



🎓 Desenvolvimento de sensores e atuadores IoT – Parte 1

Aula síncrona (10/06/2025)

Prof. Wilton Lacerda Silva

Executores:



Coordenação:



Iniciativa:



Sumário

- Objetivos
- LDR e Potenciômetro como sensores
- Uso de relés para comando
- Exemplos de aplicações

Objetivos

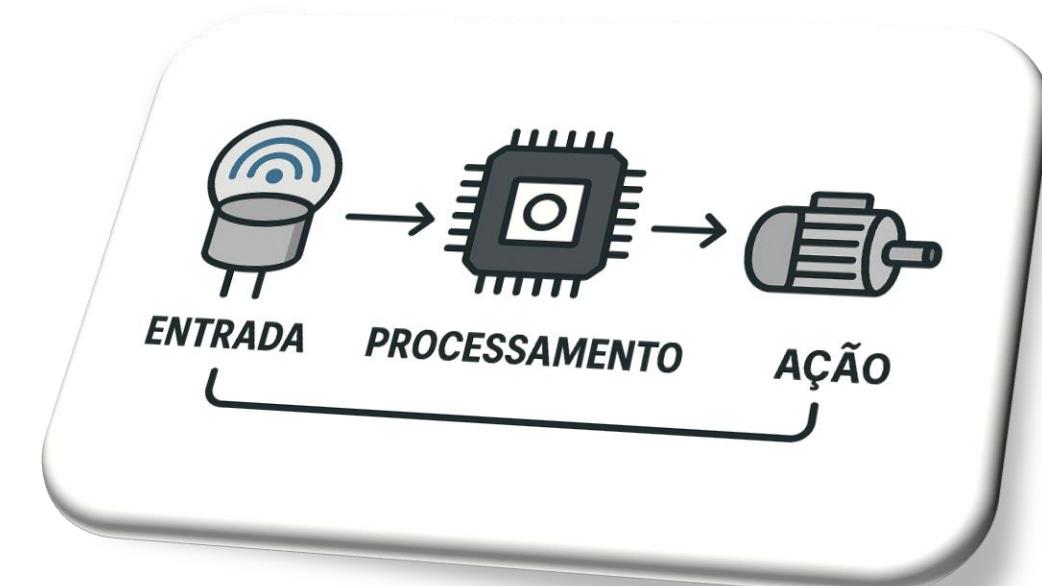
- Projetar e integrar sensores e atuadores específicos.
- Desenvolver alguns exemplos para fixação dos conceitos e realizar atividades práticas com sensores e atuadores.



O que é sensor?

Do ponto de vista da **eletrônica, instrumentação e automação**, um sensor é um dispositivo que detecta uma grandeza física ou química e a converte em um sinal elétrico que pode ser medido, interpretado e processado por sistemas eletrônicos, como **microcontroladores, CLPs** ou sistemas de aquisição de dados.

Sensor é um **transdutor** que converte uma variável do ambiente (como temperatura, pressão, luz, umidade, força, movimento, etc.) em um sinal elétrico utilizável por um sistema de controle ou monitoramento.



Função:

