

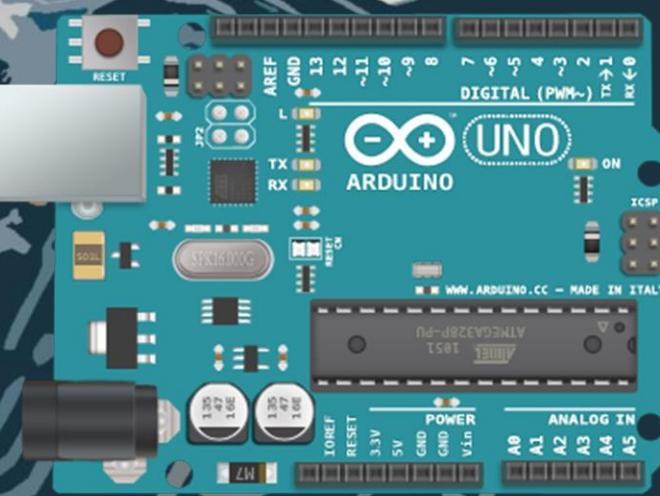
MINICURSO DE ARDUINO



Filipe Augusto



<https://github.com/filipeaocastro/Minicurso-de-Arduino>



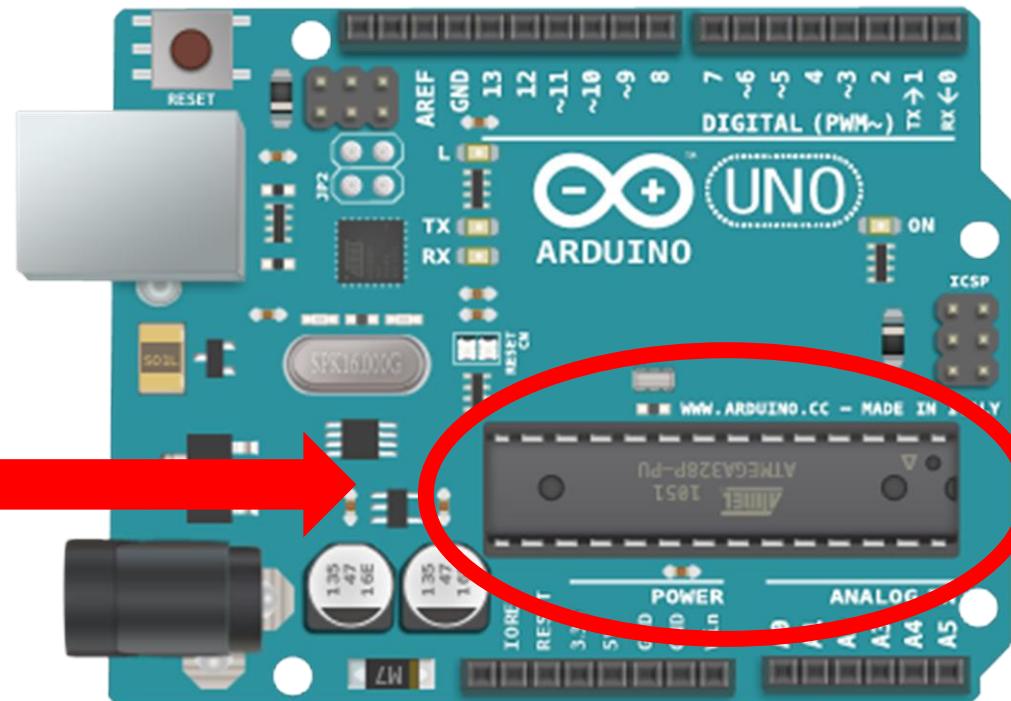
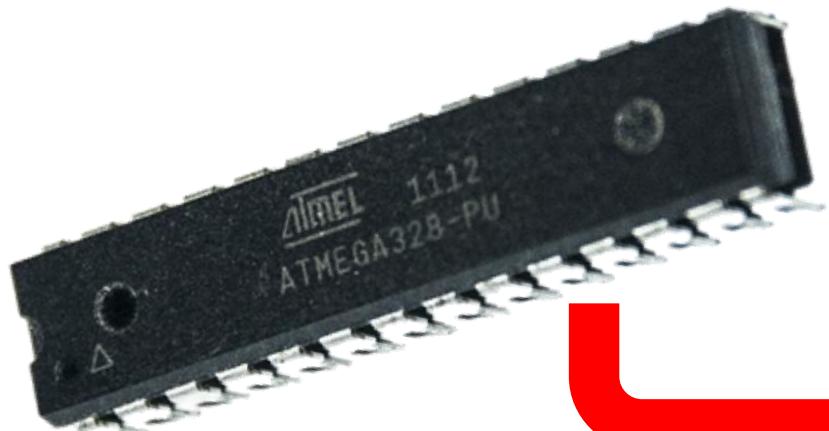
O que é Arduino?

Microcontroladores

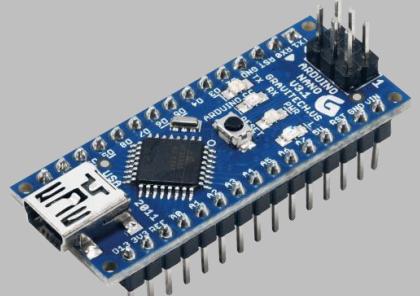


O que é Arduino?

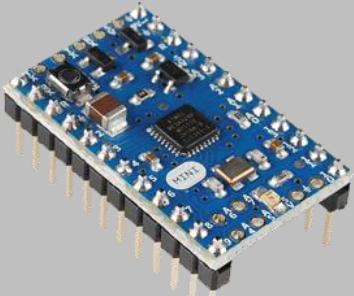
Microcontroladores



Tipos de Arduino



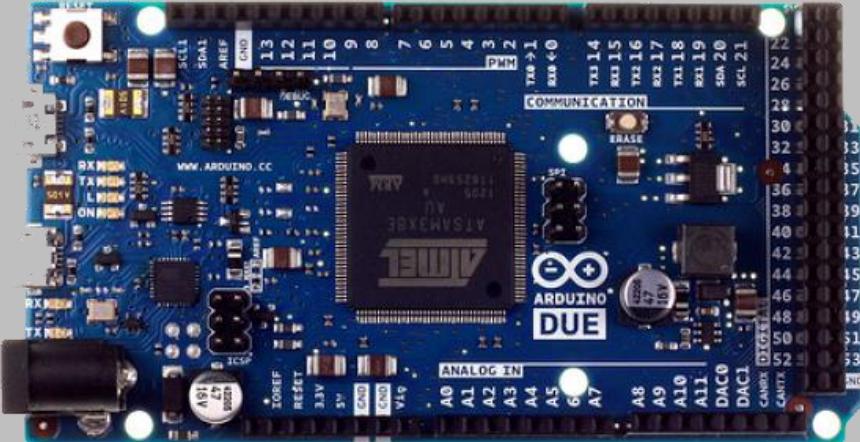
Arduino Nano



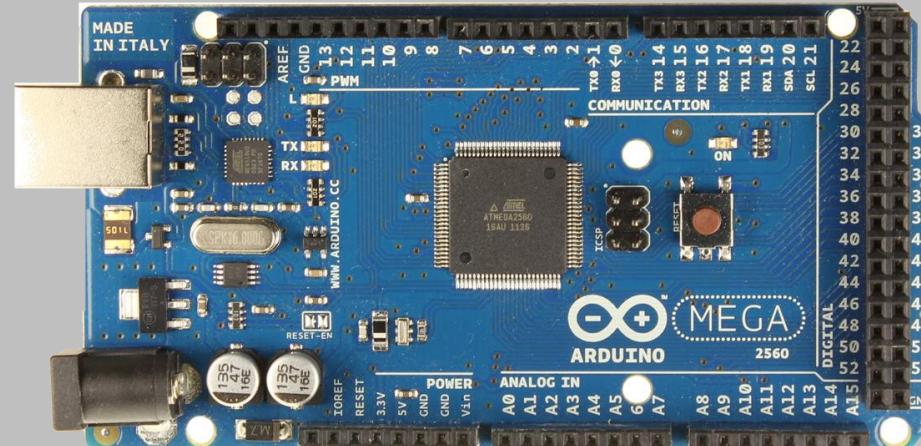
Arduino Pro Mini



Arduino UNO

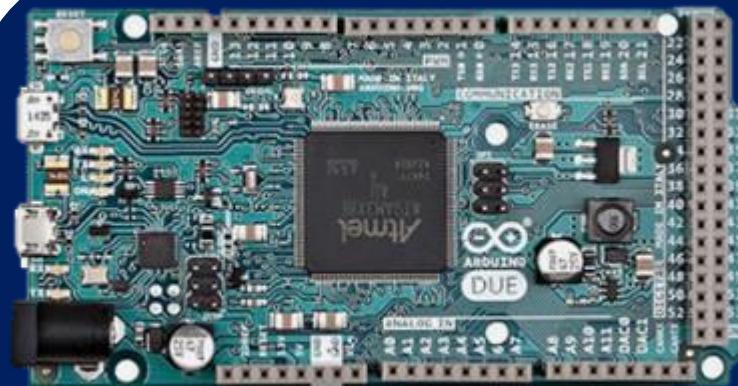
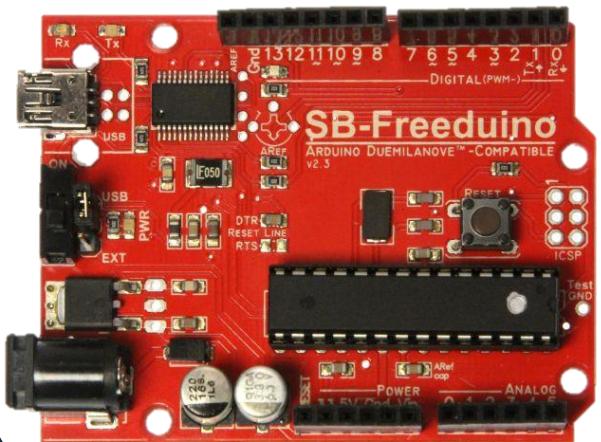
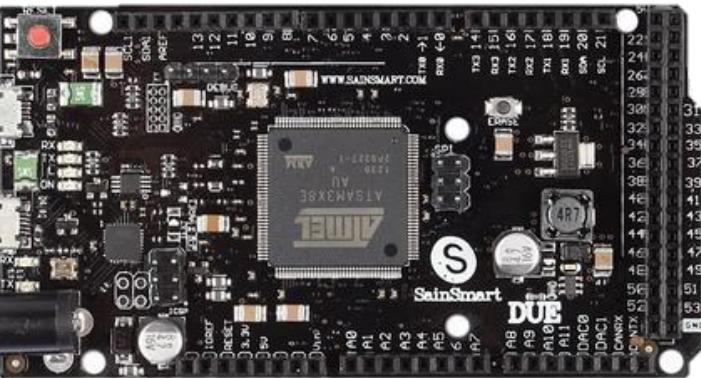


Arduino DUE



Arduino MEGA

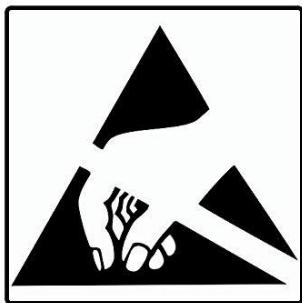
Hardware Open-Source



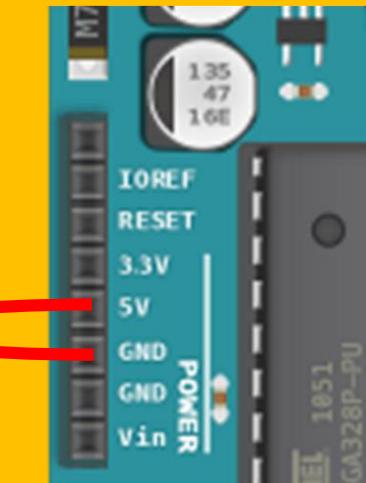
Originais

Cuidados com o Arduino

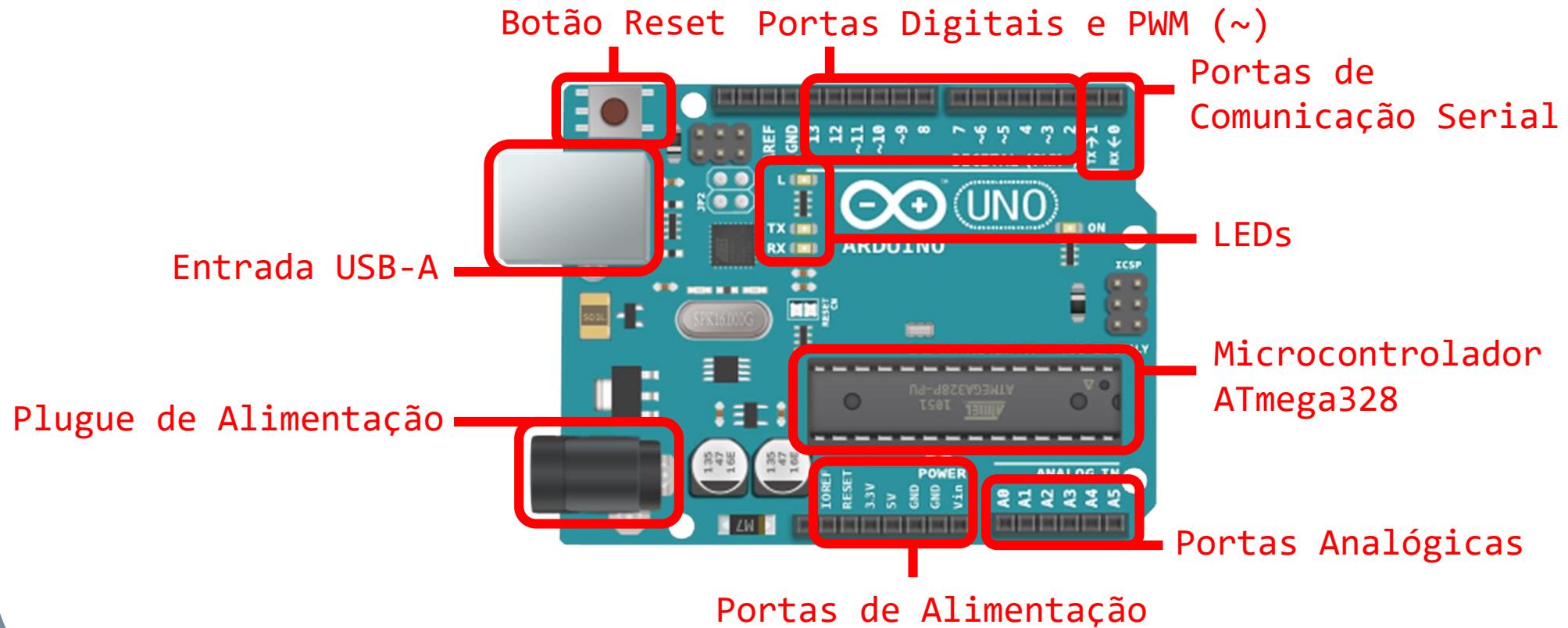
Eletricidade
Estática



Curto
Círcuito



O que é cada parte da placa?



E suas Aplicações?

Robótica

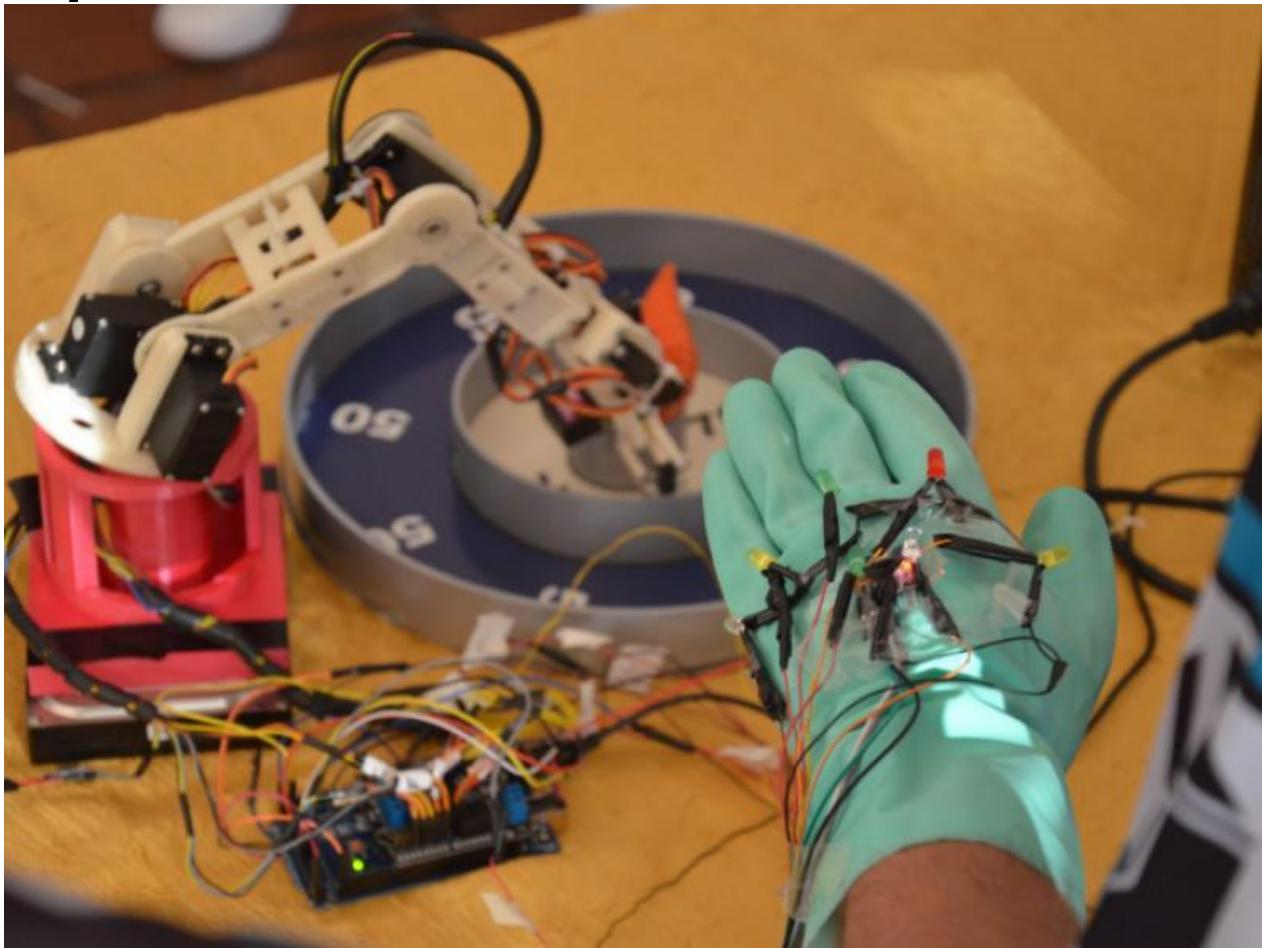


E suas Aplicações?

