



PRIMEIROS PASSOS NA PLATAFORMA

ICOMP - UFAM

marcos.lima@icomp.ufam.edu.br

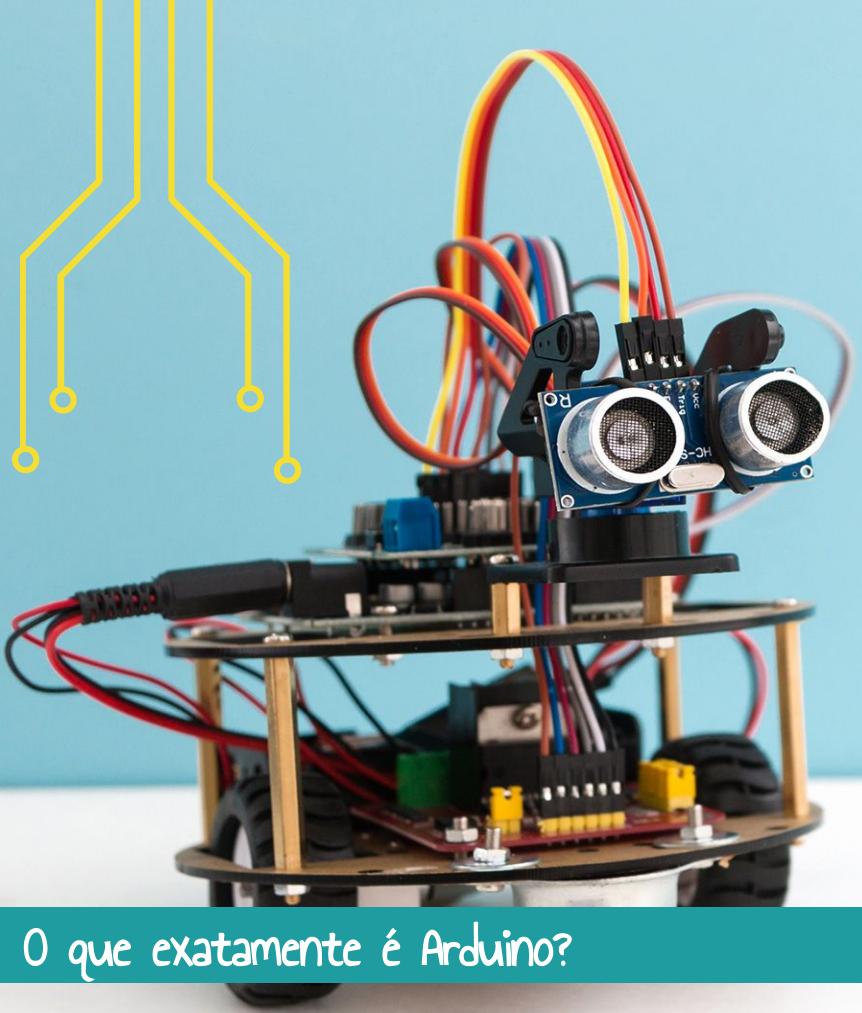
2022

Nossas Metas!!!

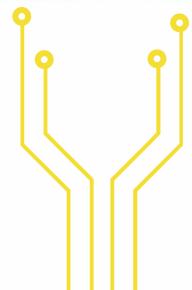
- Definir o que é a Plataforma Arduino.
- Descrever a estrutura básica de uma placa Arduino.
- Comentar sobre as “Variantes” Arduino.
- Expandir os horizontes com alguns projetos feitos com Arduino.
- Apresentar o ambiente de desenvolvimento: Arduino IDE.
- Aprender sobre alguns componentes eletrônicos básicos.
- Construir os primeiros circuitos com Arduino.
- DIVERSÃO durante todo o curso! :D



Introdução

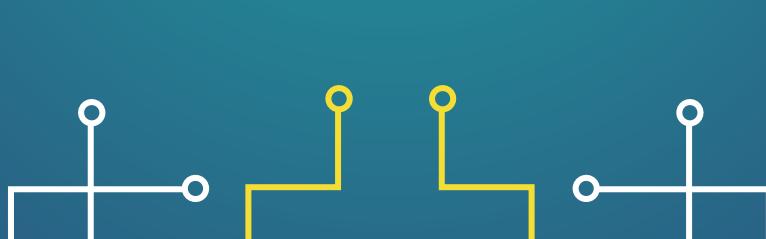


O que exatamente é Arduino?



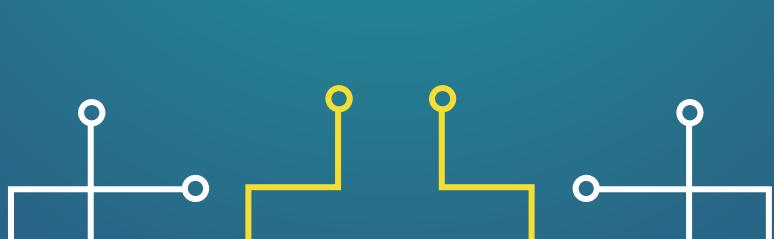


“Arduino é uma plataforma open-source de prototipagem eletrônica com hardware e software flexíveis e fáceis de usar, destinado a artistas, designers, hobbistas e qualquer pessoa interessada em criar projetos interativos.”



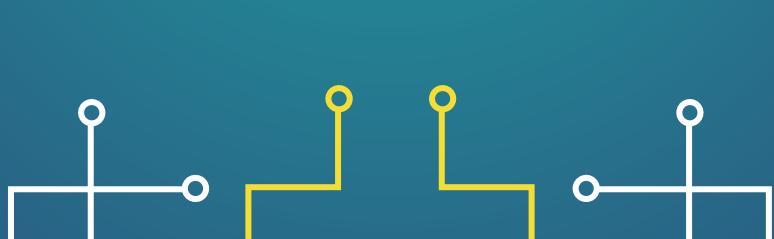


“Arduino é uma placa **microcontrolada** projetada para tornar o processo de uso de eletrônicos em projetos mais acessível.”





“Arduino é uma combinação de hardware e software, com intuito de ser facilmente acessível para aqueles com pouco ou nenhuma experiência com eletrônica.”



Um pouco de história...

Processing



Casey Reas
Benjamin Fry

Wiring



Hernando Barragán

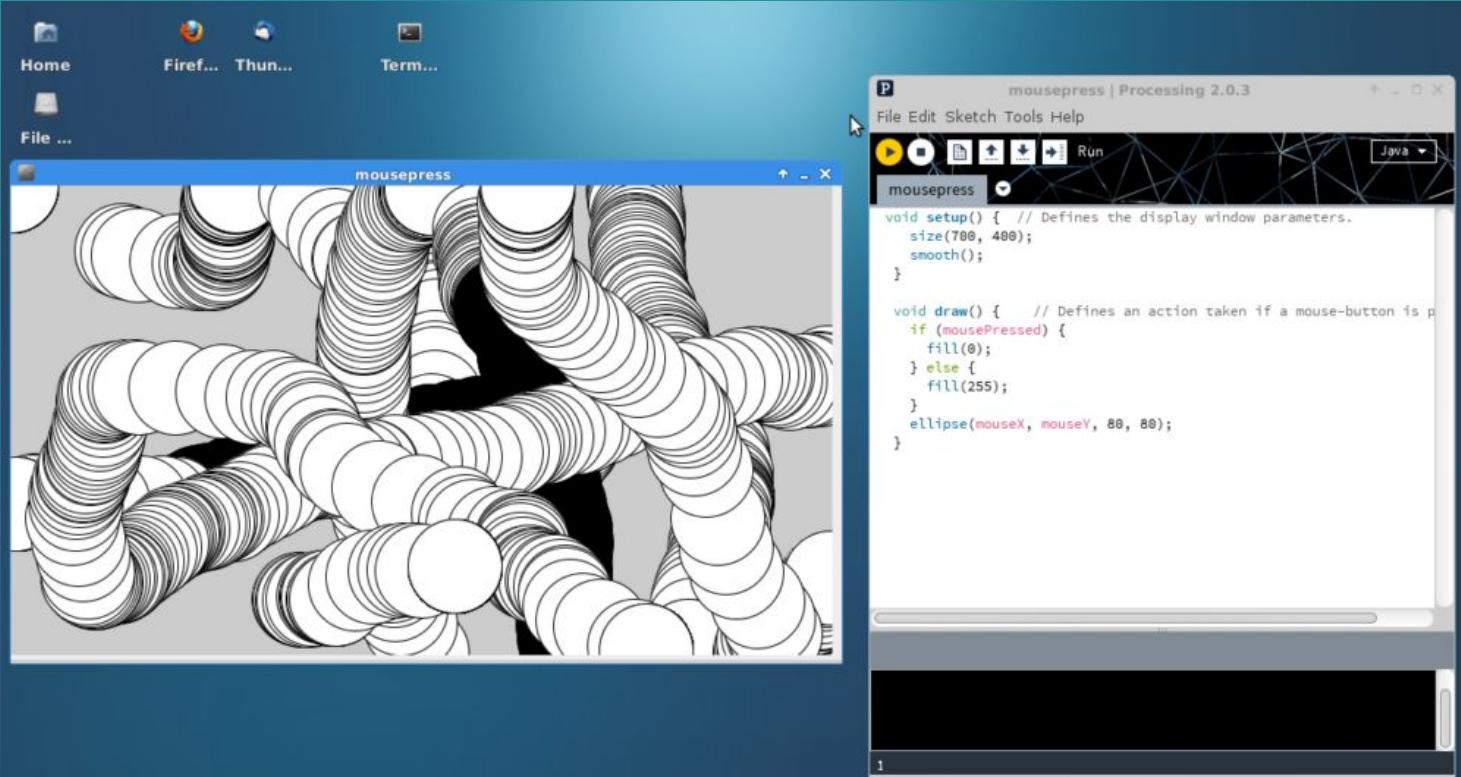
Arduino



David Cuartielles
Gianluca Martino
Tom Igoe
David Mellis
Massimo Banzi



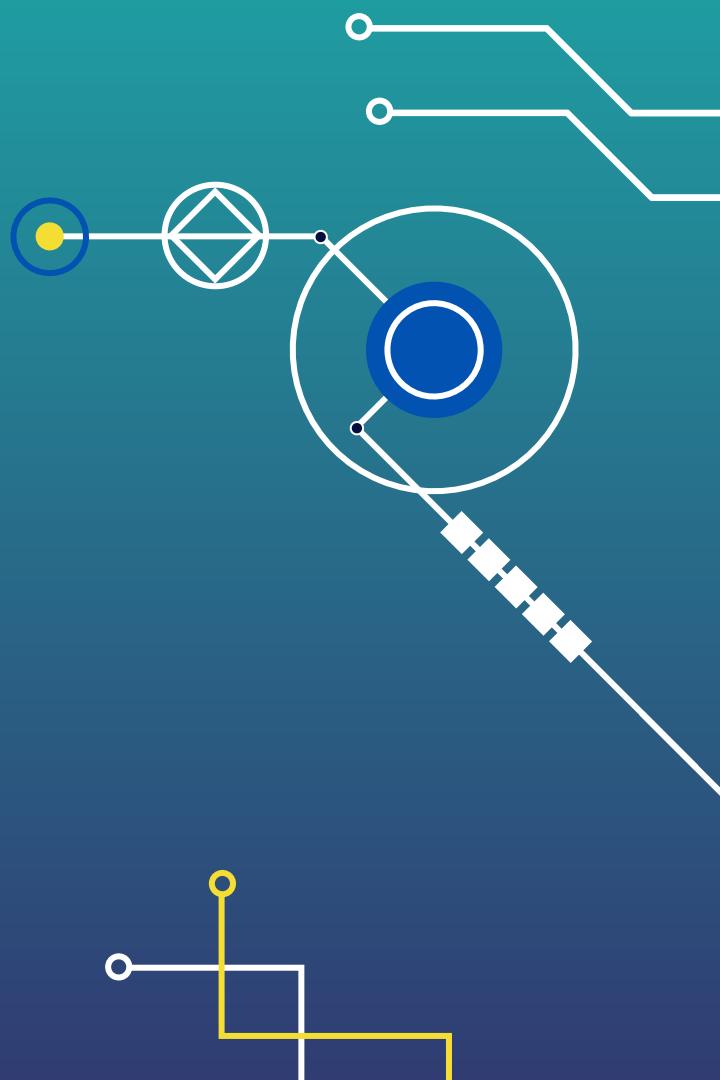
Os Criadores - David Cuartielles, Gianluca Martino, Tom Igoe, David Mellis, and Massimo Banzi.



Ambiente de Desenvolvimento Processing

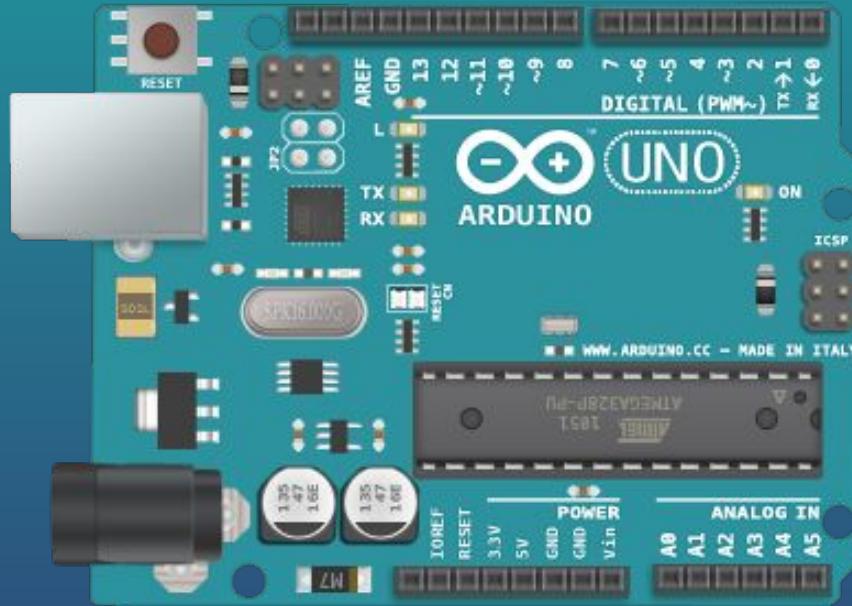
Por que utilizar Arduino?

- Baixo custo de aquisição;
- Perfeito para prototipagem (MVP);
- Fácil de usar e programar;
- Multiplataforma (Windows, Linux e Mac OS);
- Grande comunidade;
- Muitos modelos disponíveis;
- Grande quantidade de “shields” disponíveis;
- Projeto open-source;



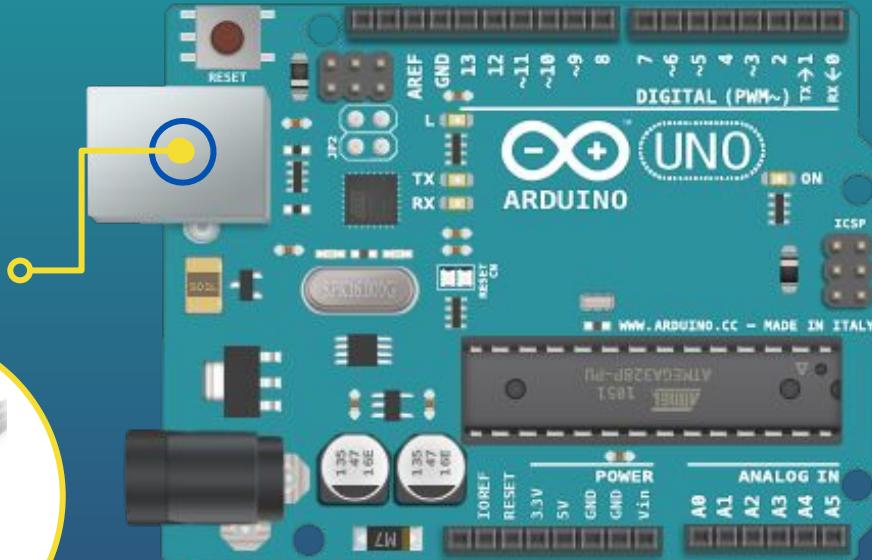


Arduino unboxing



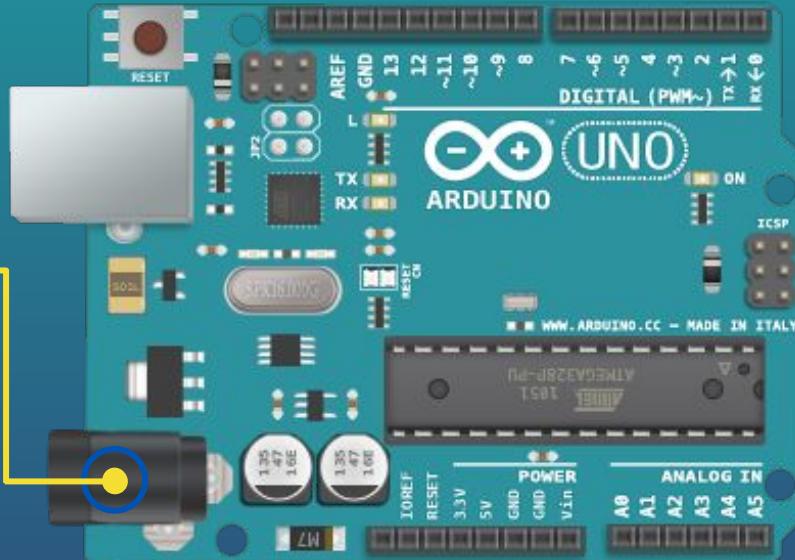
Exemplo de uma placa Arduino UNO

● Entrada USB tipo B



Utilizado para gravar código no microcontrolador e/ou alimentar a placa durante desenvolvimento

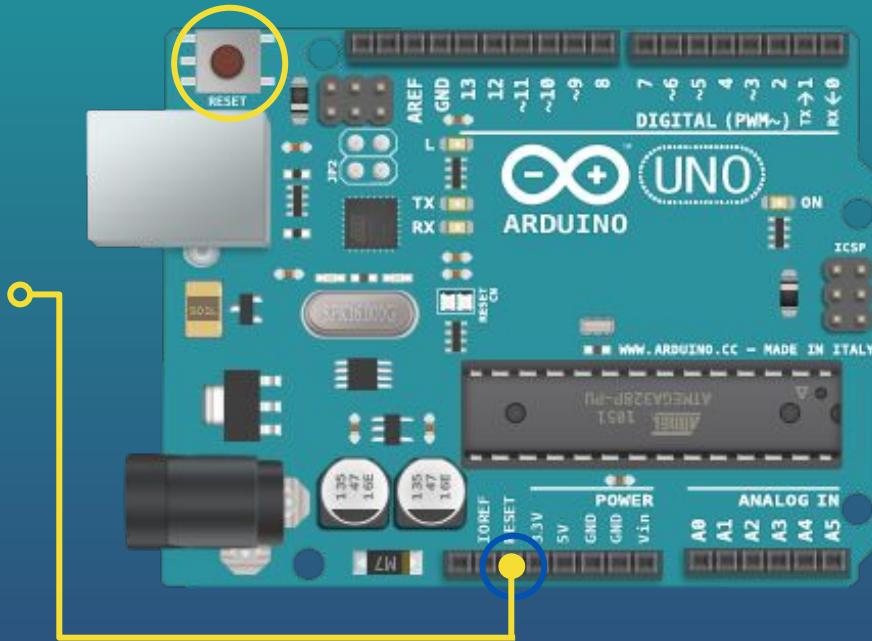
● Entrada POWER JACK^[1]



Nota 1: Tensão de Entrada (7 - 12V)

Utilizado para energizar a placa por meio de uma Fonte Externa ou uma Bateria.

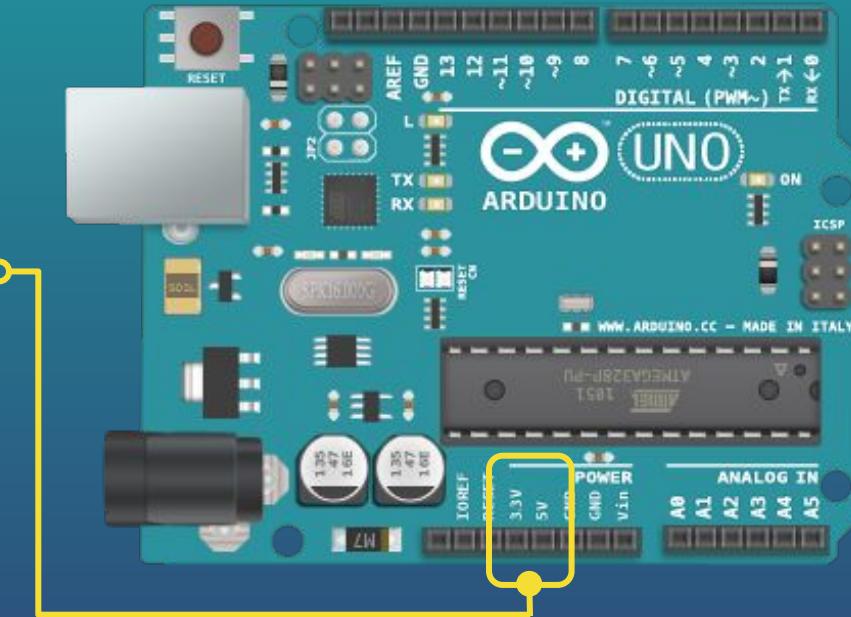
● Pinos RESET^[2]



Nota 2: **NÃO** apaga a programação!

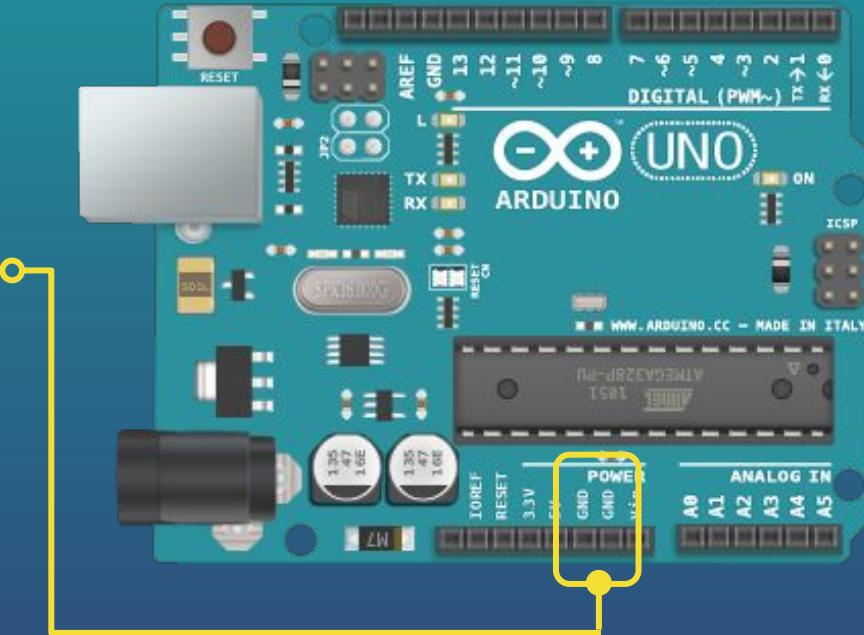
Pino utilizado para "resetar" a placa, ou seja, reiniciar o programa gravado no microcontrolador

• Pinos de 3.3V e 5V



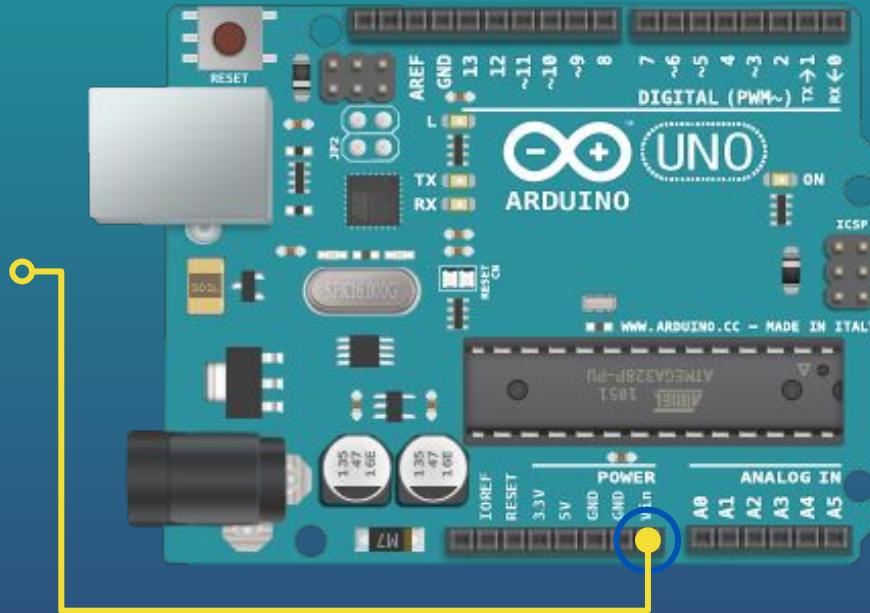
São pinos utilizados para alimentar circuitos e/ou componentes externos.

● Pinos GND



Os pinos "GND" (terra) são referenciais de tensão (zero volts) em circuitos eletrônicos.

