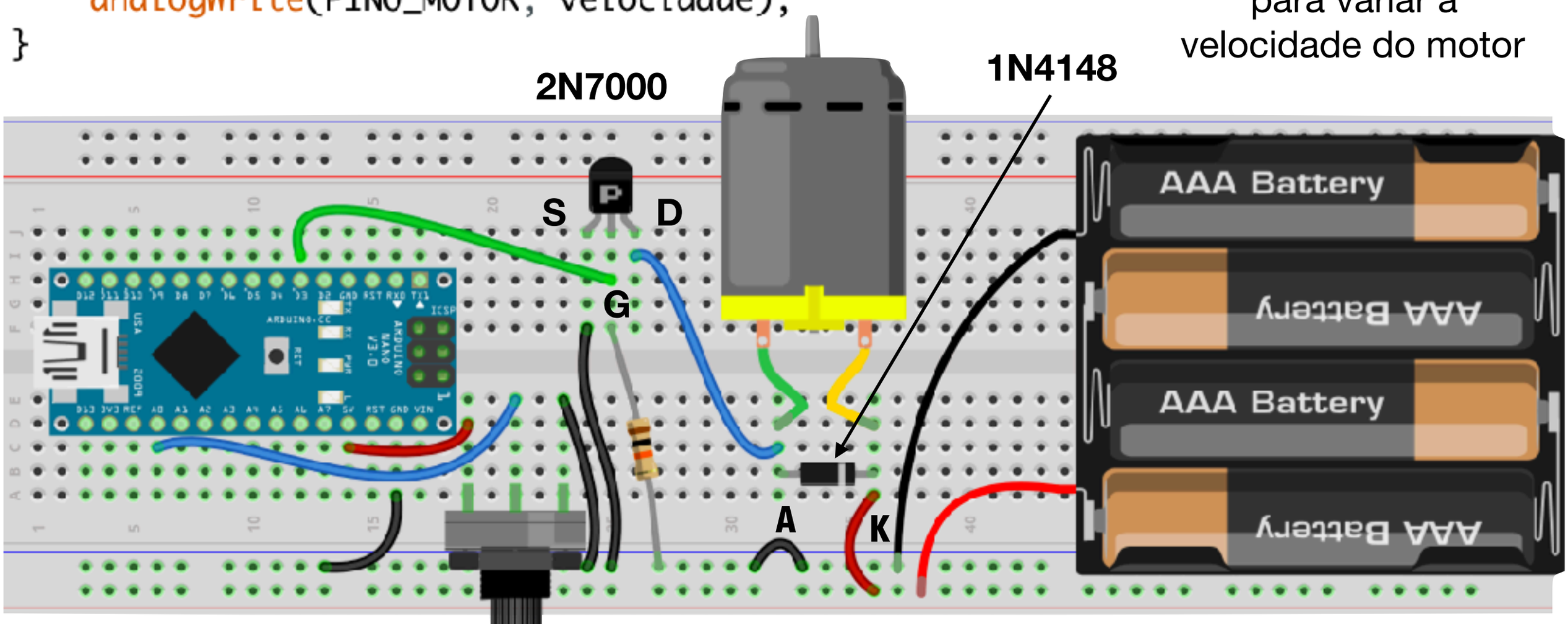


Controle de motor com PWM

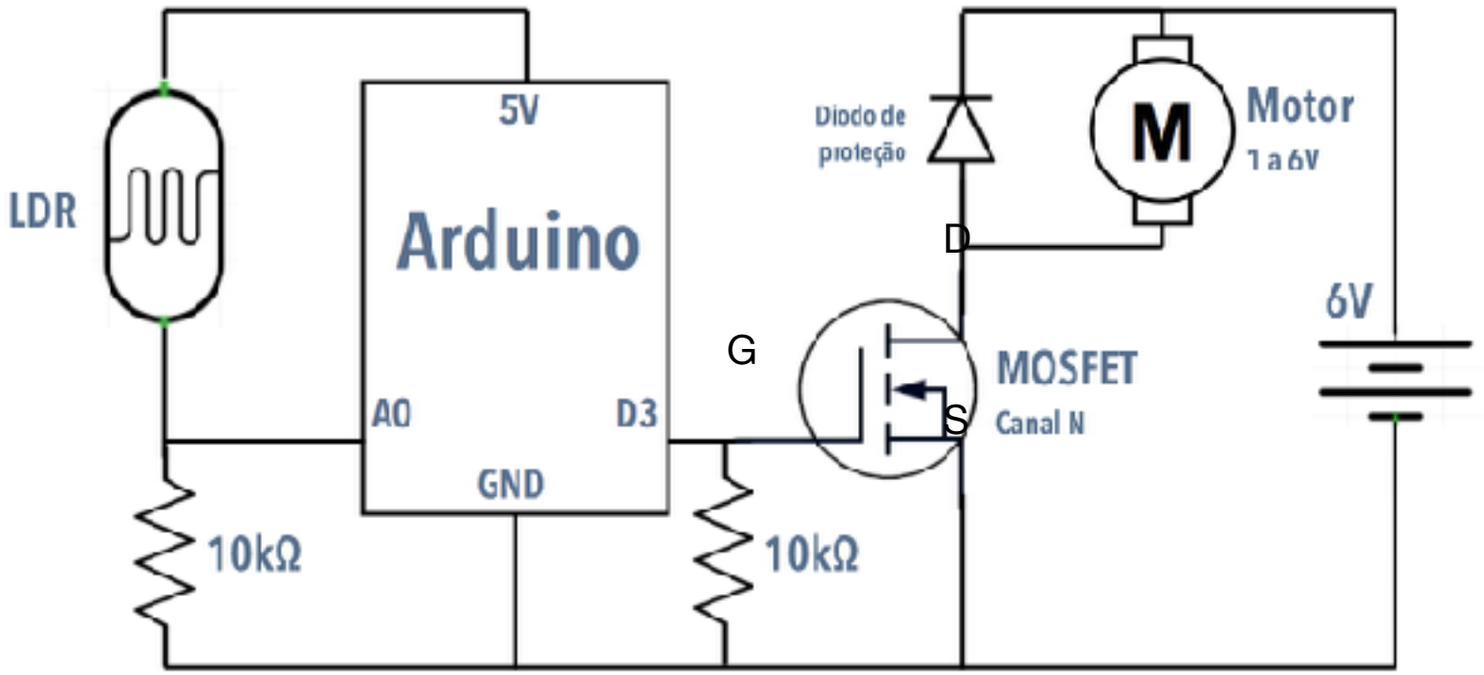
```
#define PINO_MOTOR 3;
#define PINO_POT A0;

void setup() {
  pinMode(PINO_MOTOR, OUTPUT);
}

void loop() {
    int velocidade = analogRead(PINO_POT) / 4;
    analogWrite(PINO_MOTOR, velocidade);
}
```