Robótica

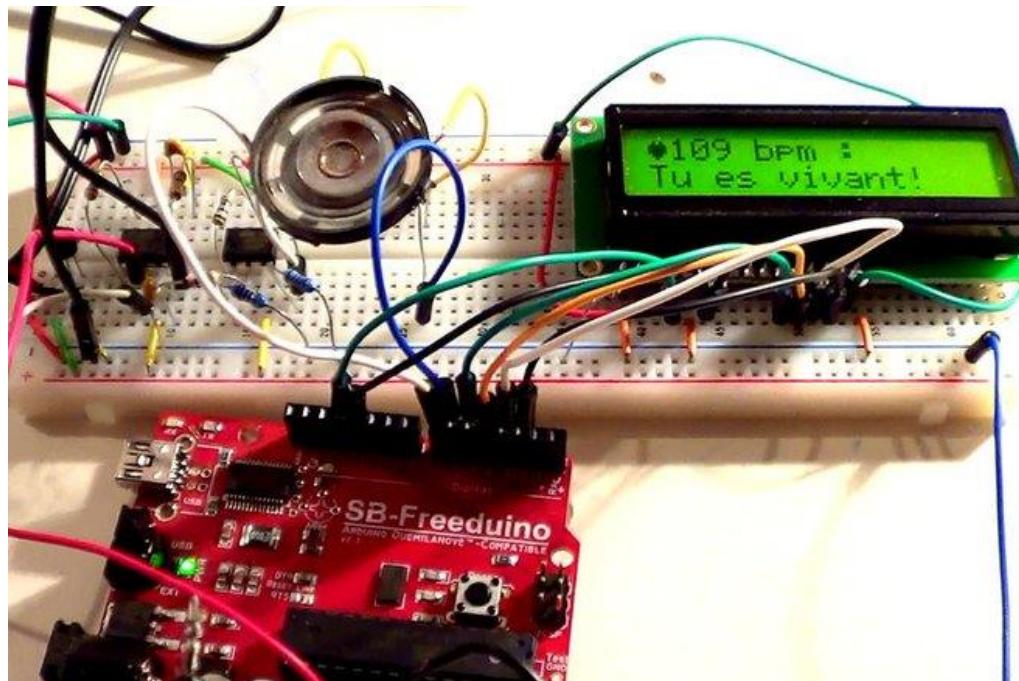
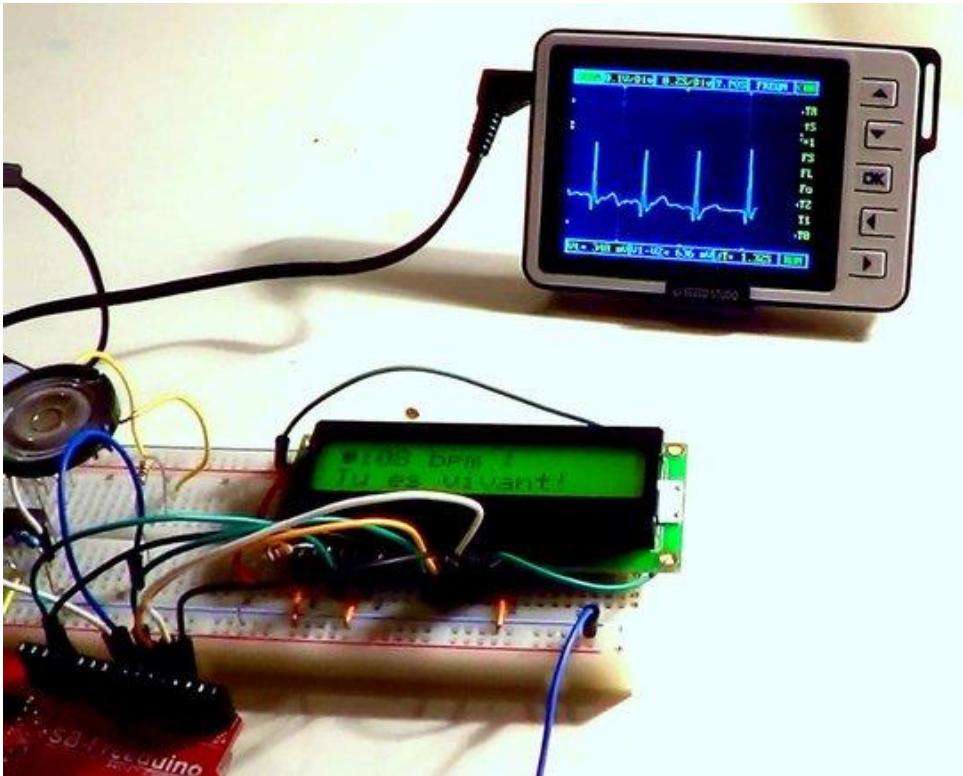


```
void PET_Engenharia_Biomédica ()
{
    DESAFIO_CAPITÃO_GANCHO;
    Edição = 1;
    start ();
}
```



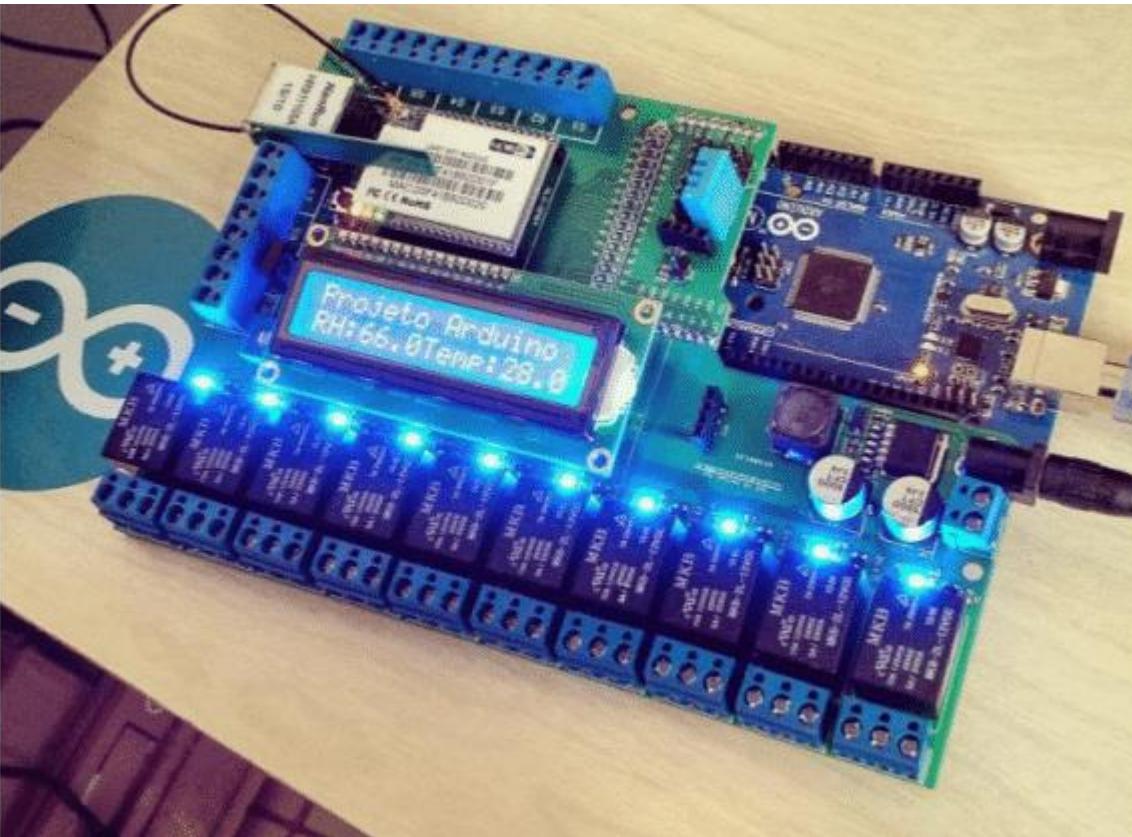
E suas Aplicações?

Monitoramento de Sinais Fisiológicos



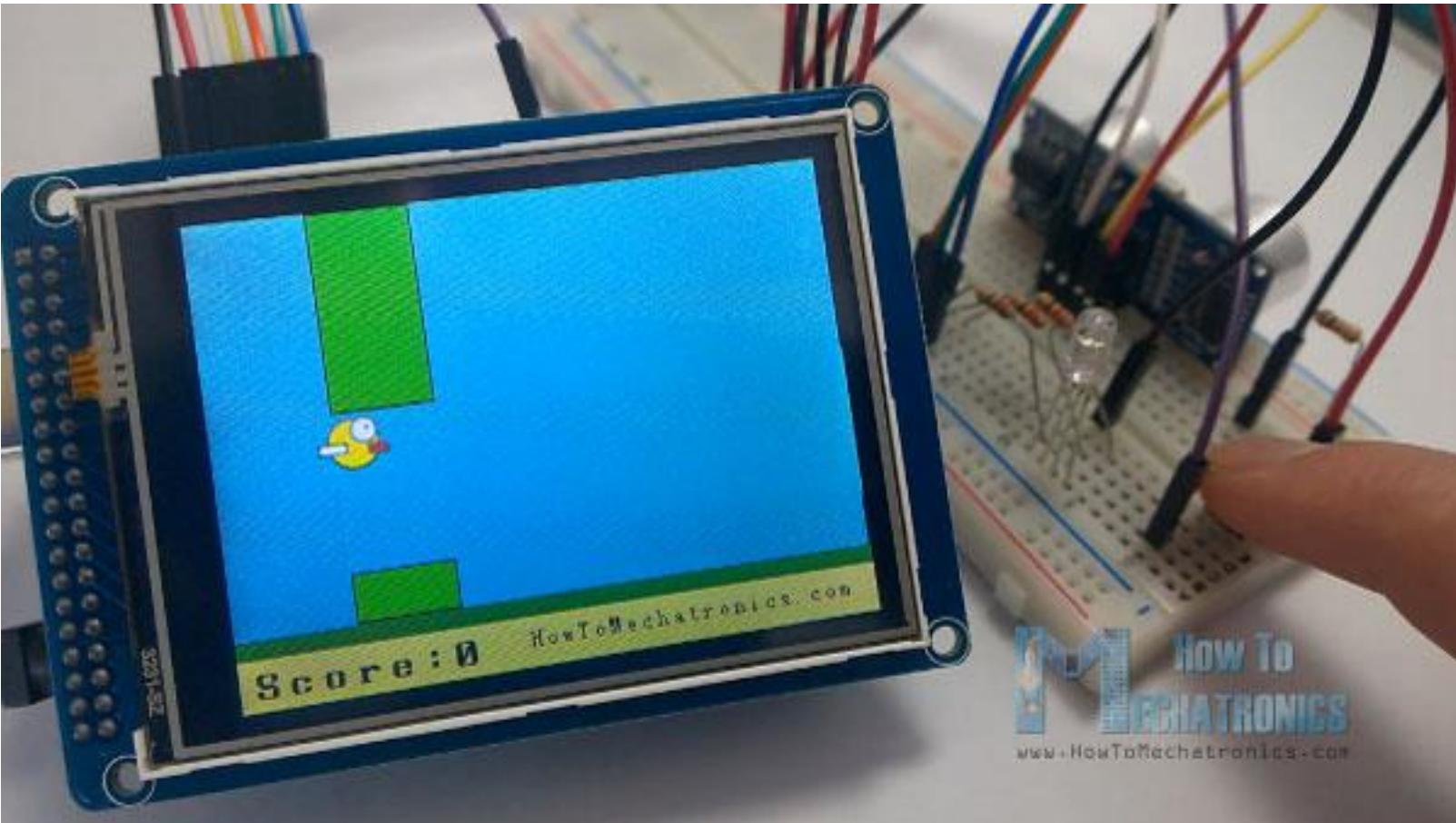
E suas Aplicações?

Projetos de Automação

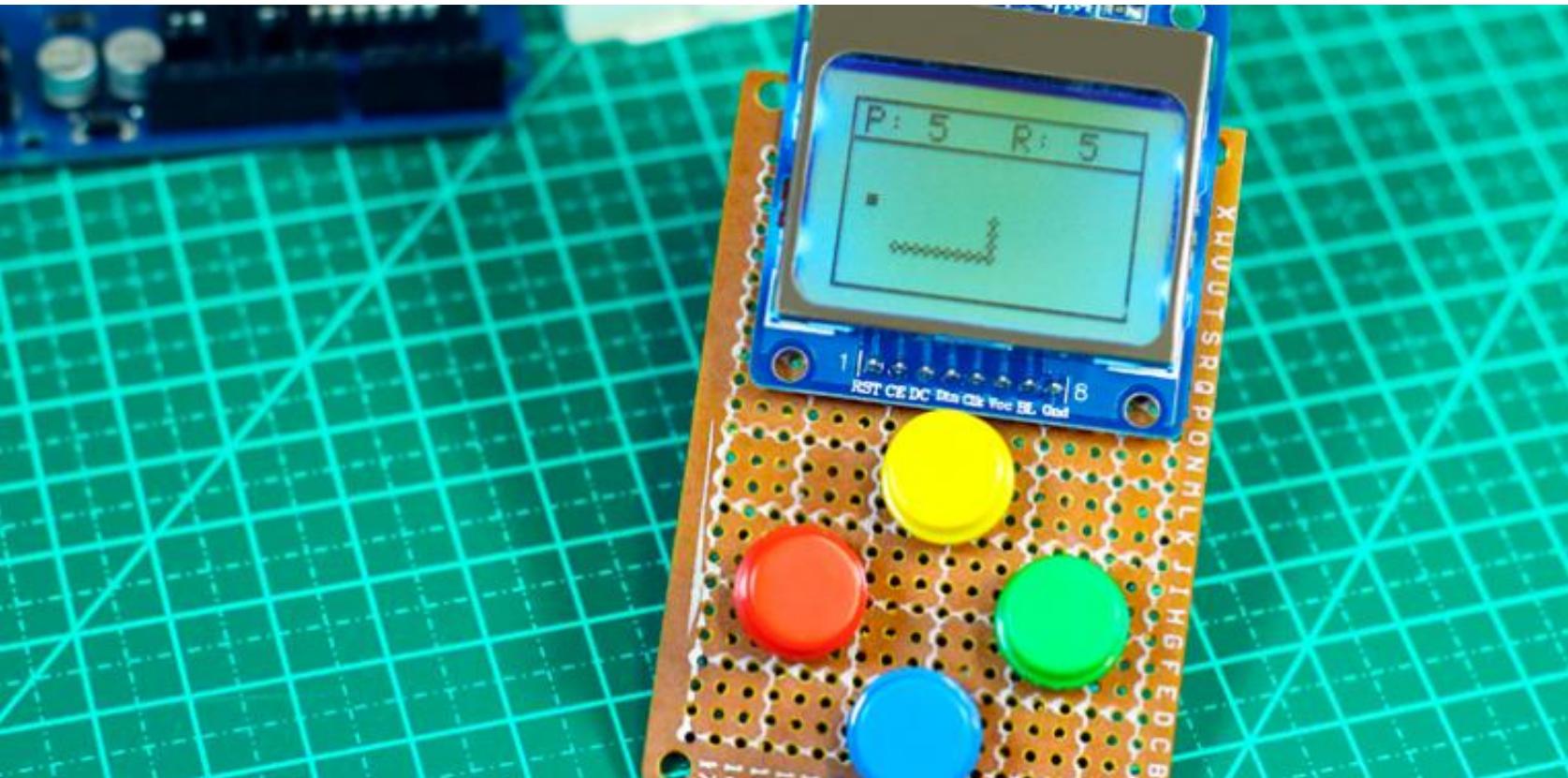


E suas Aplicações?

Jogos



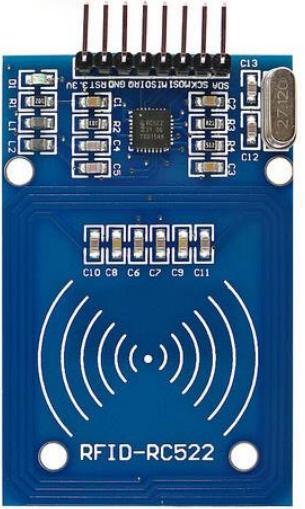
E suas Aplicações? Jogos



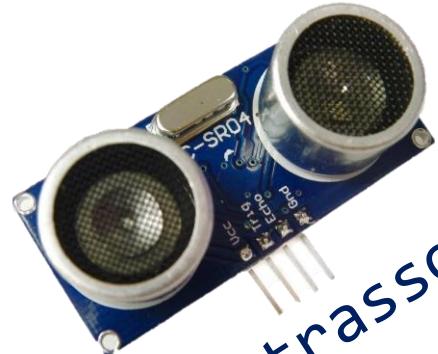
E suas Aplicações?
Coisas aleatórias -_(ツ)_/-



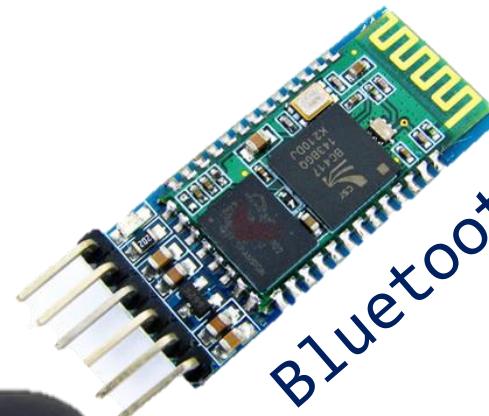
Módulos!