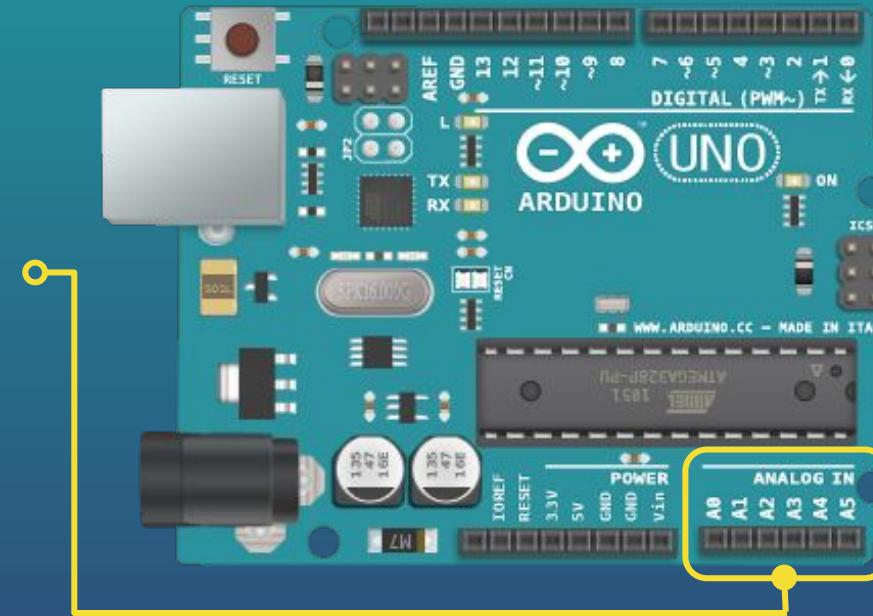
Pino Vin^[3]



Nota 3: Tensão de Entrada (7 - 12V)

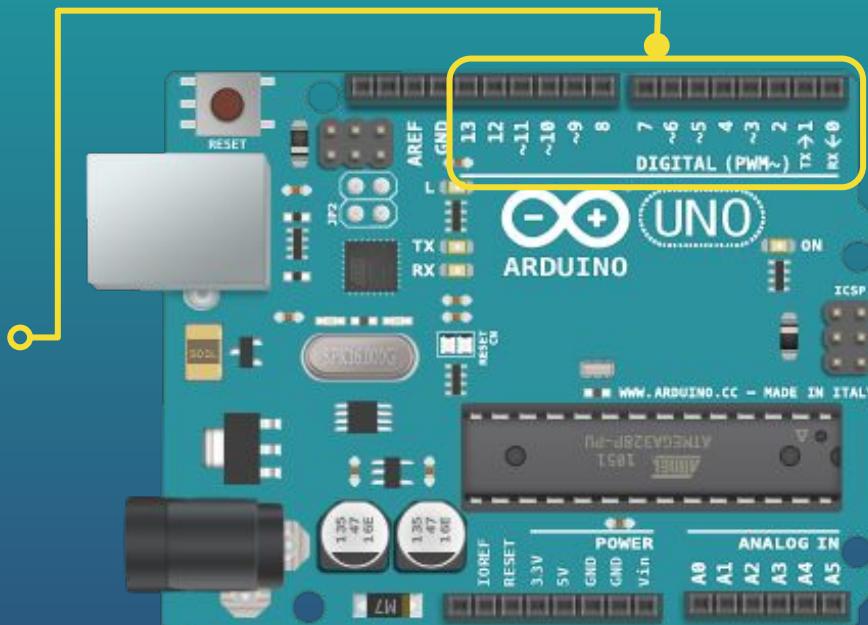
Utilizado para energizar a placa via Protoboard (placa de ensaio) ou alimentar um CKT com a mesma tensão de entrada.

● Pinos Analógicos



Podem ser utilizados para entrada/saída analógica, no modelo UNO assumem 2^{10} valores (0 - 1023).

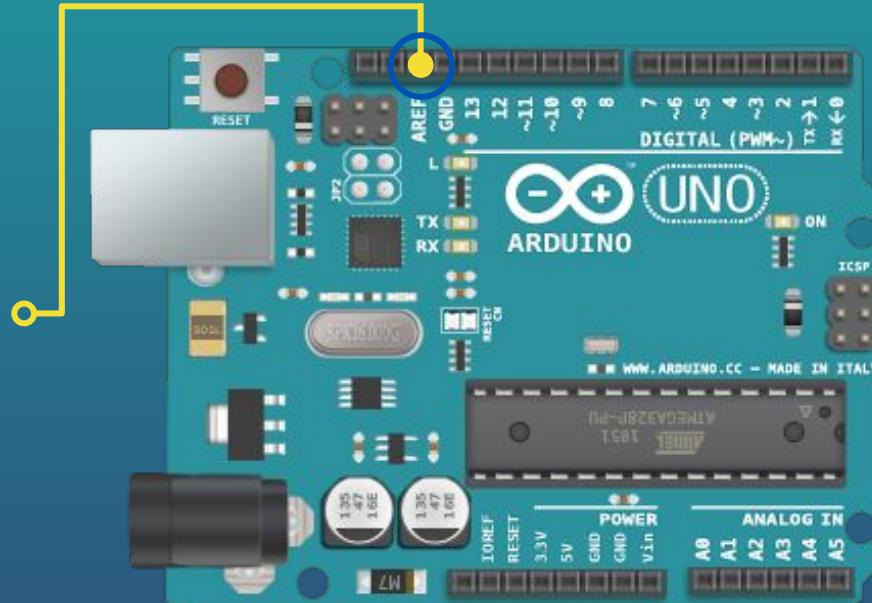
● Pinos Digitais^[4]



Nota 4: Pinos com (~) - PWM (Pulse Width Modulation).

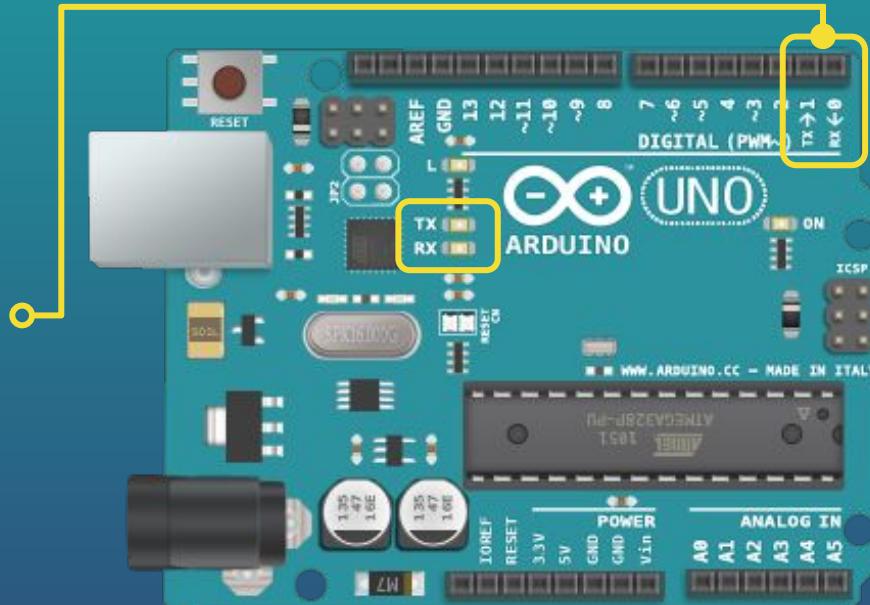
Podem ser utilizados para entrada/saída digital, no modelo UNO assumem 2^1 valores (0 ou 1).

Pino AREF
(Analogic Reference)



Pode ser utilizado para alterar a tensão padrão de operação dos pinos analógicos.

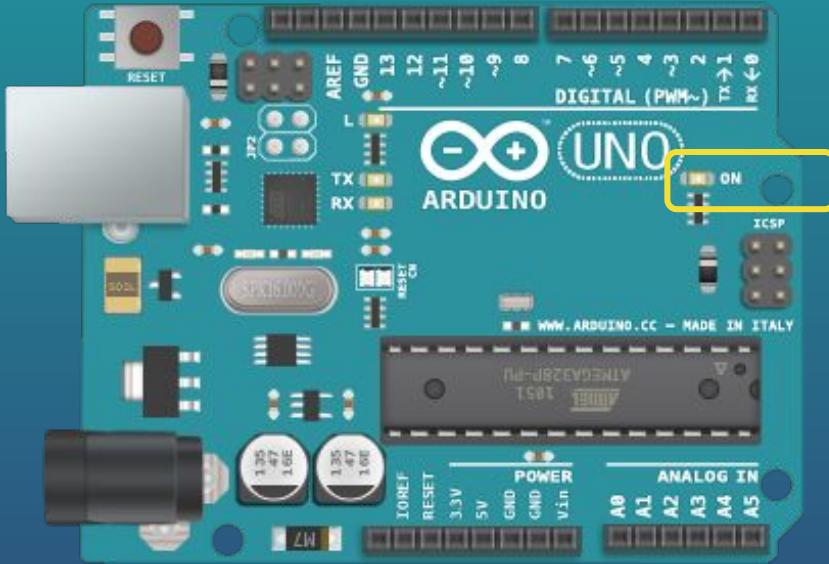
● Pinos TX e RX



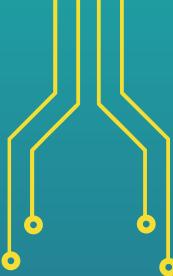
Podem ser utilizados para comunicação serial com outros dispositivos.



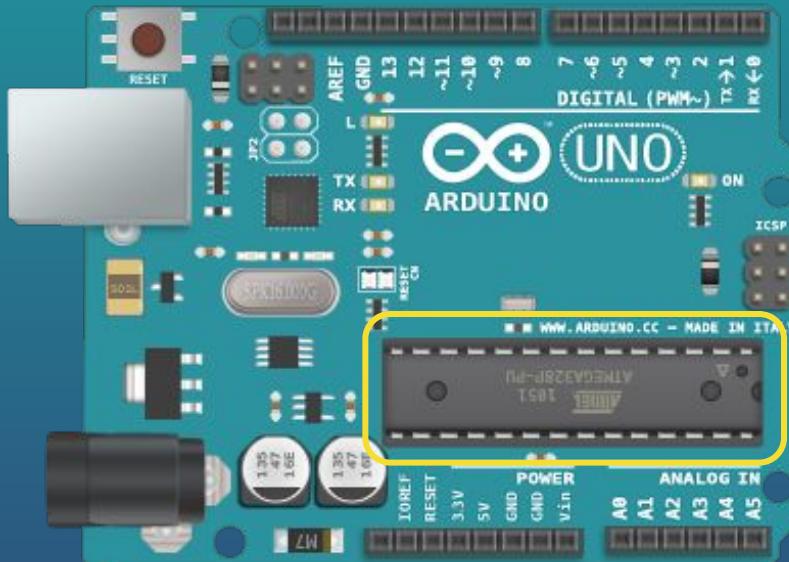
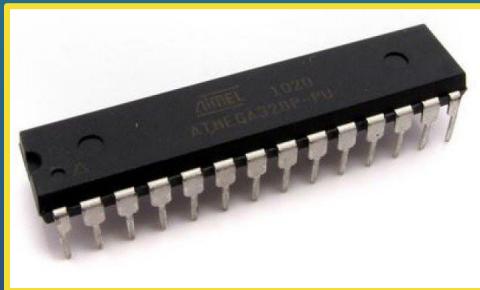
LED On



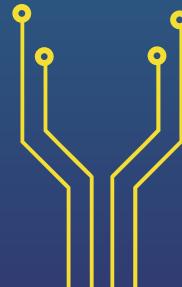
Indica se a placa está energizada e funcionando.



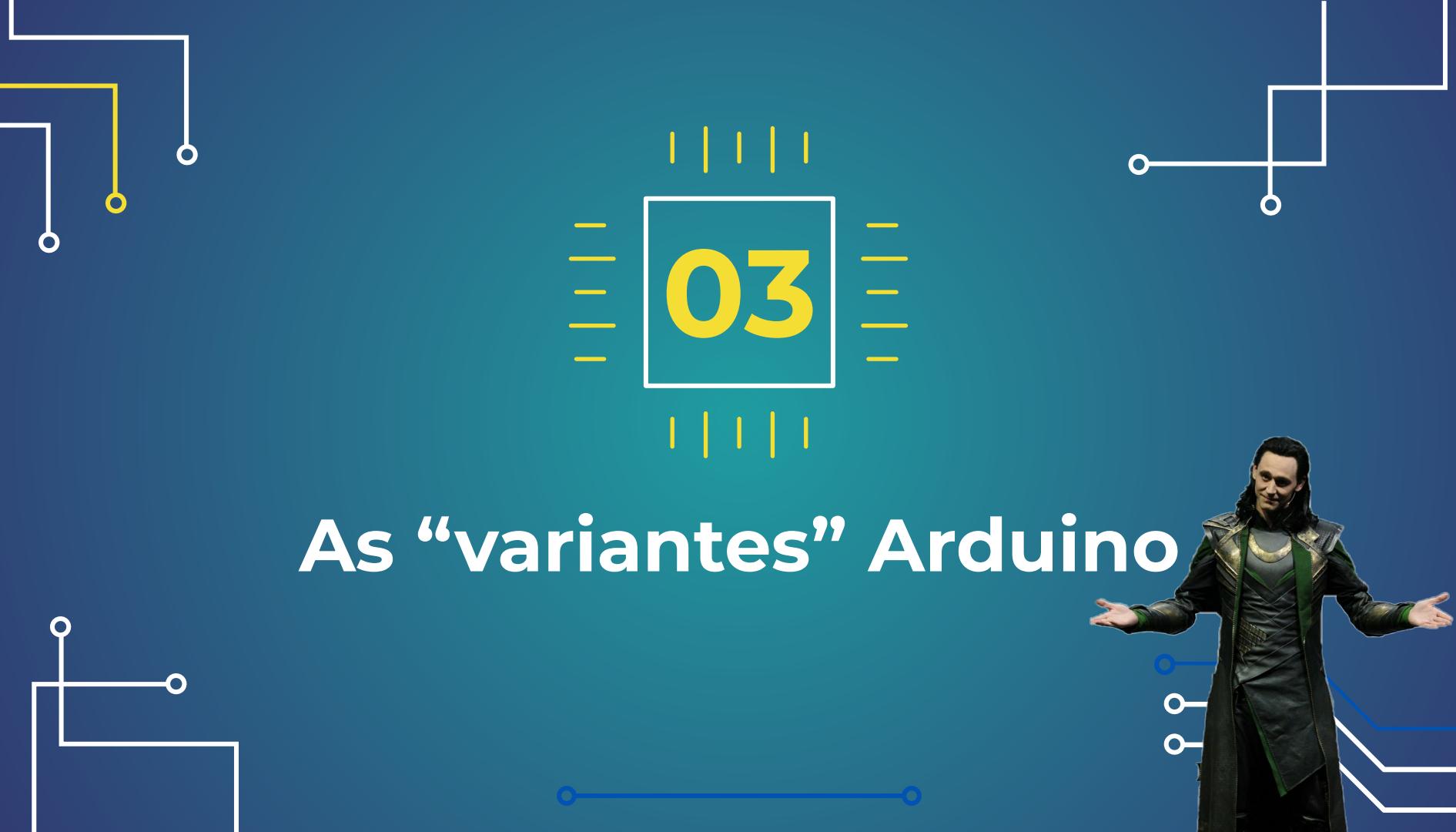
● Microcontrolador

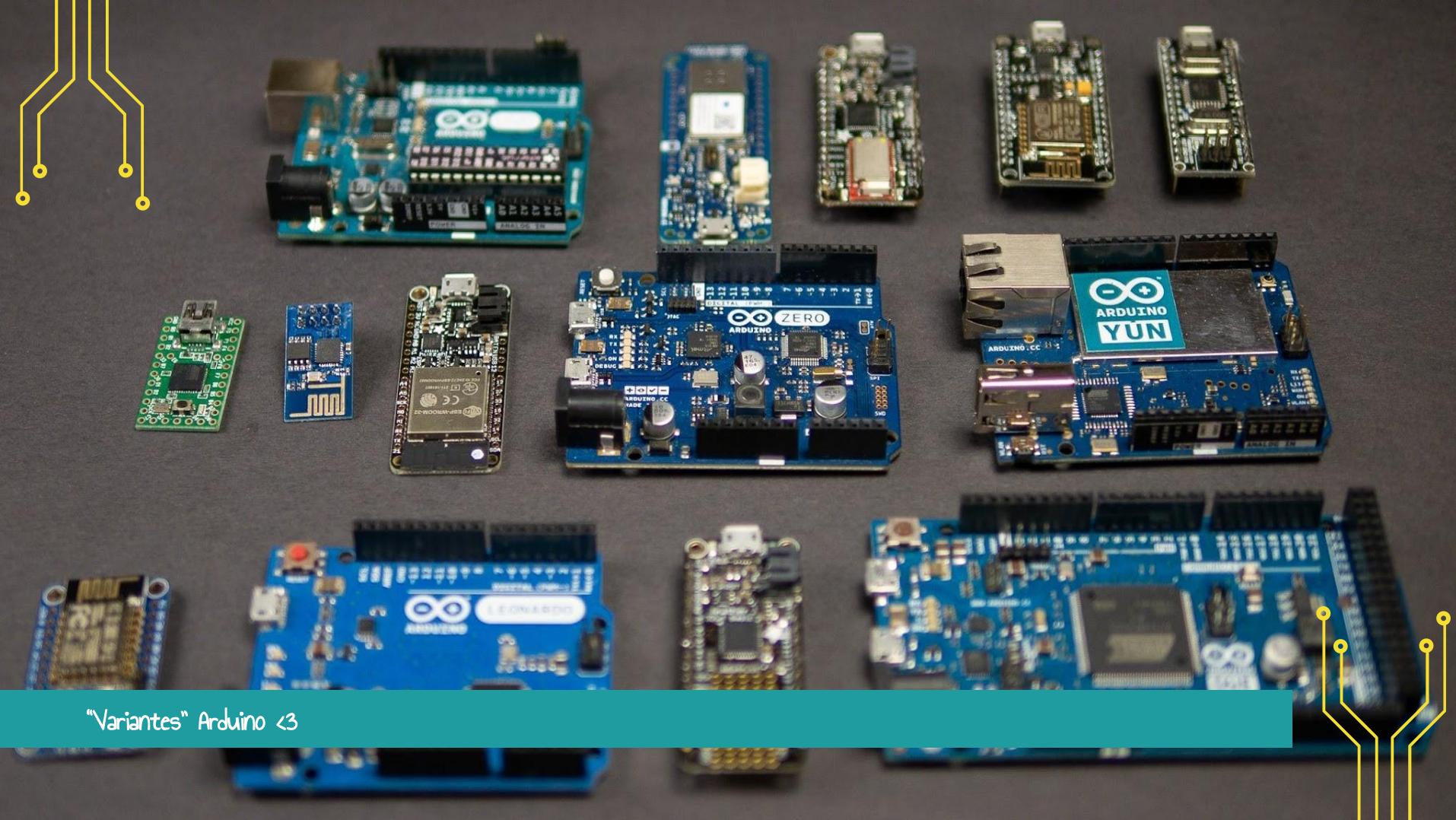


É o "cérebro" da placa, onde o código é executado. No modelo UNO ele pode ser trocado (upgrade)!

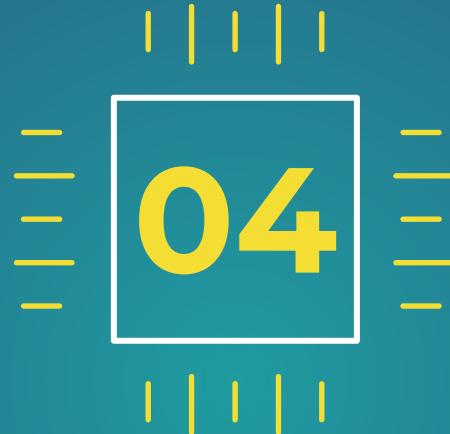


As “variantes” Arduino



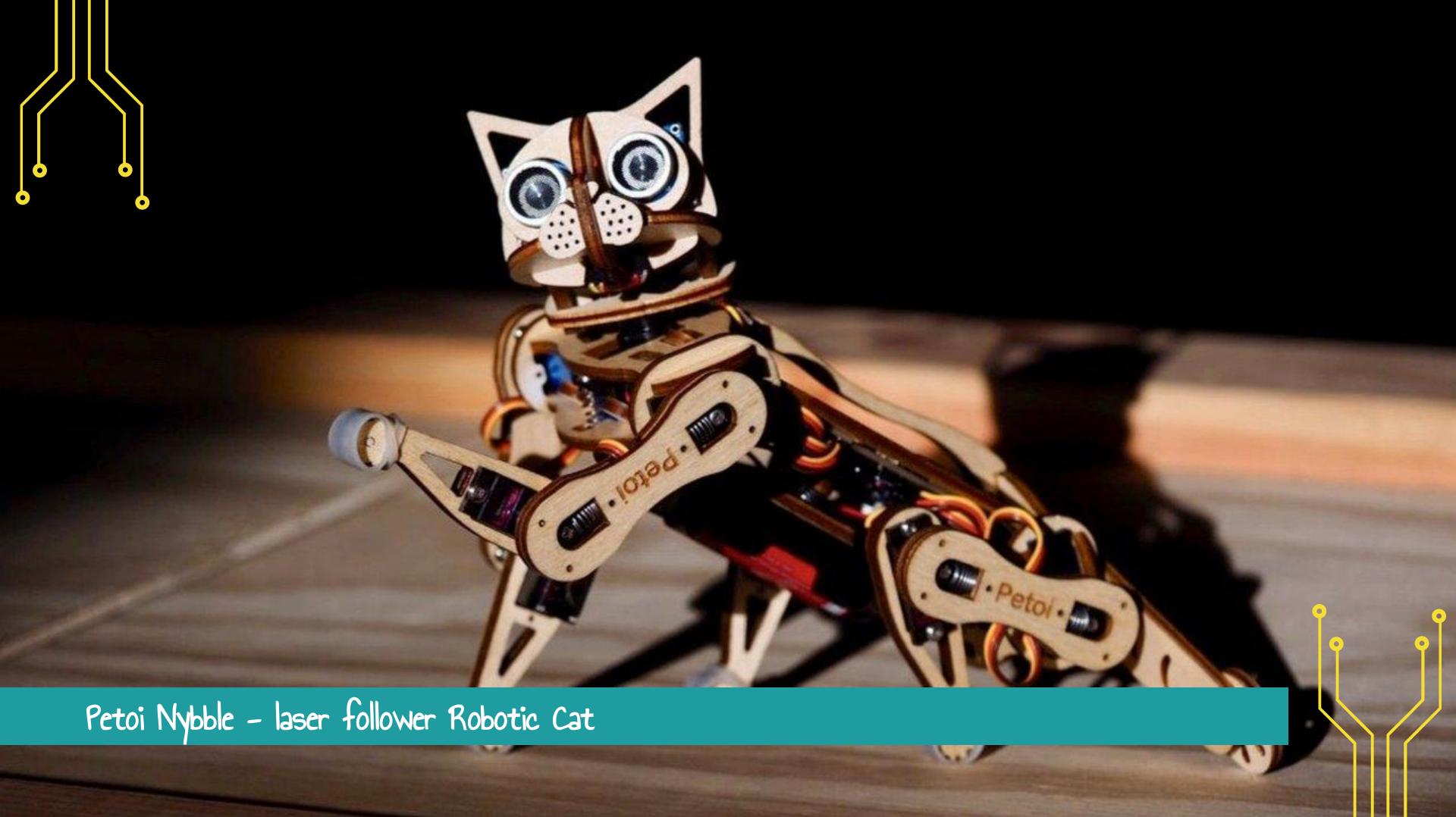


"Variantes" Arduino <3

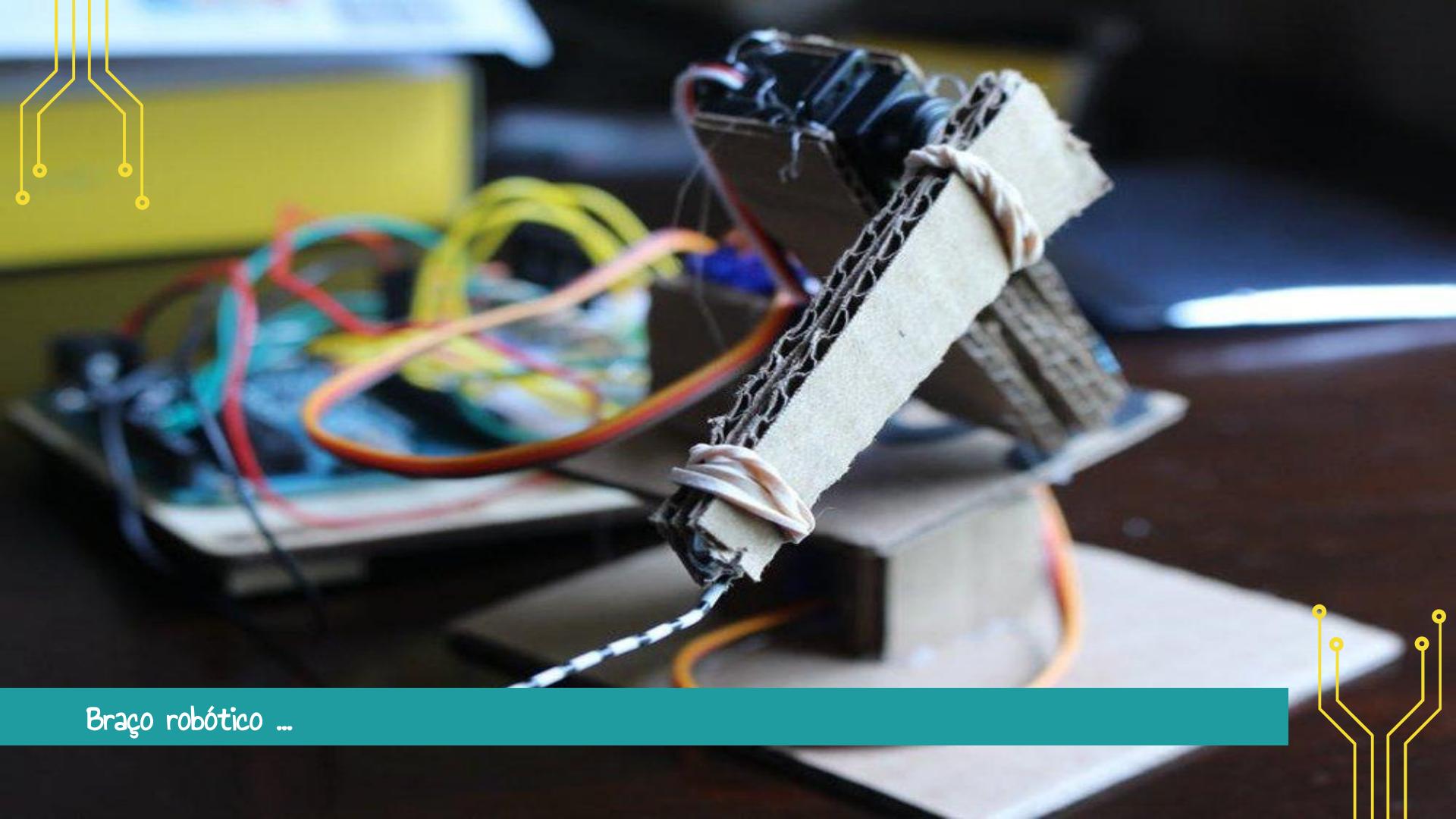


Expandindo os horizontes

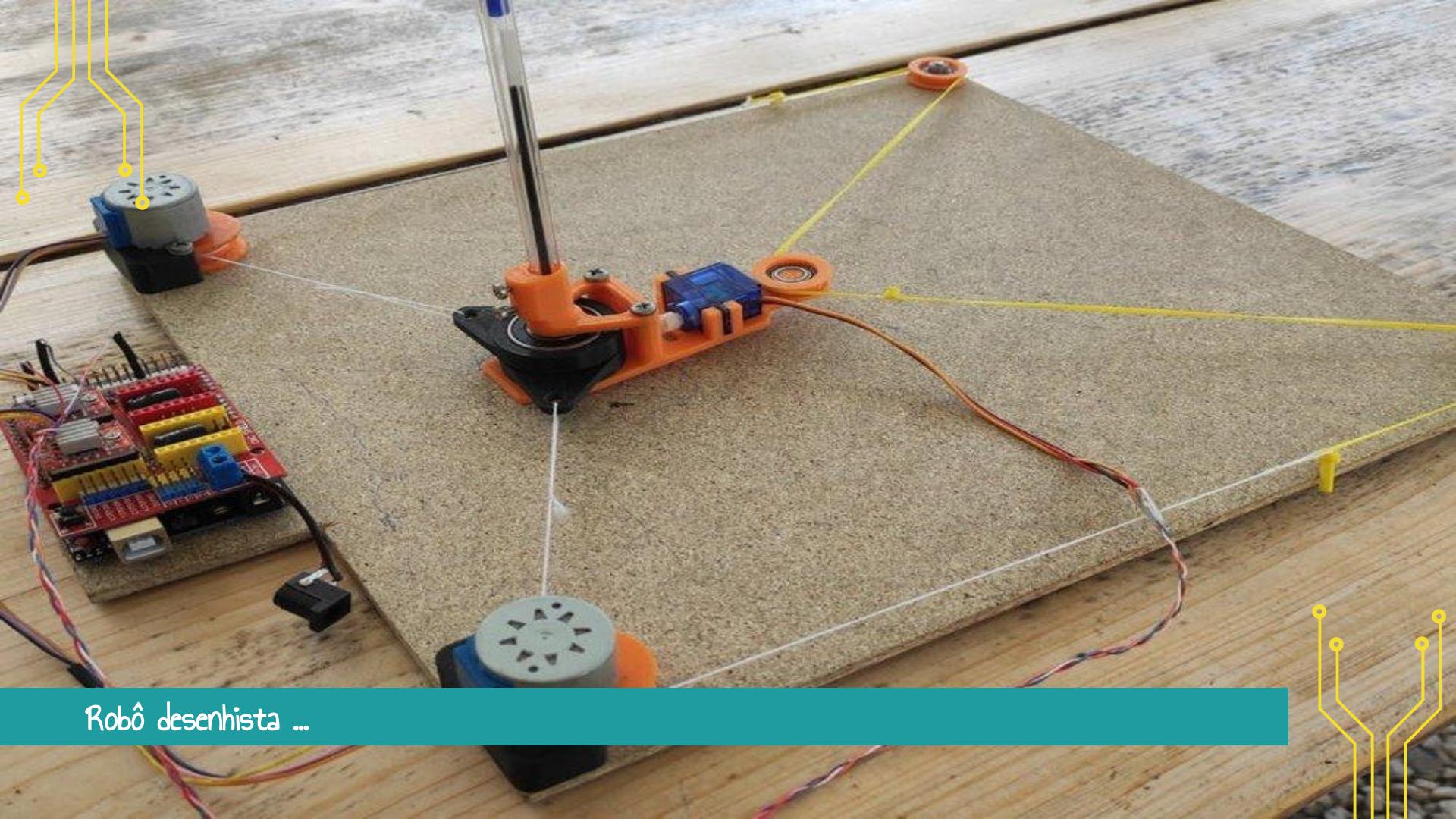
Alguns projetos feitos com Arduino!