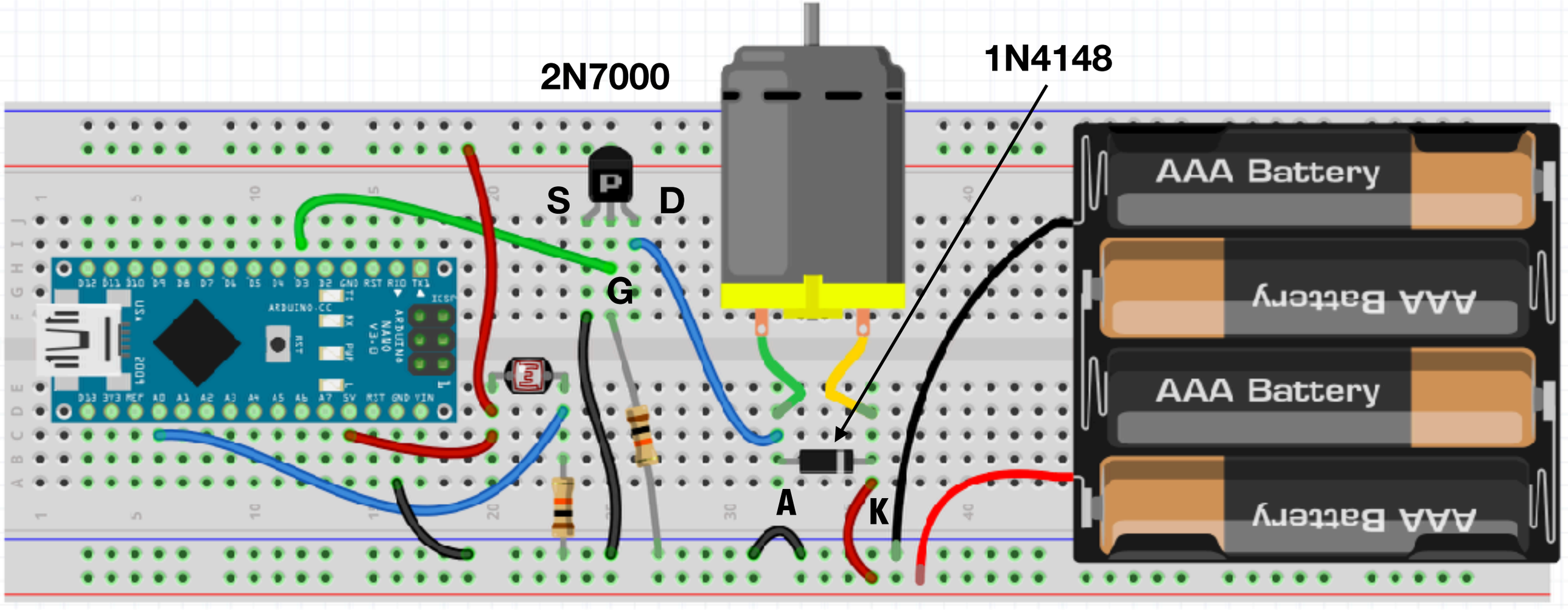
Gire o potenciômetro para variar a velocidade do motor

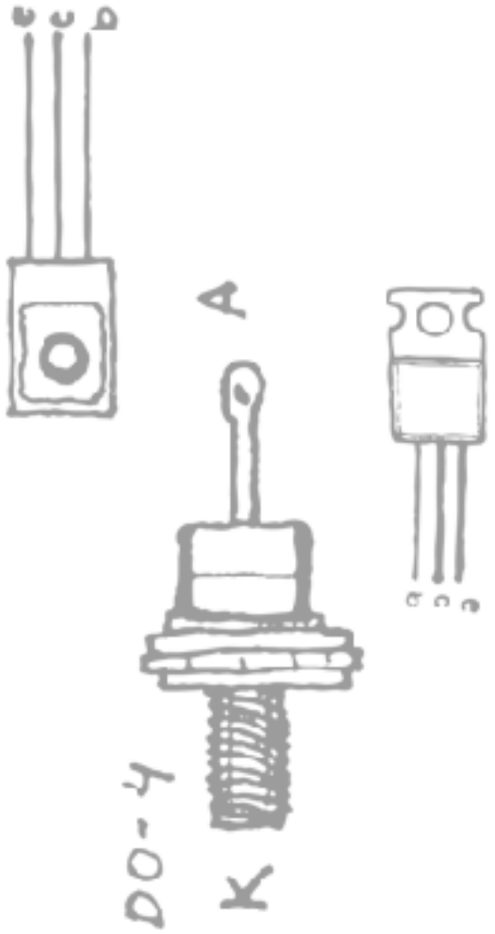


Troque o potenciômetro por um LDR

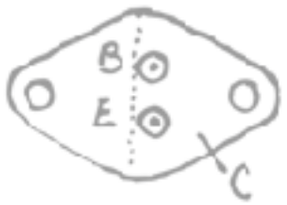


Motor gira mais rápido quando está claro e mais devagar quando está escuro (ou vice-versa - é só mudar no código)





ARDUINO 6



Dispositivos controlados por pulsos
Geração de pulsos
Leitura de pulsos

Helder da Rocha
www.eletronicaparaartistas.com.br

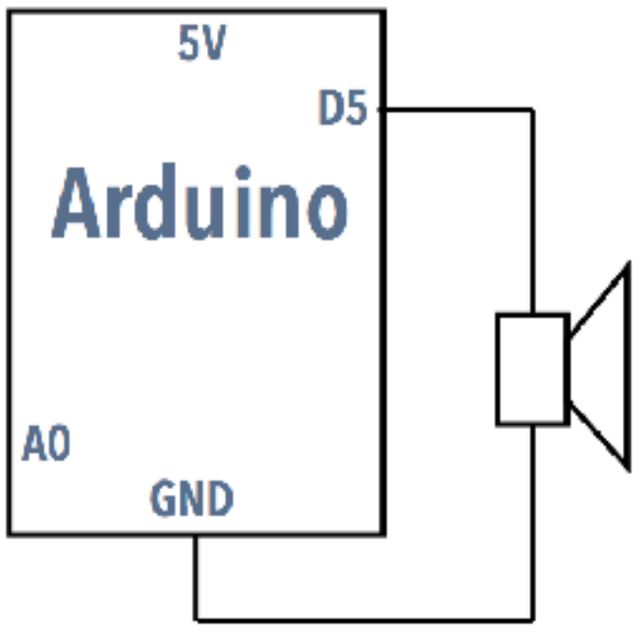
tone(pino, frequencia)