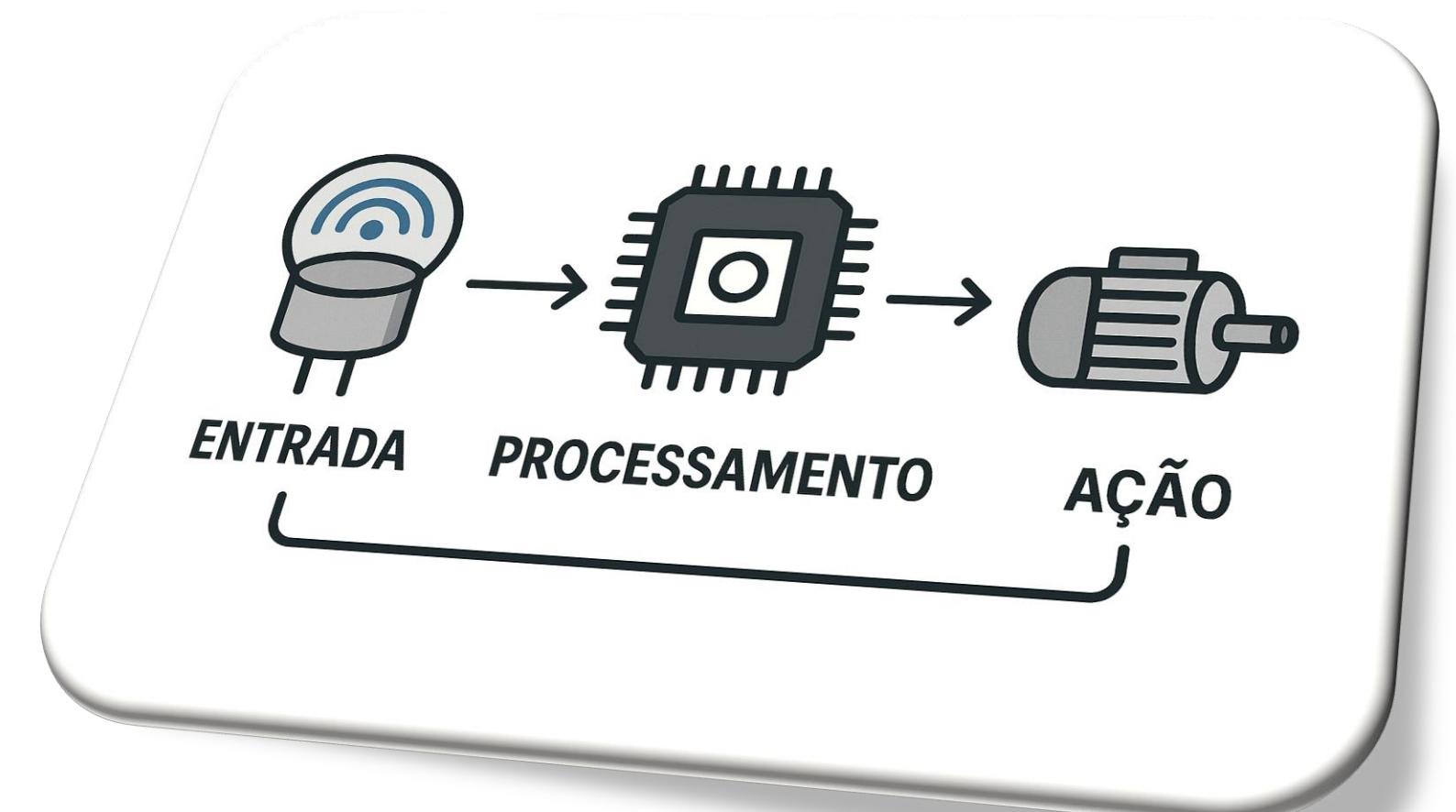
Converter uma grandeza física (analogica) em um sinal elétrico (analogico ou digital) que pode ser lido e processado por um sistema eletrônico (como um microcontrolador). São essenciais para a automação, robótica, medição e controle em diversas aplicações.

Tipo de Sensor	Grandeza Detectada	Sinal Elétrico Gerado
LM35	Temperatura	Tensão proporcional (°C)
LDR	Intensidade luminosa	Variação de resistência
Sensor ultrassônico	Distância	Pulso digital (tempo)
Sensor de pressão (MPX)	Pressão	Tensão analógica
Encoder	Posição/Velocidade	Pulsos digitais (quadratura)
Potenciômetro	Posição angular ou linear	Tensão analógica variável

Sensores na Automação/Controle

Papel no sistema automatizado:

- 1. Entrada:** O sensor representa o **elemento de entrada** no ciclo de automação.
- 2. Processamento:** O sinal do sensor é **condicionado e interpretado** por um microcontrolador.
- 3. Ação:** Com base nas leituras, o sistema decide acionar **atuadores**, como motores, válvulas ou relés.