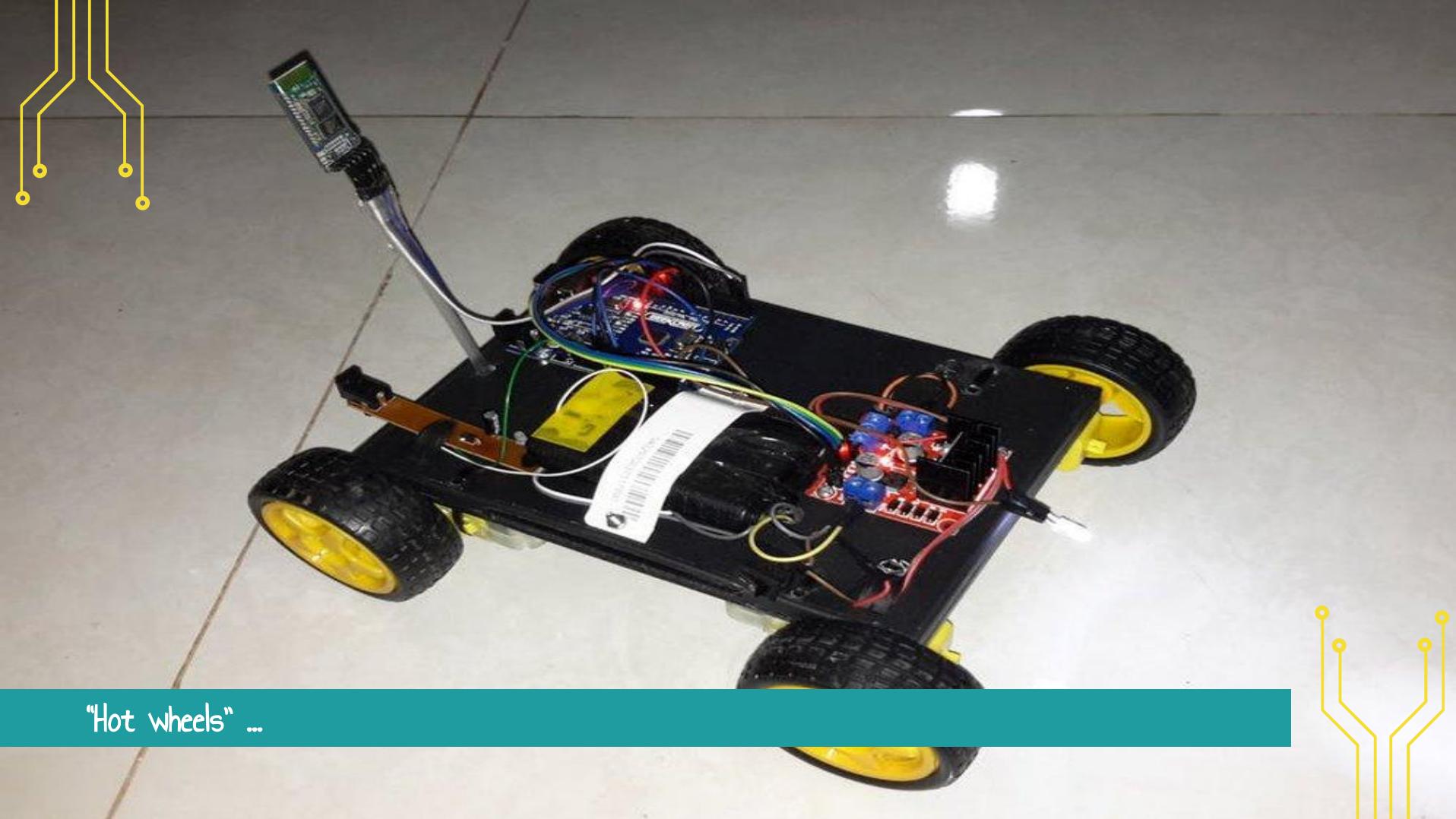
Petoil Nybble - laser follower Robotic Cat



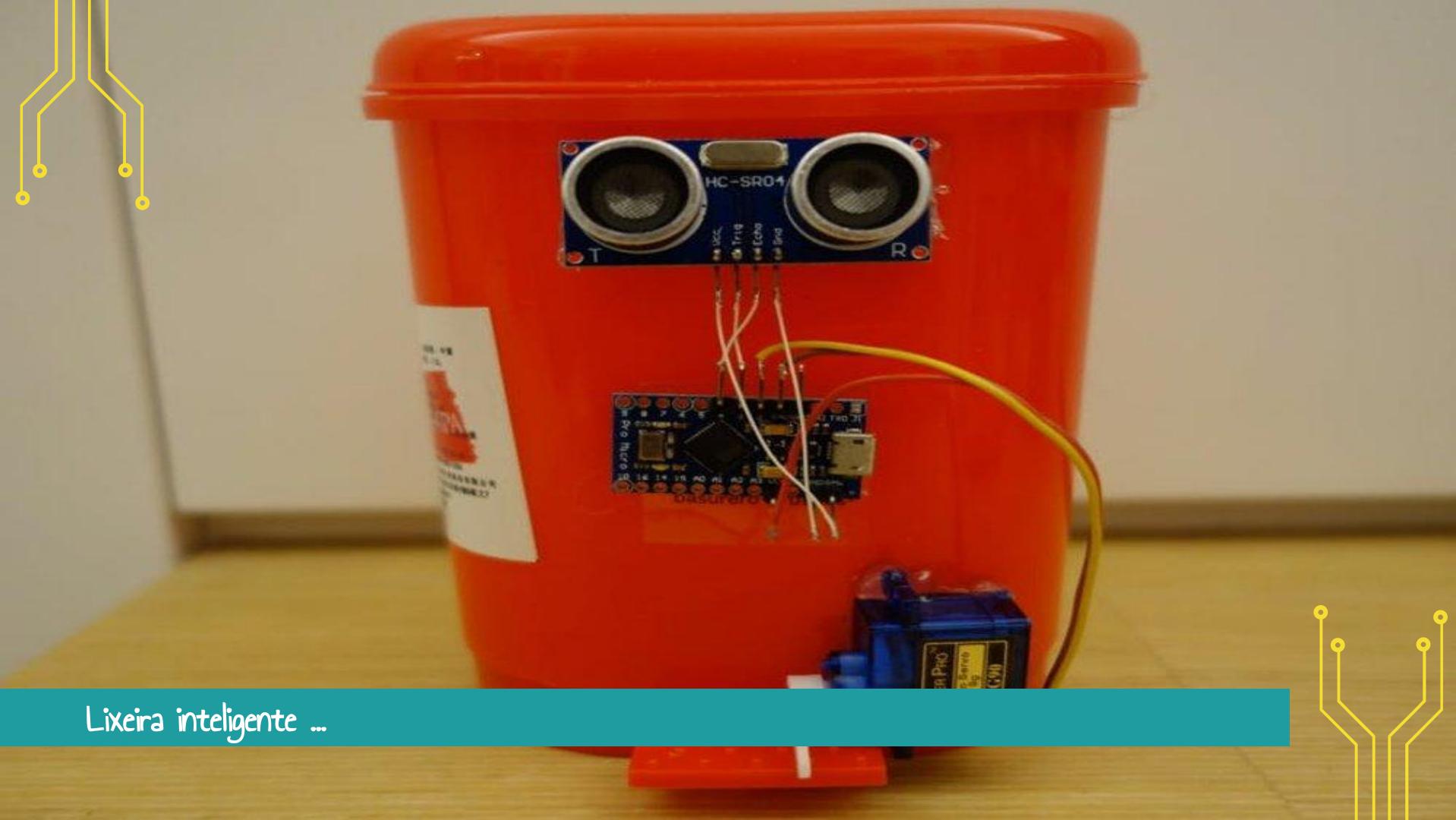
Braço robótico ...



Robô desenhista ...



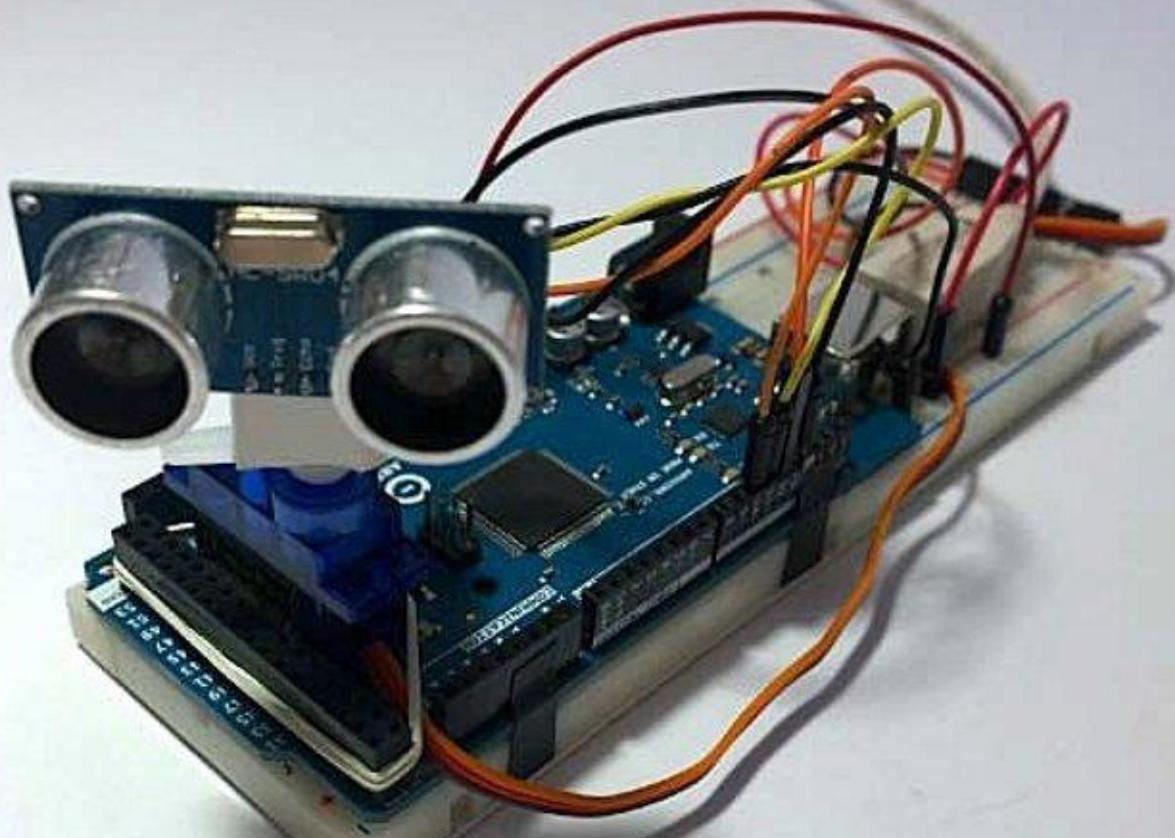
"Hot wheels" ...



Lixeira inteligente ...



Aplicações em domótica ...



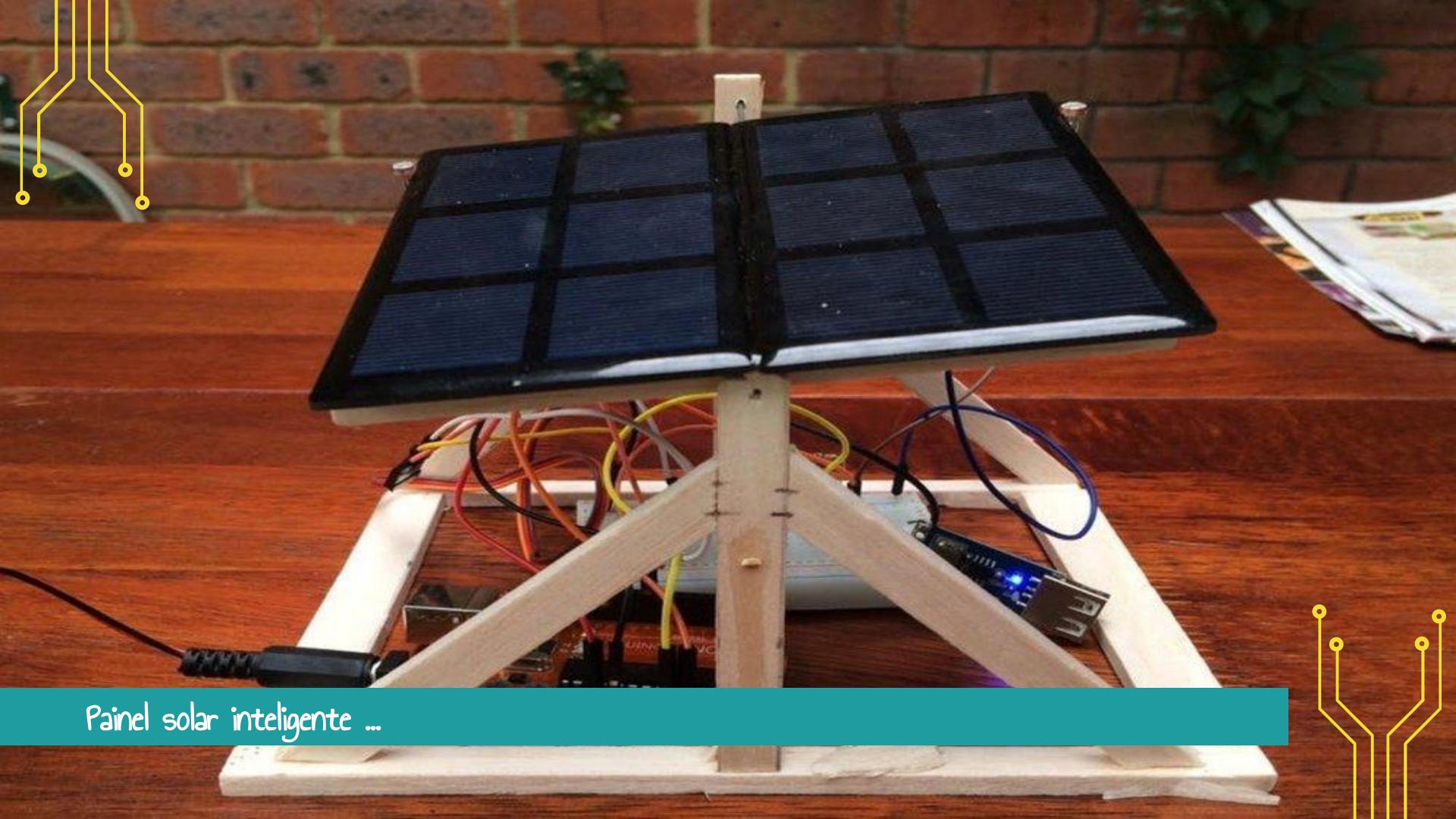
Monitoramento e detecção de presença em 360º ...



Aplicações em segurança ...



Timer Pomodoro ...



Painel solar inteligente ...



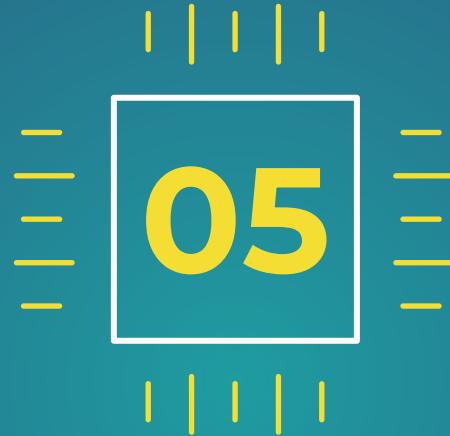
Robô aranha ...



"Xadrez de bruxo..."



Aplicações de acessibilidade ...



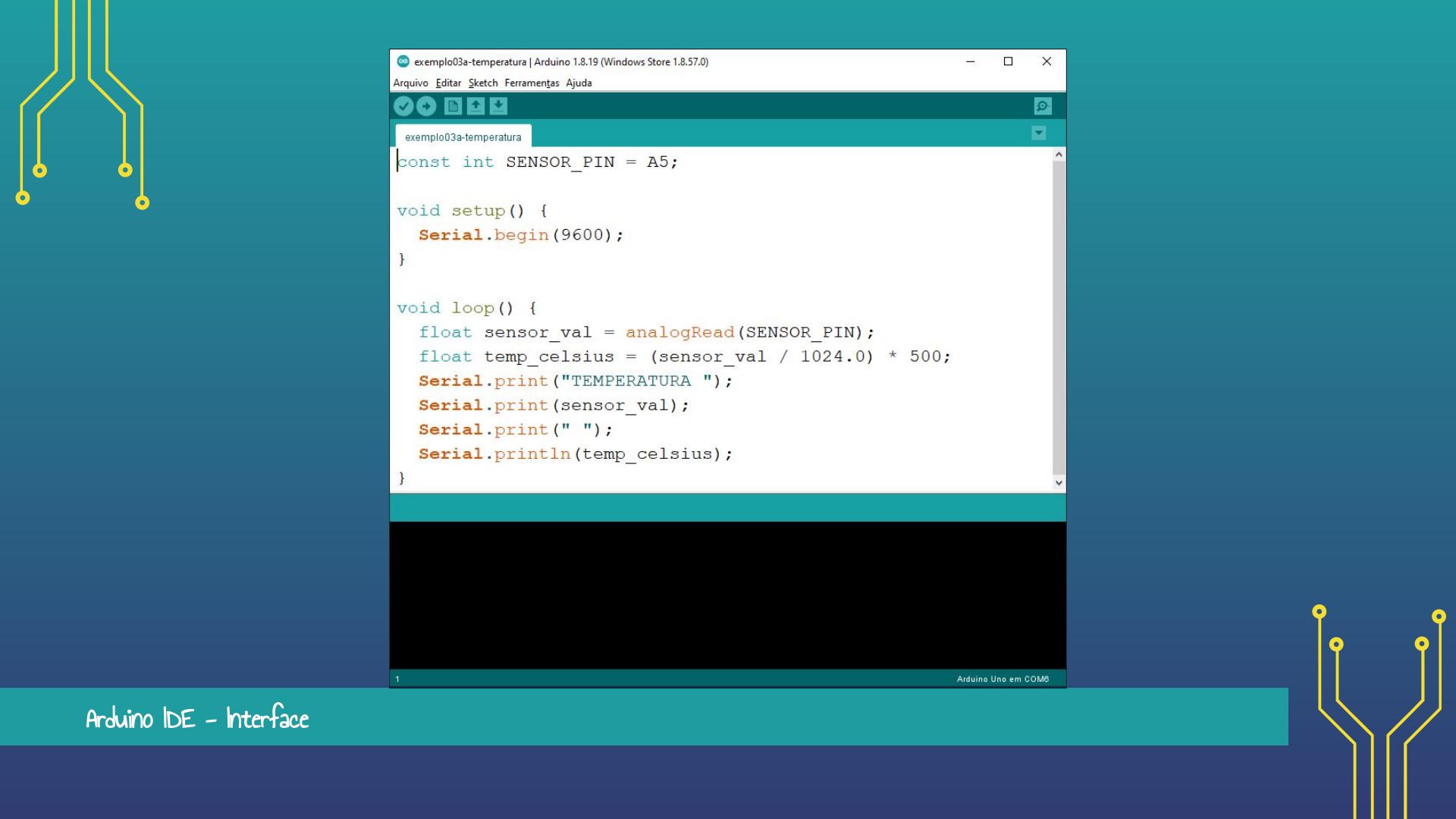
Programando para Arduino

Programando para Arduino

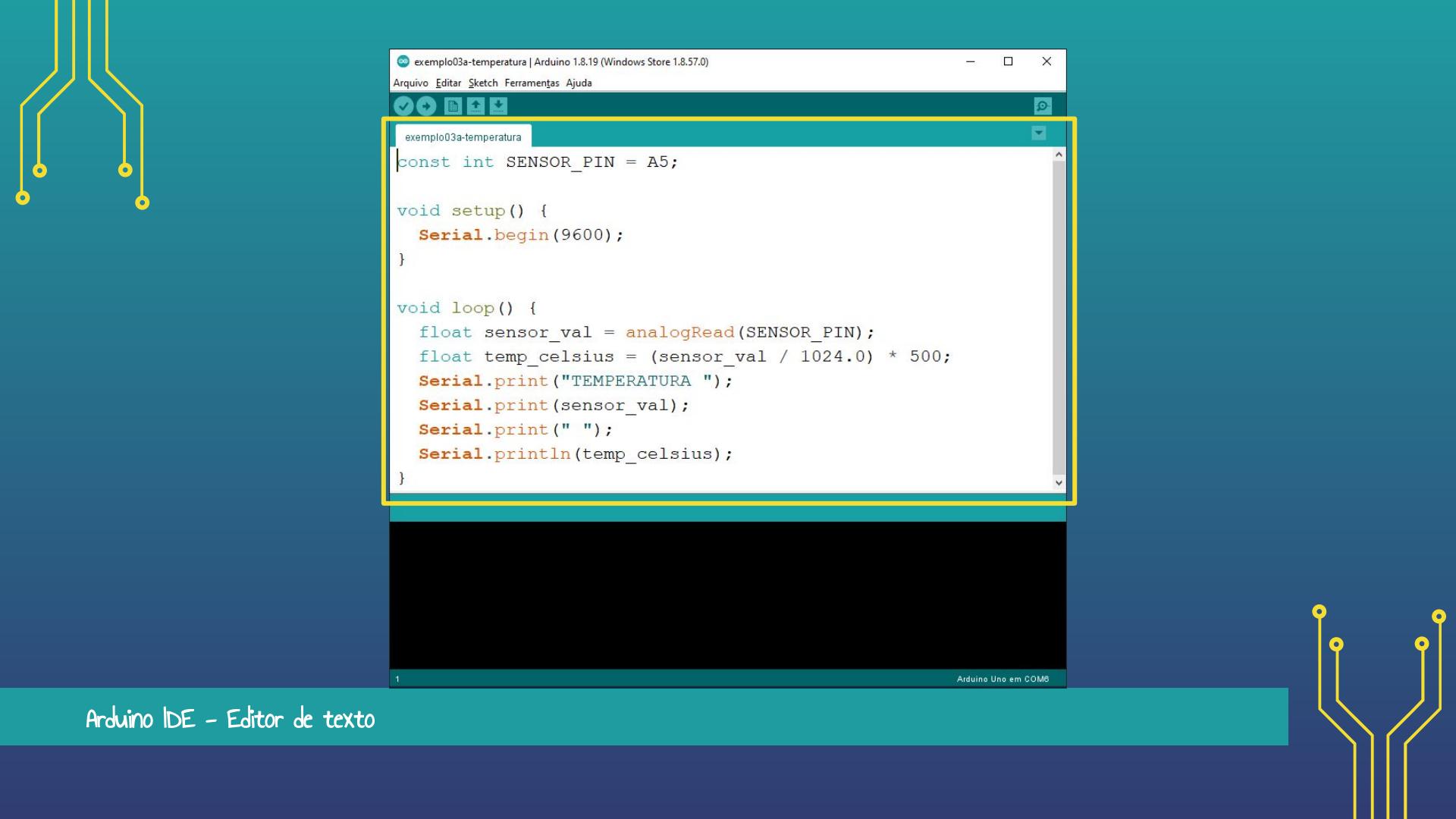
Arduino IDE (Integrated Dev Environment)

- O ambiente oficial utilizado para codificar programas para a plataforma Arduino é o **Arduino IDE**;
- É responsável por se conectar a uma placa Arduino e **carregar (upload)** programas para seu microcontrolador;
- Cada programa é chamado de **Sketch** (.ino), e uma pasta com vários programas é chamada de **Sketch Book**;
- A IDE é composta principalmente por:
 - Editor de texto;
 - Área de mensagens;
 - Console de texto;
 - Barra ferramentas (botões);
 - Barra de menus

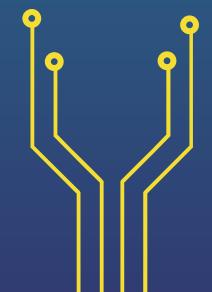
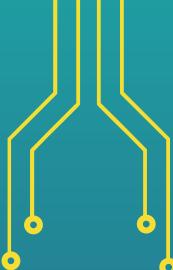
<https://docs.arduino.cc/software/ide-v1/tutorials/Environment>