RFID



Ultrassom



Bluetooth



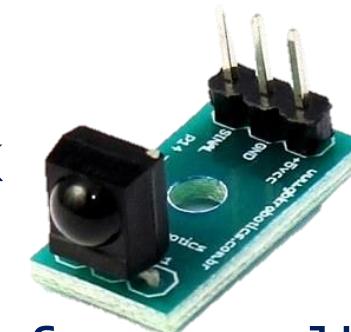
Batimentos
Cardíacos



Wi-Fi



Joystick



Infravermelho



Teclado



Motores

Como utilizar um Arduino?

IDE Arduino

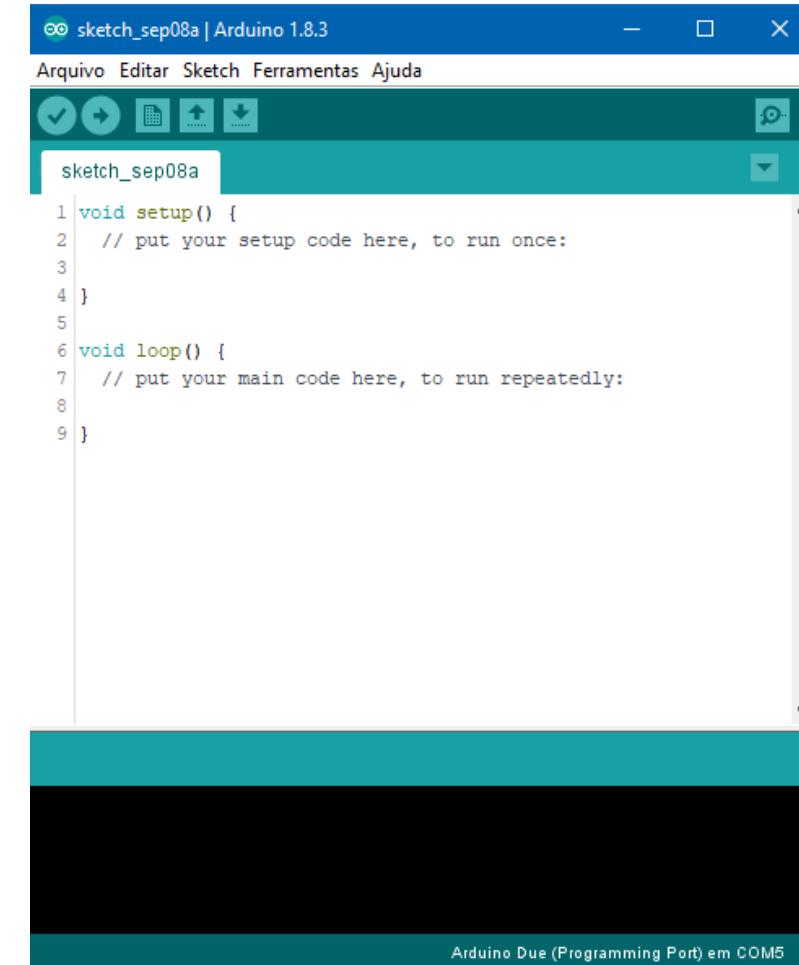
Linguagem: C++ (modificada)

Cada programa é um "sketch"



Programa

Arduino



Verificar

Arquivo Editar Sketch Ferramentas Ajuda



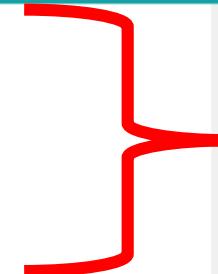
Carregar para a placa



sketch_sep08a §

Nome do Sketch

```
1 void setup() {  
2   // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7   // put your main code here, to run repeatedly:  
8  
9 }
```



Monitor
Serial

Função Setup:
É chamada
apenas na
inicialização

Função Loop:
É chamada após
o setup e
repete enquanto
estiver ligado

Nome da Placa

Porta em que a
placa está
conectada

Mão na massa!



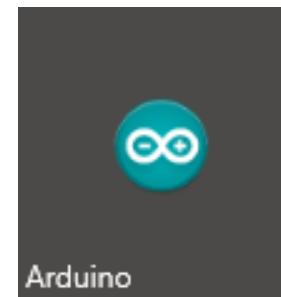


Pegue o Arduino

Conecte no PC



Abra a IDE do Arduino



Arquivo Editar Sketch **Ferramentas** Ajuda

sketch_apr02a

```
void setup() {  
  // put your setup  
}  
  
void loop() {  
  // put your main  
}
```

Autoformatação Ctrl+T

Arquivar Sketch

Corrigir codificação e recarregar

Gerenciar Bibliotecas... Ctrl+Shift+I

Monitor serial Ctrl+Shift+M

Plotter serial Ctrl+Shift+L

WiFi101 Firmware Updater

Placa: "Arduino/Genuino Uno"

Porta

Obter informações da Placa

Programador: "AVRISP mkII"

Gravar Bootloader

Portas seriais

COM10 (Arduino/Genuino Uno)

Arquivo Editar Sketch Ferramentas Ajuda



sketch_apr02a

```
void setup() {  
    // put your setup code here  
}  
  
void loop() {  
    // put your main loop code here  
}
```

Autoformatação Ctrl+T

Arquivar Sketch

Corrigir codificação e recarregar

Gerenciar Bibliotecas... Ctrl+Shift+I

Monitor serial Ctrl+Shift+M

Plotter serial Ctrl+Shift+L

WiFi101 Firmware Updater

Placa: "Arduino/Genuino Uno"

Porta

Obter informações da Placa

Programador: "AVRISP mkII"

Gravar Bootloader

Gerenciador de Placas...

Placas Arduino AVR

Arduino Yún

● Arduino/Genuino Uno

Arduino Duemilanove or Diecimila

Arduino Nano

Arduino/Genuino Mega or Mega 2560

Arduino Mega ADK

Arduino Leonardo

Arduino Leonardo ETH

Arduino/Genuino Micro

Arduino Esplora

Arduino Mini

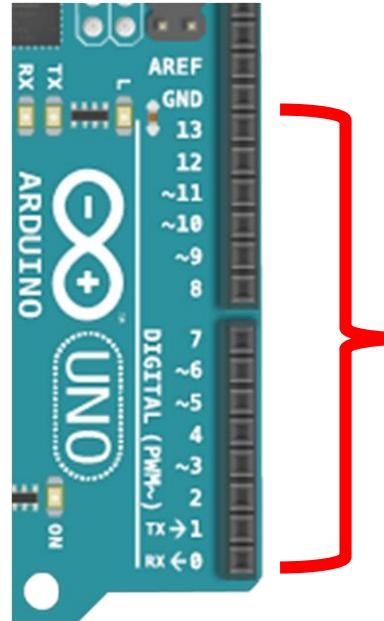
Arduino Ethernet

Arduino Fio

1º Programa: Blink

Objetivo:

Fazer um led piscar com intervalos de tempo definidos



Portas
Digitais

Conceito: Saída de Portas Digitais

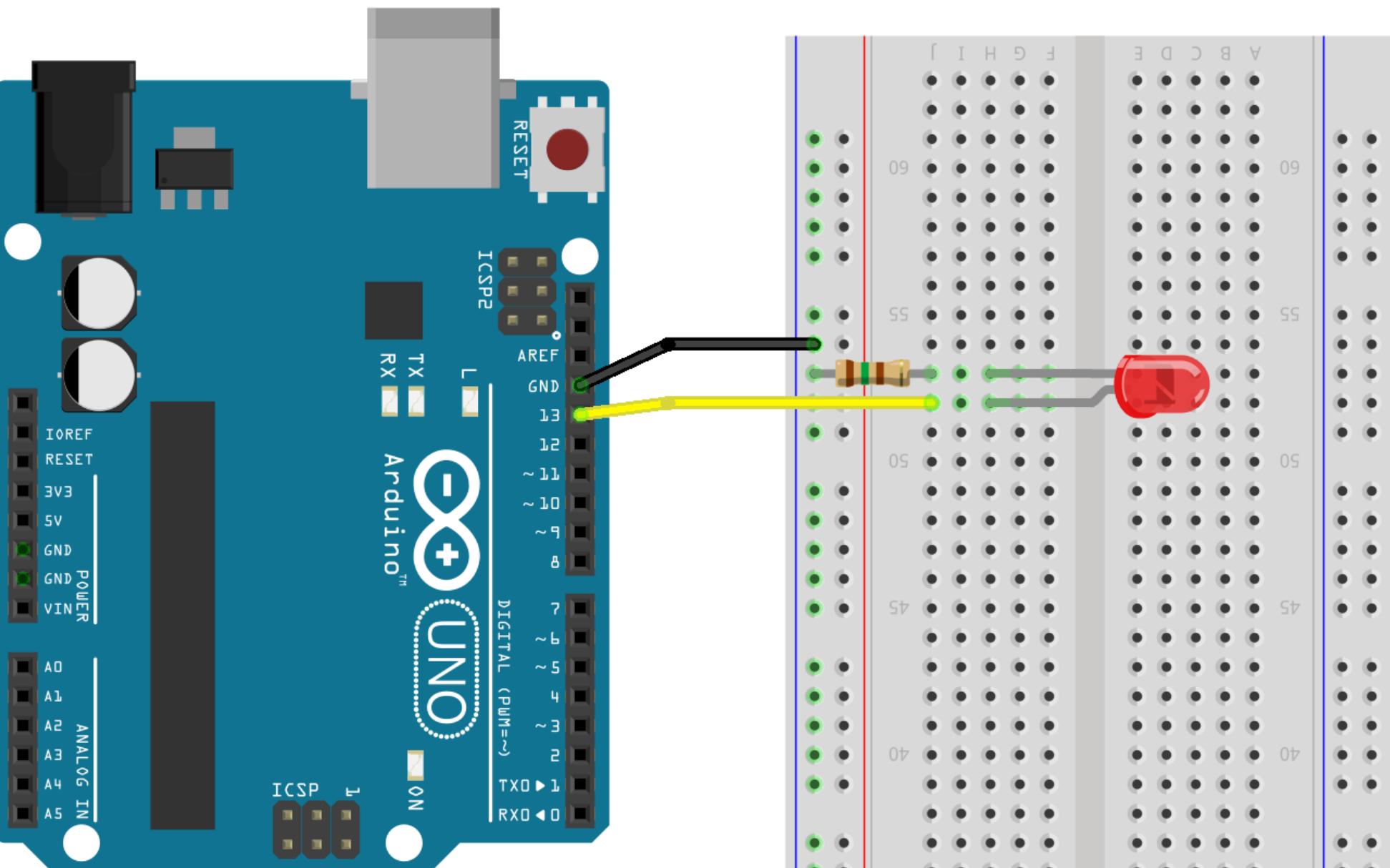
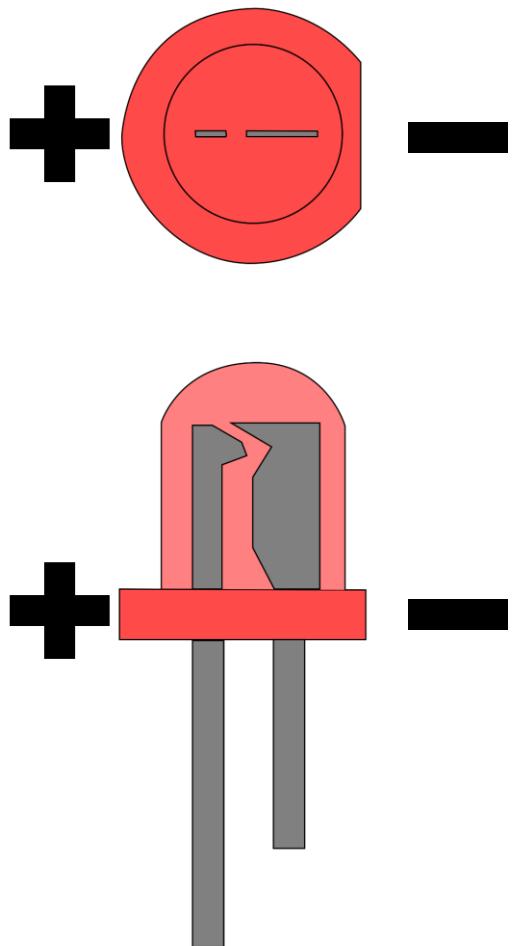
Portas digitais fornecem apenas dois valores de tensão:

Alto: 5V e **Baixo:** 0V (GND)

1º Programa: Blink

Montagem:

Obs.:



1º Programa: Blink

Funções:

`pinMode(pino, modo)`

Nº do pino



INPUT: "Recebe" energia
OUTPUT: "Fornece" energia

`digitalWrite(pino, nível)`

Nº do pino



HIGH: 5V (nível alto)
LOW: 0V (nível baixo)

`delay(tempo)`



Tempo em **milissegundos**
(1000 ms = 1 segundo)

2º Programa: LED controlado por botão

Objetivo:

Fazer um led piscar quando um botão for apertado

Conceito: Entrada de Portas Digitais
Portas digitais identificam apenas dois valores de tensão:

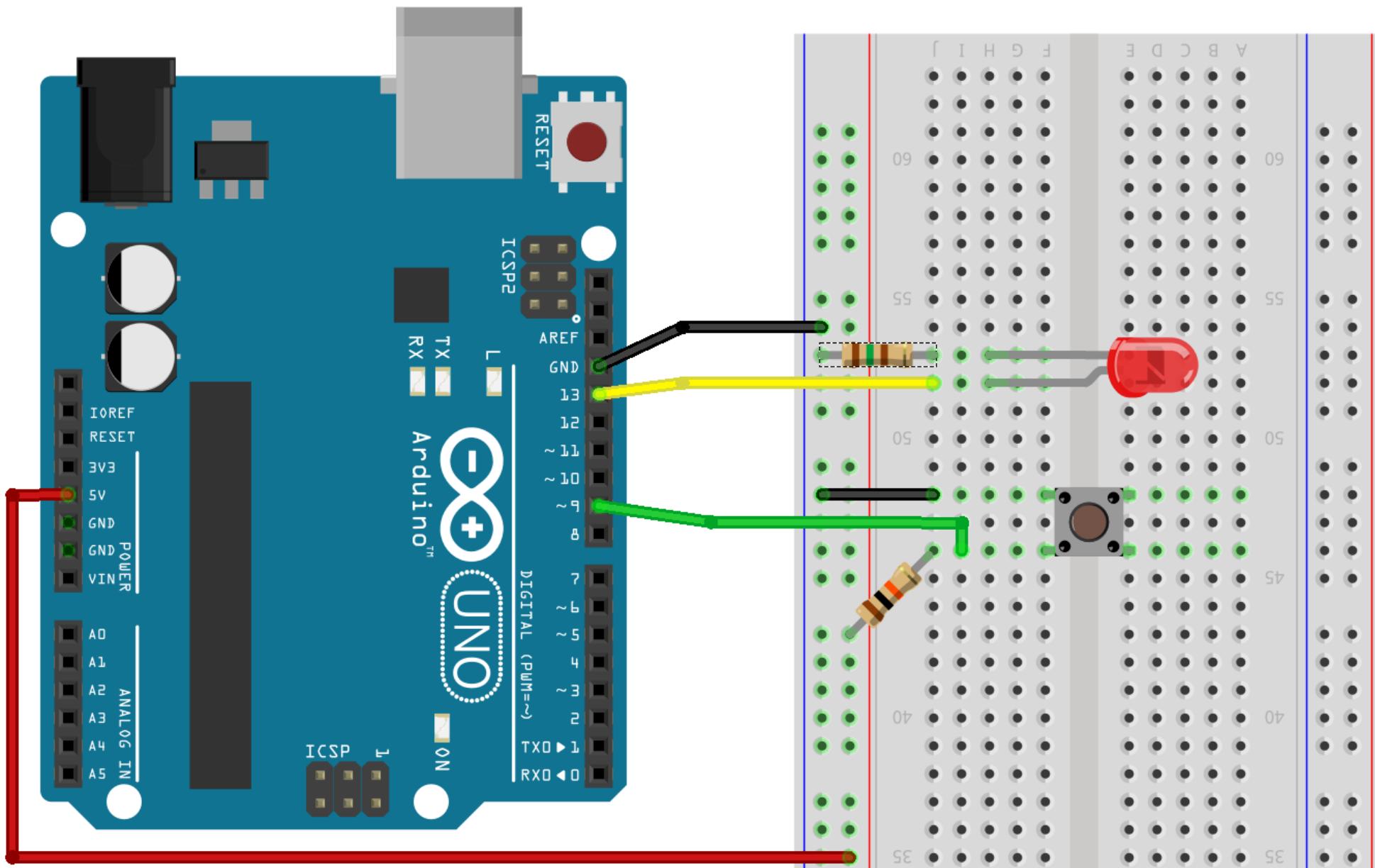
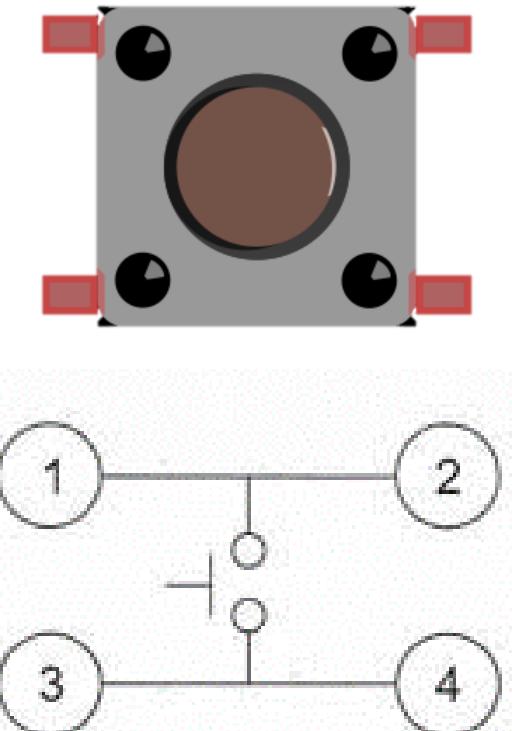
Alto: 5V e Baixo: 0V (GND)

Valores intermediários retornam valores aleatórios!