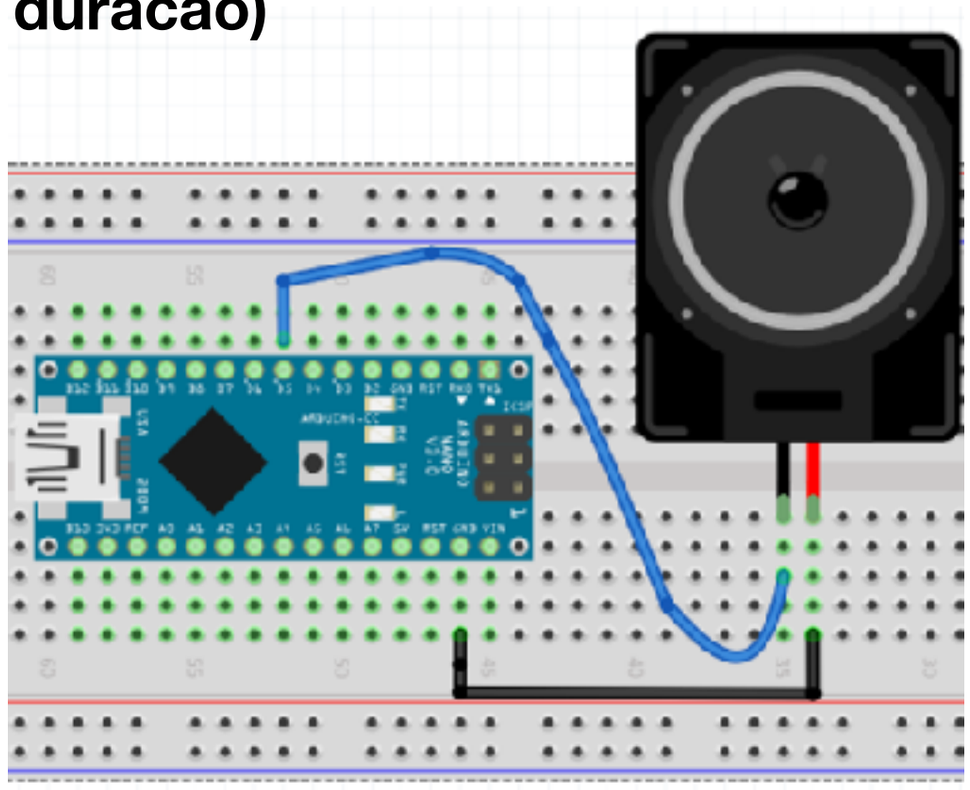
Gera pulsos de ondas quadradas
em determinada **frequência**
(em Hertz - pulsos por segundo)

- O Arduino gera apenas um tom de cada vez, que continua até que haja uma instrução **noTone(pino)**

```
void setup() {
  tone(5, 392, 250);
  delay(260);
  tone(5, 392, 250);
  delay(260);
  tone(5, 392, 250);
  delay(260);
  tone(5, 311, 1000);
  delay(1250);
```

- Duração também pode ser especificada (em ms):
  tone(pino, frequencia, duracao)





```
  tone(5, 349, 250);
  delay(260);
  tone(5, 349, 250);
  delay(260);
  tone(5, 349, 250);
  delay(260);
  tone(5, 294, 1000);
  delay(1250);
}

void loop() {}
```

Gerando pulsos

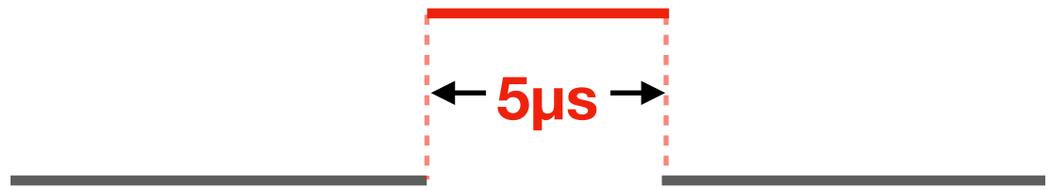
Diversos dispositivos e sensores são controlados por **pulsos.**
Um pulso é um **sinal** com **duração** especificada no **estado oposto** ao inicial/final.
Pode ser gerado usando **digitalWrite() + delay() ou delayMicroseconds()**

- Pulso **LOW** no pino 12 com **5ms** de duração (estado inicial/final é **HIGH**)

```
digitalWrite(12, LOW);
delay(5);
digitalWrite(12, HIGH);
```

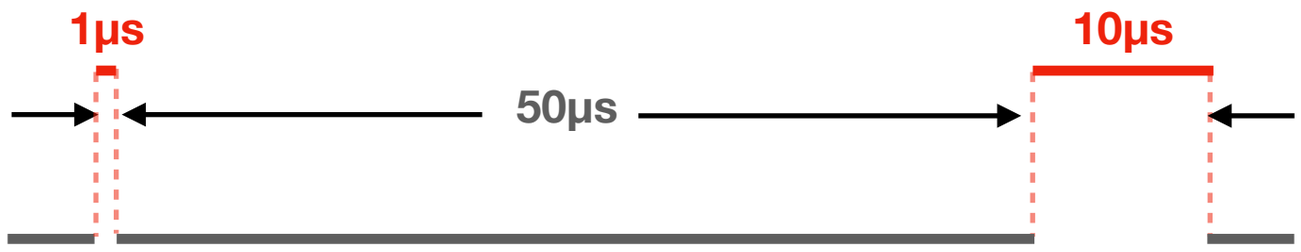
- Pulso **HIGH** no pino 7 com **5µs** de duração (estado inicial/final é **LOW**)

```
digitalWrite(7, HIGH);
delayMicroseconds(5);
digitalWrite(7, LOW);
```



- Dois pulsos **HIGH** no pino 9, com **1µs** e **10µs**, com intervalo de **50µs**

```
digitalWrite(9, HIGH);
delayMicroseconds(1);
digitalWrite(9, LOW);
delayMicroseconds(50);
digitalWrite(9, HIGH);
delayMicroseconds(5);
digitalWrite(9, LOW);
```



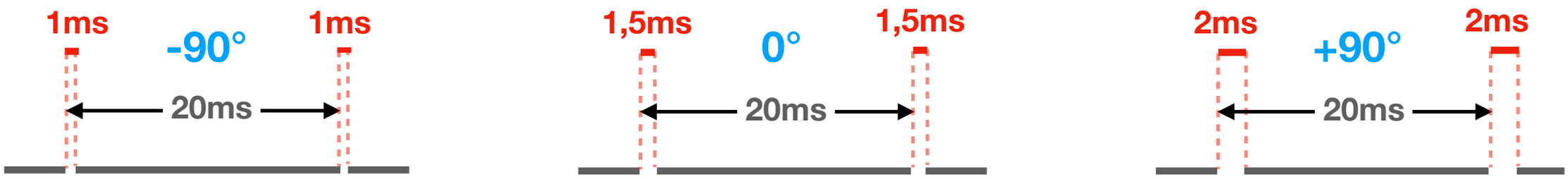
Controle com PWM

- **PWM** - Pulse Width Modulation é a produção de pulsos contínuos variando a largura do pulso HIGH.