Arduino IDE - Interface



Arduino IDE - Editor de texto



exemplo03a-temperatura | Arduino 1.8.19 (Windows Store 1.8.57.0)

Arquivo Editar Sketch Ferramentas Ajuda

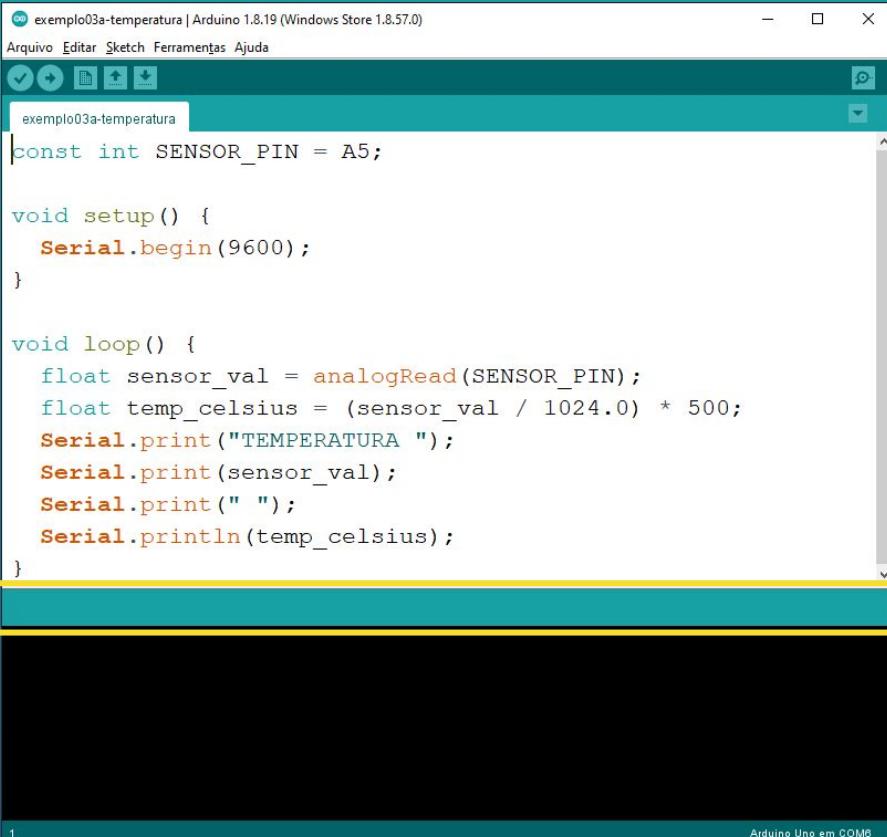
```
const int SENSOR_PIN = A5;

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

void loop() {
    float sensor_val = analogRead(SENSOR_PIN);
    float temp_celsius = (sensor_val / 1024.0) * 500;
    Serial.print("TEMPERATURA ");
    Serial.print(sensor_val);
    Serial.print(" ");
    Serial.println(temp_celsius);
}
```

1 Arduino Uno em COM6

Arduino IDE - Área de mensagens (erro)

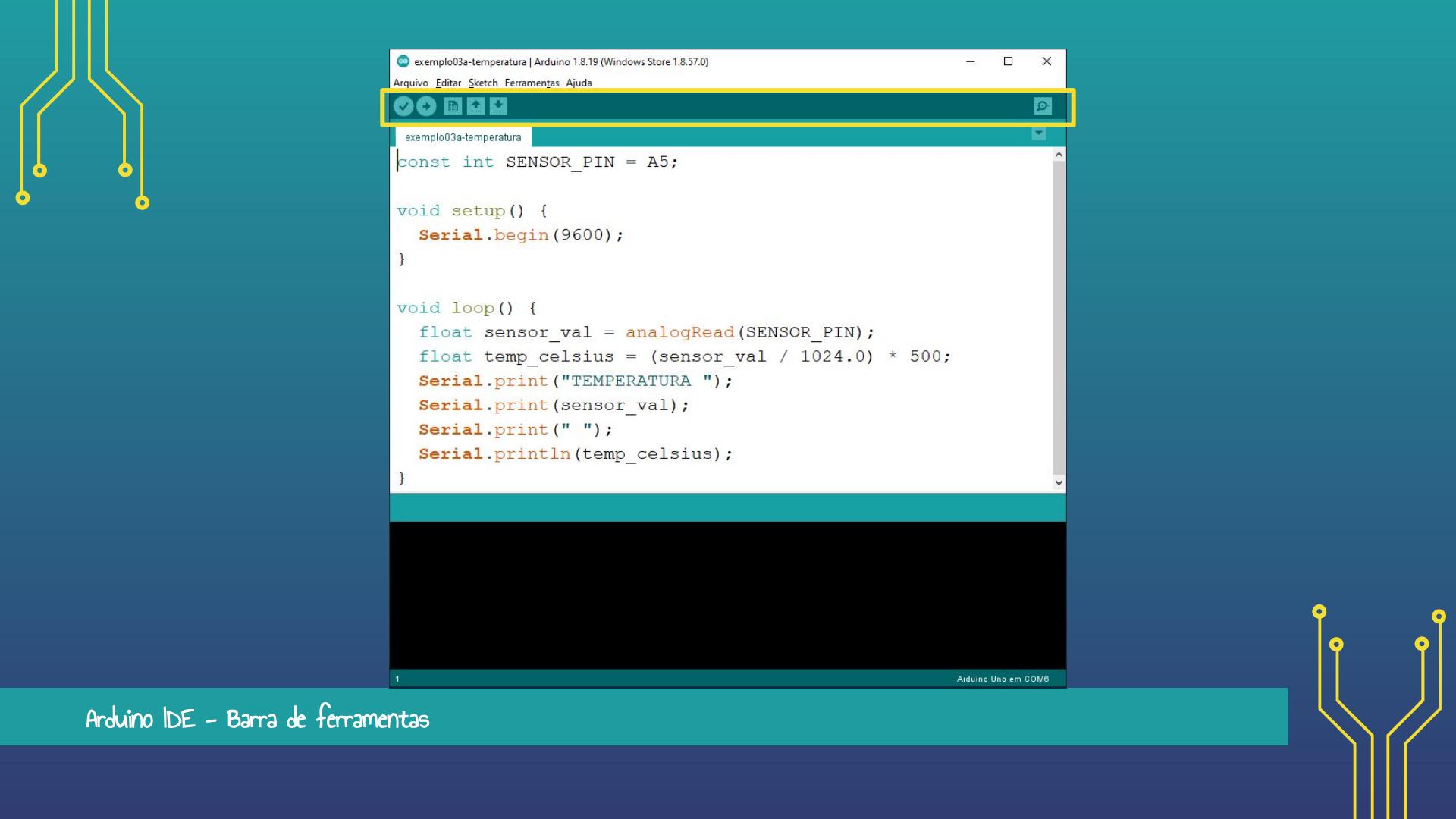


```
const int SENSOR_PIN = A5;

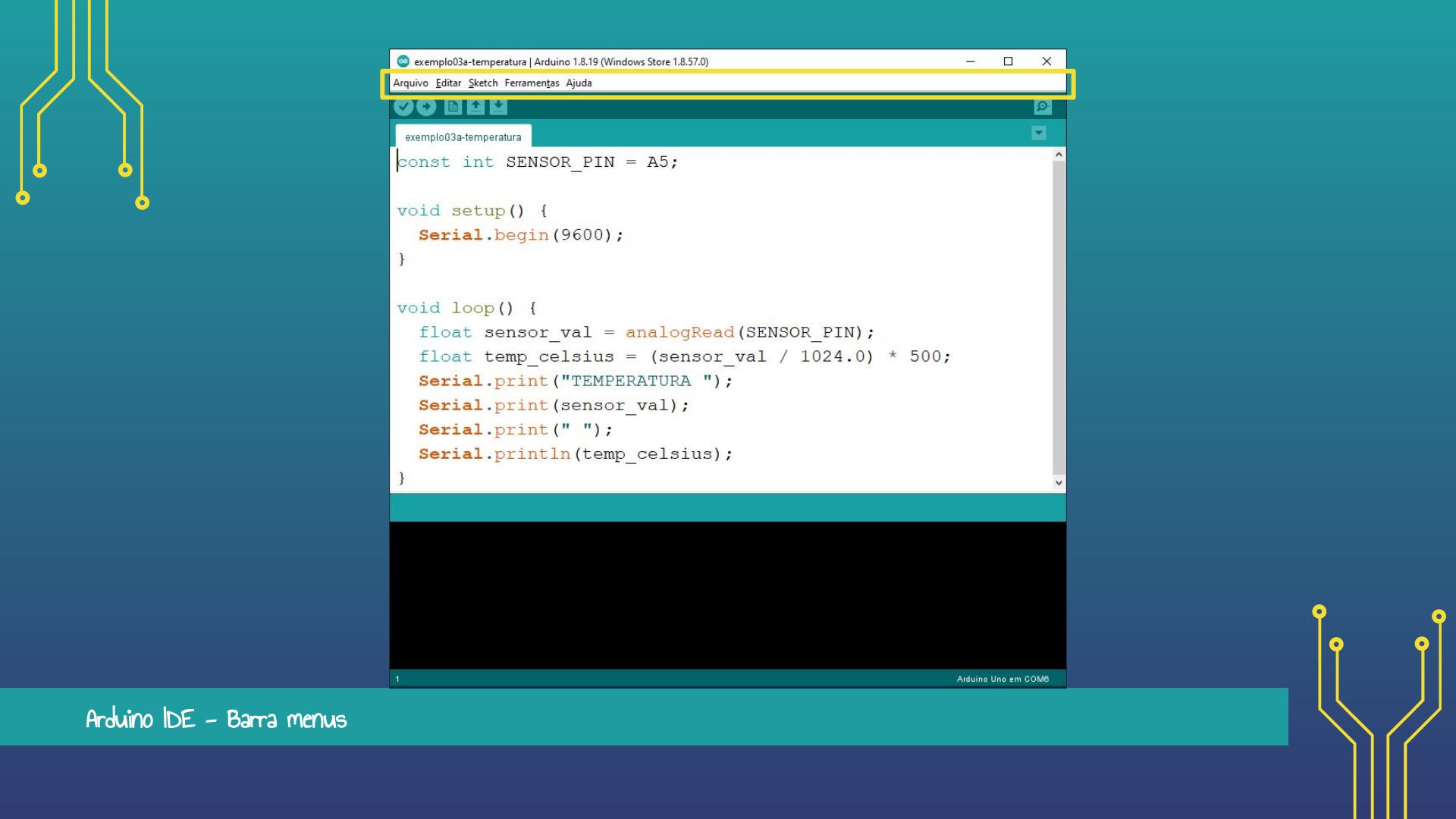
void setup() {
    Serial.begin(9600);
}

void loop() {
    float sensor_val = analogRead(SENSOR_PIN);
    float temp_celsius = (sensor_val / 1024.0) * 500;
    Serial.print("TEMPERATURA ");
    Serial.print(sensor_val);
    Serial.print(" ");
    Serial.println(temp_celsius);
}
```

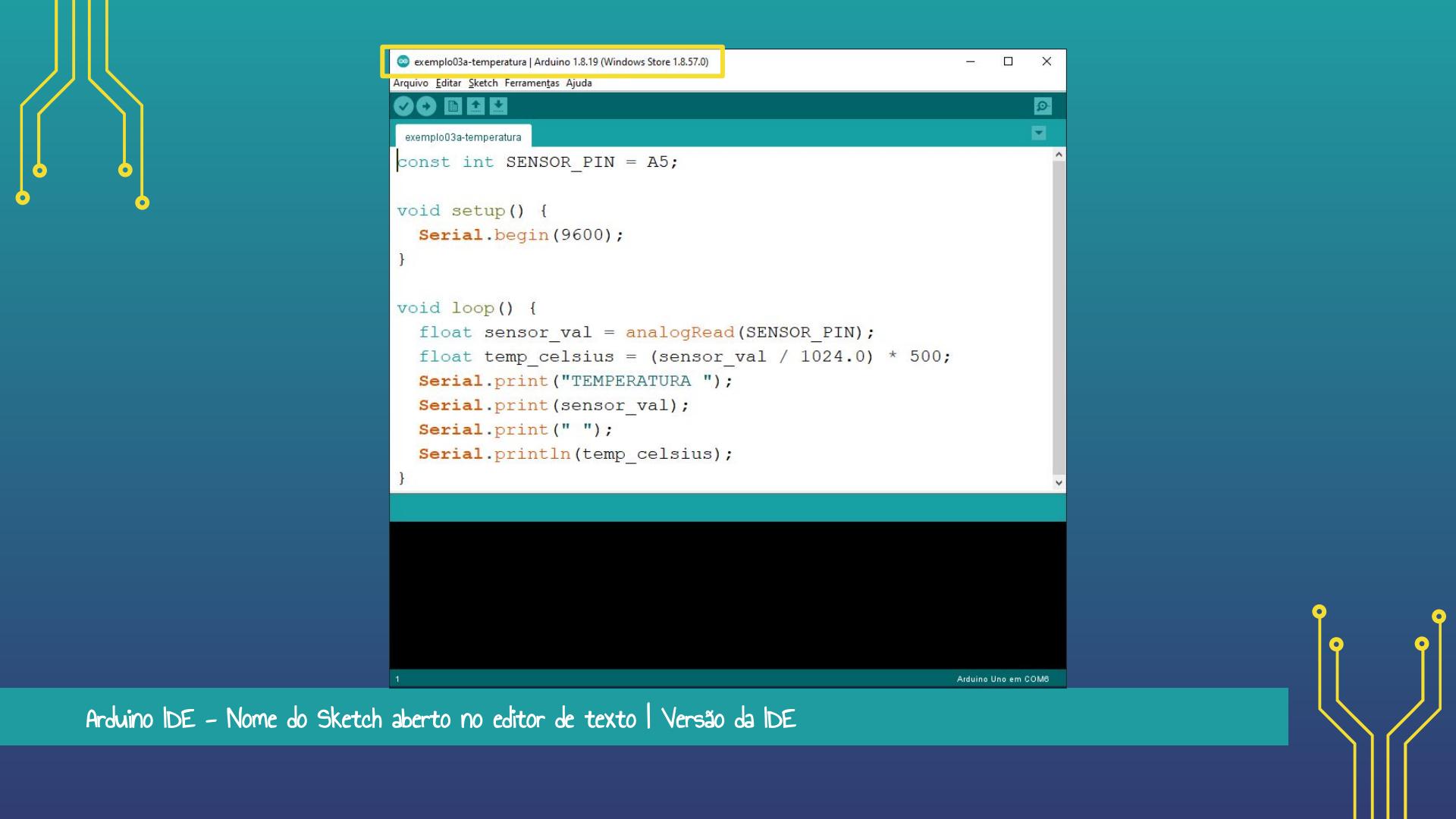
Arduino IDE - Barra de status



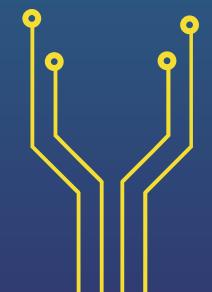
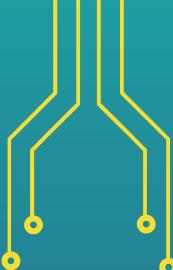
Arduino IDE - Barra de ferramentas



Arduino IDE - Barra menus



Arduino IDE - Nome do Sketch aberto no editor de texto | Versão da IDE



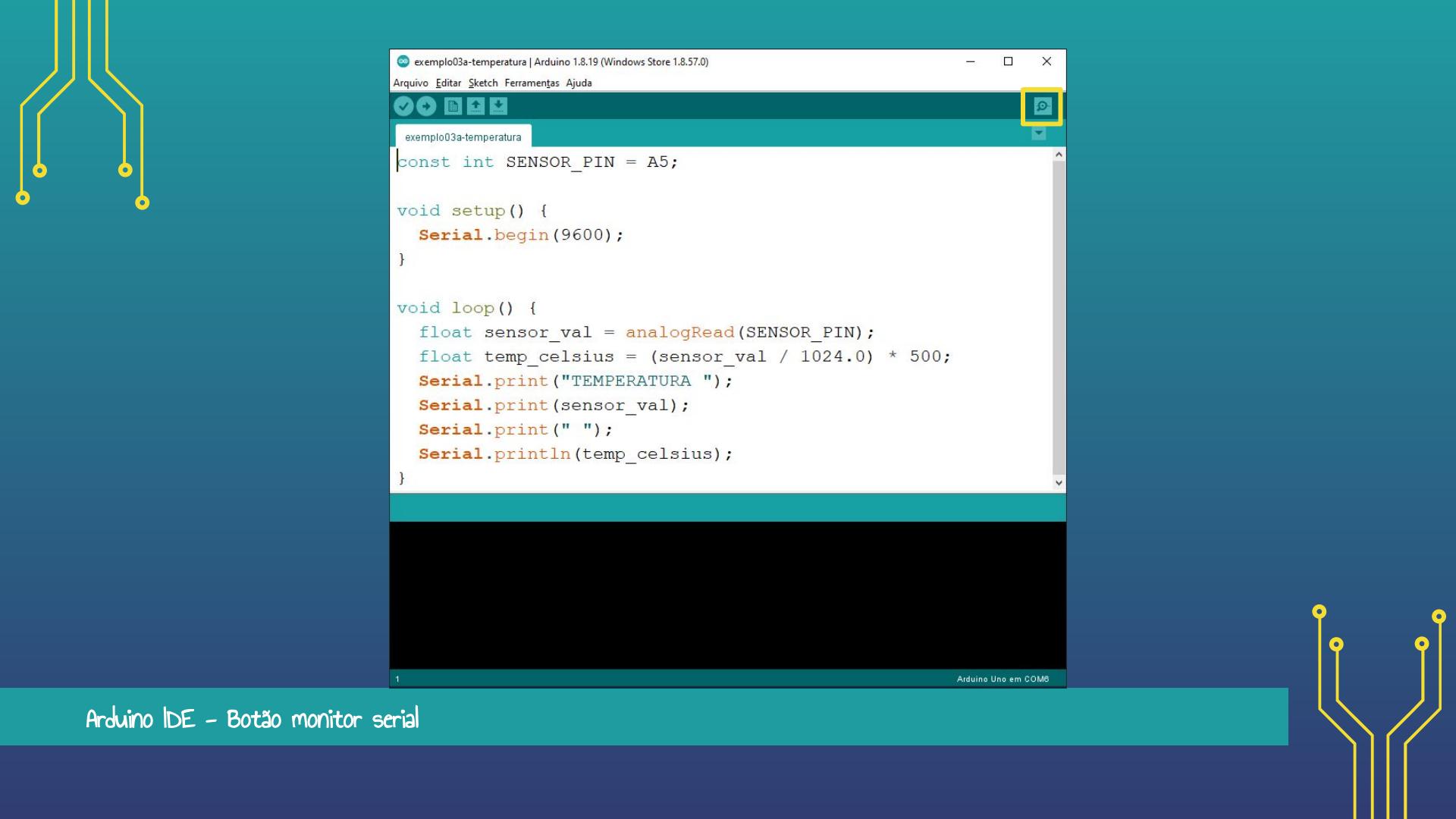
```
exemplo03a-temperatura | Arduino 1.8.19 (Windows Store 1.8.57.0)
Arquivo Editar Sketch Ferramentas Ajuda
exemplo03a-temperatura
const int SENSOR_PIN = A5;

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

void loop() {
    float sensor_val = analogRead(SENSOR_PIN);
    float temp_celsius = (sensor_val / 1024.0) * 500;
    Serial.print("TEMPERATURA ");
    Serial.print(sensor_val);
    Serial.print(" ");
    Serial.println(temp_celsius);
}
```

Arduino IDE - Placa conectada e a porta utilizada

Arduino Uno em COM6



```
const int SENSOR_PIN = A5;

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

void loop() {
    float sensor_val = analogRead(SENSOR_PIN);
    float temp_celsius = (sensor_val / 1024.0) * 500;
    Serial.print("TEMPERATURA ");
    Serial.print(sensor_val);
    Serial.print(" ");
    Serial.println(temp_celsius);
}
```

Arduino IDE - Botão monitor serial

Principais atalhos e comandos

Ctrl + R

Verifica/Compilar

Verifica erros na sintaxe do Sketch atual.

Ctrl + K

Abrir pasta do Sketch

Abre o local do Sketch atual no explorador de arquivos.

Ctrl + U

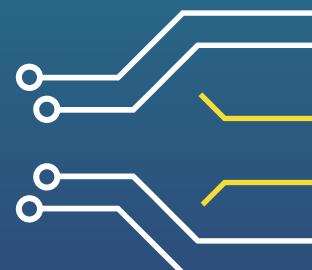
Carregar

Faz o upload do Sketch atual para o Arduino.

Ctrl + S

Salvar

Salva o Sketch atual no SketchBook.



Principais atalhos e comandos



Ctrl + N

Novo

Abre um novo Sketch (em branco) numa nova janela.

Ctrl + O

Abrir

Abre a janela com a lista de SketchBooks.

Ctrl + Vírgula

Preferências

Abre a janela de preferências da IDE.

Ctrl + Q

Sair

Fechá a janela atual da IDE.

Principais atalhos e comandos

Ctrl + Barra

Comentar

Converte a linha atual de código no editor, numa linha comentário.

TAB

Aumentar indentação

Aumenta em um nível a indentação (recuo) da linha de código atual.

Shift + TAB

Diminuir indentação

Diminui em um nível a indentação (recuo) da linha de código atual.

Ctrl + F

Localizar

Abre a janela Localizar.



Principais atalhos e comandos

Ctrl + Shift + I

Gerenciar Bibliotecas

Abre a janela Gerenciador de Bibliotecas.

Ctrl + T

Autoformatação

Formata automaticamente todo o código no editor de texto.

Ctrl + Shift + M

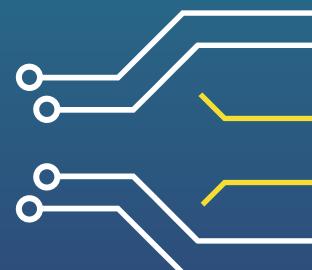
Monitor Serial

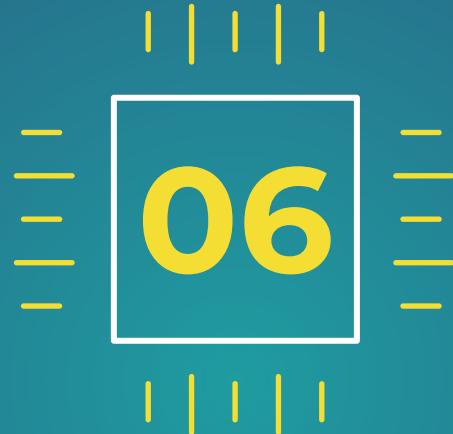
Abre a janela Monitor Serial.

Ctrl + Shift + L

Plotter Serial

Abre a janela Plotter Serial.



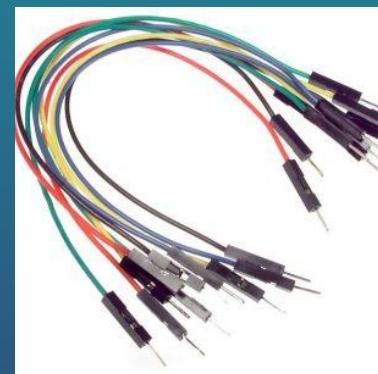


Projetos Eletrônicos Básicos

COMPONENTES ELETRÔNICOS

FIOS JUMPERS

- São essenciais na construção de circuitos numa placa de ensaio;
- Tem como função **conduzir energia elétrica** entre um barramento e outro, auxiliando na montagem do circuito;
- São facilmente comprados em casas de eletrônica e na internet;
- Os cabos de rede podem ser cortados, desenrolados e utilizados como jumpers;
- Suas extremidades podem ser do tipo “**macho**” ou “**fêmea**”;



COMPONENTES ELETRÔNICOS

RESISTOR

- Resistores são componentes usados para controlar a passagem de corrente elétrica em circuitos elétricos por meio do Efeito Joule.
- Alguns resistores conseguem manter a resistência elétrica constante (resistores ôhmicos). A grande maioria dos resistores reais não são ôhmicos;
- O valor da resistência é identificado por um código de cores;



Cor	1º. Algarismo Significativo	2º. Algarismo Significativo	3º. Algarismo Significativo	Múltiplo	Tolerância
Preto		0	0	x 1	
Marrom	1	1	1	x 10	± 1%
Vermelho	2	2	2	x 10²	± 2%
Laranja	3	3	3	x 10³	
Amarelo	4	4	4	x 10⁴	
Verde	5	5	5	x 10⁵	
Azul	6	6	6	x 10⁶	
Violeta	7	7	7		
Cinza	8	8	8		
Branco	9	9	9		
Ouro				x 10⁻¹	± 5%
Prata				x 10⁻²	± 10%
Ausência					± 20%

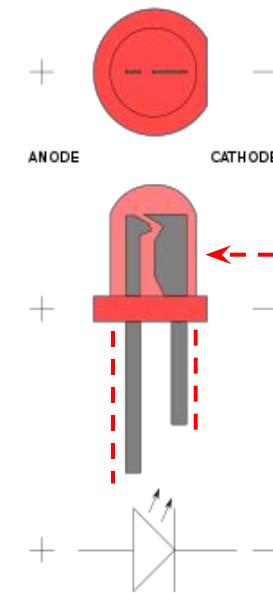
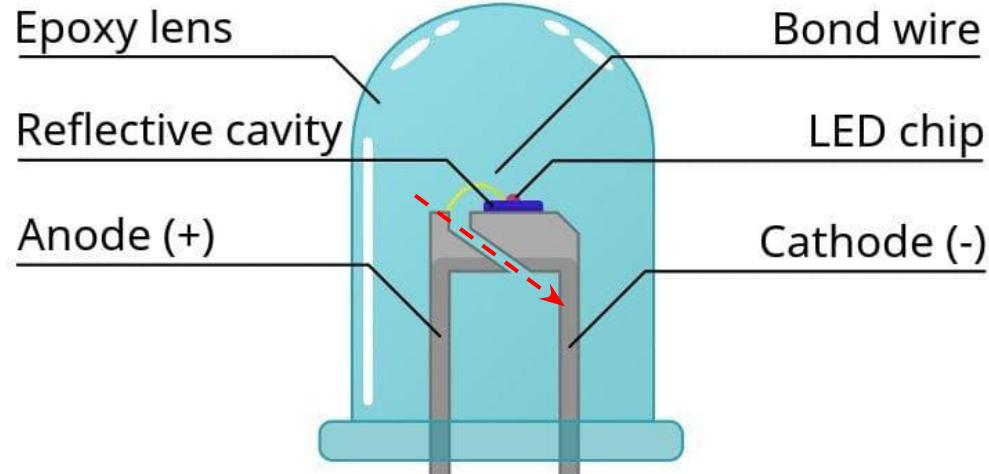
Esquema de cores para cálculo da resistência.