2º Programa: LED controlado por botão

Montagem:

Obs.:



2º Programa: LED controlado por botão

Função:

`digitalRead(pino)`



Nº do pino

Retorna:

1 para 5V e 0 para 0V

3º Programa: Semáforo com controle de tempo

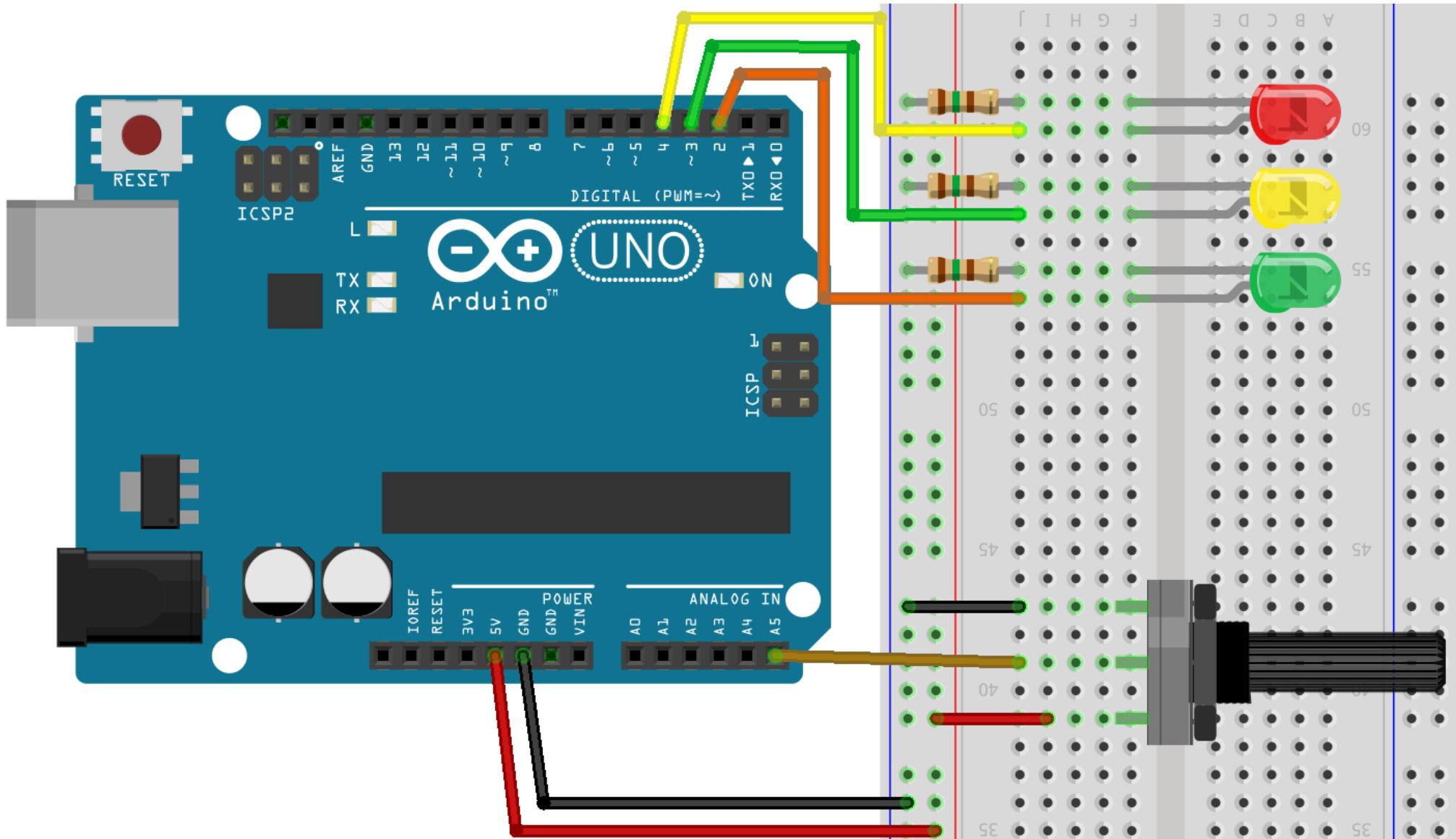
Objetivo:

Elaborar um semáforo com 3 LEDs cujo tempo de operação varie de acordo com a tensão de saída do potenciômetro.

Conceito: Entrada de Portas Analógicas
Portas analógicas conseguem identificar valores de entrada tensão entre 0 e 5 Volts.

3º Programa: Semáforo com controle de tempo

Montagem:



3º Programa: Semáforo com controle de tempo

Função:

`analogRead(pino)`



Nº do pino

Retorna:

1023 para 5V e 0 para 0V

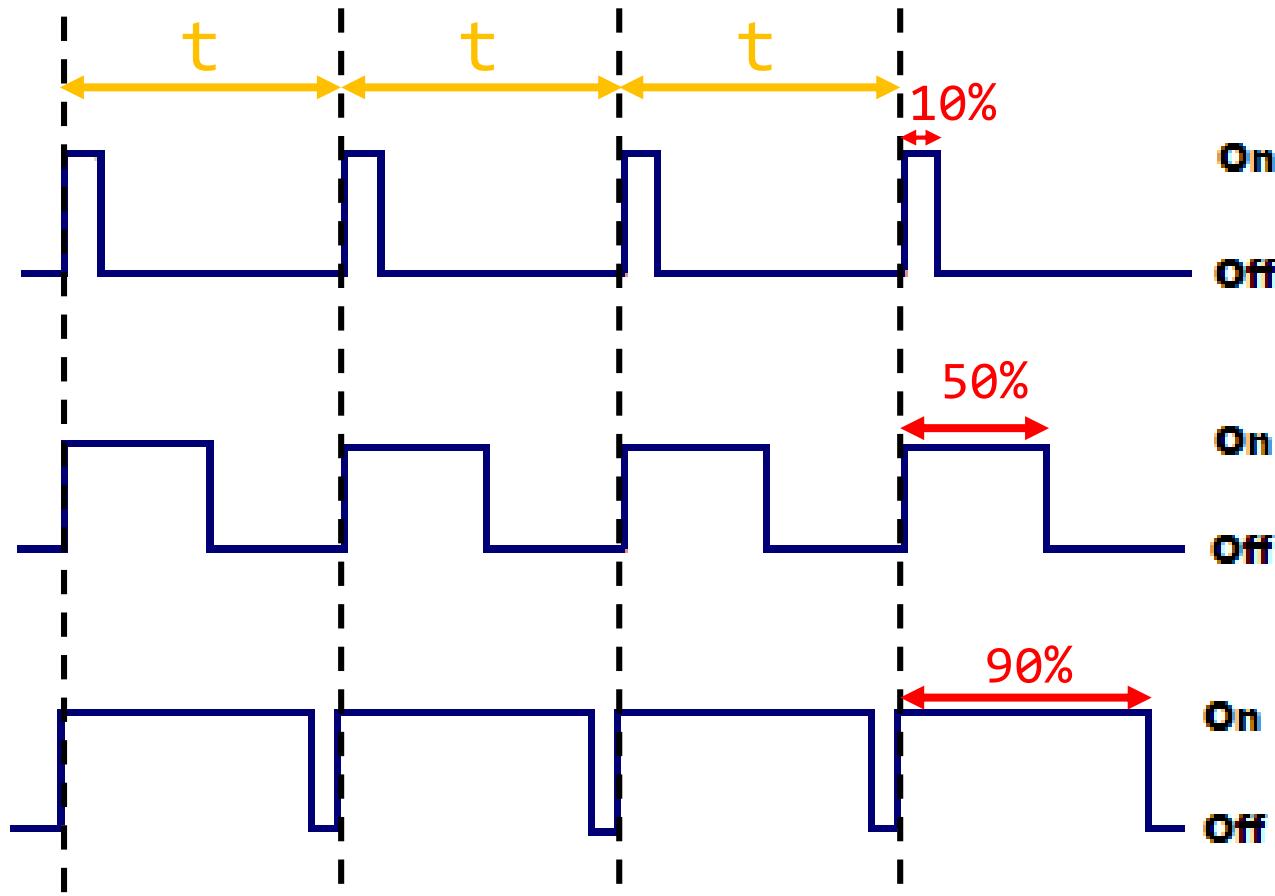
Arduino UNO: resolução de 10 bits

10 bits = armazena até 1023 em valores decimais

$$\frac{5V}{1023} \approx 0,005 = 5mV \quad \text{Portanto, cada bit equivale a } 5mV.$$

Introdução às saídas PWM

- Todas as saídas do Arduino são digitais
- Diferentes larguras de pulso, mesma frequência
- As saídas PWM são controladas pela função analogWrite



As saídas PWM são identificadas por um til (~)

`analogWrite(pino, valor)`

↓
Nº do pino

Qualquer valor entre
255 e 0, onde:
255: 100% largura de
pulso (5V)
0: 0% de largura de
pulso (0V)

Exemplo: `analogWrite(10, 127)`
escreve aprox. 2,5 V no pino 10



4º Programa: Dimmer

Objetivo:

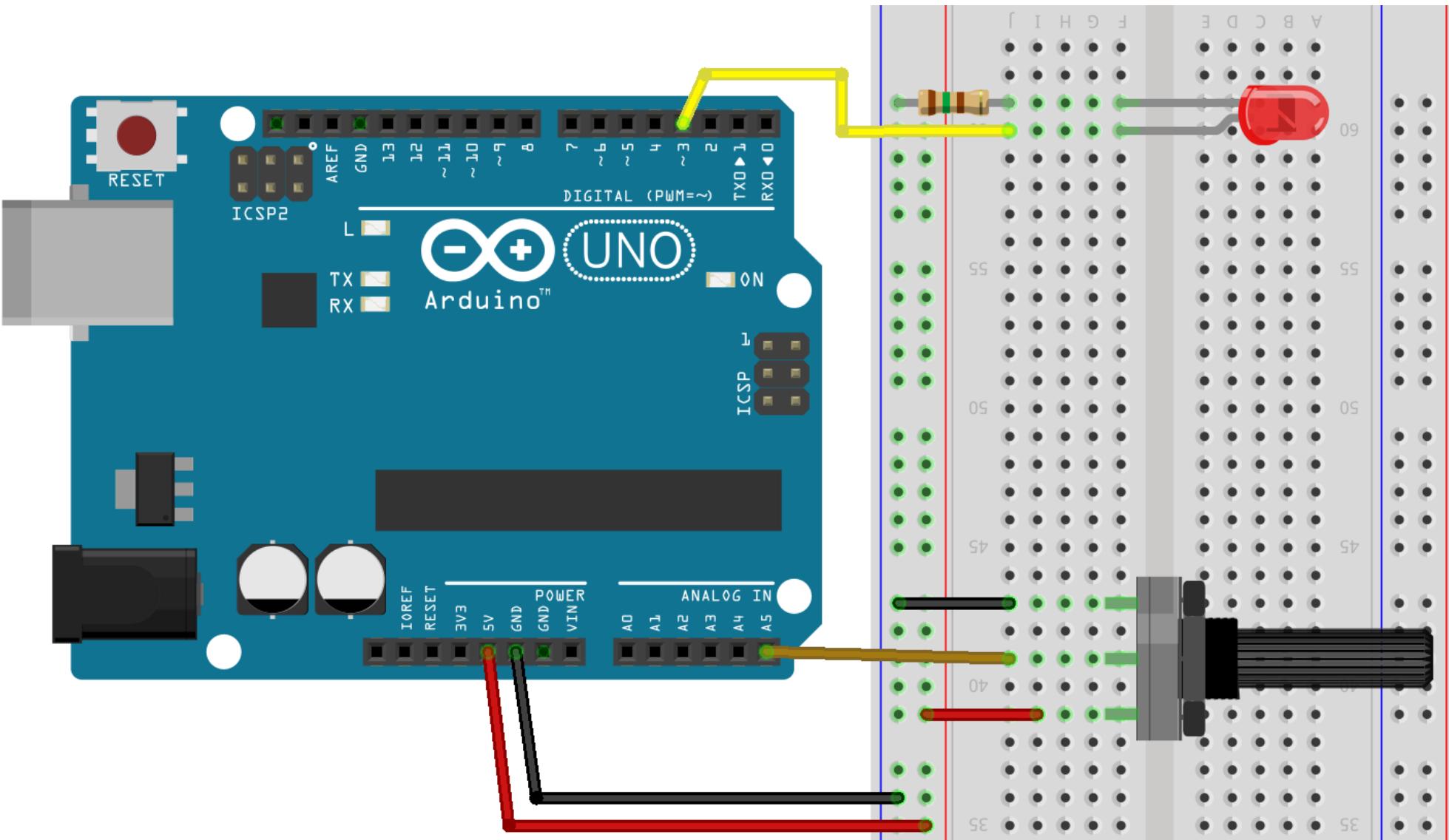
Controlar o brilho de um LED a partir da leitura analógica da saída de um potenciômetro

Conceito: Saídas PWM

Portas PWM fornecem valores de tensão de 0 a 5V de acordo com a mudança a largura de pulso.

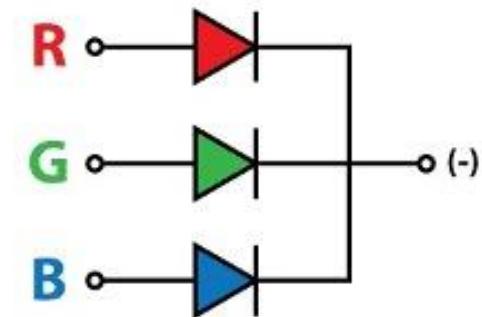
4º Programa: Dimmer

Montagem:



LED RGB

- O LED RGB é caracterizado por possuir o equivalente a 3 LEDs no mesmo encapsulamento, onde suas cores são **vermelho**, **verde** e **azul**;
- Cada cor é controlada por um terminal (“perninha”);
- O maior terminal é o Ground (-).



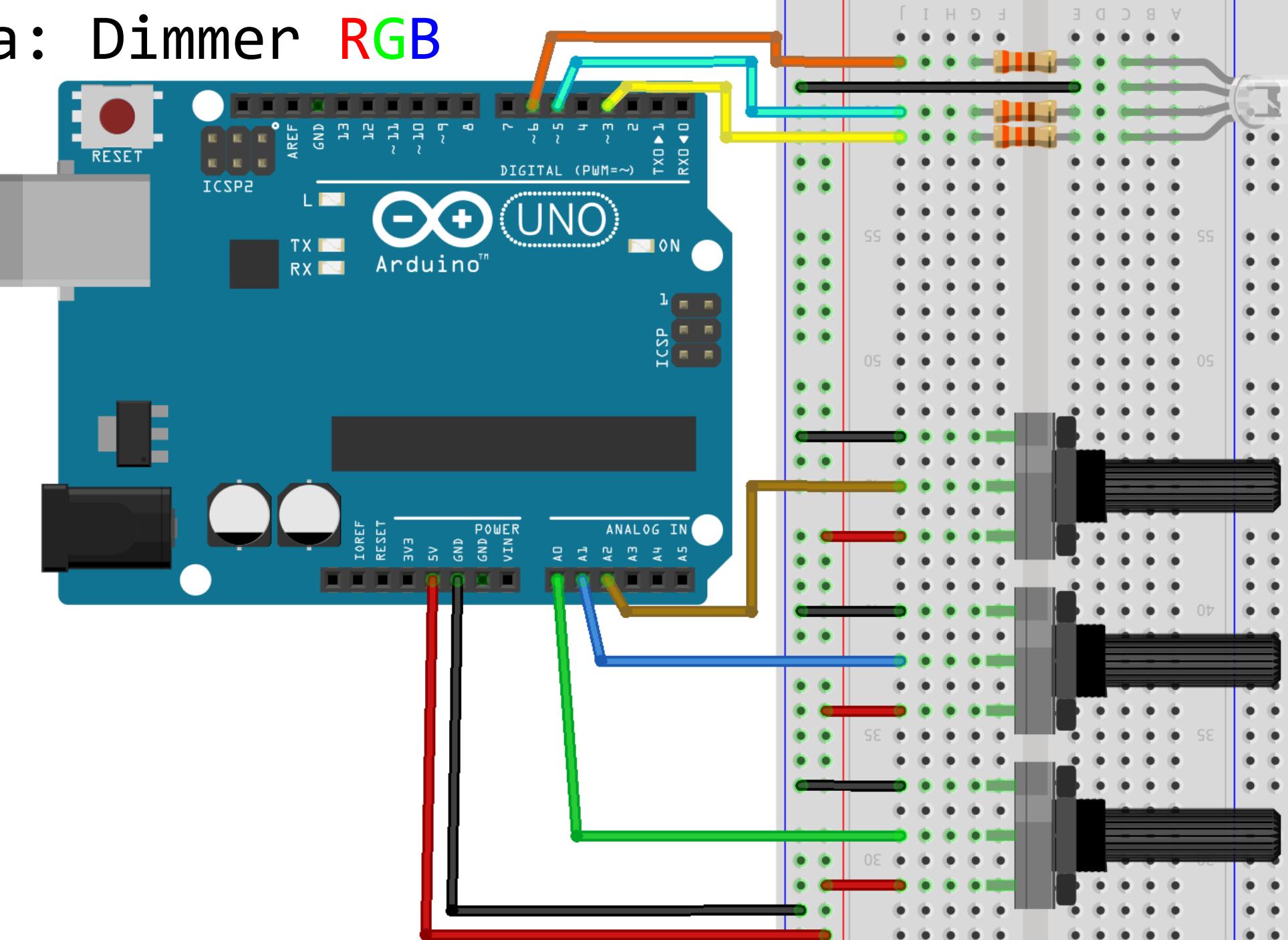
5º Programa: Dimmer RGB

Objetivo:

Controlar o brilho de cada cor de um LED RGB de forma independente através de potenciômetros