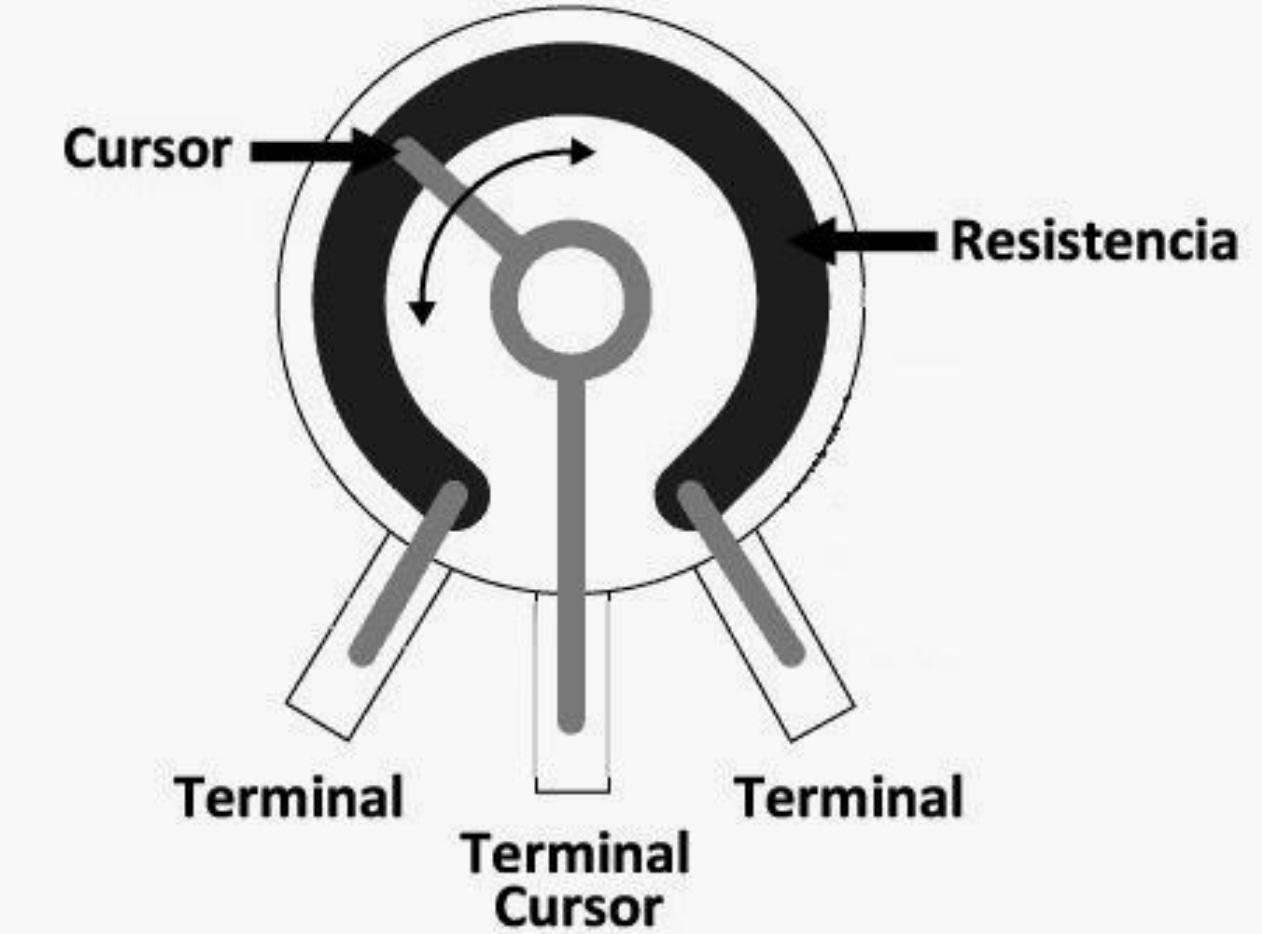
Potenciômetro - O que é?

Definição:

Um potenciômetro é um resistor variável com três terminais. Ele funciona como um divisor de tensão ajustável.

Componentes Principais:

- **Elemento Resistivo:** Uma trilha de material resistivo (ex: carbono, metal).
- **Cursor (Wiper):** Um contato deslizante que se move ao longo do elemento resistivo.
- **Terminais:** Dois terminais fixos nas extremidades do elemento resistivo e um terminal conectado ao cursor.
- **Tipos Comuns:** Rotativo (para rotação) e Linear (para deslizamento)



<https://eletricaesuasduvidas.blogspot.com/2014/02/o-que-e-um-potenciometro.html>

Potenciômetro - Tipos

Potenciômetro Linear:

A resistência varia de forma linear com a posição do cursor.

Potenciômetro Logarítmico:

A resistência varia de forma logarítmica com a posição do cursor, o que é comumente usado para controlar o volume.

Trimpot:

Potenciômetros de ajuste fino, geralmente usados para calibração e ajuste preciso.

Potenciômetro Deslizante:

A resistência é ajustada por um cursor que se move ao longo de uma pista resistiva.