COMPONENTES ELETRÔNICOS

LEDs (Light-Emitting Diode)

- Utilizado para emissão de luz em locais e instrumentos onde não é adequado o uso de uma lâmpada;
- Possui dois tipos de terminais: anodo (+) e cátodo (-);
- O LED é um diodo semicondutor, de junção P-N, que quando é energizado emite luz visível, resultado dos elétrons que passam por um fio muito fino que conecta os terminais;
- Em geral operam com tensão de 1.6V a 3.3V;
- Alguns LEDs redondos apresentam um pequeno “achatamento” na lateral para identificar o terminal cátodo;
- Além disso, os terminais são separados por uma ranhura, no sentido descendente, do anodo para o cátodo;





Identificando os terminais de um LED.

Hands On

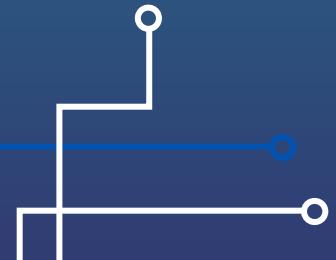
Exemplo 01 - Blink

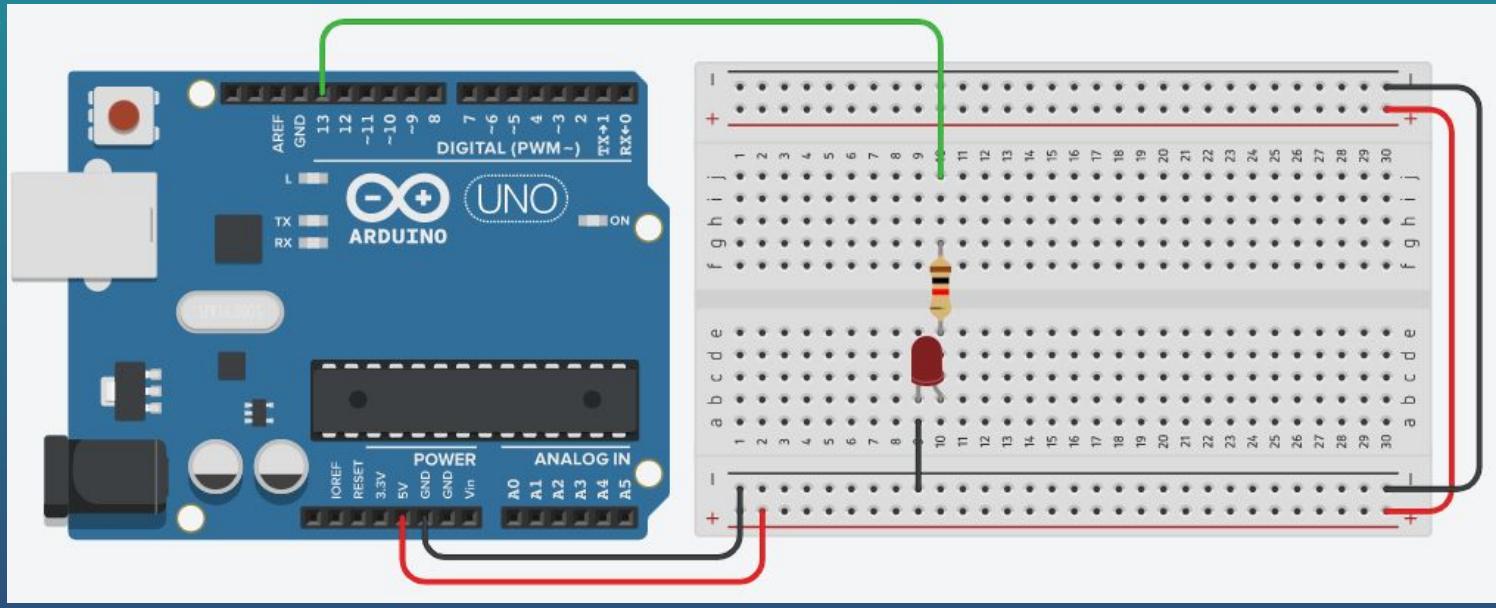


EXEMPLO 01 - BLINK

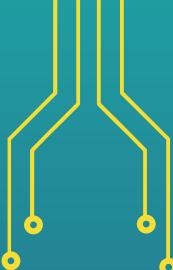
Materiais

- Este exemplo ensina como acender e apagar um LED pela placa Arduino;
- Para este exemplo serão necessários:
 - 01 Placa Arduino UNO;
 - 01 LED;
 - 01 Resistor 100 ohm;
 - 01 Placa de ensaio;

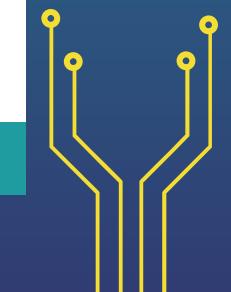




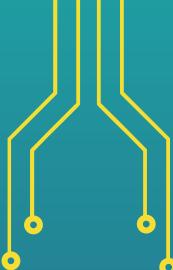
Exemplo 01 - Montagem do circuito



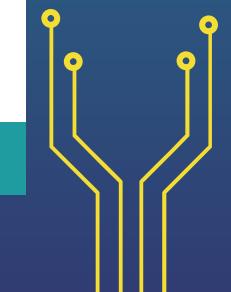
```
1. void setup() {
2.   pinMode(13, OUTPUT);    // configura o pino 13 como pino de saída
3. }
4.
5. void loop() {
6.   digitalWrite(13, HIGH); // liga o LED enviando sinal HIGH
7.   delay(1000);           // aguarda por 1000 ms ou 1 segundo
8.   digitalWrite(13, LOW); // desligar o LED enviando sinal LOW
9.   delay(1000);           // aguarda novamente 1 segundo
10.}
11.
12.
13.
14.
15.
16.
17.
18.
```



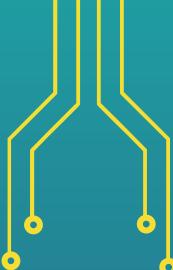
Exemplo 01 - Código v1



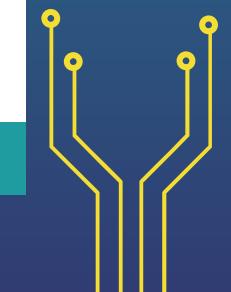
```
1. void setup() {
2.   pinMode(13, OUTPUT);      // configura o pino 13 como pino de saída
3. }
4.
5. void loop() {
6.   digitalWrite(13, HIGH);   // liga o LED enviando sinal HIGH
7.   delay(250);             // aguarda por 1000 ms ou 1 segundo
8.   digitalWrite(13, LOW);   // desligar o LED enviando sinal LOW
9.   delay(5000);            // aguarda novamente 1 segundo
10.}
11.
12.
13.
14.
15.
16.
17.
18.
```



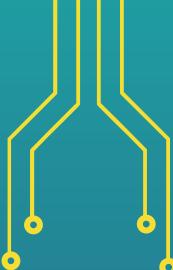
Exemplo 01 - Código v2 (variando os intervalos)



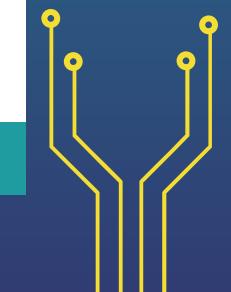
```
1. void setup() {
2.   pinMode(13, OUTPUT);      // configura o pino 13 como pino de saída
3. }
4.
5. void loop() {
6.   digitalWrite(13, HIGH);   // liga o LED enviando sinal HIGH
7.   delay(2000);             // aguarda por 1000 ms ou 1 segundo
8.   digitalWrite(13, LOW);    // desligar o LED enviando sinal LOW
9.   delay(500);              // aguarda novamente 1 segundo
10. }
11.
12.
13.
14.
15.
16.
17.
18.
```



Exemplo 01 - Código v3 (variando os intervalos)



```
1. void setup() {
2.   pinMode(13, OUTPUT);      // configura o pino 13 como pino de saída
3. }
4.
5. void loop() {
6.   digitalWrite(13, HIGH);   // liga o LED enviando sinal HIGH
7.   delay(50);               // aguarda por 1000 ms ou 1 segundo
8.   digitalWrite(13, LOW);    // desligar o LED enviando sinal LOW
9.   delay(50);               // aguarda novamente 1 segundo
10.}
11.
12.
13.
14.
15.
16.
17.
18.
```



Exemplo 01 - Código v4t (variando os intervalos)

Hands On

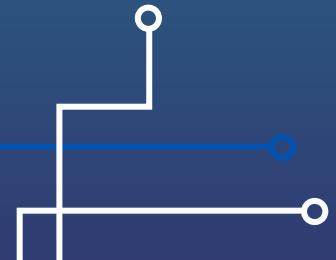
Exemplo 02 - PWM



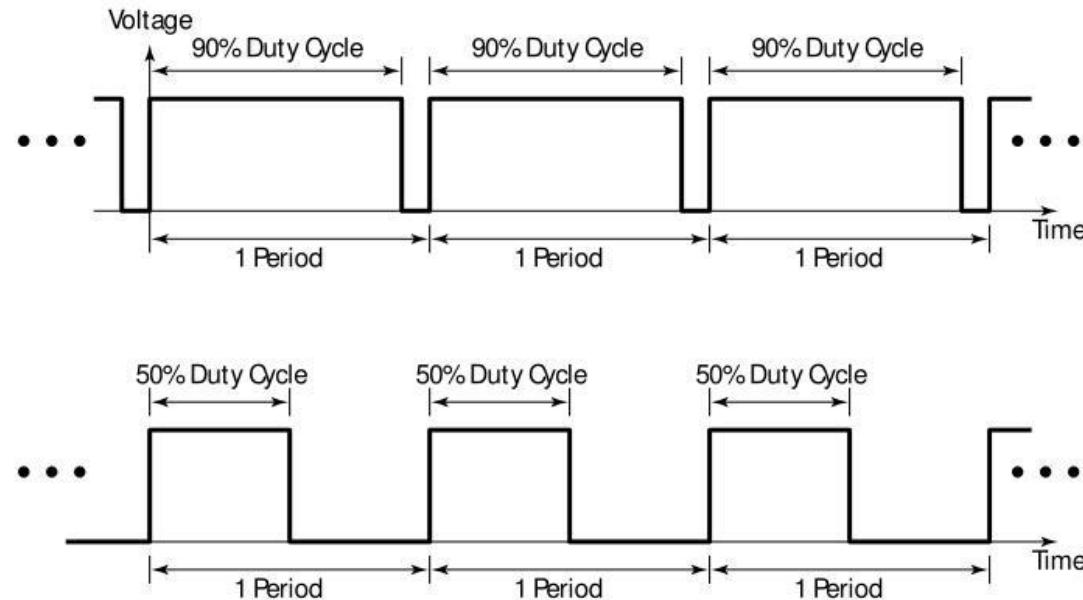
EXEMPLO 02 - PWM

Materiais

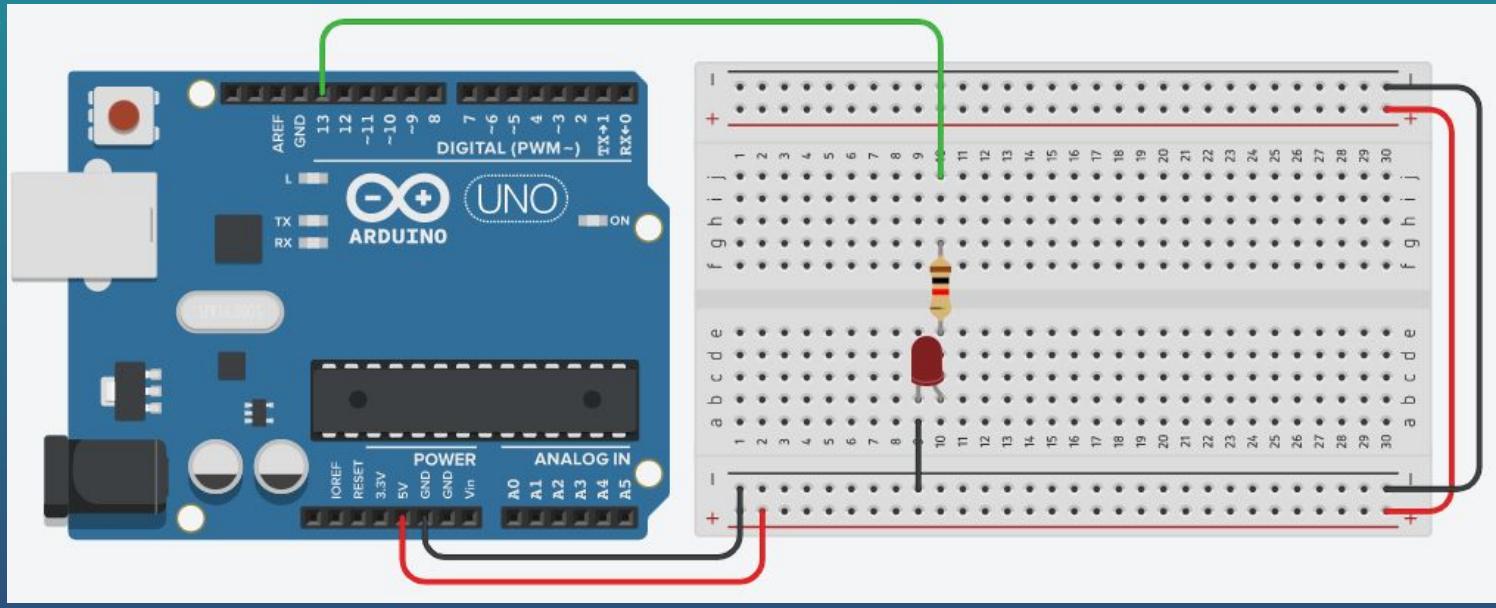
- Este exemplo ensina como utilizar sinais PWM nos pinos digitais da placa Arduino;
- Para este exemplo serão necessários:
 - 01 Placa Arduino UNO;
 - 01 LED;
 - 01 Resistor 220 ohm;
 - 01 Placa de ensaio;
 - Fios jumpers;



Pulse Width Modulation (PWM)



Funcionamento de sinais PWM



Exemplo 02 - Montagem do circuito

```
1. int brilho;
2. const int LED_PIN = 13;
3.
4. void setup() {
5.     pinMode(LED_PIN, OUTPUT); // configura o pino 13 como pino de saída
6. }
7.
8. void loop() {
9.     for (brilho = 0; brilho < 256; brilho++) { // incrementa o valor de brilho
10.         analogWrite(LED_PIN, brilho); // sinal PWM de acordo com o valor de brilho
11.         delay(30);
12.     }
13.     for (brilho = 255; 0 <= brilho; brilho--) { // decrementa o valor de brilho
14.         analogWrite(LED_PIN, brilho); // sinal PWM de acordo com o valor de brilho
15.         delay(30);
16.     }
17. }
```

Exemplo 02 - Código

Hands On

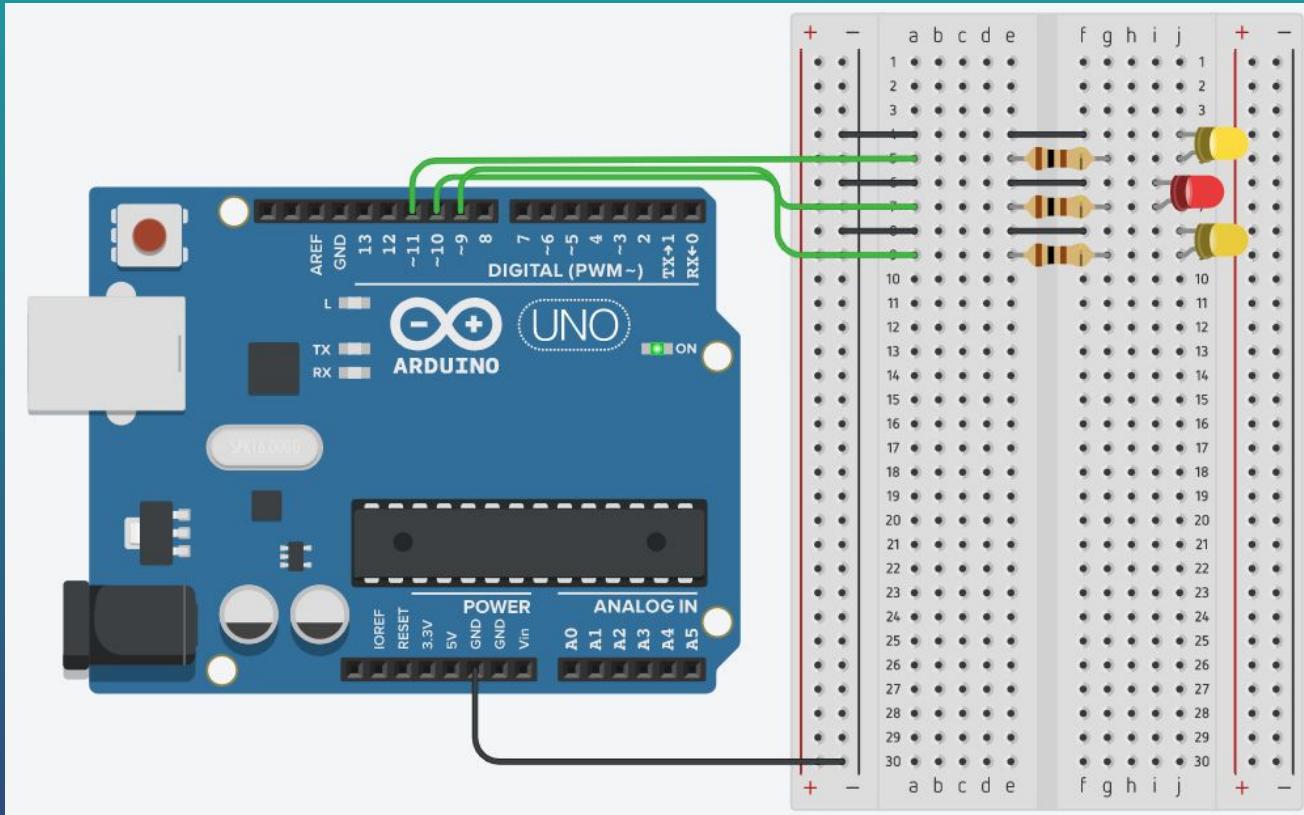
Exemplo 03 - Simulando Fogo



EXEMPLO 03 - Simulando Fogo

Materiais

- Este exemplo ensina como gerar números aleatórios utilizando a função random e como criar vetores (arrays);
- Para este exemplo serão necessários:
 - 01 Placa Arduino UNO;
 - 02 LED amarelos;
 - 01 LED vermelho;
 - 03 Resistor 100 ohms;
 - 01 Placa de ensaio;
 - Fios jumpers;



Exemplo 03 – Montagem do circuito

```
1. #define LED_PIN1 9;
2. #define LED_PIN2 10;
3. #define LED_PIN3 11;
4.
5. void setup() {
6.   pinMode(LED_PIN1, OUTPUT);
7.   pinMode(LED_PIN2, OUTPUT);
8.   pinMode(LED_PIN3, OUTPUT);
9. }
10.
11. void loop() {
12.   analogWrite(LED_PIN1, random(185) + 70);
13.   analogWrite(LED_PIN2, random(185) + 70);
14.   analogWrite(LED_PIN3, random(185) + 70);
15.   delay(random(100));
16. }
```

Exemplo 03 - Código v1

```
1. const byte pins[] = {9, 10, 11};  
2.  
3. void setup() {  
4.     for (unsigned int x = 0; x < 3; x++)  
5.         pinMode(pins[x] , OUTPUT);  
6. }  
7.  
8. void loop() {  
9.     for (unsigned int x = 0; x < 3; x++)  
10.         analogWrite(pins[x] , random(185) + 70);  
11.     delay(random(100));  
12. }  
13.  
14.  
15.  
16.  
17.  
18.
```

Exemplo 03 - Código v2

Hands On

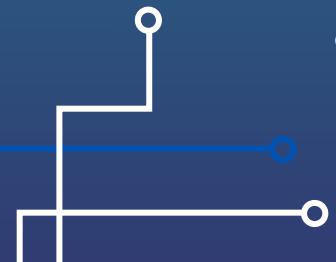
Exemplo 04 - Semáforo



EXEMPLO 04 - Semáforo

Materiais

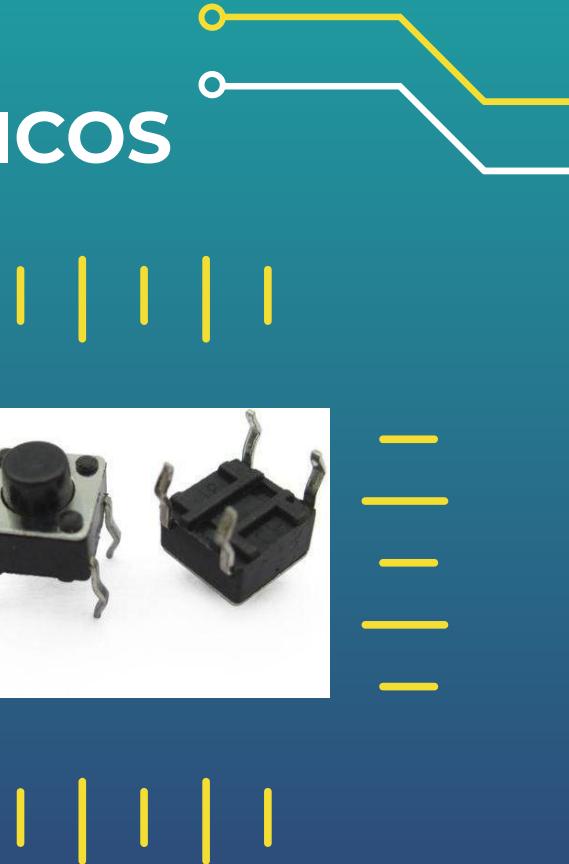
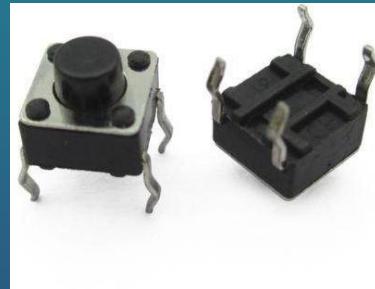
- Este exemplo ensina como utilizar uma chave táctil para interagir com a placa Arduino:
 - 01 Placa Arduino UNO;
 - 01 Chave táctil;
 - 02 LEDs vermelhos;
 - 02 LEDs verdes;
 - 01 LED amarelo;
 - 05 Resistor 150 ohms;
 - 01 Resistor 10 KOhms;
 - 01 Placa de ensaio;
 - Fios jumpers;

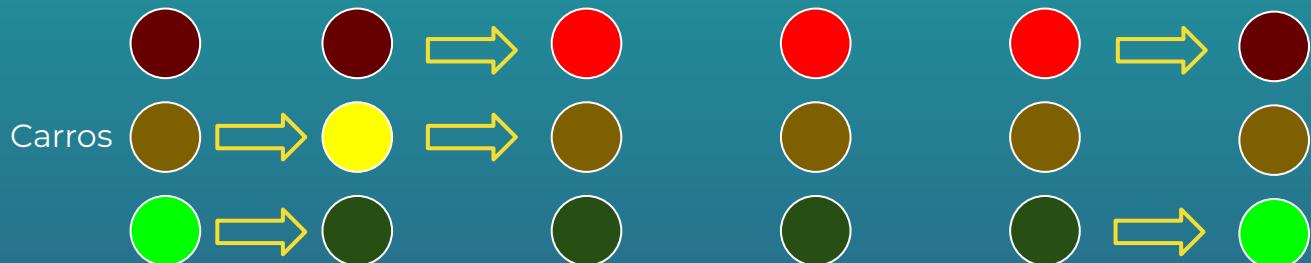


COMPONENTES ELETRÔNICOS

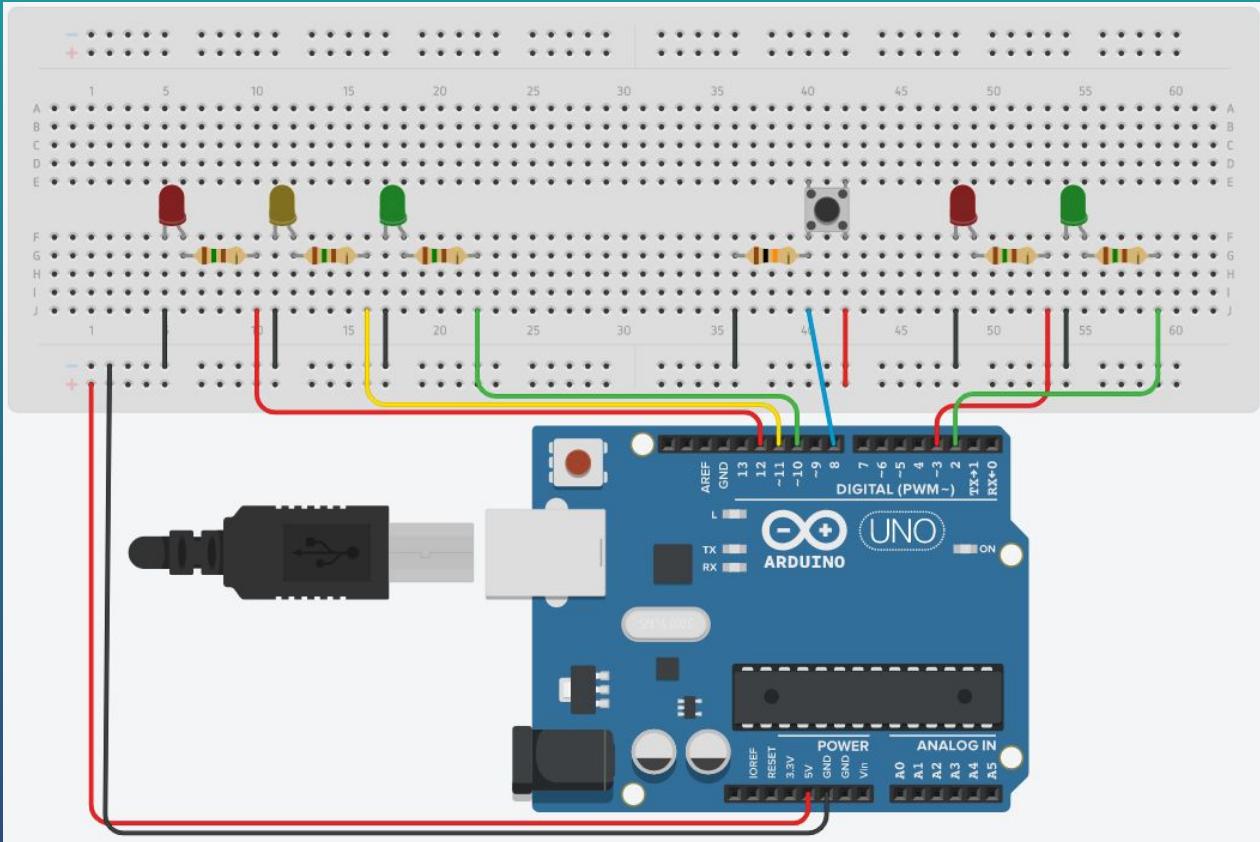
CHAVE TÁCTIL (PUSH BUTTON)