5º Programa: Dimmer RGB

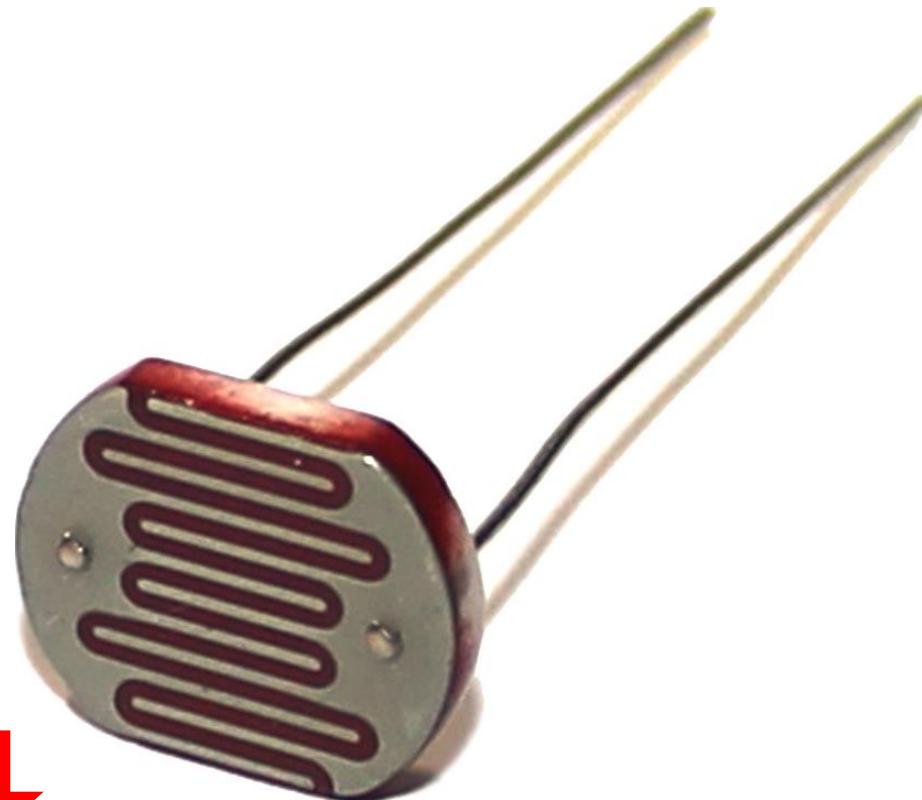
Montagem:

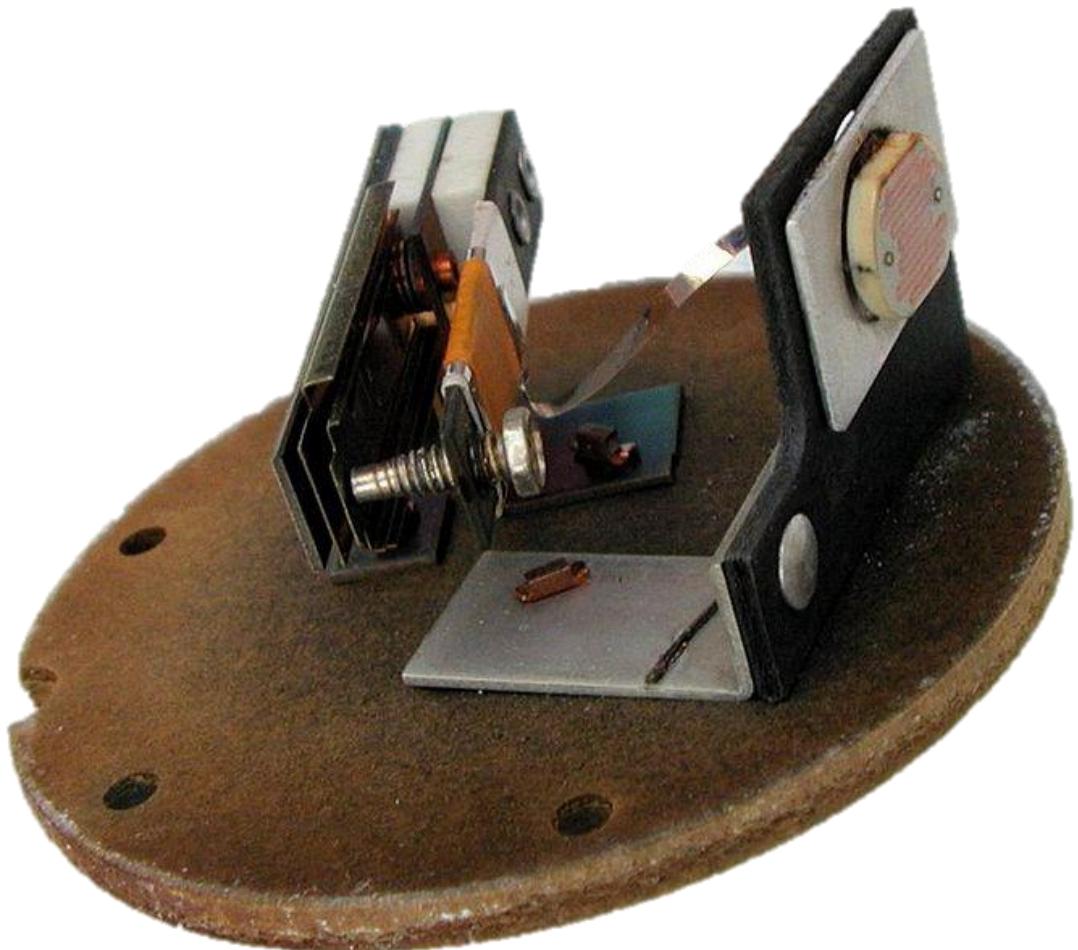


LDR – Sensor de Luz

- LDR – Light Dependent Resistor (Resistor Dependente de Luz)
- É um resistor que varia sua resistência em função da luz incidente
- Resistência inversamente proporcional à luz

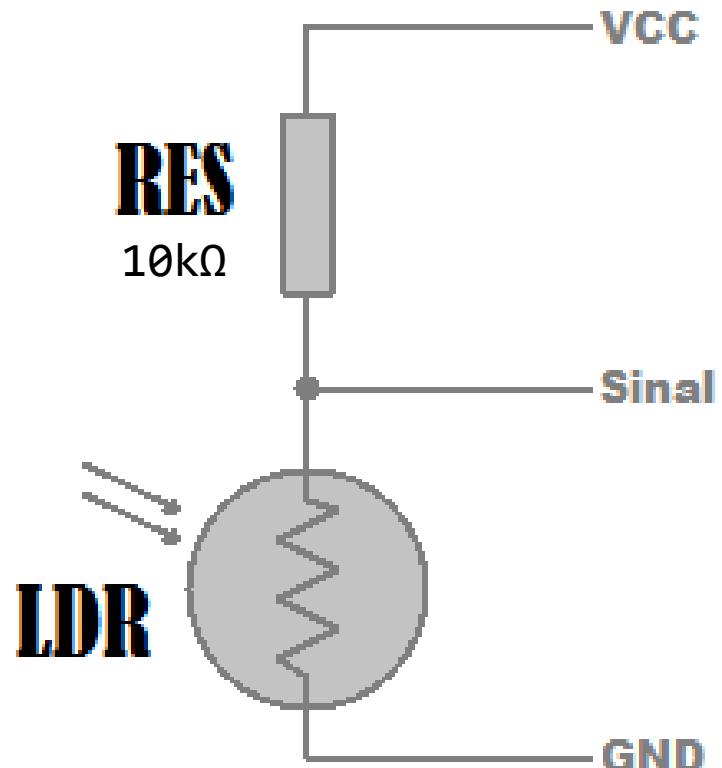
↑ Luz \approx Resistência ↓





<https://github.com/filipeaocastro/Minicurso-de-Arduino>

Como utilizá-lo?



- Divisor de tensão resistivo

$$Sinal = \frac{R_{LDR} \times Vcc}{10k + R_{LDR}}$$

- $Vcc = 5V$
- Quanto maior R_{LDR} , maior a amplitude do sinal
- Como R_{LDR} aumenta quando a luz diminui, conclui-se que:

[Calculadora de Divisor de Tensão](#)

Luz \approx Sinal

6º Programa: LED controlado por LDR

Objetivo:

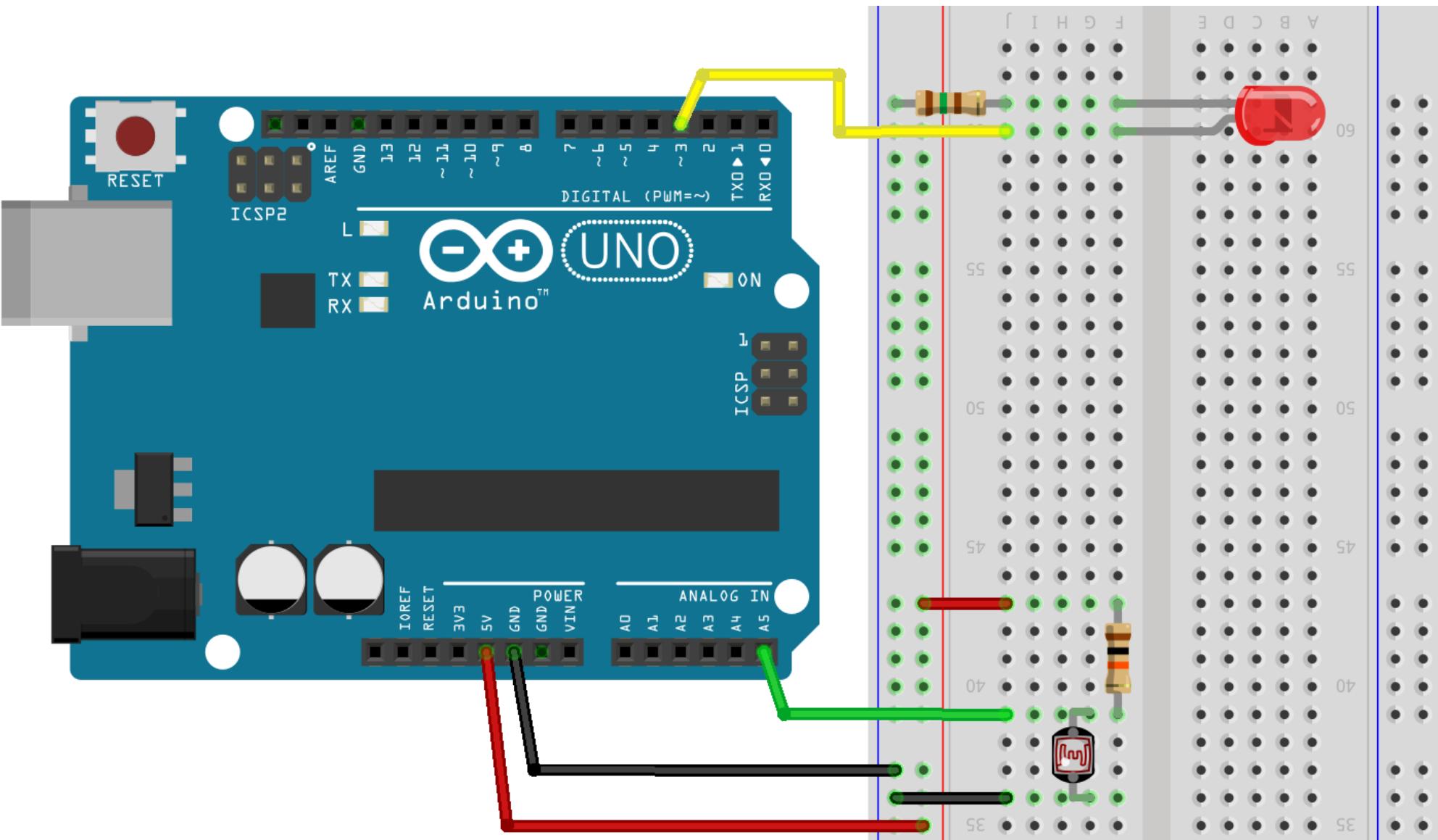
Controlar o brilho de um LED a partir da intensidade luminosa indicada por um LDR.

Conceito: Funcionamento do LDR

O LDR, no divisor de tensão já mostrado, faz com que o sinal de tensão de saída seja inversamente proporcional à intensidade luminosa

6º Programa: Controle de Luminosidade por LDR

Montagem:



Buzzer



- Produz efeito sonoro através da vibração de uma pastilha piezoelétrica
- Barulho bem chato

7º Programa: Sirene com Buzzer

Funções:

`tone(pino, frequência)`

Faz com que o buzzer vibre numa **frequência definida**

`sin(valor em radianos)`

Retorna o valor da função **seno** para determinado ângulo

7º Programa: Sirene com Buzzer

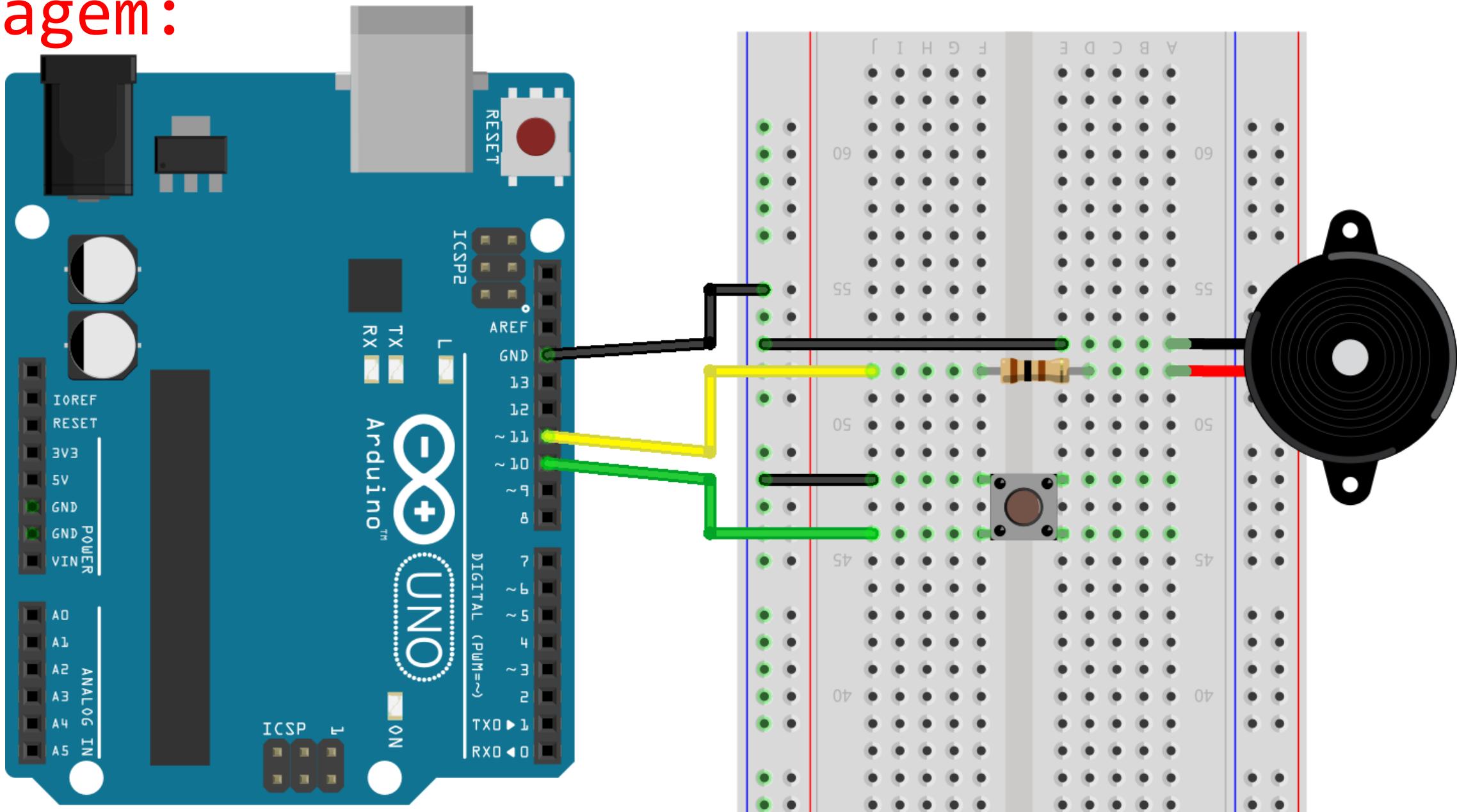
Objetivo:

Fazer uma sirene no buzzer utilizando uma onda senoidal alterar sua frequência apertando um botão.