- **analogWrite(pino, largura)** varia o **duty-cycle**, que é a **proporção** da largura do pulso HIGH em relação ao LOW = 0 (0%) a 255 (100%) e independe da frequência

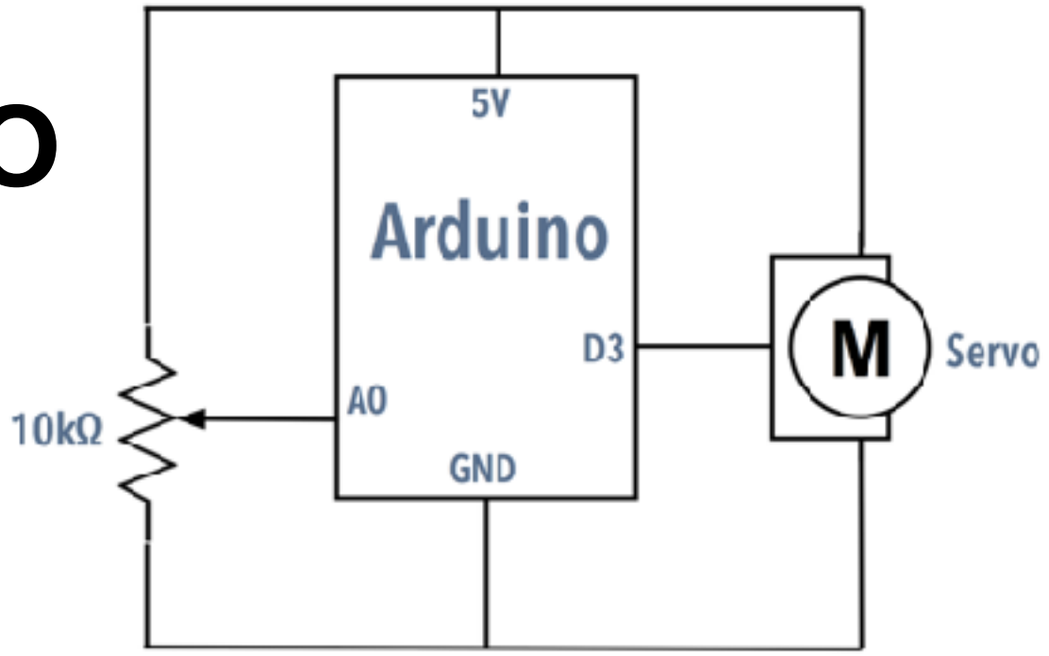
- Controle de dispositivos como **servos** usam outra forma de PWM que independe do duty-cycle e utiliza pulsos com **largura específica** (não pode ser feito com analogWrite)

- Para controlar um **servo motor** típico é preciso gerar no máximo um pulso **HIGH** a cada 20ms com largura variando de 1 a 2ms, onde **1ms** = -90 graus, **2ms** = +90 graus e **1,5ms** = 0 graus (centro)



Controle de um Servo

A forma mais fácil de controlar um servo é
incluir a **biblioteca Servo.h**
disponibilizada pelo portal arduino.cc



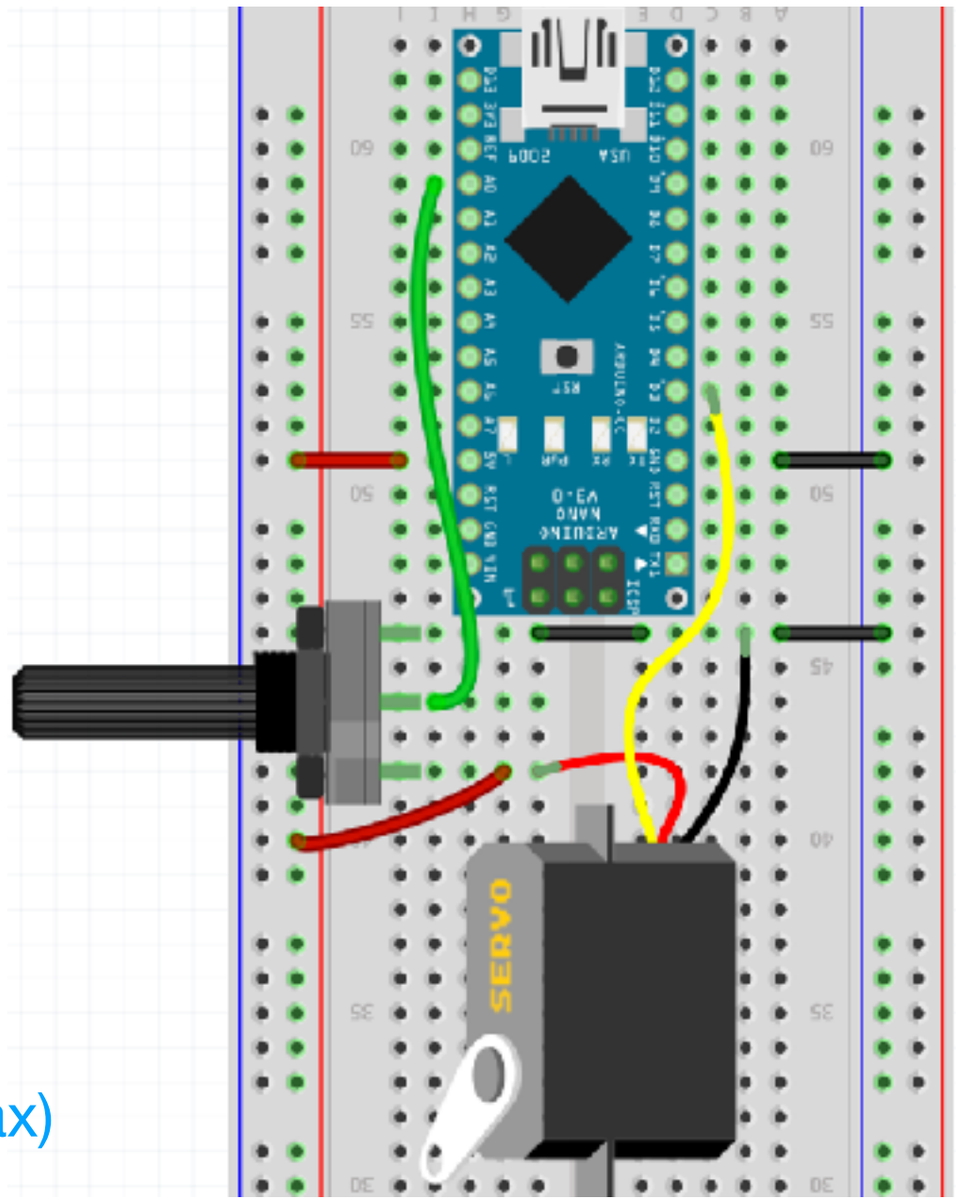

```
#include <Servo.h>

Servo servo;

void setup() {
  servo.attach(3);
  servo.write(90); // centraliza
}

void loop() {
  int pot = analogRead(A0); // 0 a 1023
  int posicao = map(pot, 0, 1023, 0, 179);
  servo.write(posicao);
  delay(100);
}
```