- Chaves e botões estão presentes em diversas aplicações de nosso cotidiano, por exemplo o botão ON/OFF de um aparelho eletrônico;
- São **contatos mecânicos** que têm como objetivo **controlar a passagem de corrente** em um sistema a partir de uma força externa;
- Existem diversos modelos disponíveis no mercado:
 - Chaves momentâneas;
 - Chaves memorizadas;
 - Chaves magnéticas;





Funcionamento do projeto semáforo



Exemplo 04 - Montagem do circuito

```
1. #define CAR_VERM 12
2. #define CAR_AMAR 11
3. #define CAR_VERD 10
4. #define PED_VERM 3
5. #define PED_VERD 2
6. #define BUTTON_PIN 8
7. #define CROSS_TIME 5000 // tempo de cruzamento em ms
8.
9. unsigned long changeTime = 0;
10.
11. void setup()
12. {
13.     pinMode(CAR_VERM, OUTPUT);
14.     pinMode(CAR_AMAR, OUTPUT);
15.     pinMode(CAR_VERD, OUTPUT);
16.     pinMode(PED_VERM, OUTPUT);
17.     pinMode(PED_VERD, OUTPUT);
18.     pinMode(BUTTON_PIN, INPUT);
19.     digitalWrite(CAR_VERD, HIGH);
20.     digitalWrite(PED_VERM, HIGH);
21. }
22.
23.
```

Exemplo 04 - Código parte 1

```
24. void loop()
25. {
26.   if (digitalRead(BUTTON_PIN) && (millis() - changeTime) > 5000) {
27.     abrirPedestre();
28.   }
29. }
30.
31. void abrirPedestre() {
32.   trocarLEDS(CAR_VERD, CAR_AMAR);
33.   delay(2000);
34.   trocarLEDS(CAR_AMAR, CAR_VERM);
35.   delay(1000);
36.   trocarLEDS(PED_VERM, PED_VERD);
37.   delay(CROSS_TIME);
38.   blinkLED(PED_VERD, 250, 10);
39.   trocarLEDS(PED_VERD, PED_VERM);
40.   delay(500);
41.   trocarLEDS(CAR_VERM, CAR_VERD);
42.   changeTime = millis();
43. }
44.
45.
46.
```

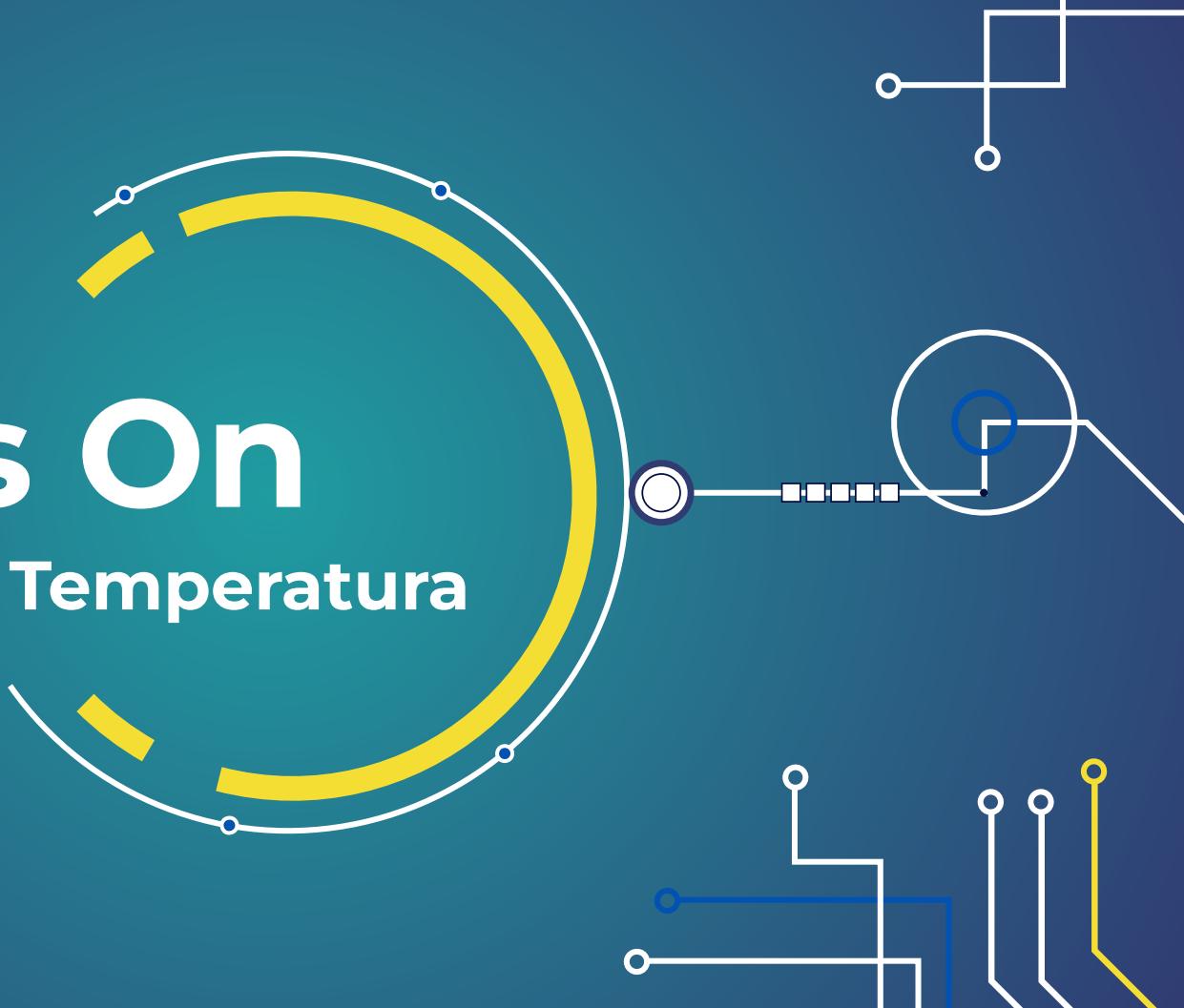
Exemplo 04 - Código parte 2

```
47. void blinkLED(unsigned int ledPin, unsigned int blinkInterval, int reps) {
48.     for (unsigned int x = 0; x < reps; x++) {
49.         digitalWrite(ledPin, HIGH);
50.         delay(blinkInterval);
51.         digitalWrite(ledPin, LOW);
52.         delay(blinkInterval);
53.     }
54. }
55.
56. void trocarLEDS(unsigned int desativarLed, unsigned int ativarLed) {
57.     digitalWrite(desativarLed, LOW);
58.     digitalWrite(ativarLed, HIGH);
59. }
60.
61.
62.
63.
64.
65.
66.
67.
68.
69.
```

Exemplo 04 - Código parte 3

Hands On

Exemplo 05 - Temperatura



EXEMPLO 05 - Temperatura

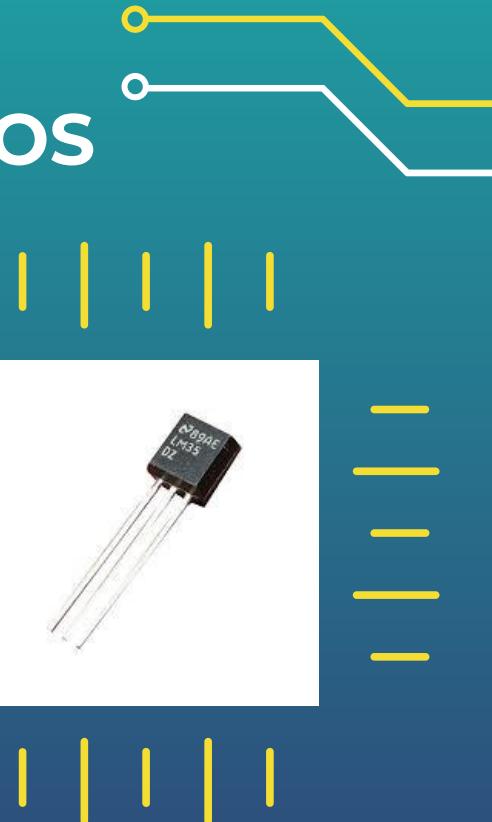
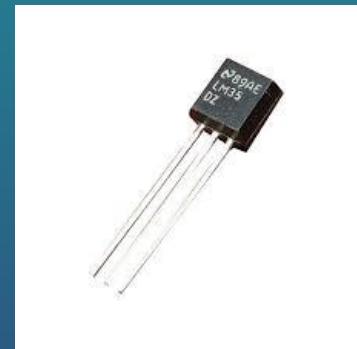
Materiais

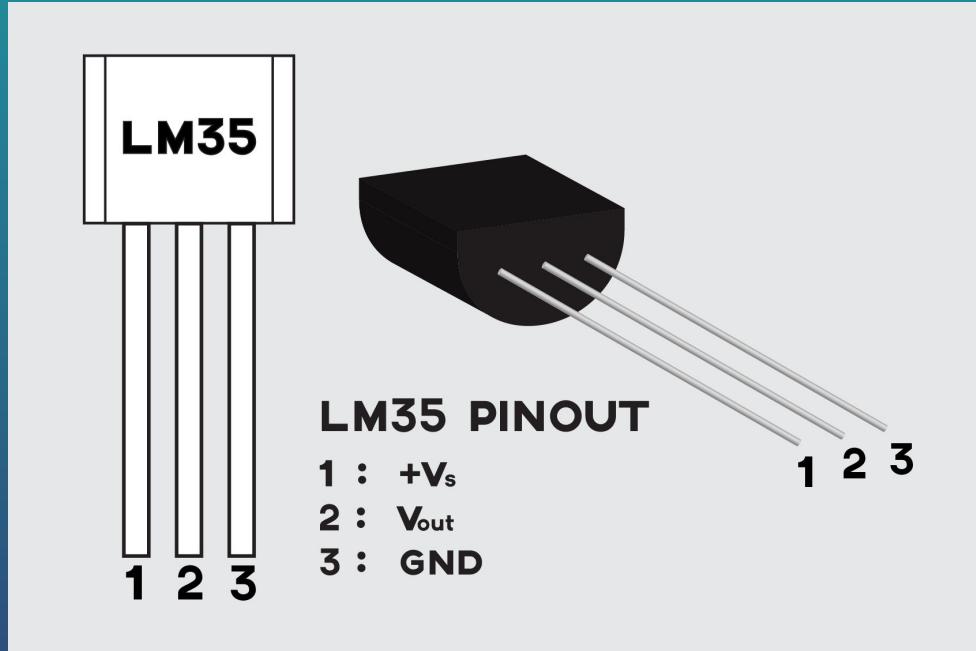
- Este exemplo aborda como fazer leitura de informações provenientes de sensores por meio de pinos analógicos;
- Para este exemplo serão necessários:
 - 01 Placa Arduino UNO;
 - 01 Sensor LM35;
 - 01 Placa de ensaio;
 - Fios jumpers;

COMPONENTES ELETRÔNICOS

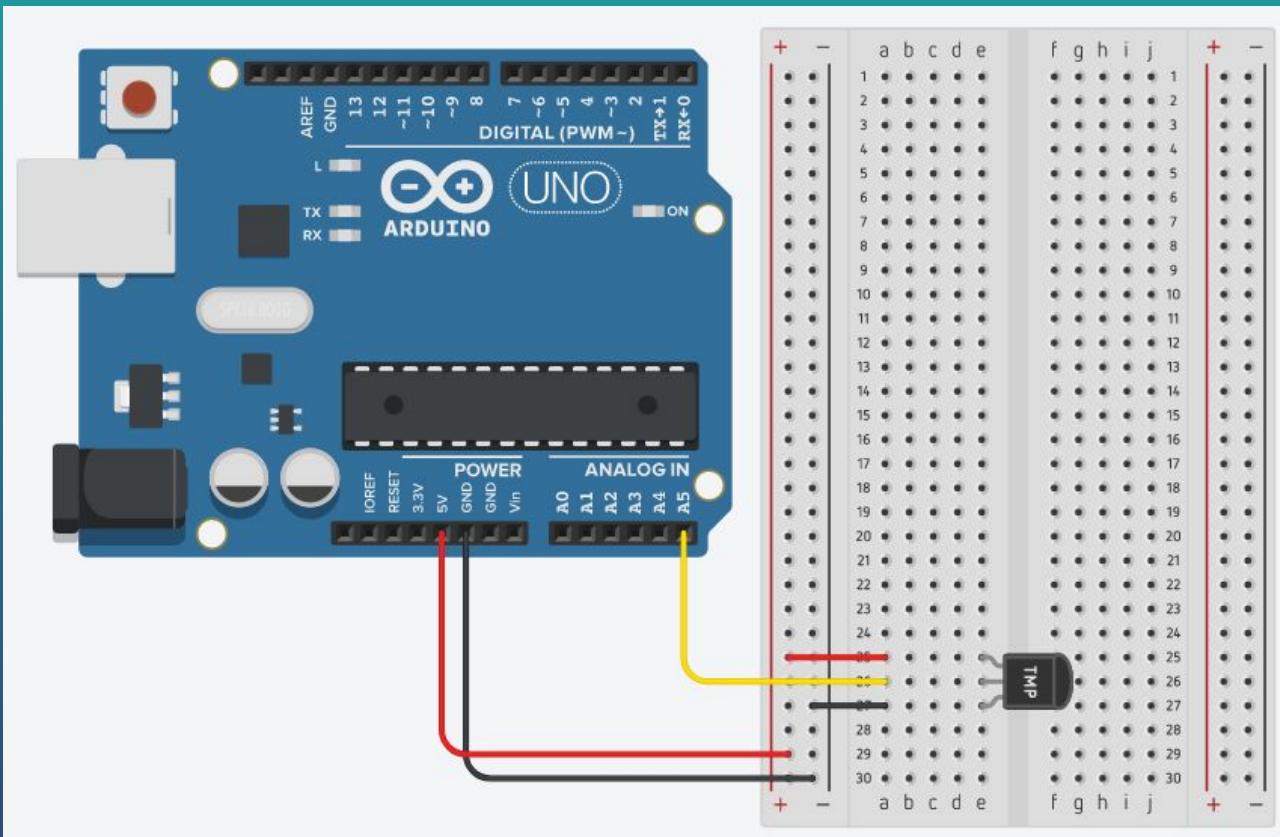
SENSOR TEMPERATURA LM35

- O sensor de Temperatura LM35 é um sensor de precisão;
- Apresenta uma saída de tensão linear relativa à temperatura em que ele se encontrar no momento;
- Sua leitura retorna um sinal variável de 10mV para cada grau celsius de temperatura;
- Sua faixa de operação é entre -55°C e 150°C





Sensor temperatura LM35 – Pinout



Exemplo 5 – Montagem do circuito

```
1. const int SENSOR_PIN = A5;
2.
3. void setup() {
4.   pinMode(SENSOR_PIN, OUTPUT); // configura o pino A5 como pino de saída
5.   Serial.begin(9600);
6. }
7.
8. void loop() {
9.   sensor_val = analogRead(SENSOR_PIN); // lê voltagem do sensor
10.  Serial.print(sensor_val); // imprime a voltagem lida no Monitor Serial
11.  delay(1000);
12. }
```

Exemplo 05 - Código v1

```
1. const int SENSOR_PIN = A5;
2.
3. void setup() {
4.   pinMode(SENSOR_PIN, OUTPUT); // configura o pino A5 como pino de saída
5.   Serial.begin(9600);
6. }
7.
8. void loop() {
9.   sensor_val = analogRead(SENSOR_PIN); // lê voltagem do sensor
10.  Serial.print(sensor_val * 0.48828125); // converte para celsius
11.  delay(1000);
12. }
```

Exemplo 05 - Código v2 (conversão da voltagem para graus celsius)

```
1. const int SENSOR_PIN = A5;
2.
3. void setup() {
4.   pinMode(SENSOR_PIN, OUTPUT); // configura o pino A5 como pino de saída
5.   Serial.begin(9600);
6. }
7.
8. void loop() {
9.   sensor_val = analogRead(SENSOR_PIN); // lê voltagem do sensor
10.  Serial.print("TEMPERATURA = ");
11.  Serial.print(sensor_val * 0.48828125); // converte para celsius
12.  delay(1000);
13. }
14.
15.
16.
17.
18.
```

Exemplo 05 - Código v3 (formatação da mensagem no monitor serial)

```
1. const int SENSOR_PIN = A5;
2.
3. void setup() {
4.   pinMode(SENSOR_PIN, OUTPUT); // configura o pino A5 como pino de saída
5.   Serial.begin(9600);
6. }
7.
8. void loop() {
9.   sensor_val = analogRead(SENSOR_PIN); // lê voltagem do sensor
10.  sensor_val = map(sensor_val, 0, 1023, -55, 150); // converte c/ map
11.  Serial.print("TEMPERATURA = ");
12.  Serial.print(sensor_val);
13.  delay(1000);
14. }
```

Exemplo 05 - Código Vt (função map)

Hands On

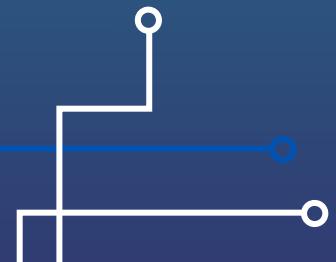
Exemplo 06 - Potenciômetro



EXEMPLO 06 - Potenciômetro

Materiais

- Este exemplo aborda como fazer leitura do valor de um Potenciômetro;
- Para este exemplo serão necessários:
 - 01 Placa Arduino UNO;
 - 01 Potenciômetro;
 - 05 LEDs;
 - 05 Resistores de 100 ohms;
 - 01 Placa de ensaio;
 - Fios jumpers;

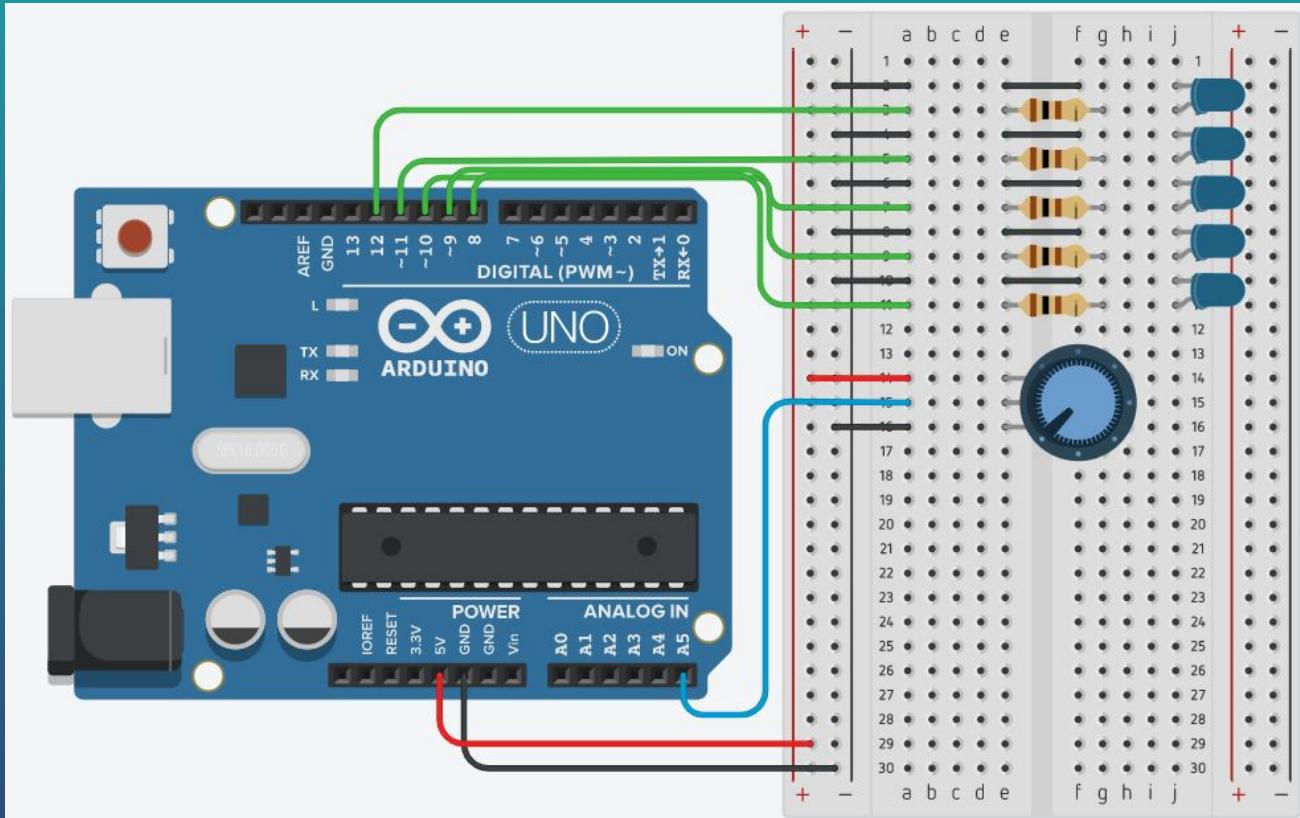


COMPONENTES ELETRÔNICOS

POTENCIÔMETRO

- É um componente eletrônico que possui resistência elétrica ajustável;
- Possui normalmente três terminais onde a conexão central é deslizante e manipulável via cursor móvel;
- Existem variados tipos de potenciômetros:
 - Lineares;
 - Logarítmico;
 - Simples;
 - Duplo;
 - Multi Voltas;





Exemplo 6 – Montagem do circuito

```
1. #define POT_PIN A5
2. #define N_LEDS 5
3. #define LED_DELAY 100
4. const byte ledPins[] = {12, 11, 10, 9, 8};
5. unsigned int ledPos;
6. unsigned long changeTime;
7.
8. void setup() {
9.     for (unsigned int x = 0; x < N_LEDS; x++) {
10.         pinMode(ledPins[x], OUTPUT);
11.     }
12.     changeTime = millis();
13. }
14.
15. void loop() {
16.     ledPos = map(analogRead(POT_PIN), 0, 1023, 0, N_LEDS - 1);
17.     if ((millis() - changeTime) > LED_DELAY) {
18.         changeLED(ledPos);
19.         changeTime = millis();
20.     }
21. }
```

Exemplo 06 - Código parte 1

```
23.  
24. void changeLED(unsigned int ledPos) {  
25.     for (unsigned int x = 0; x < N_LEDS; x++) {  
26.         digitalWrite(ledPins[x], LOW);  
27.     }  
28.     digitalWrite(ledPins[ledPos], HIGH);  
29. }  
30.  
31.  
32.  
33.  
34.  
35.  
36.  
37.  
38.  
39.  
40.  
41.  
42.  
43.  
44.
```

Exemplo 06 - Código parte 2

Hands On

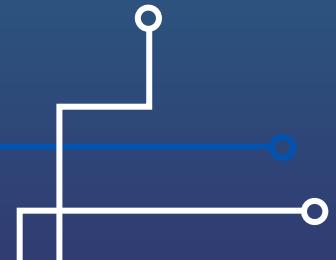
Exemplo 07 - Led RGB



EXEMPLO 07 - LED RGB

Materiais

- Este exemplo aborda como gerar cores diferentes em um LED RGB;
- Para este exemplo serão necessários:
 - 01 Placa Arduino UNO;
 - 01 LED RGB;
 - 03 Resistores de 220 ohms;
 - 01 Placa de ensaio;
 - Fios jumpers;

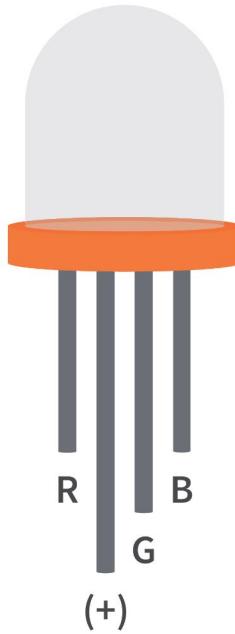
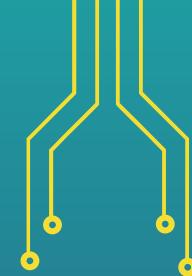


COMPONENTES ELETRÔNICOS

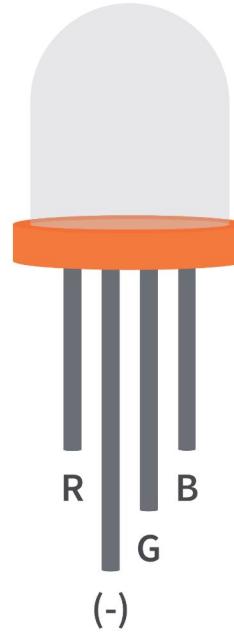
LED RGB

- Assim como o LED tradicional, o LED RGB emite luz por meio da passagem de corrente elétrica;
- É uma **combinação** de três LEDs de cores diferentes:
 - Vermelho (RED);
 - Verde (GREEN);
 - Azul (BLUE);
- Cada LED pode ser controlado de maneira individual, conectando apenas os seus terminais em alguma fonte de energia;
- Podem ser de **dois tipos**:
 - Cártodo (-) comum;
 - Anodo (+) comum;