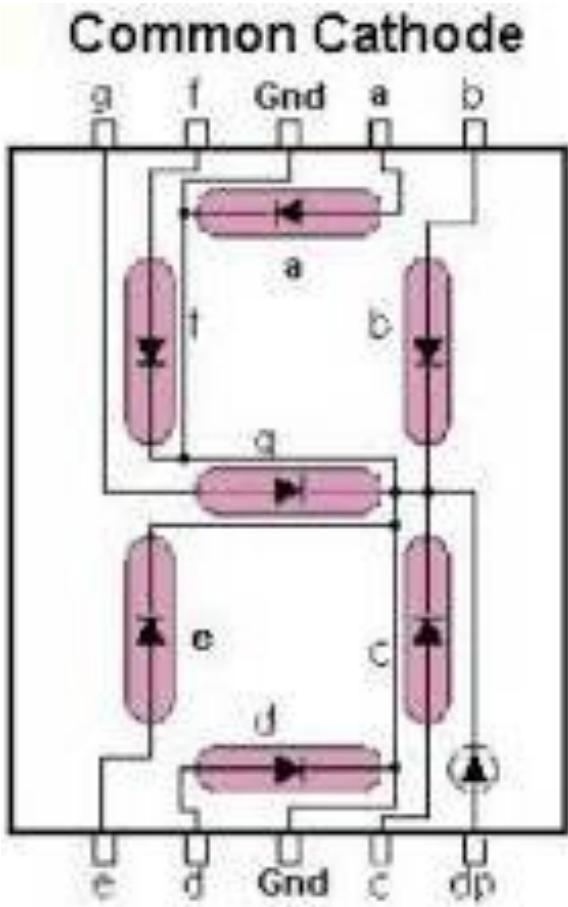
Conceito: Utilização das funções sin() e tone()

7º Programa: Sirene com Buzzer

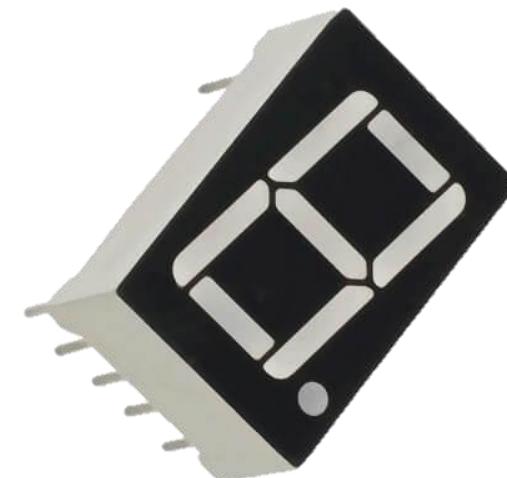
Montagem:



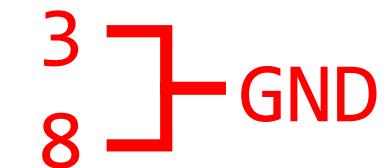
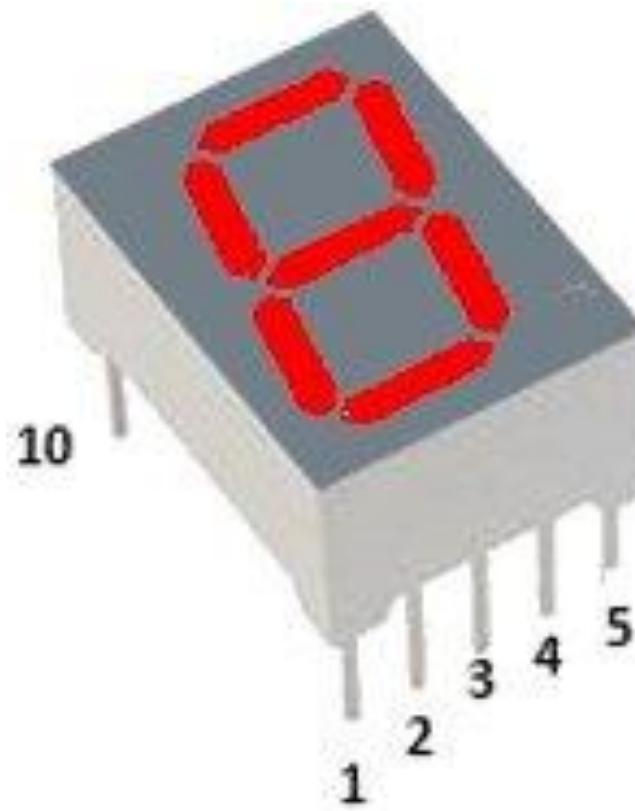
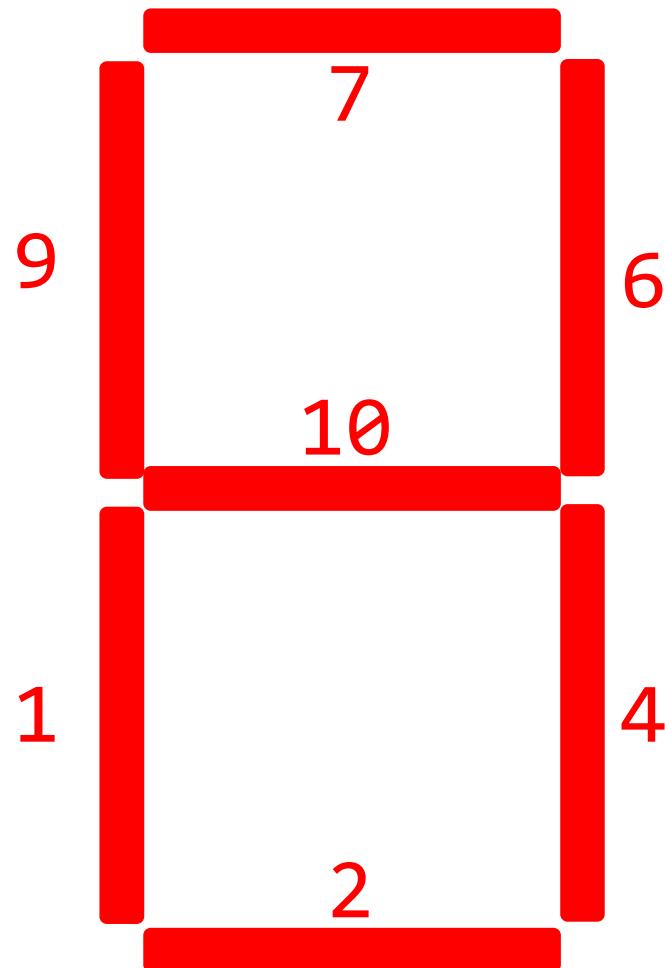
Display de 7 segmentos



- Display composto com 8 LEDs em paralelo (contando o ponto)
- Cada LED é ativado individualmente



Display de 7 segmentos



8º Programa: Contagem com Display de 7 Segmentos

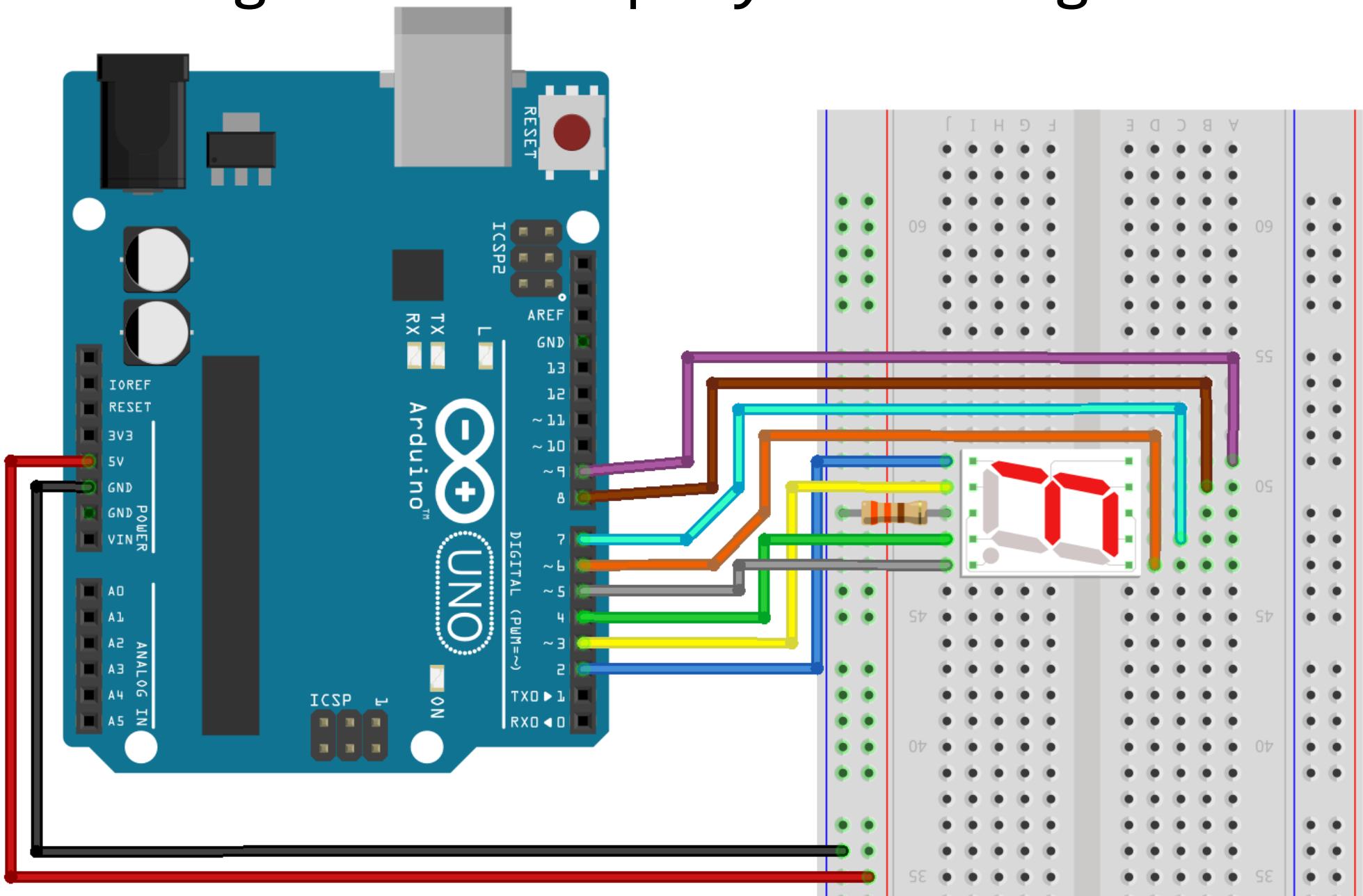
Objetivo:

Contar de 0 a 9 com o Display de 7 Segmentos

Conceito: Utilização de matrizes

8º Programa: Contagem com Display de 7 Segmentos

Montagem:



9º Programa: Despertador

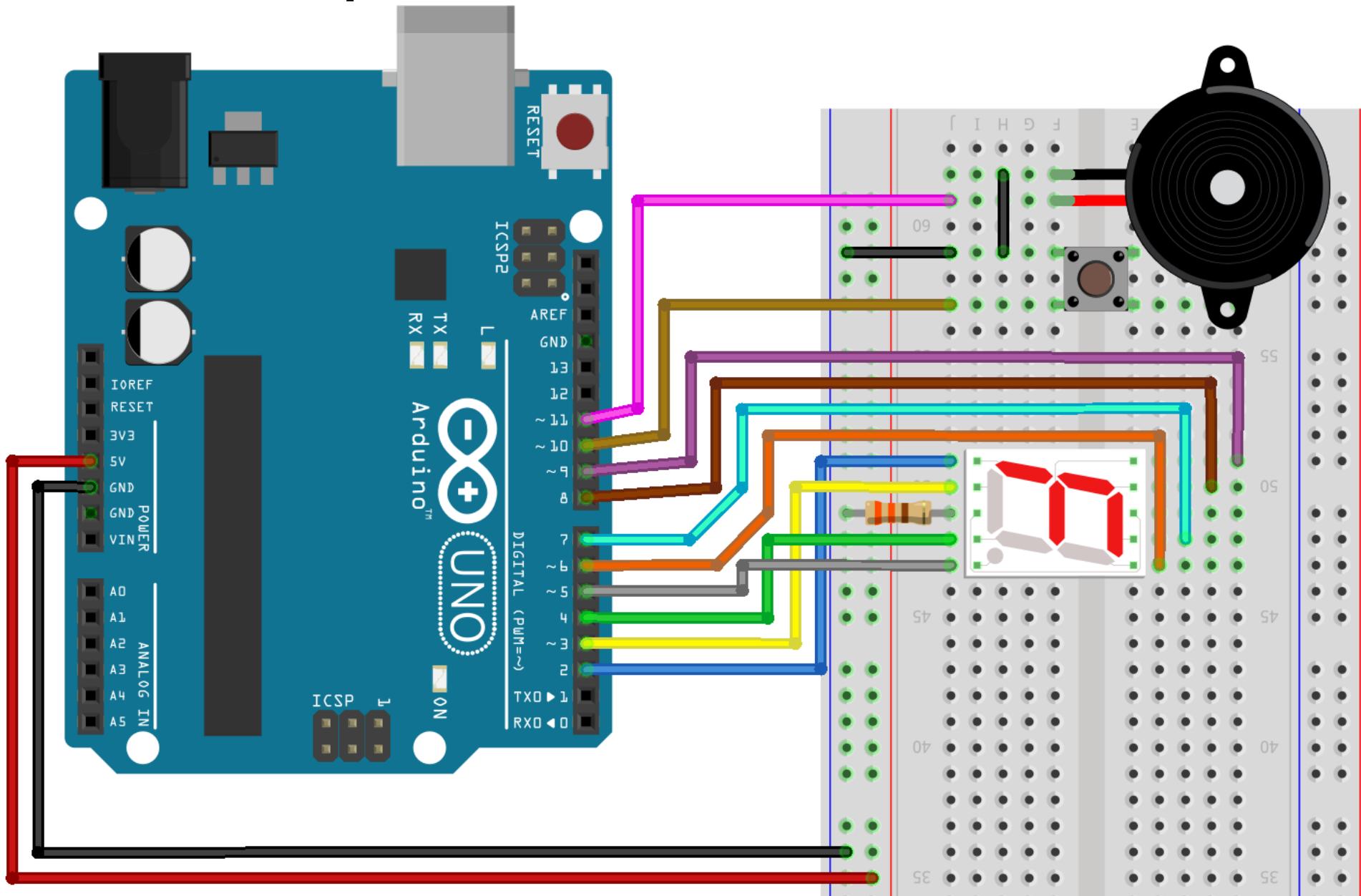
Objetivo:

Fazer uma contagem regressiva com um display de 7 segmentos, e ao final acionar um buzzer que pode ser desligado por um botão

Conceito: Aplicação dos conceitos de PWM, entrada e saída digitais.

9º Programa: Despertador

Montagem:



Display LCD

- LCD (Liquid Crystal Display) - Display de Cristal Líquido
- Resolução: 16x2
- É utilizado com a biblioteca <LiquidCrystal.h>



10º Programa: Hello World - LCD

Funções:

`lcd.begin(colunas, linhas)`

Indica para o programa quantas **colunas** e **linhas** o LCD possui

`lcd.print(variável ou string)`

Imprime na tela um **texto** ou uma **Variável**

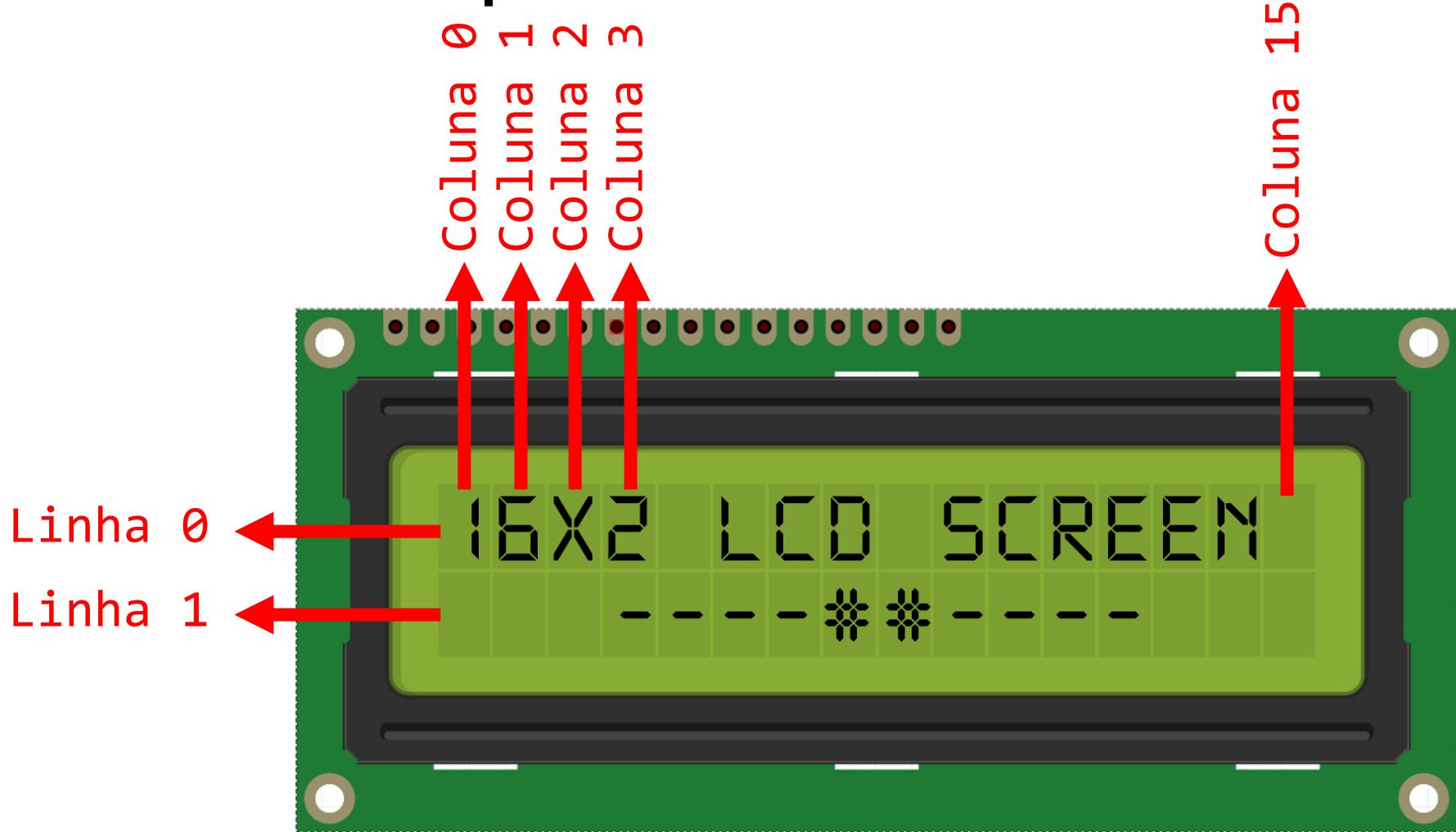
`lcd.clear()`

Limpa a tela e volta o cursor para a posição **(0,0)**

`lcd.setCursor(coluna, linha)`

Move o cursor para a **coluna** e **linha** desejadas

Ressaltando que...