Inclui a biblioteca

Declara objeto (variável)

Informa pino PWM onde está o servo

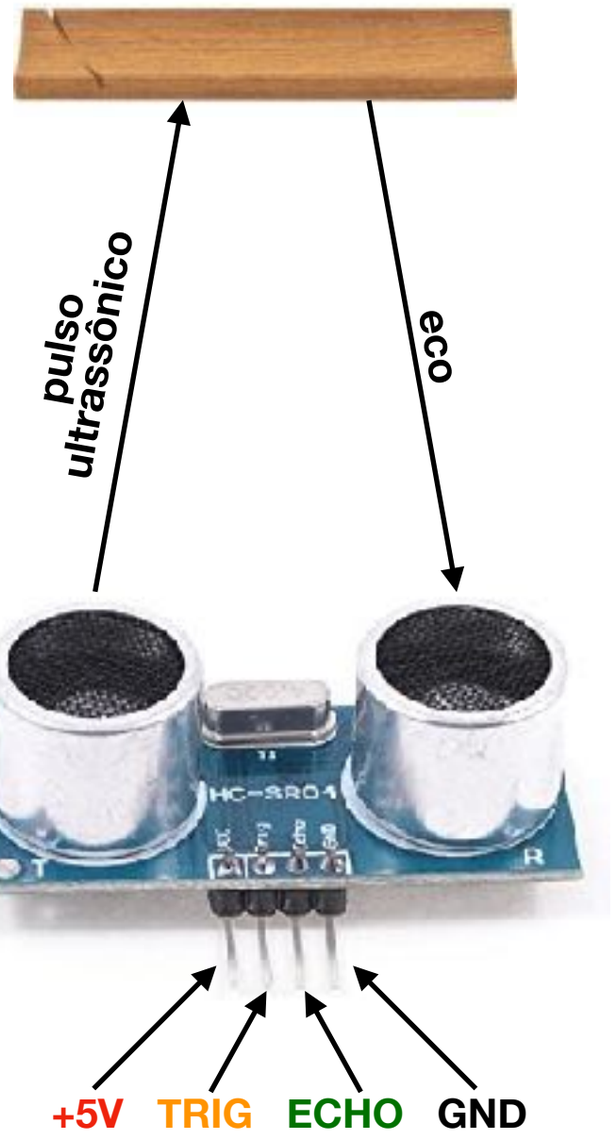
Gira 0 a 180 graus (0 = min, 90 = centro, 179 = max)

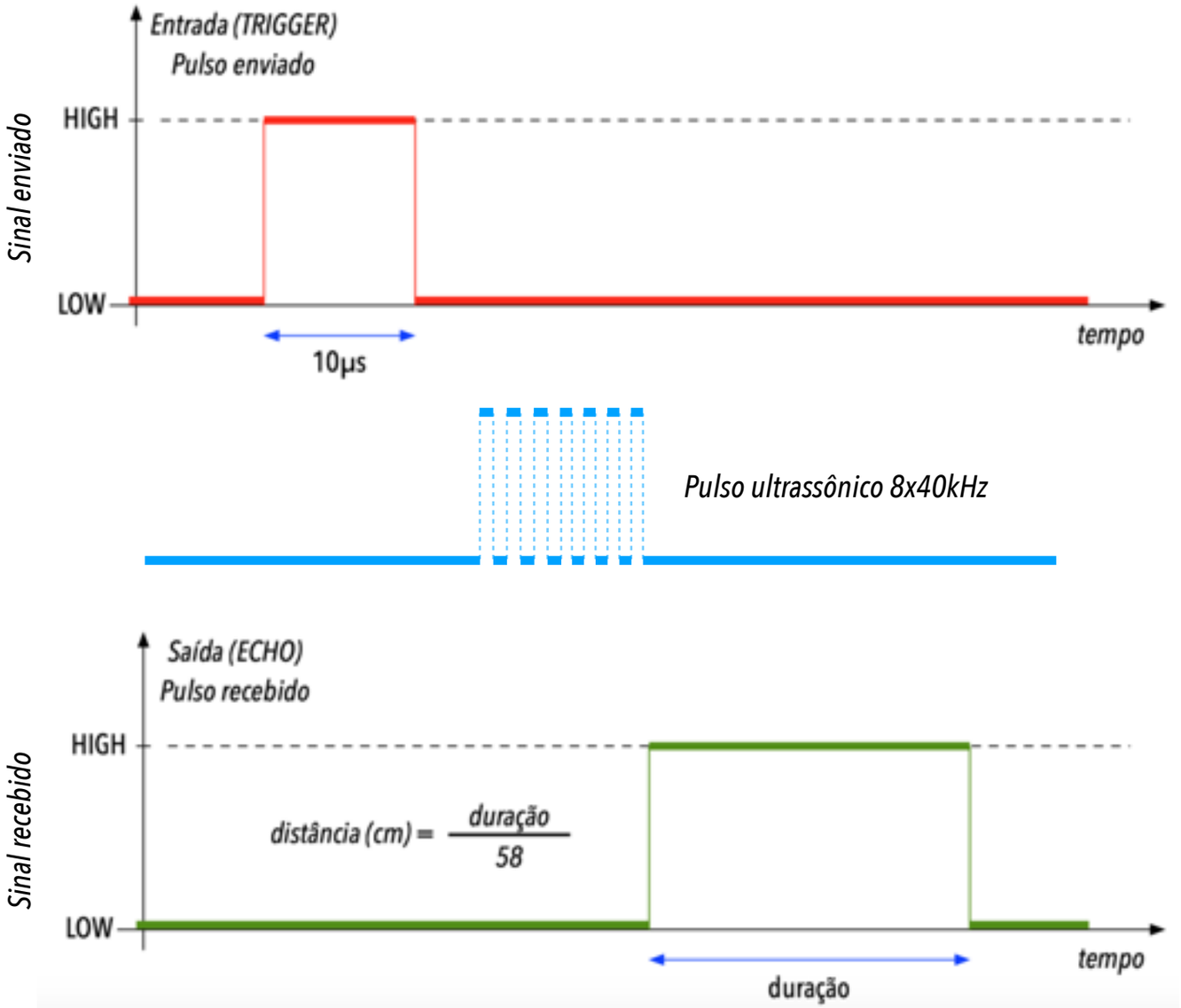


Sensor ultrassônico HC-SR04

O sensor é ativado através do envio de um pulso HIGH de 10µs ao pino **TRIG**, provocando o envio de um **pulso ultrassônico** (a 40kHz) que deverá refletir em uma superfície situada entre 1cm e 2m de distância.