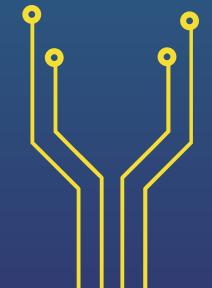


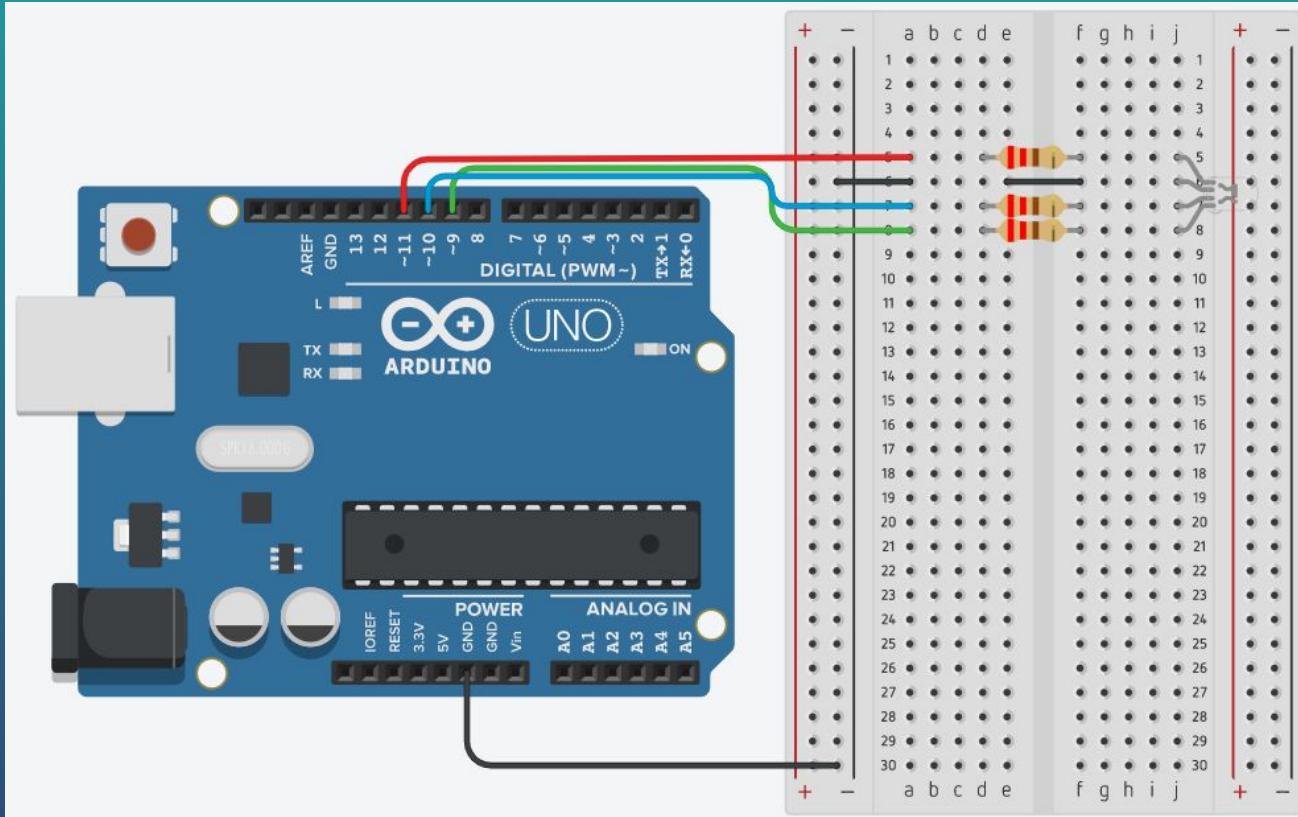
Common Anode



Common Cathode

LED RGB - Pinout de ambos os tipos





Exemplo 7 - Montagem circuito

```
1. #define RLED_PIN 11
2. #define GLED_PIN 9
3. #define BLED_PIN 10
4.
5. const byte ledPins[] = {RLED_PIN, GLED_PIN, BLED_PIN};
6.
7. void setup() {
8.     for (byte x = 0; x < 3; x++)
9.         pinMode(ledPins[x], OUTPUT);
10. }
11.
12. void loop() {
13.     for (byte x = 0; x < 3; x++) {
14.         analogWrite(ledPins[x], random(255));
15.     }
16. }
```

Exemplo 07 - Código v1 (Cores aleatórias)

```
1. #define RLED_PIN 11
2. #define GLED_PIN 9
3. #define BLED_PIN 10
4.
5. String str = ",";
6. const byte ledPins[] = {RLED_PIN, GLED_PIN, BLED_PIN};
7.
8. void setup() {
9.     for (byte x = 0; x < 3; x++)
10.         pinMode(ledPins[x], OUTPUT);
11.     Serial.begin(9600);
12. }
13.
14. void loop() {
15.     for (unsigned int r = 0; r < 256; r += 25)
16.         for (unsigned int g = 0; g < 256; g += 25)
17.             for (unsigned int b = 0; b < 256; b += 25)
18.                 rgb_color(r, g, b);
19. }
20.
```

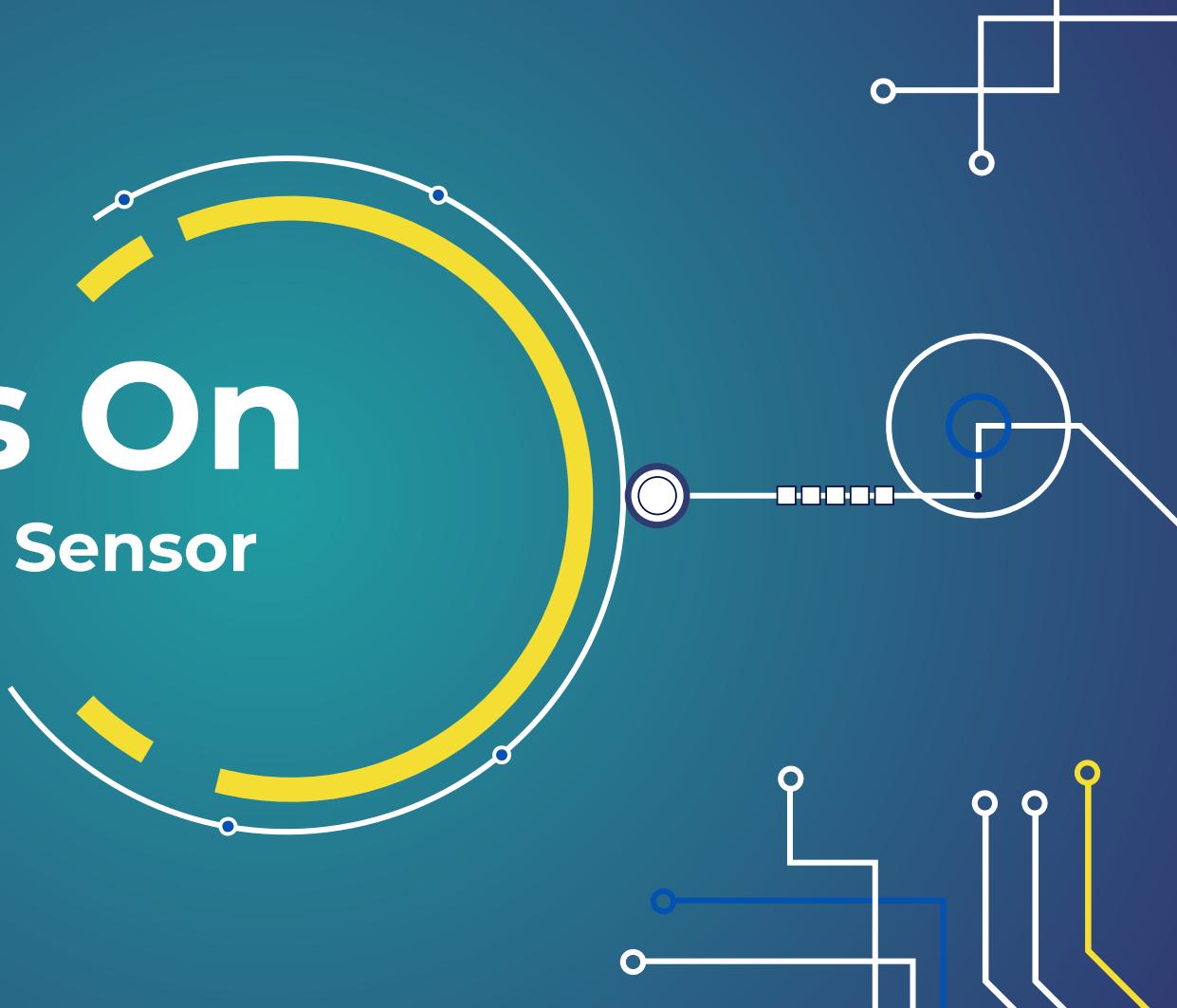
Exemplo 07 - Código v2 (Testando todo o espectro de cores)

```
1.
2. void rgb_color(unsigned int r, unsigned int g, unsigned int b) {
3.     analogWrite(RLED_PIN, r);
4.     analogWrite(GLED_PIN, g);
5.     analogWrite(BLED_PIN, b);
6.     Serial.print("RGB = ");
7.     Serial.println(r + str + g + str + b);
8. }
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
```

Exemplo 07 - Código v2 (Testando todo o espectro de cores)

Hands On

Exemplo 08 - Sensor ultrassônico



EXEMPLO 08 - Sensor ultrassônico

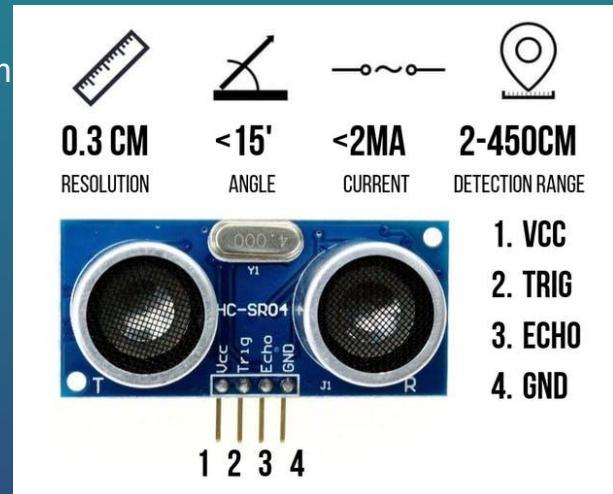
Materiais

- Este exemplo aborda como fazer leitura de informações provenientes de sensores de modo analógico;
- Para este exemplo serão necessários:
 - 01 Placa Arduino UNO;
 - 01 Sensor HC-SR04;
 - 01 Placa de ensaio;
 - Fios Jumpers;

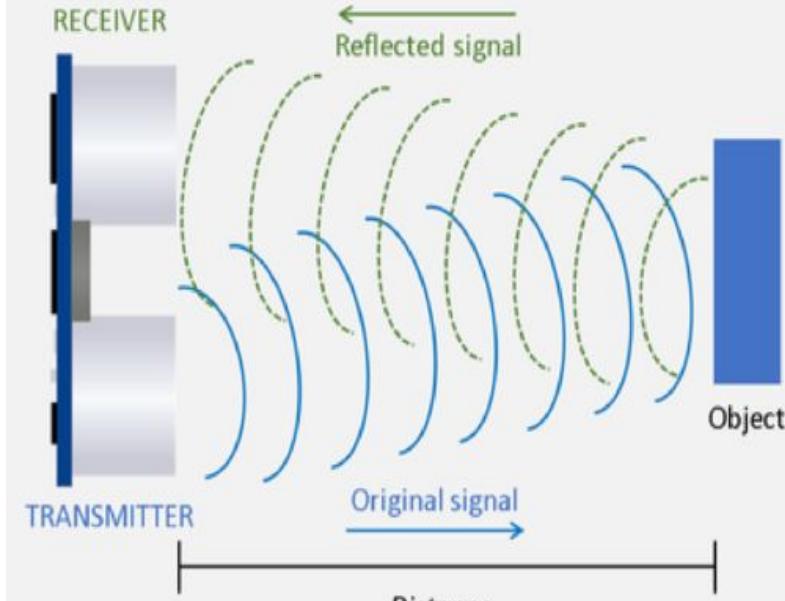
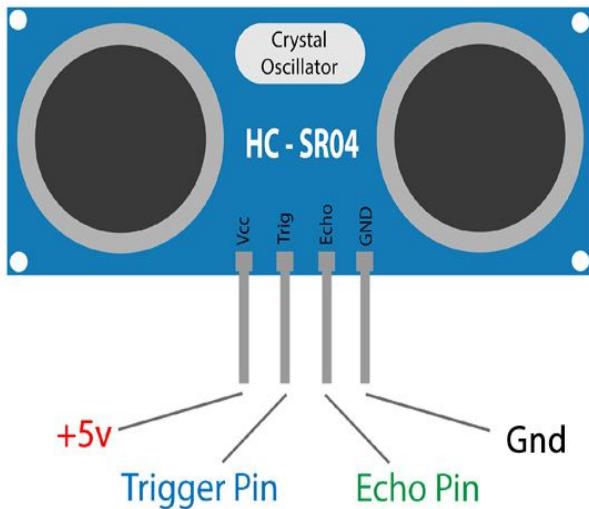
COMPONENTES ELETRÔNICOS

SENSOR ULTRASSÔNICO

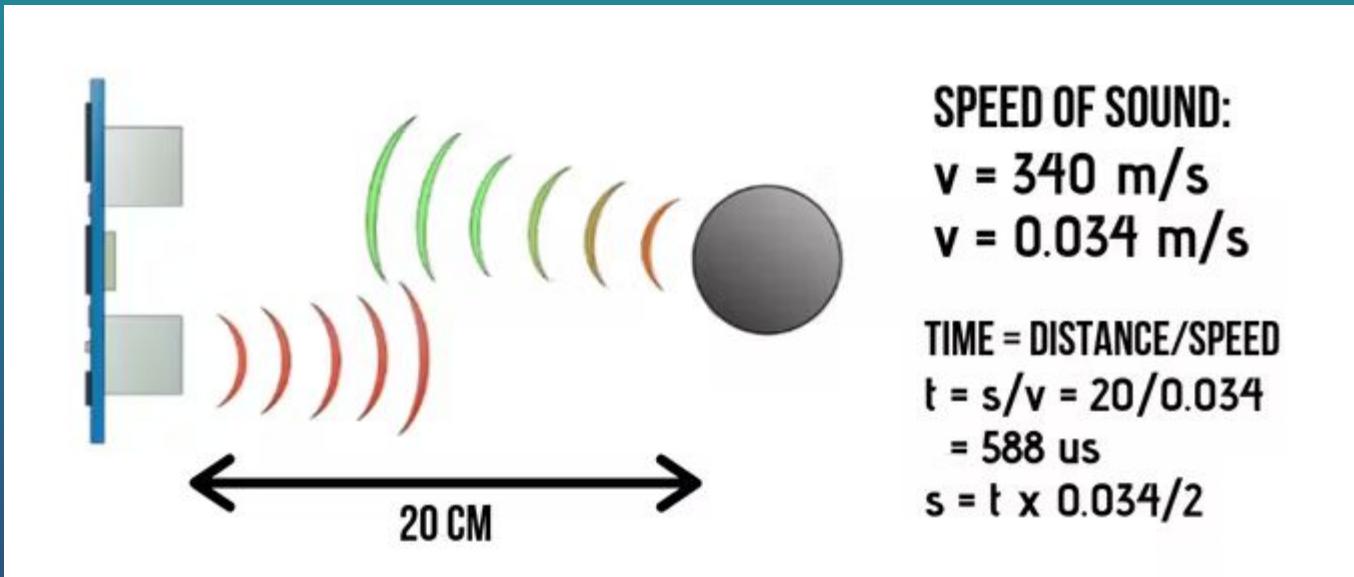
- O Sensor HC-SR04 é um sensor de distância baseado em ondas sonoras;
- Funciona numa angulação de $\pm 15^\circ$;
- Possui um alcance de até 450 cm ($\pm 0.3\text{cm}$);
- O pulso de onda é mensurado em milisegundos;
- Possui 4 pinos:
 - 1 - VCC;
 - 2 - TRIGGER (disparo);
 - 3 - ECHO (leitura);
 - 4 - GND;



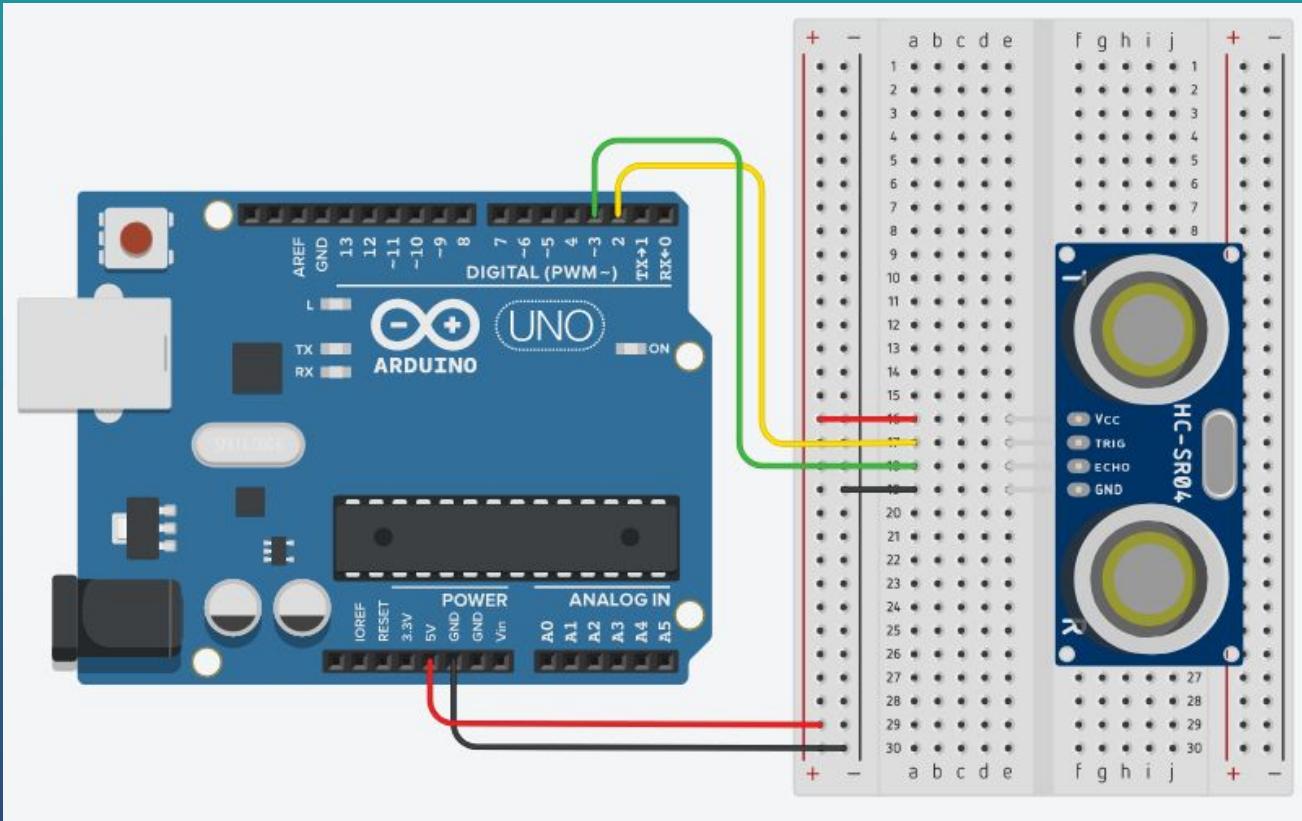
HC-SR04 Pinout



Sensor ultrassônico – Funcionamento do cálculo de distância



Sensor ultrassônico – Funcionamento do cálculo de distância



Exemplo 8 – Montagem circuito

```
1. #define ECHO_PIN 3;
2. #define TRIGGER_PIN 2;
3.
4. unsigned long duracao;
5. float distancia;
6.
7. void setup() {
8.     pinMode(ECHO_PIN, INPUT);
9.     pinMode(TRIGGER_PIN, OUTPUT);
10.    Serial.begin(9600);
11. }
12.
13. void loop() {
14.     digitalWrite(TRIGGER_PIN, LOW);      // desativa o sensor
15.     delayMicroseconds(2);
16.     digitalWrite(TRIGGER_PIN, HIGH);     // ativa o sensor por 10ms
17.     delayMicroseconds(10);
18.     digitalWrite(TRIGGER_PIN, LOW);      // desativa o sensor
19.     duracao = pulseIn(ECHO_PIN, HIGH); // lê a duracao da onda em ms
20.     distancia = duracao * 0.034 / 2;   // calcula a distancia em cm
21.     Serial.println(distancia);
22. }
```

Exemplo 08 - Código v1 (para abrir o Monitor Serial pressione Ctrl + Shift + M)

```
1. #include <NewPing.h>
2. #define ECHO_PIN 3;
3. #define TRIGGER_PIN 2;
4. #define MAX_DISTANCE 400 // distancia maxima a ser mensurada (em cm)
5.
6. NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing object
7.
8. void setup() {
9.     pinMode(ECHO_PIN, INPUT);
10.    pinMode(TRIGGER_PIN, OUTPUT);
11.    Serial.begin(9600);
12. }
13.
14. void loop() {
15.     delay(50); // aguarda 50ms entre as leituras (20 pings/sec).
16.     unsigned long distancia = sonar.ping_cm(); // retorna a distancia em cm
17.     Serial.print("DISTÂNCIA: ");
18.     Serial.print(distancia );
19.     Serial.println("cm");
20. }
```

Exemplo 08 - Código v2 (para abrir o Monitor Serial pressione Ctrl + Shift + M)

Hands On

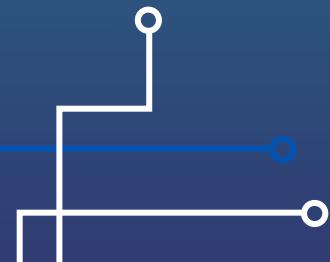
Exemplo 09 - Plotter Serial

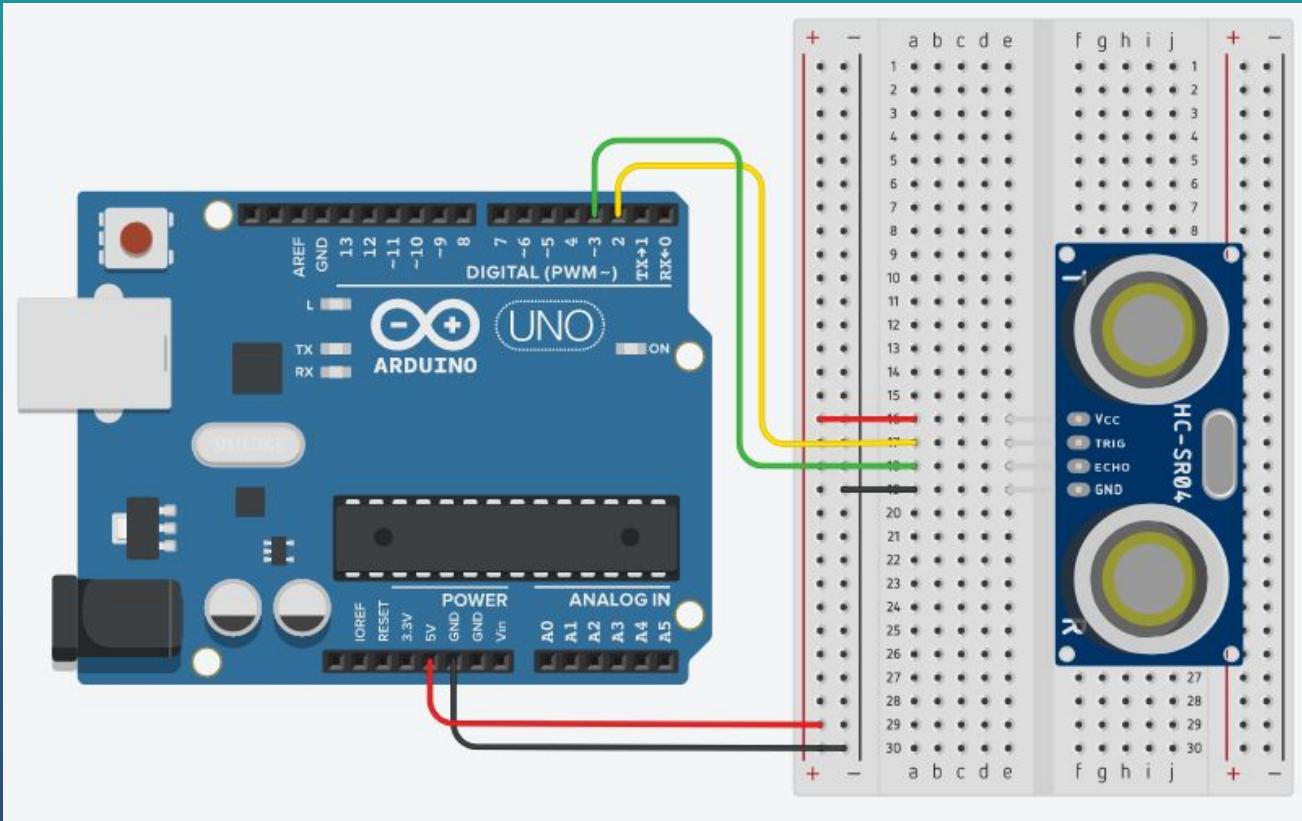


EXEMPLO 09 - Plotter Serial

Materiais

- Este exemplo aborda como fazer leitura de informações provenientes de sensores de modo analógico e exibir um gráfico desses valores;
- Para este exemplo serão necessários:
 - 01 Placa Arduino UNO;
 - 01 Sensor HC-SR04;
 - 01 Placa de ensaio;
 - Fios Jumpers;





Exemplo 9 – Montagem circuito

```
1. #include <NewPing.h>
2. #define ECHO_PIN 3;
3. #define TRIGGER_PIN 2;
4. #define MAX_DISTANCE 400 // distancia maxima a ser mensurada (em cm)
5.
6. NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE); // NewPing object
7.
8. void setup() {
9.     pinMode(ECHO_PIN, INPUT);
10.    pinMode(TRIGGER_PIN, OUTPUT);
11.    Serial.begin(9600);
12. }
13.
14. void loop() {
15.     delay(50); // aguarda 50ms entre as leituras (20 pings/sec).
16.     unsigned long distancia = sonar.ping_cm(); // retorna a distancia em cm
17.     Serial.println(distancia);
18. }
19.
20.
```

Exemplo 09 - Código (para abrir o Plotter Serial pressione Ctrl + Shift + L)

Hands On

Exemplo 10 - Funções matemáticas



EXEMPLO 10 - Funções matemáticas

Materiais

- Este exemplo aborda como utilizar funções matemáticas prontas da biblioteca Arduino;
- Para este exemplo serão necessários:
 - 01 Placa Arduino UNO;
 - 01 Alarme de Som Piezo (ou Buzzer);
 - 01 Resistor de 100 ohms;
 - 01 Placa de ensaio;
 - Fios Jumpers;

Biblioteca Arduino

Funções disponíveis

- Além da definição da linguagem, operadores, construções e palavras-chave, a biblioteca da linguagem Arduino trás algumas funções matemáticas que podem ser utilizadas;
- Podem ser divididas em:
 - Funções de E/S avançadas;
 - Funções para temporização;
 - Funções matemáticas;
 - Funções trigonométricas;
 - Funções para manipulação de caracteres;
 - Funções de aleatoriedade;
 - Funções para bits e bytes;
 - Funções de comunicação.

Funções matemáticas

abs(x)

Módulo

Retorna o módulo (valor absoluto) do número X.

max(x, y)

Maior

Retorna o maior valor entre “x” e “y”.

map(valor, x, y, z, w)

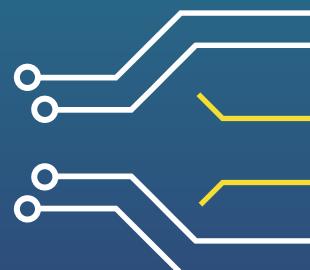
Mapeamento

Faz o mapeamento de “valor” do intervalo [x … y] para o intervalo [z … w].

min(x, y)

Menor

Retorna o menor valor entre “x” e “y”.