10º Programa: Hello World - LCD

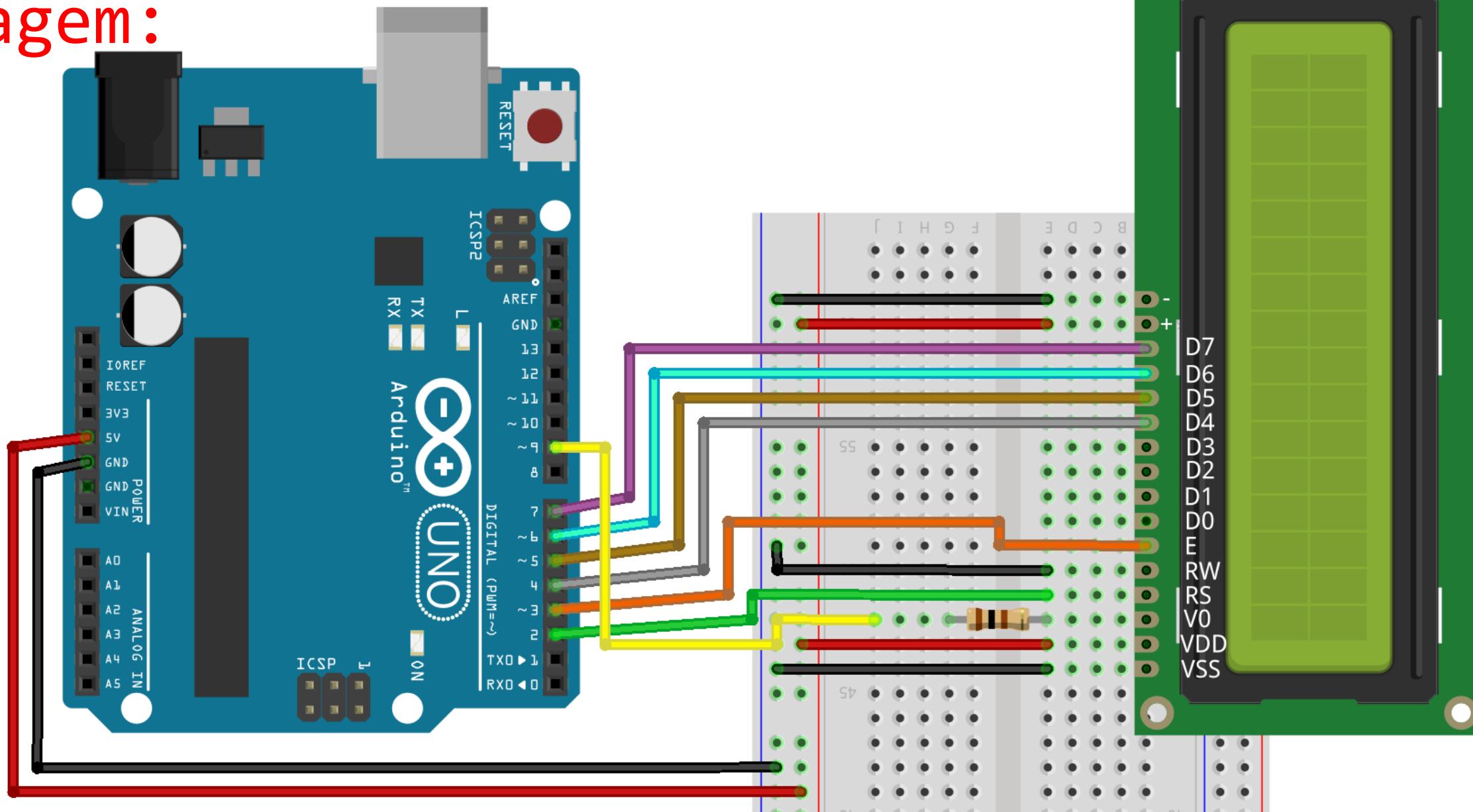
Objetivo:

Escrever "Hello World!" no centro da linha do LCD

Conceito: Funções do LCD

10º Programa: Hello World - LCD

Montagem:



Termíster NTC

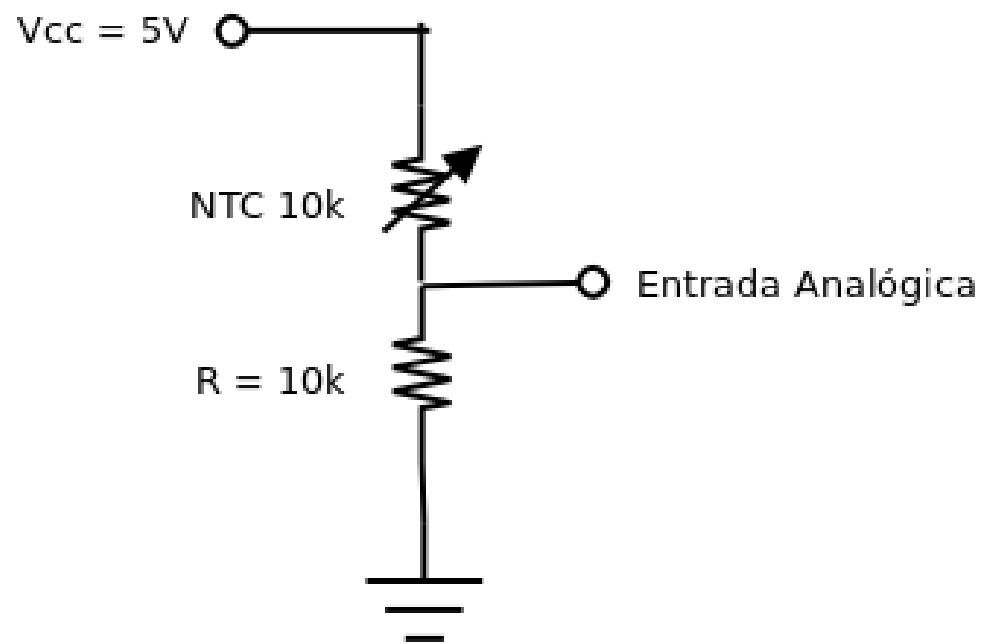
- Termíster – Resistência que varia de acordo com sua temperatura;
- NTC (Negative Temperature Coefficient) – Termíster com Coeficiente de Temperatura Negativo, ou seja, a resistência abaixa quando a temperatura aumenta;
- A relação Temperatura x Resistência é não-linear.

↑ Temperatura \approx Resistência ↓



Termíster NTC

- De forma similar ao LDR, o NTC pode ser colocado num divisor de tensão para que a temperatura seja descoberta.



11º Programa: Exibindo a temperatura no LCD

Biblioteca: "thermistor.h"

Funções:

`thermistor(pino, resNominal, coefBeta, resSérie)`
Define os parâmetros do NTC que está sendo utilizado.

`pino` = Pino em que o NTC está conectado

`resNominal` = Resistência nominal do NTC a 25°C

`coefBeta` = Coeficiente Beta do NTC

`resSérie` = Valor da resistência em série com o NTC

`thermistor.read()`

Retorna o valor da temperatura, em graus Celsius

11º Programa: Exibindo a temperatura no LCD

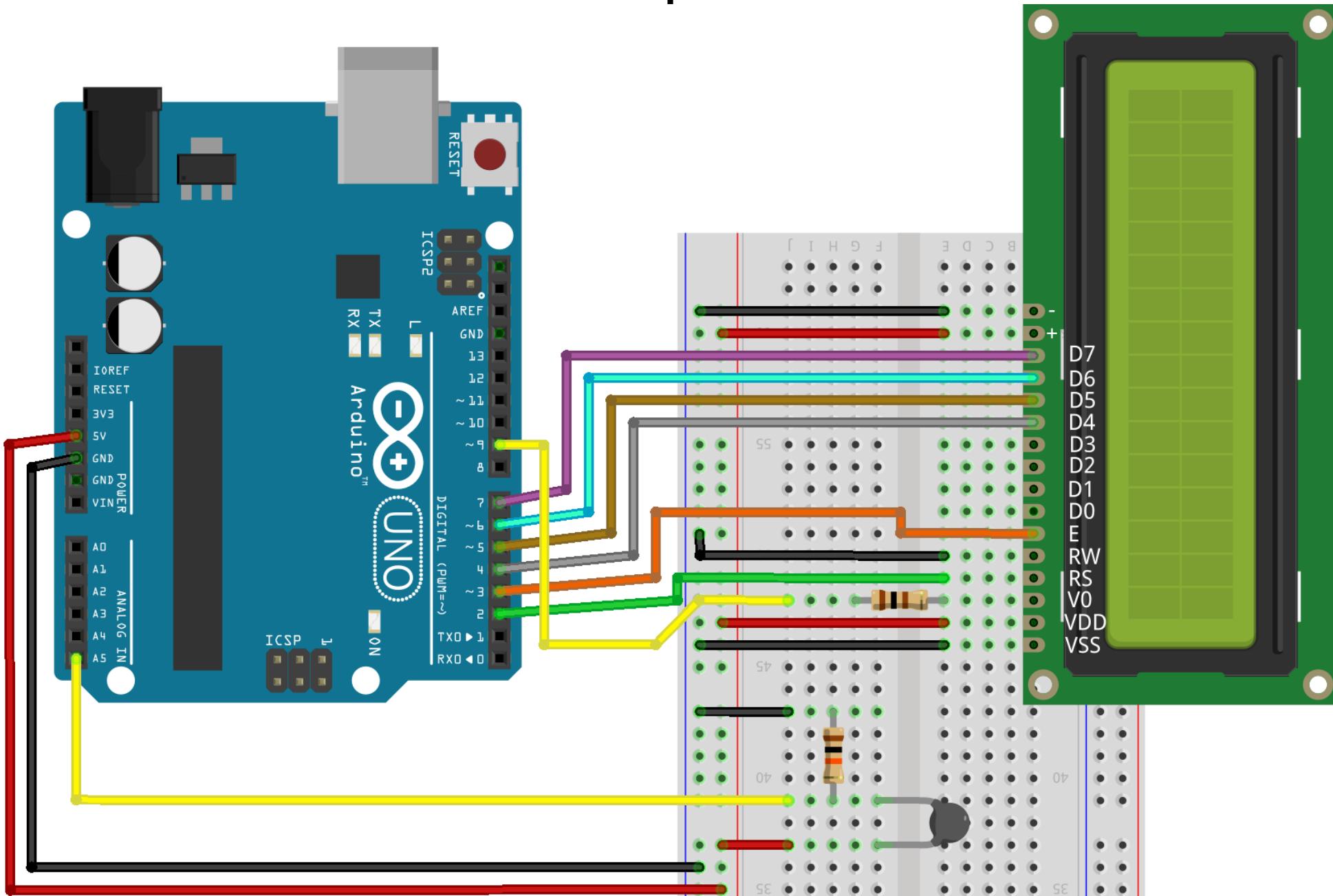
Objetivo:

Obter a temperatura do NTC e imprimí-la na tela do LCD.

Conceito: Funções do LCD e funcionamento do NTC

11º Programa: Exibindo a temperatura no LCD

Montagem:



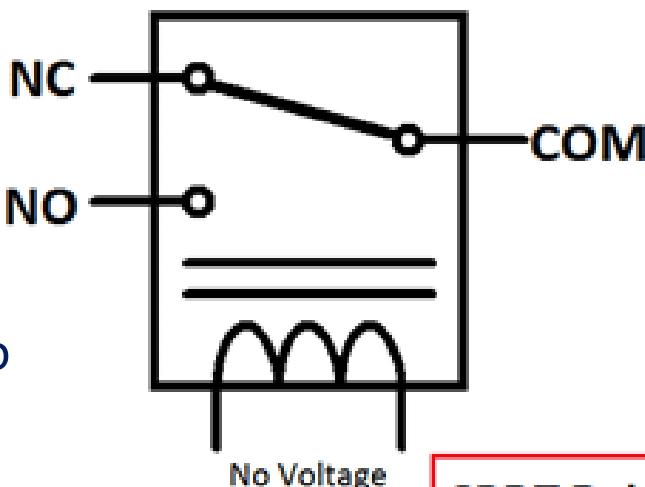
Relé

- É um componente utilizado para controlar o chaveamento de fontes de energia externas ao Arduino
- Aplicando uma pequena tensão na bobina é possível mudar a posição da chave do relé