O **eco** do pulso é recebido pelo sensor que mede o intervalo de tempo entre o envio e o recebimento. Um pulso HIGH com duração proporcional à distância é enviado ao pino **ECHO**.





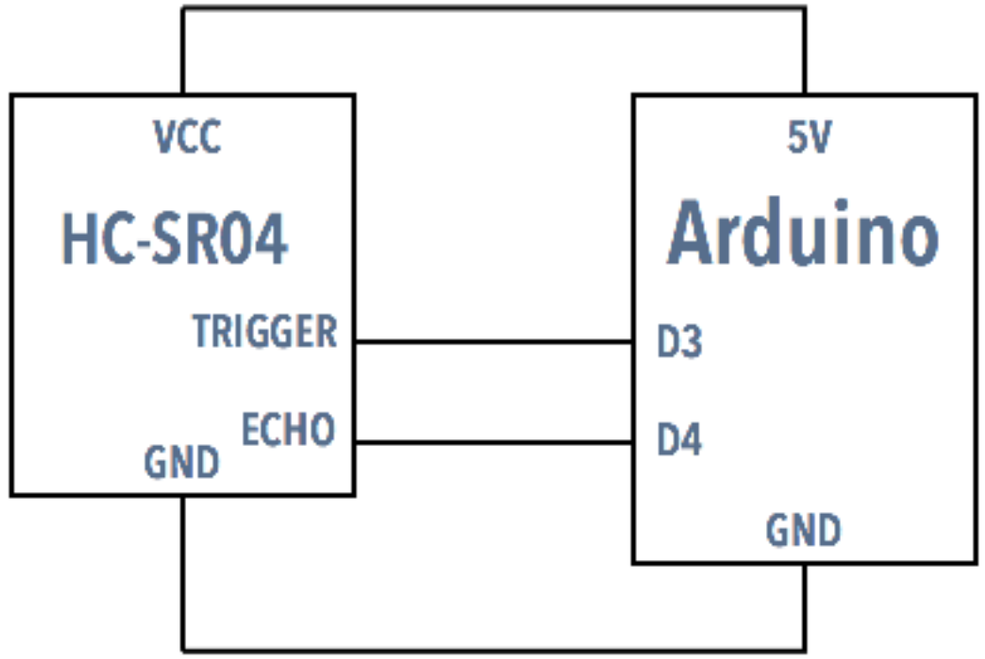
Para ler a duração de um pulso em µs:

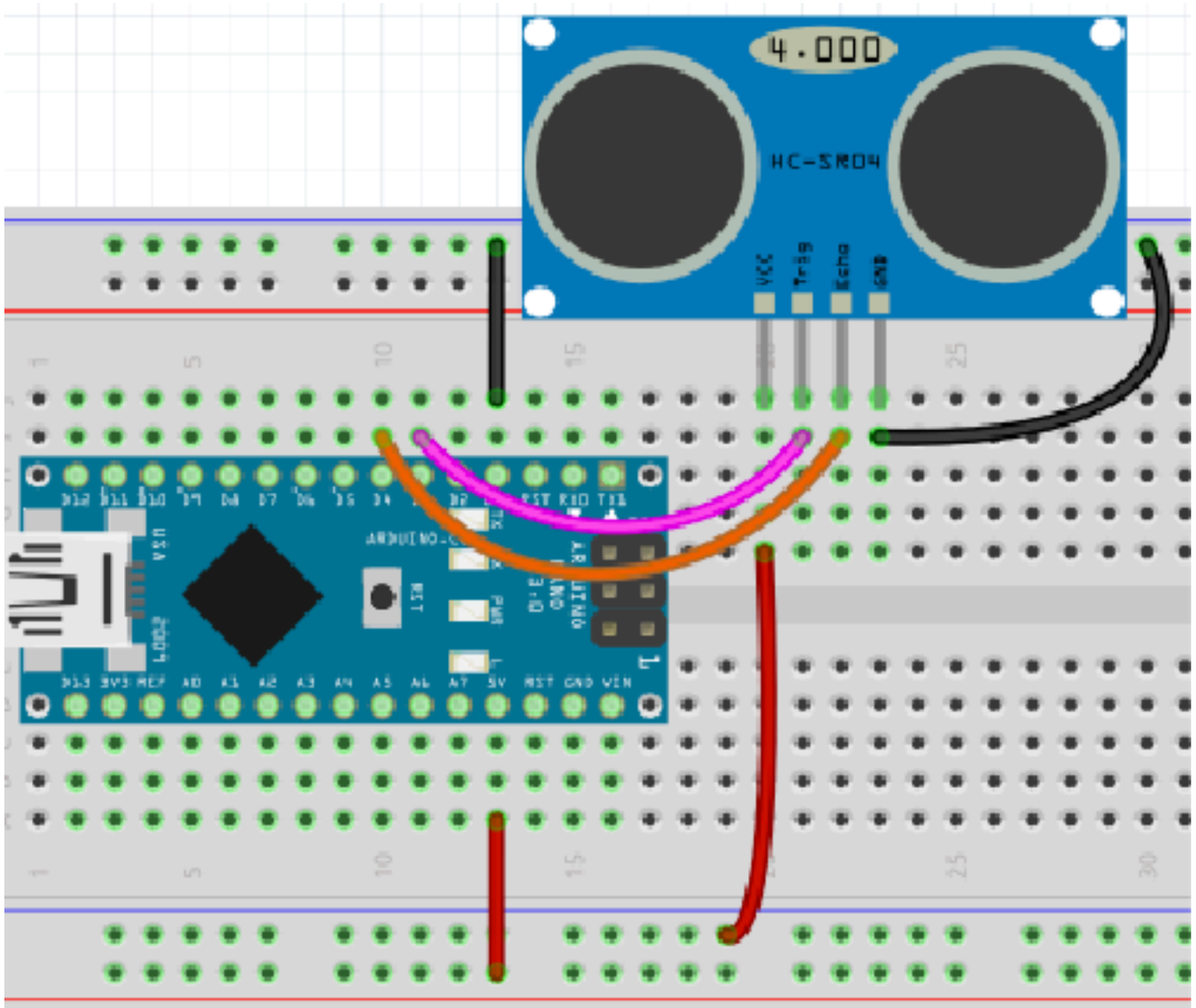
```
pulseIn(pino, nível)
pulseIn(pino, nível, timeout_µs)
```