Funções matemáticas

pow(x, y)

Potência

Retorna o valor de “x” elevado a potência “y”.

sqrt(X)

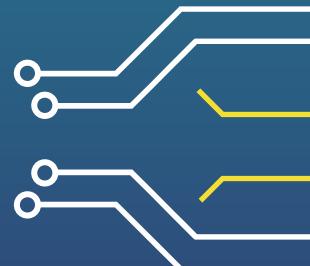
Raiz quadrada

Retorna o valor da raiz quadrada de “x”

sq(x)

Quadrado

Retorna o valor de “x” elevado ao quadrado.



Funções trigonométricas

cos(x)

Cosseno

Retorna o valor do cosseno do ângulo X (em radianos).

tan(X)

Tangente

Retorna o valor da tangente do ângulo X (em radianos).

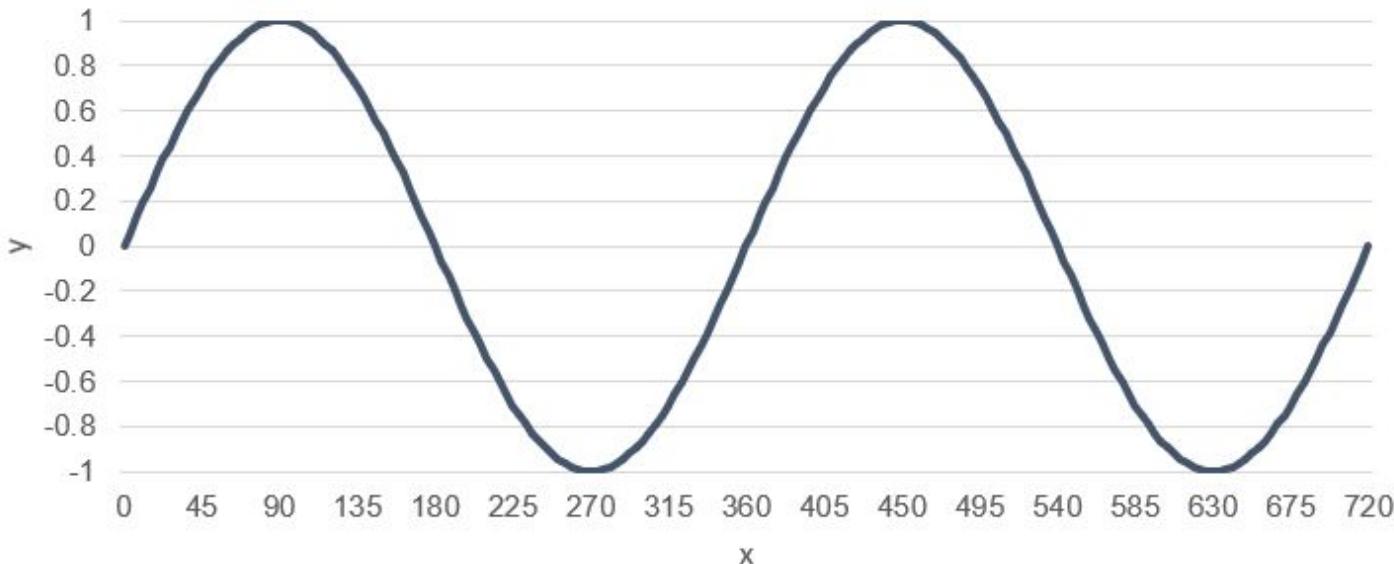
sin(x)

Seno

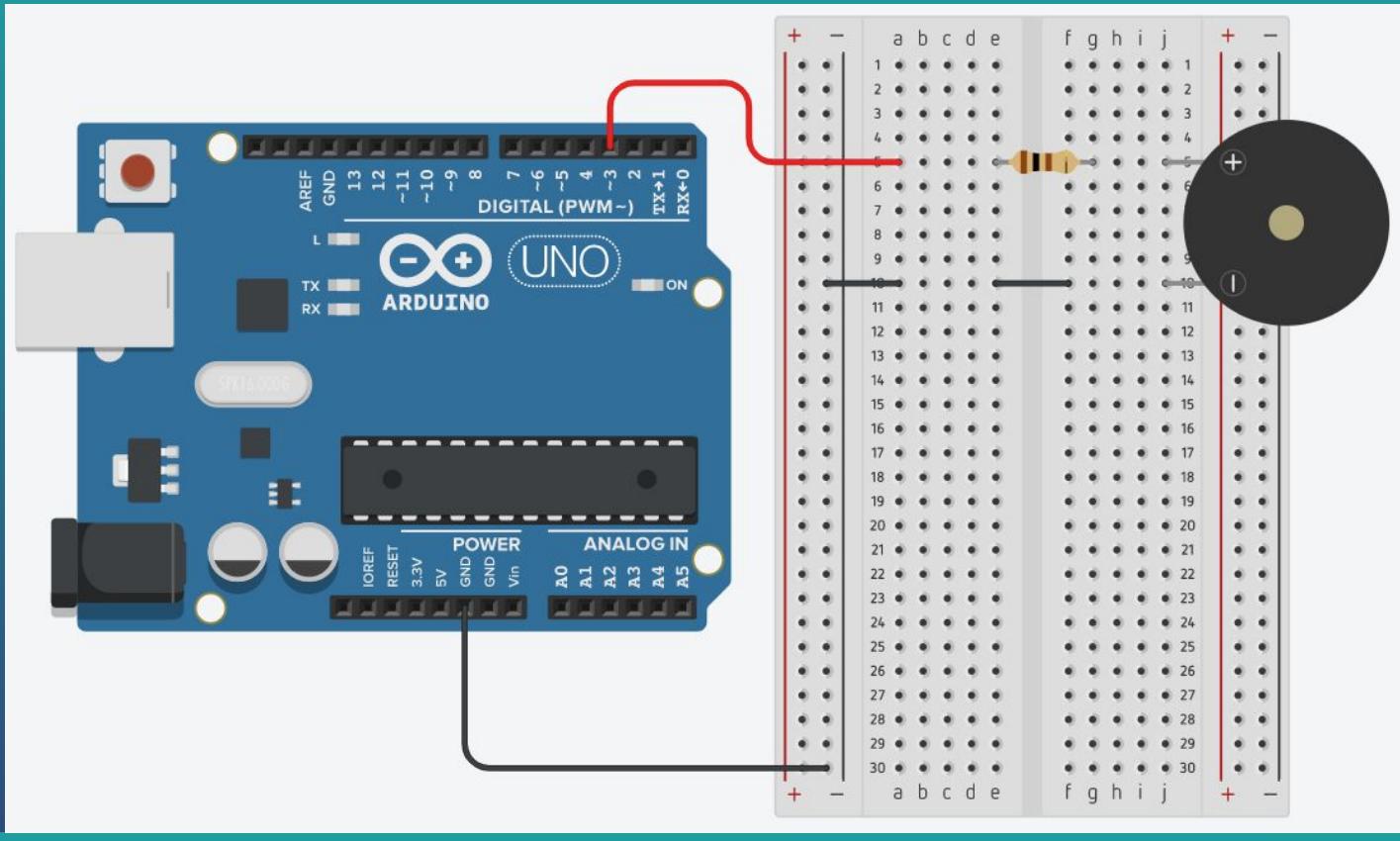
Retorna o valor do seno do ângulo X (em radianos).



Graph of $y = \sin(x)$



Formato da função seno



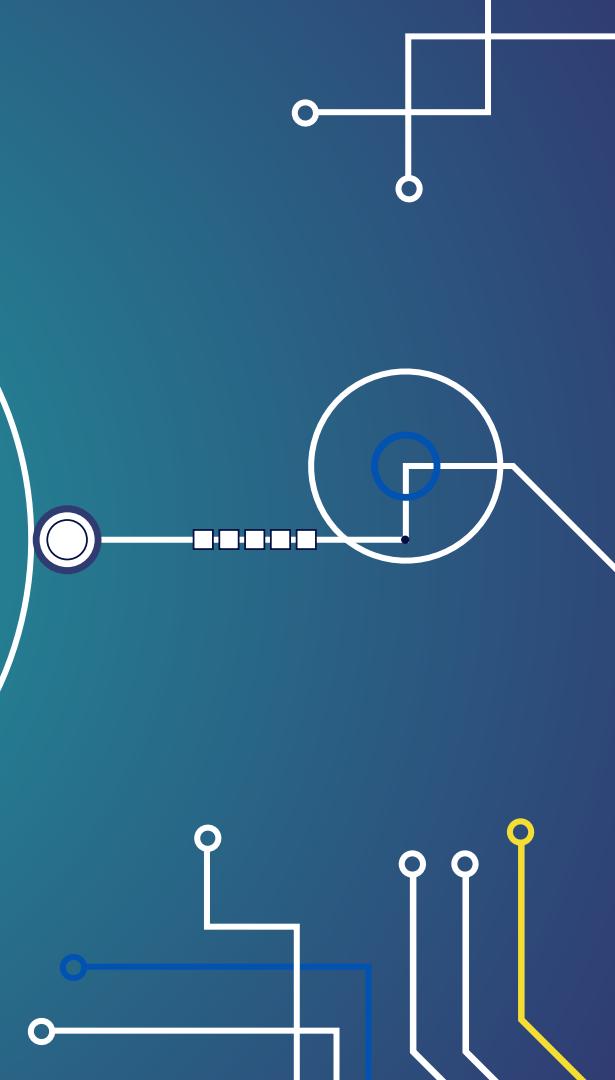
Exemplo 10 – Montagem circuito

```
1. #define PIEZO_PIN 3
2. float sinVal;
3. int toneVal;
4.
5. void setup() {
6.   pinMode(PIEZO_PIN, OUTPUT);
7.   Serial.begin(9600);
8. }
9.
10. void loop() {
11.   for (byte x = 0; x < 180; x++) {
12.     // converte o angulo para radiano e calcula o valor do seno
13.     sinVal = sin(x * (3.1412 / 180));
14.     // gera um valor de frequênci para o valor do seno
15.     toneVal = 2000 + int(sinVal * 1000);
16.     tone(PIEZO_PIN, toneVal);
17.     delay(2);
18.   }
19. }
```

Exemplo 10 - Código

Hands On

Exemplo 11 - Trabalhando com motores



EXEMPLO 11 - Trabalhando com motores

Materiais

- Este exemplo aborda como utilizar motores (mais especificamente servo motores) em aplicações com Arduino;
- Para este exemplo serão necessários:
 - 01 Placa Arduino UNO;
 - 01 Servo motor;
 - 01 Resistor de 100 ohms;
 - 01 Placa de ensaio;
 - Fios Jumpers;

TIPOS DE MOTORES



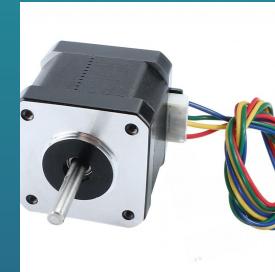
MOTOR DC

Motor que converte energia elétrica em mecânica.



SERVO MOTOR

Motor de precisão em ângulo com velocidade e torque quase constante.



MOTOR DE PASSO

Motor de alta precisão em ângulo com perda de torque em altas rotações.

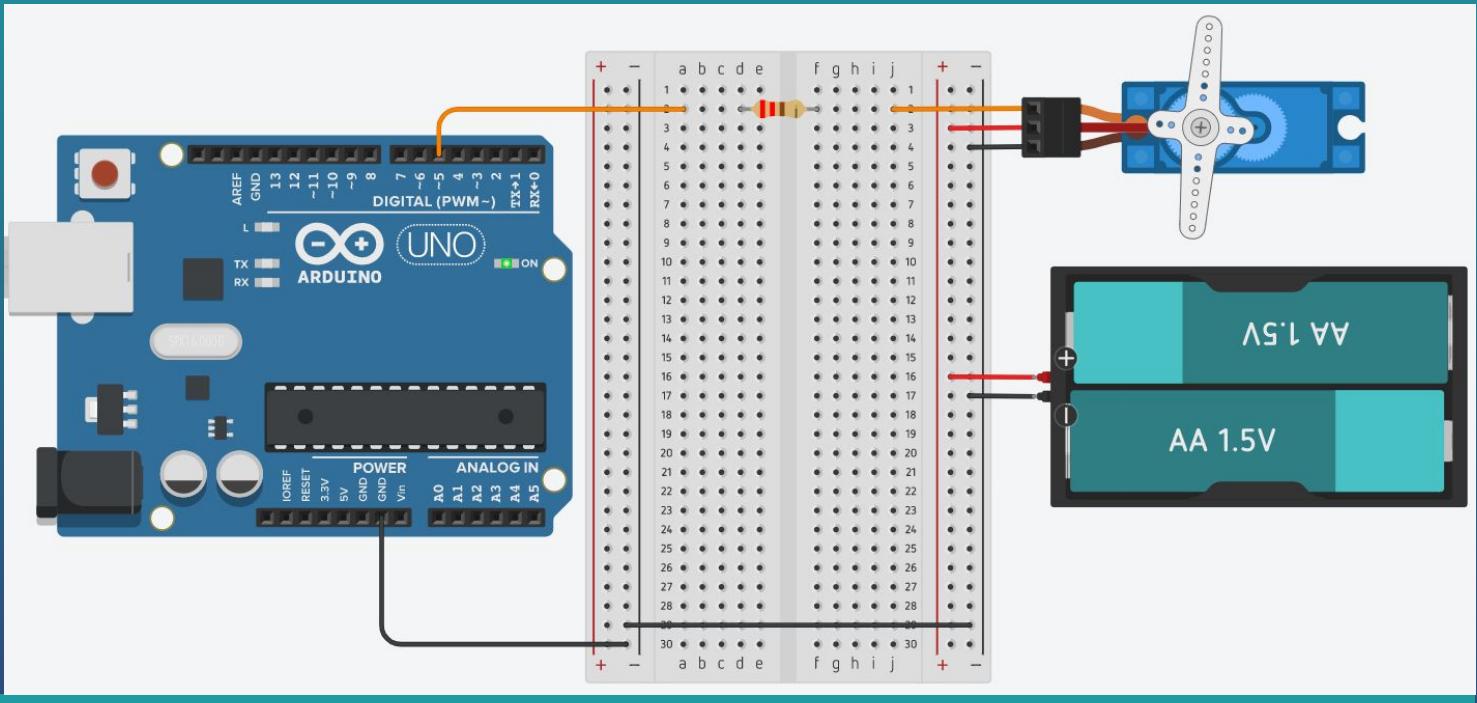


COMPONENTES ELETRÔNICOS

SERVO MOTOR

- “Servos” são motores com controle angular de posição;
- Trabalham tipicamente com 180°, embora existam alguns modelos de rotação contínua (360°);
- São geralmente utilizados nas articulações móveis de projetos robóticos e para controle de animatrônicos;
- Embora a placa Arduino possa fornecer energia para um servo motor, isso geralmente gera aquecimento e danos à placa. Sendo por isso recomendado uma alimentação externa;

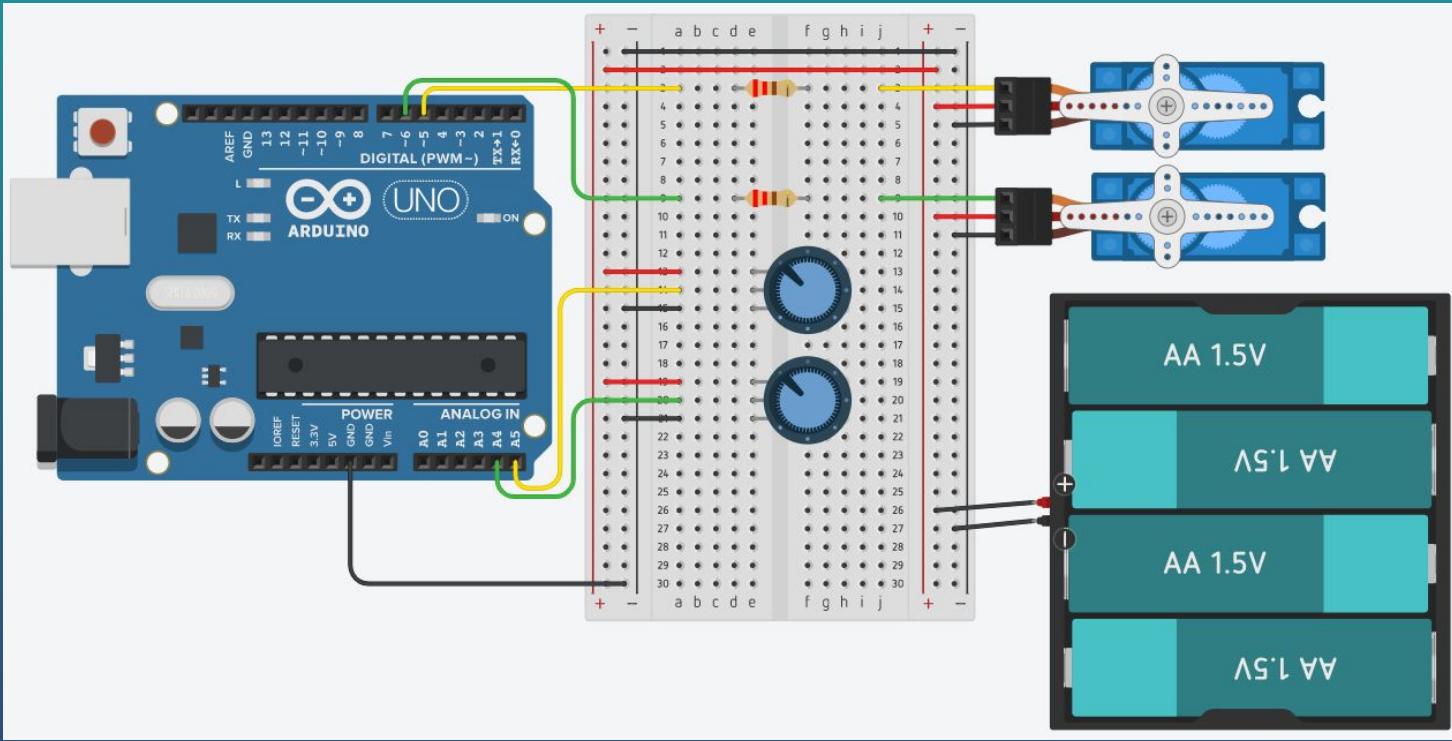




Exemplo 11a – Montagem circuito

```
1. #include <Servo.h>
2. #define SERVO_PIN 5
3.
4. Servo servo; // cria um objeto para controlar o servo motor
5. int servoPos = 90; // posicao inicial do servo motor
6.
7. void setup() {
8.     servo.attach(SERVO_PIN); // define que o servo sera operado pelo pino 5
9.     servo.write(servoPos); // coloca o servo na posicao inicial
10.    delay(25);
11. }
12.
13. void loop() {
14.     for (; servoPos < 180; servoPos++) {
15.         servo.write(servoPos);
16.         delay(20);
17.     }
18.     for (; 0 < servoPos; servoPos--) {
19.         servo.write(servoPos);
20.         delay(20);
21.     }
22. }
```

Exemplo 11a - Código



Exemplo 11b – Montagem circuito

```
1. #include <Servo.h>
2. #define SERVO1_PIN 5
3. #define SERVO2_PIN 6
4. #define POT1_PIN A5
5. #define POT2_PIN A4
6.
7. Servo servo1, servo2; // cria os objetos para controlar os servo motores
8. int pot1, pot2 // variaveis para leitura dos potenciometros
9.
10. void setup() {
11.     servo1.attach(SERVO1_PIN); // define que o servo sera operado pelo pino 5
12.     servo2.attach(SERVO2_PIN); // define que o servo sera operado pelo pino 6
13.
14.     servo1.write(0); // coloca o servo 1 na posicao inicial
15.     servo2.write(0); // coloca o servo 2 na posicao inicial
16.
17.     delay(25);
18. }
19.
```

Exemplo 11b - Código parte 1

```
20.  
21. void loop() {  
22.     pot1 = analogRead(POT1_PIN); // le o valor indicado no potenciometro 1  
23.     pot2 = analogRead(POT2_PIN); // le o valor indicado no potenciometro 2  
24.  
25.     // converte para um angulo o valor lido no potenciometro 1  
26.     pot1 = map(pot1, 0, 1023, 0, 180);  
27.     // converte para um angulo o valor lido no potenciometro 2  
28.     pot2 = map(pot2, 0, 1023, 0, 180);  
29.  
30.     // movimenta os servo motores para seus respectivos angulos  
31.     servo1.write(pot1);  
32.     servo2.write(pot2);  
33.  
34.     delay(15);  
35. }
```

Exemplo 11b - Código parte 2

REFERÊNCIAS

BIBLIOGRÁFICAS



LIVROS

Primeiros Passos com o Arduino – 2a. Edição

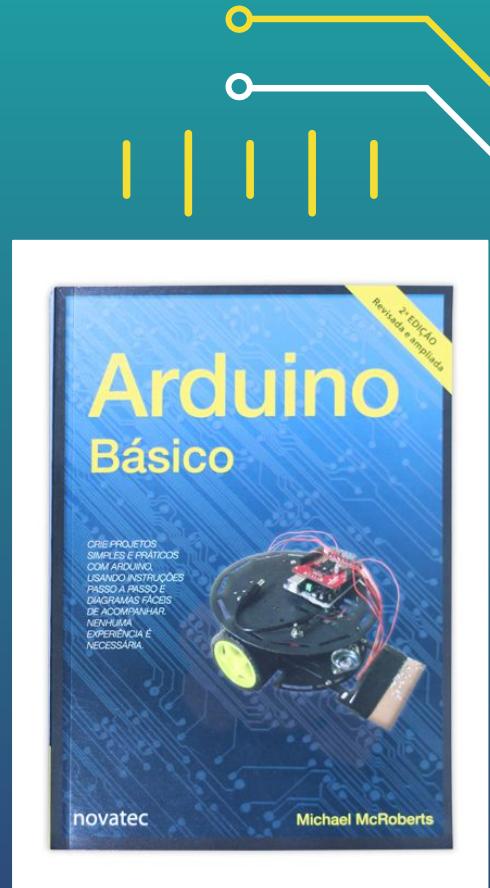
- O livro Primeiros Passos com o Arduino é um dos mais indicados para quem não sabe nada sobre Arduino.
- Escrito em conjunto com Massimo Banzi, um dos criadores do Arduino;
- O livro dá uma visão geral do que é a plataforma, onde os projetos com Arduino podem se encaixar, e aborda itens básicos mas interessantes da utilização, como por exemplo a identificação de portas COM, instalação da IDE e exemplos de programação com leds, botões, sensores, até chegar à itens mais complexos, como o uso de portas PWM.



LIVROS

Arduino Básico – 2a. Edição

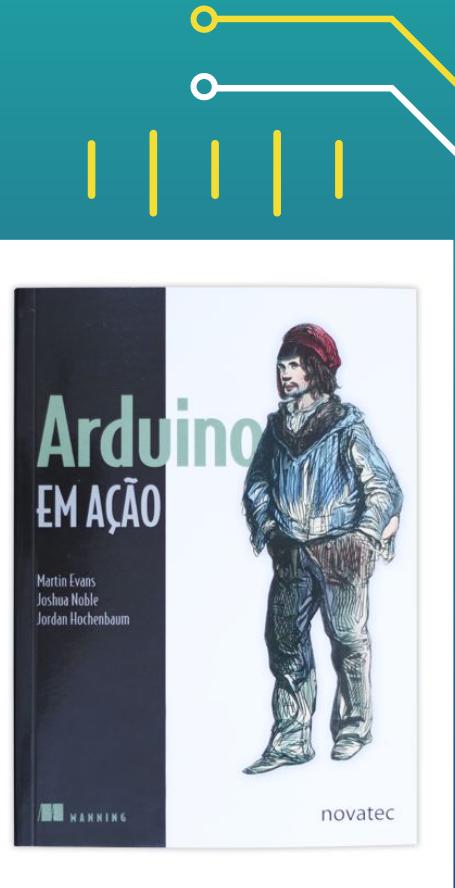
- Um dos livros mais abrangentes sobre o assunto, Arduino Básico é dividido em 50 projetos com diferentes graus de dificuldade;
- Para cada projeto é apresentada a lista de material, o esquema de montagem e o código de programação acompanhado de uma explicação detalhada sobre a lógica do programa;
- O livro aborda não só o básico sobre Arduino, mas também como trabalhar com motores, sensores, contadores, registradores de deslocamento, shields e muitos outros componentes.



LIVROS

Arduino em Ação

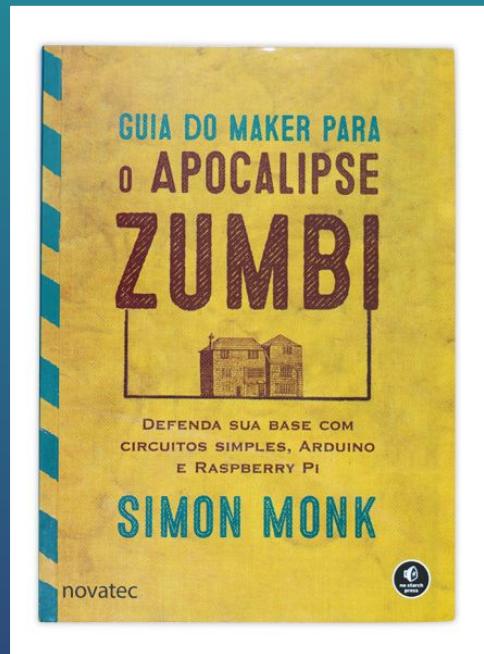
- O livro Arduino em Ação possui um capítulo inicial sobre o Arduino, como instalar a IDE, configuração do ambiente de trabalho e outros pré-requisitos para utilização da placa;
- Contém explicações mais detalhadas sobre cada item apresentado, mas é indicado para usuários um pouco mais avançados ou com noções de eletrônica, já que mostra o diagrama esquemático de ligação dos projetos, sem apresentar uma imagem detalhada do circuito em uma protoboard, como é comum em outros livros;
- De qualquer forma, é mais uma excelente opção de informação para quem deseja saber um pouco mais sobre o Arduino.



LIVROS

Guia do Maker para o Apocalipse Zumbi

- O livro Guia do Maker para o Apocalipse Zumbi ensina de maneira bem humorada a construção de pequenos projetos usando Arduino e Raspberry para enfrentar um ataque de zumbis em um cenário pós-apocalíptico;
- Com ele você vai aprender a montar circuitos geradores para conseguir sua própria energia, sistemas de alarme, monitoração de ambiente e sistemas de comunicação para conversar com outros “sobreviventes”;
- Os capítulos são apresentados com esquemas de montagem, lista de material e explicações detalhadas sobre como testar cada circuito.



OBRIGADO!

Alguma dúvida?

 marcos.lima@icomp.ufam.edu.br

 (92) 98221 0642

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik