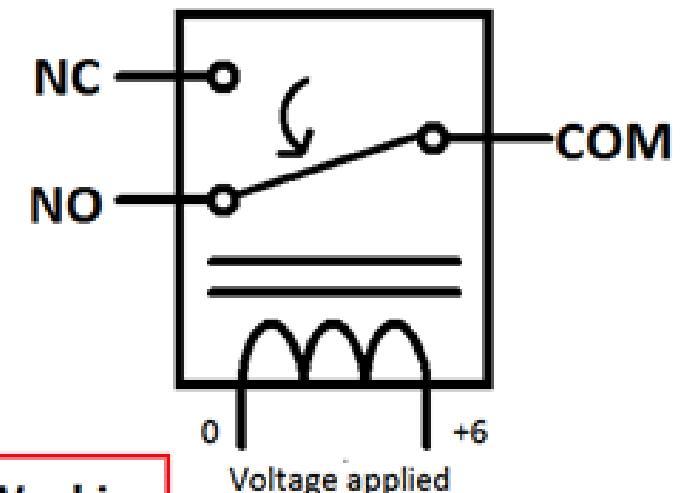
NC = Normalmente Fechado

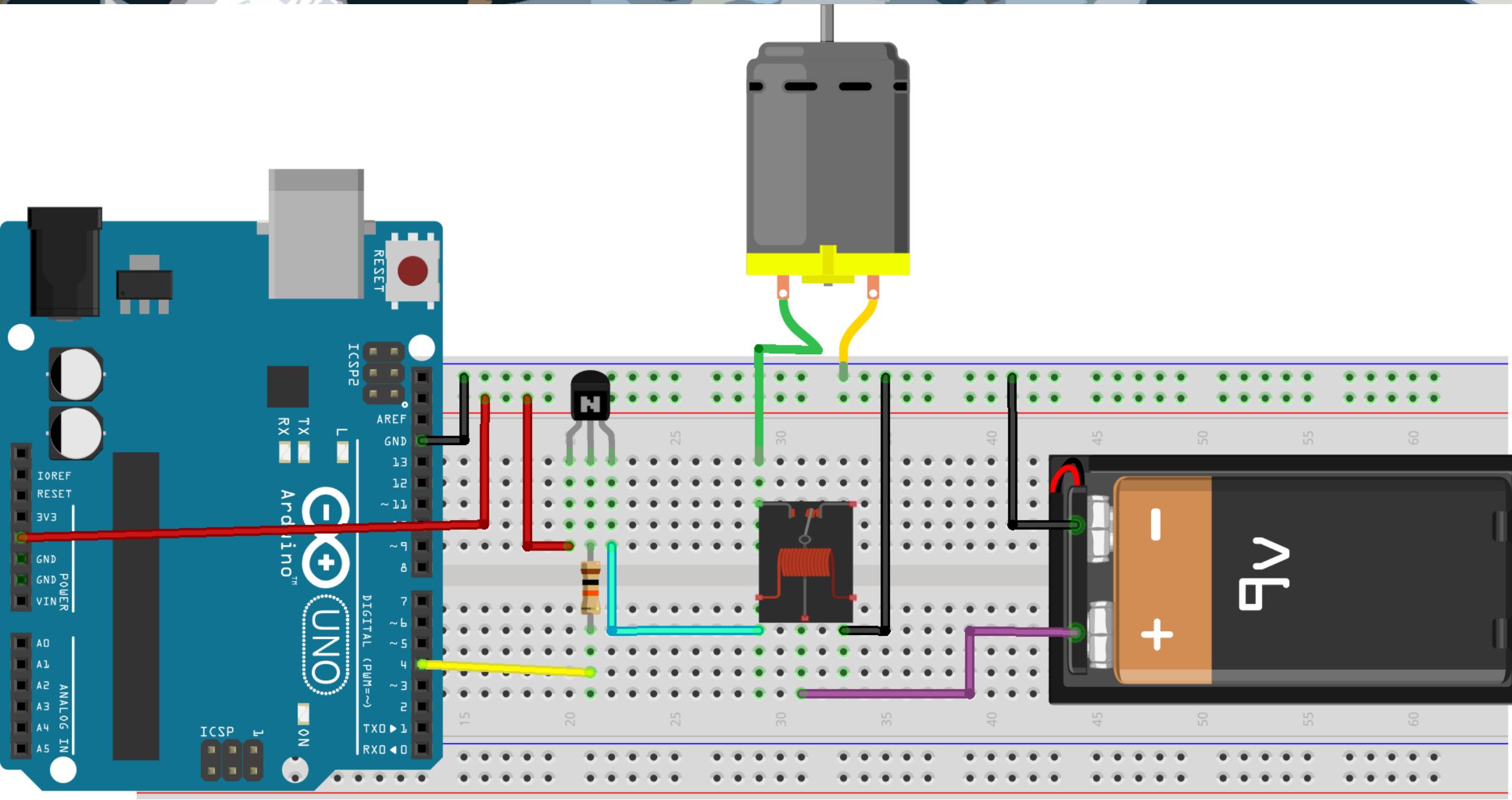
NO = Normalmente Aberto

COM = Comum



SPDT Relay Working





12º Programa: Acendendo um LED utilizando o Relé

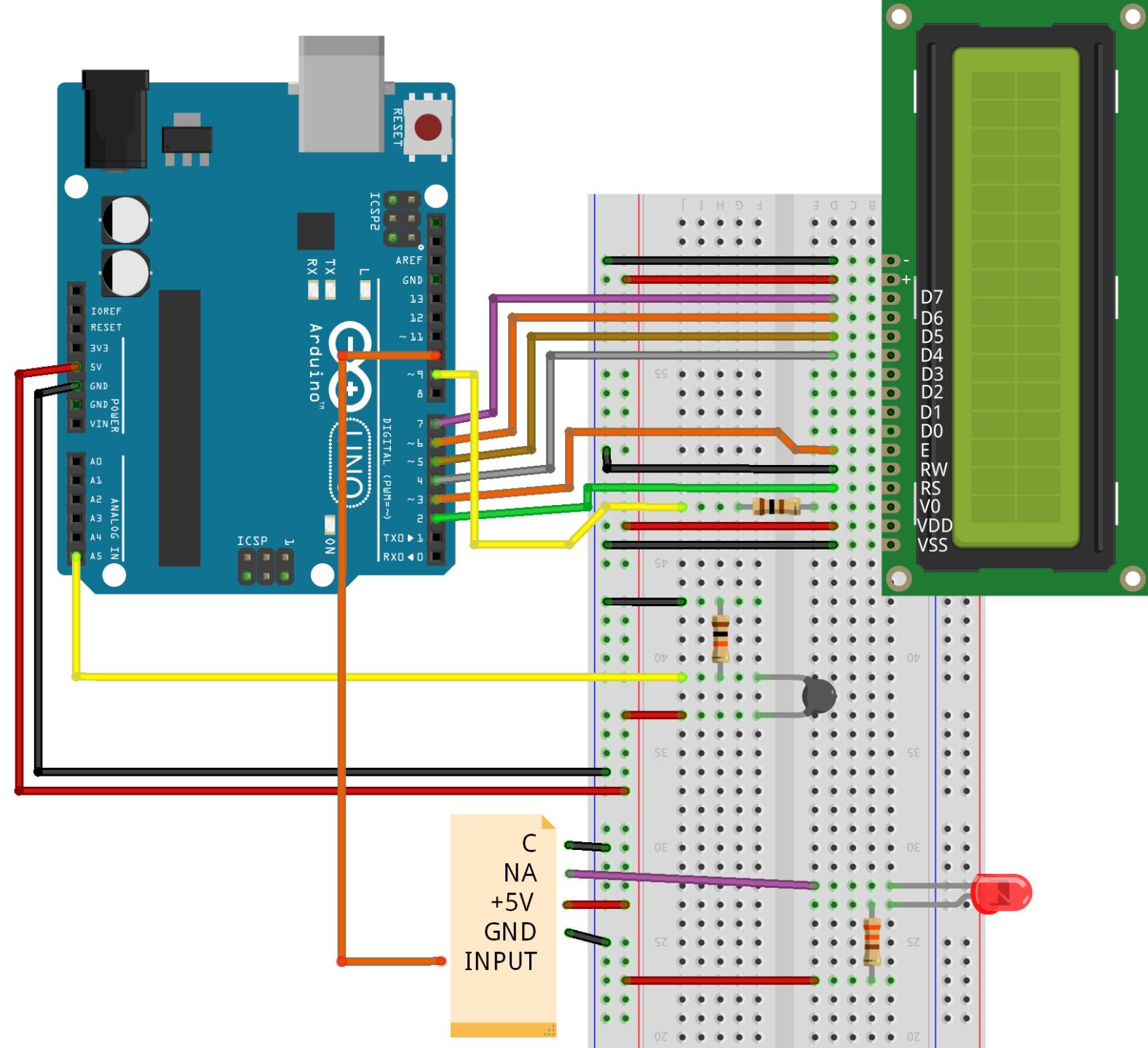
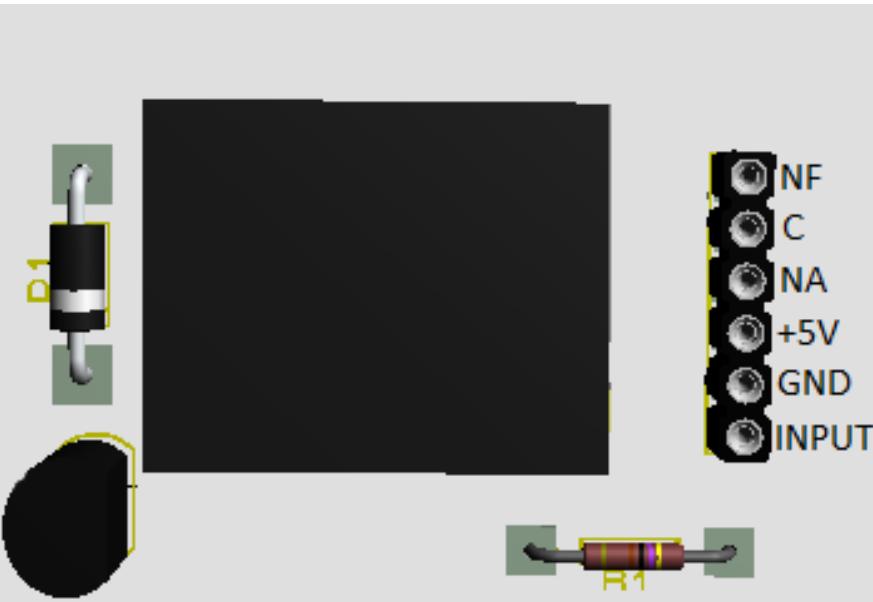
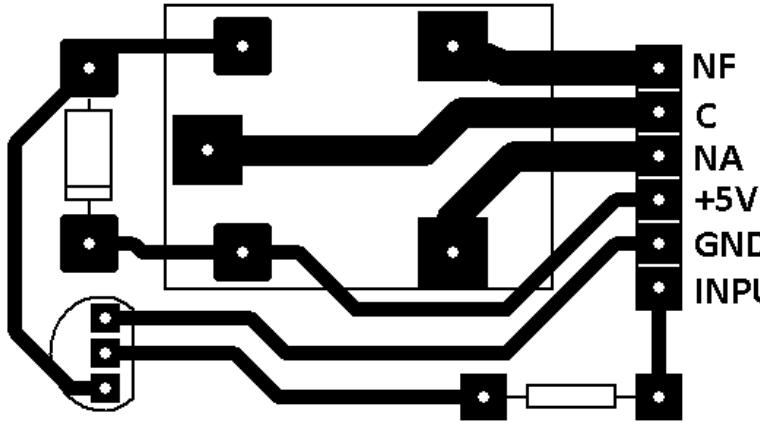
Objetivo:

Obter a temperatura do NTC, imprimí-la na tela do LCD, e se passar de 30 graus um LED é aceso pelo Relé

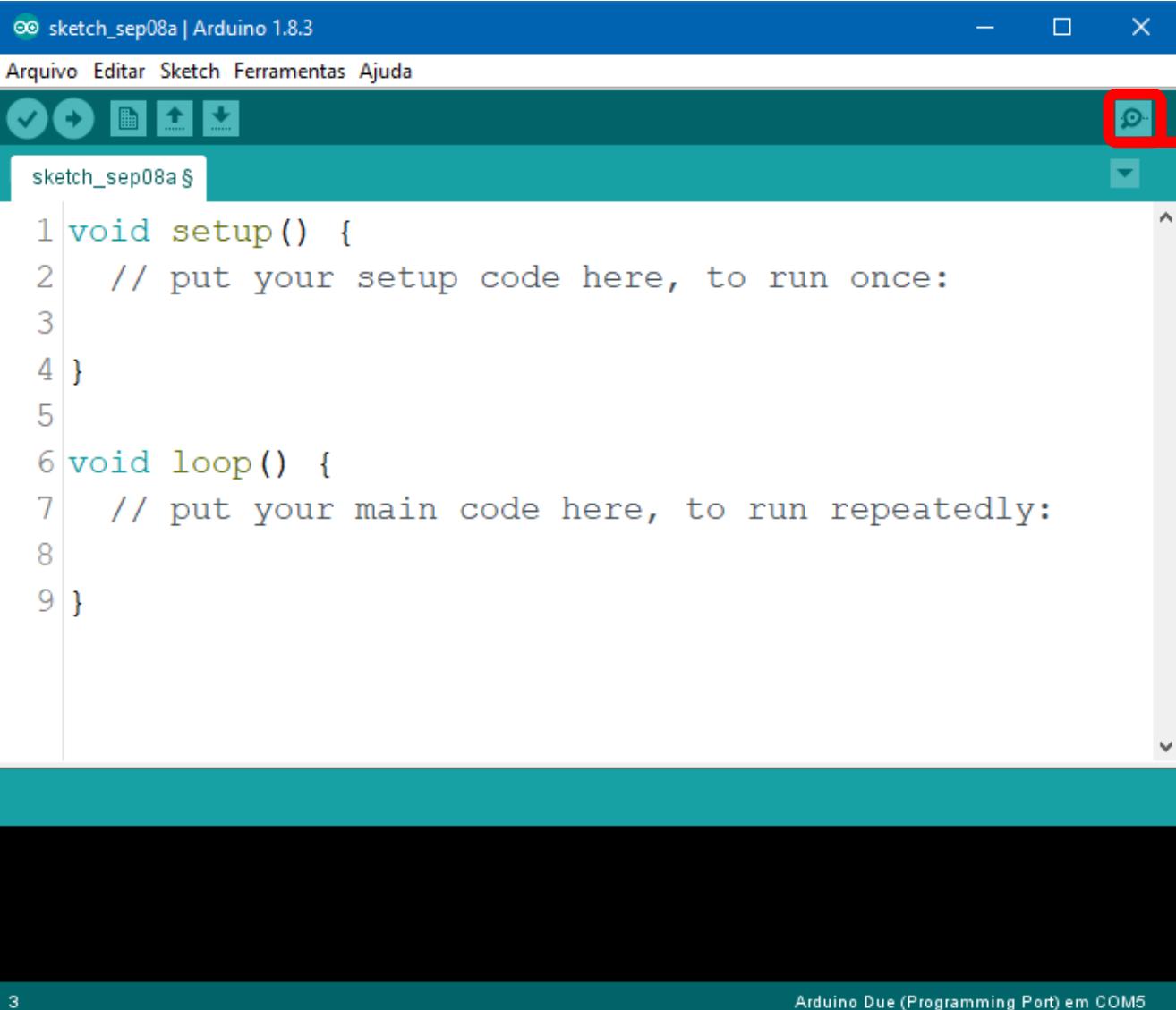
Conceito: Funcionamento do Relé

12º Programa:
Acendendo um LED
utilizando o Relé

Montagem:



Monitor Serial



The screenshot shows the Arduino IDE interface. The title bar reads "sketch_sep08a | Arduino 1.8.3". The menu bar includes "Arquivo", "Editar", "Sketch", "Ferramentas", and "Ajuda". Below the menu is a toolbar with icons for file operations. A red box highlights the "Serial Monitor" icon (a blue speech bubble with a white 'm') located in the top right corner of the toolbar. The main area displays the code for "sketch_sep08a":

```
1 void setup() {  
2     // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7     // put your main code here, to run repeatedly:  
8 }  
9
```

The status bar at the bottom indicates "Arduino Due (Programming Port) em COM5".

Monitor Serial

- Exibe informações vindas do Arduino
- Envia informações ao Arduino

Olar

Velocidade de
comunicação

Quebra
de linha

Nova-linha

57600 velocidade

Clear output

Auto-rolagem



10:30
12/09/2018

13º Programa: Enviando informações do Arduino ao computador

Funções:

`Serial.begin(velocidade)`

Inicia a comunicação do Arduino com o computador

`Serial.print(variável ou texto)`

Imprime no monitor serial alguma informação

`Serial.println(variável ou texto)`

Imprime no monitor serial alguma informação separada em uma linha

13º Programa: Enviando informações do Arduino ao computador

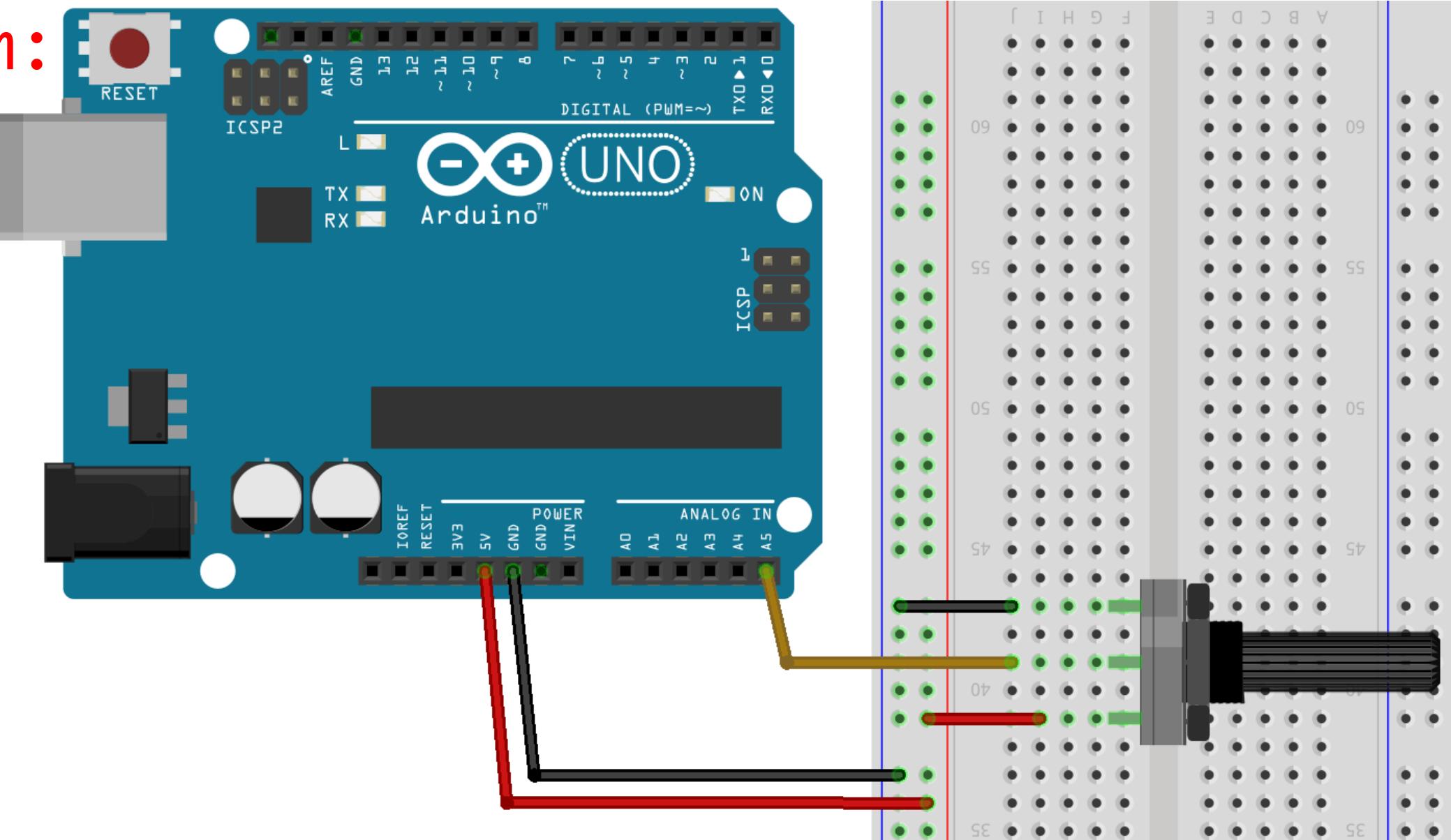
Objetivo:

Enviar o valor de tensão lido de um potenciômetro para o computador

Conceito: Utilização do Monitor Serial

13º Programa: Enviando informações do Arduino ao computador

Montagem:



Olar

Enviar

Informação a ser
enviada

 Auto-rolagem

Nova-linha

57600 velocidade

Clear output



10:30

12/09/2018

1

14º Programa: Enviando informações do computador para o Arduino

Funções:

`Serial.available()`

Retorna a **quantidades de bytes** disponíveis para leitura

`Serial.read()`

Lê **um byte** de informação

14º Programa: Enviando informações do computador para o Arduino

Objetivo:

Enviar algum dado para o Arduino e retornar o valor

Conceito: Utilização do Monitor Serial

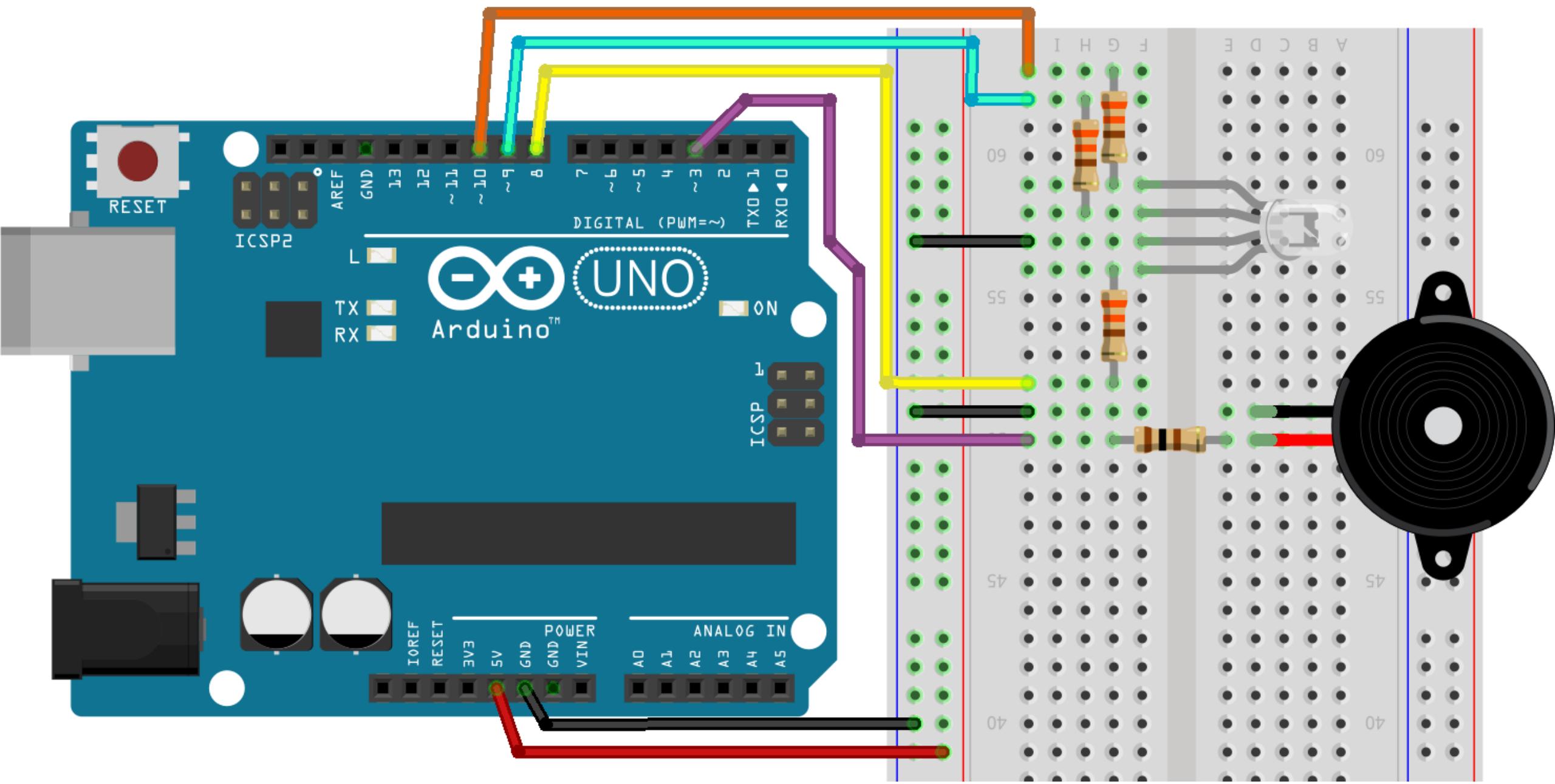
15º Programa: Controlando dispositivos conectados ao Arduino pelo computador

Objetivo:

Controlar um LED e um Buzzer pelo computador

Conceito: Utilização do Monitor Serial

Montagem:



Sugestões para estudos futuros:

- Função millis() ao invés do delay()
- Interrupções Externas
- Comunicação Serial Avançada
- Integração com outros programas
- Conversor DA (apenas Arduino DUE)
- Criação de bibliotecas
- Outros microcontroladores
 - Raspberry Pi
 - STM32



FIM!

Obrigado!