Sensor de distância

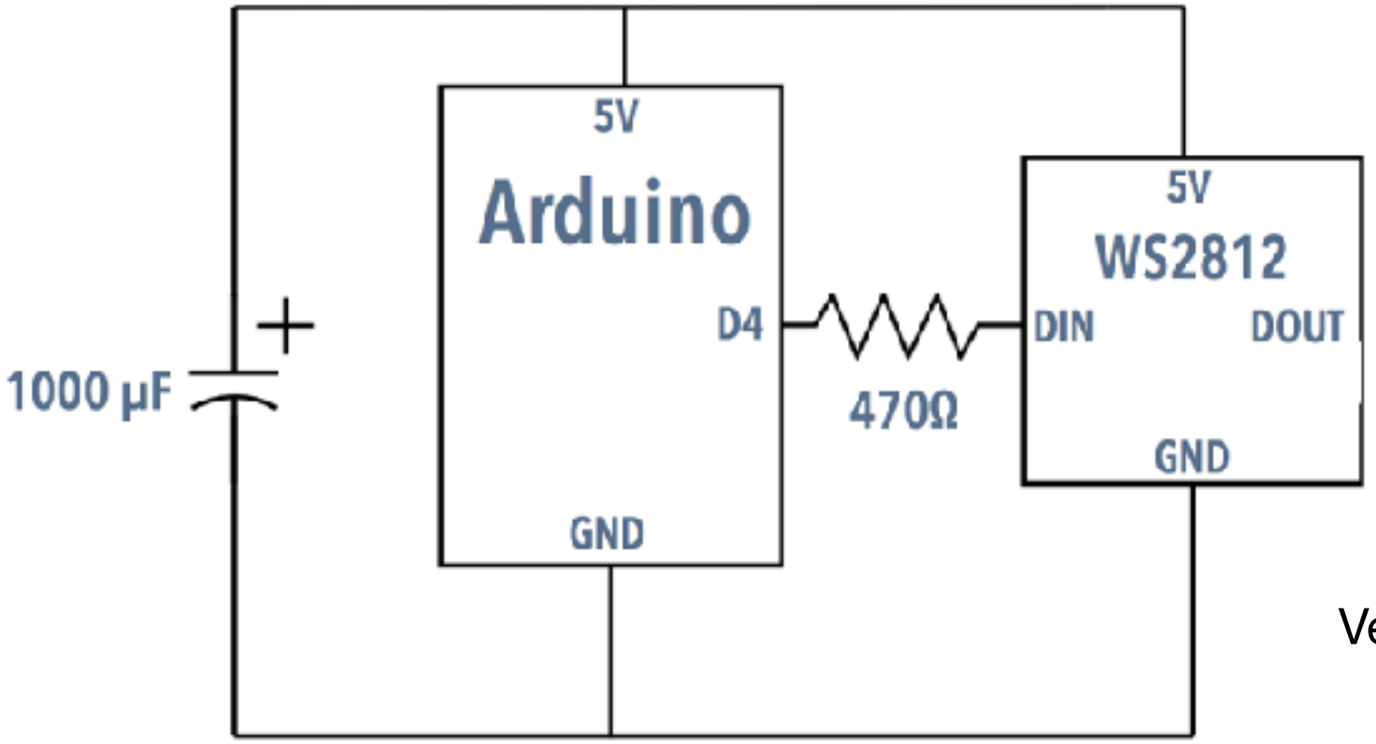
```ino
const int TRIGGER = 3;
const int ECHO = 4;

void setup() {
  pinMode(TRIGGER, OUTPUT);
  pinMode(ECHO, INPUT);
  Serial.begin(9600);
}

void loop() {
  float distance = pulse();
  Serial.println(distance);
  delay(500); // must be over 60ms
}

float pulse() {
  digitalWrite(TRIGGER, LOW);
  delayMicroseconds(10);
  digitalWrite(TRIGGER, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER, LOW);

  long duration = pulseIn(ECHO, HIGH);
  return duration / 58.0;
}
```

Veja resultados no Monitor Serial





Instrução **`pulseIn(pino, nível)`** *retorna* ***duração*** *do pulso em* ***µs*** *(ou 0 se estourar o timeout)*



LEDs endereçáveis

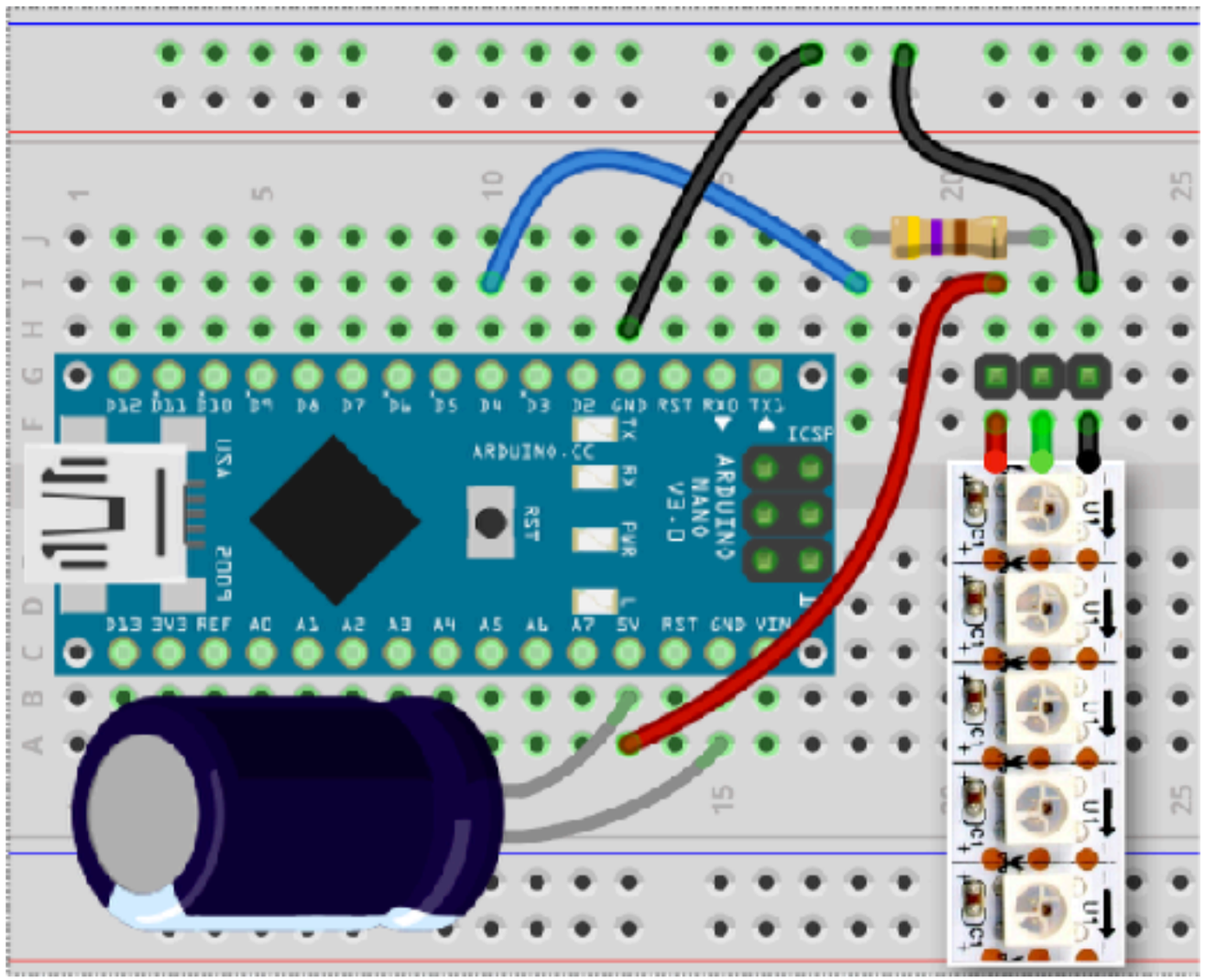
LEDs **RGB WS2812** são pixels endereçáveis
Veja como usar usando exemplos das bibliotecas
FastLED e AdaFruit NeoPixel

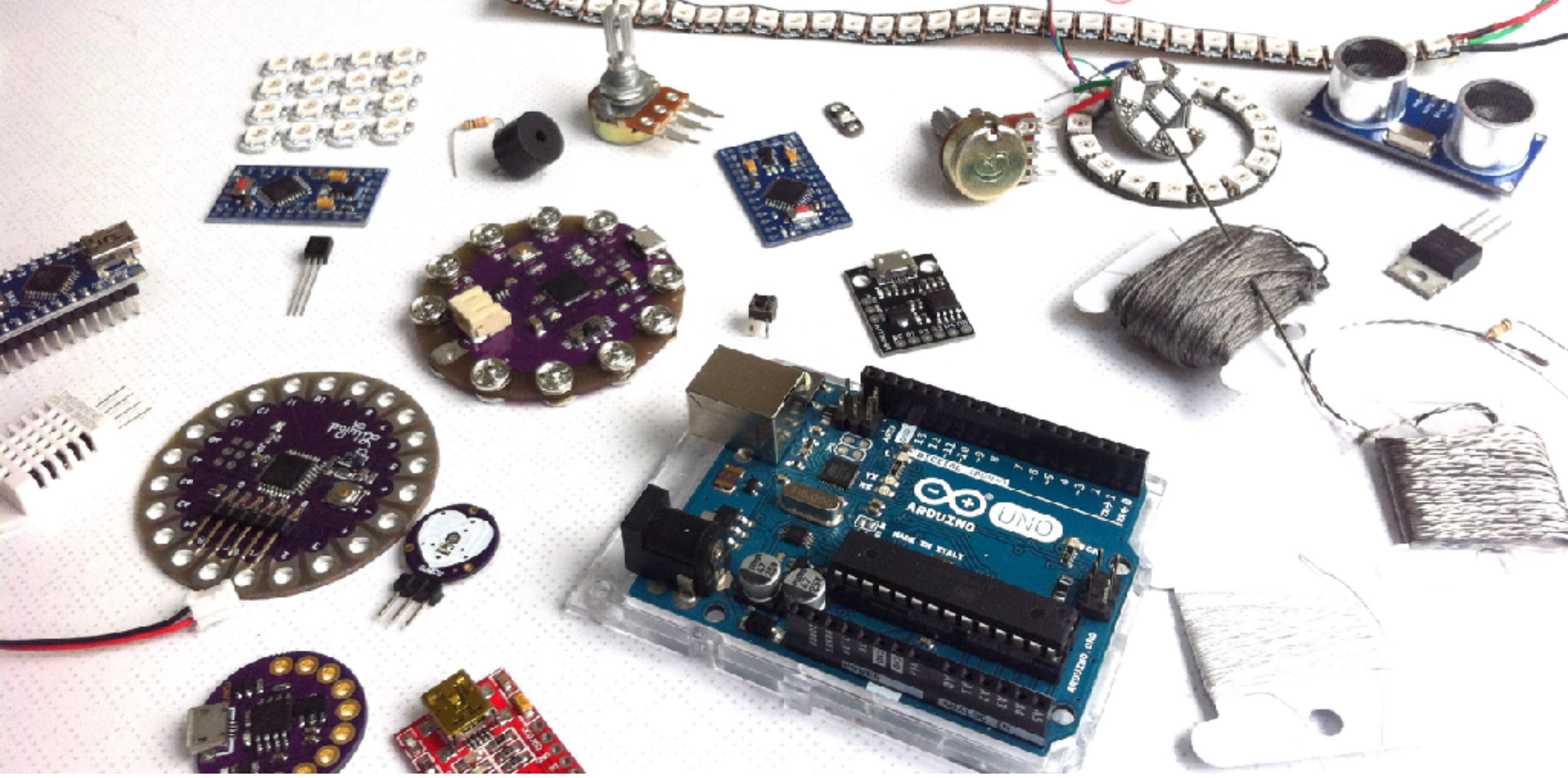
Instale as bibliotecas:
FastLED e/ou **AdaFruit NeoPixel**

Abra, altere e rode os exemplos:

1) NeoPixel: **RGBWstrandtest**
Mude as linhas
```
#define PIN 4
#define NUM_LEDS 5
```

2) FastLED: **DemoReel100**
Mude as linhas
```
#define DATA_PIN 4
#define NUM_LEDS 5
```





Introdução ao Arduino

Helder da Rocha

(agosto 2017)

argonavis

Veja mais em

www.eletronicaparaartistas.com.br