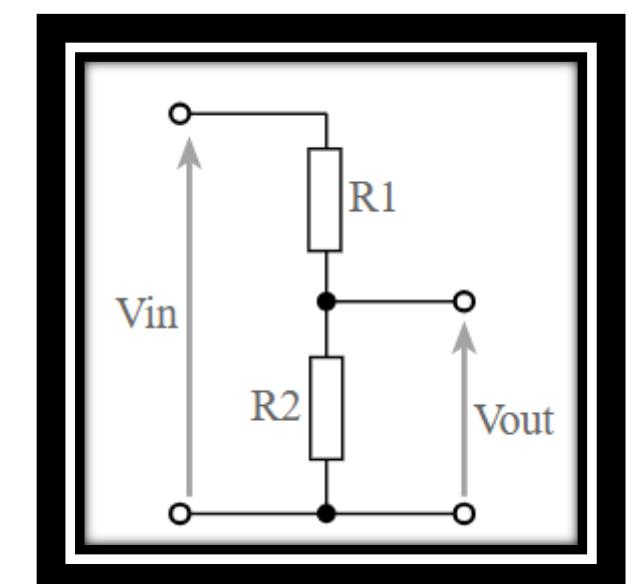
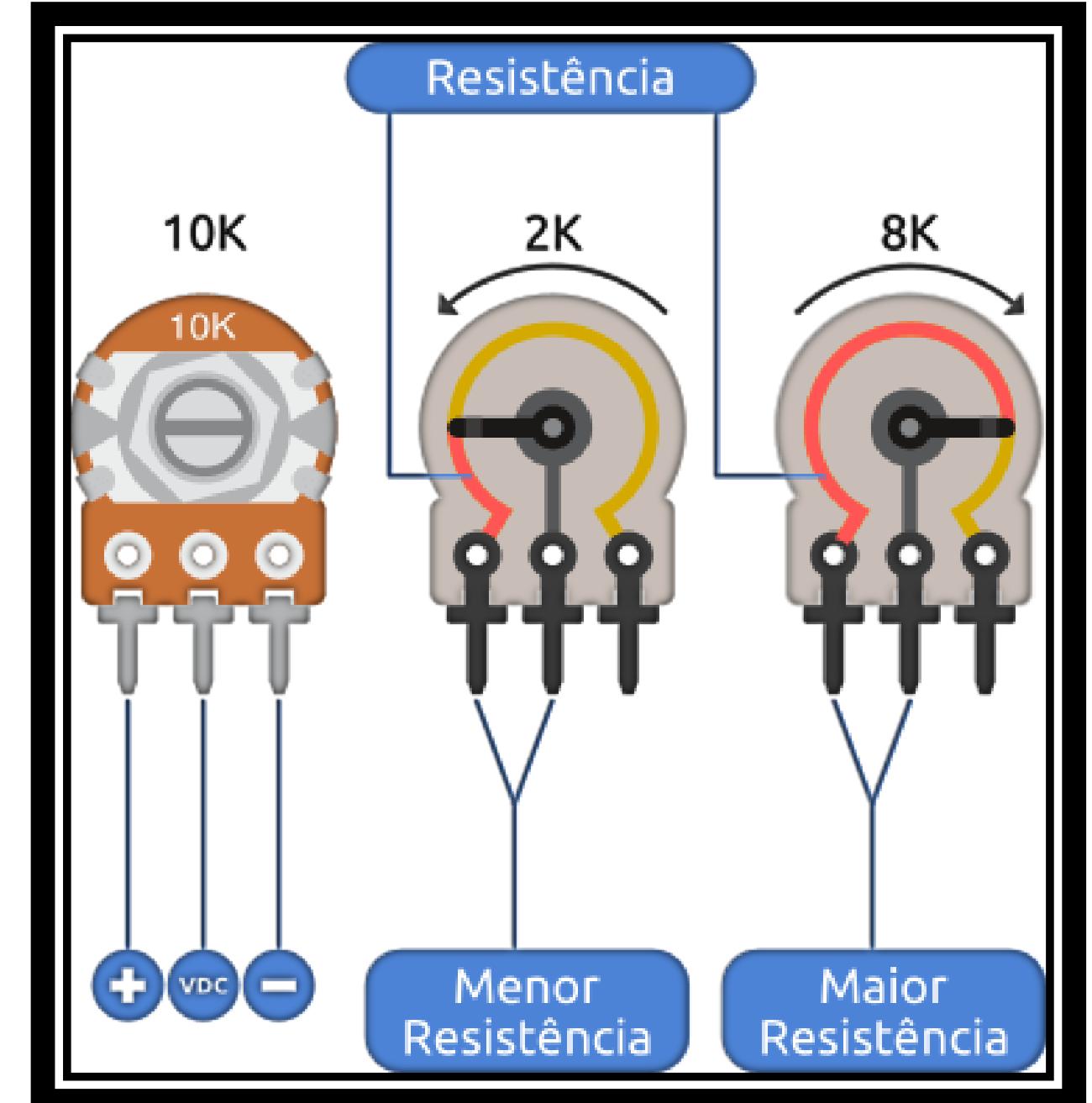
<https://www.kitmakers.com.br/cmp/potenciometro.html>

Potenciômetro: Funcionamento

Ao girar o eixo (ou deslizar o cursor), a resistência entre o terminal central (cursor) e os terminais externos varia.

Divisor de Tensão: Quando os três terminais são usados, o potenciômetro atua como um divisor de tensão. Uma tensão de entrada é aplicada aos terminais externos, e a tensão de saída é medida entre um terminal externo e o cursor.

Saída Analógica: A tensão de saída é proporcional à posição do cursor, fornecendo um sinal analógico que pode ser lido por um microcontrolador.



Potenciômetro: Aplicações

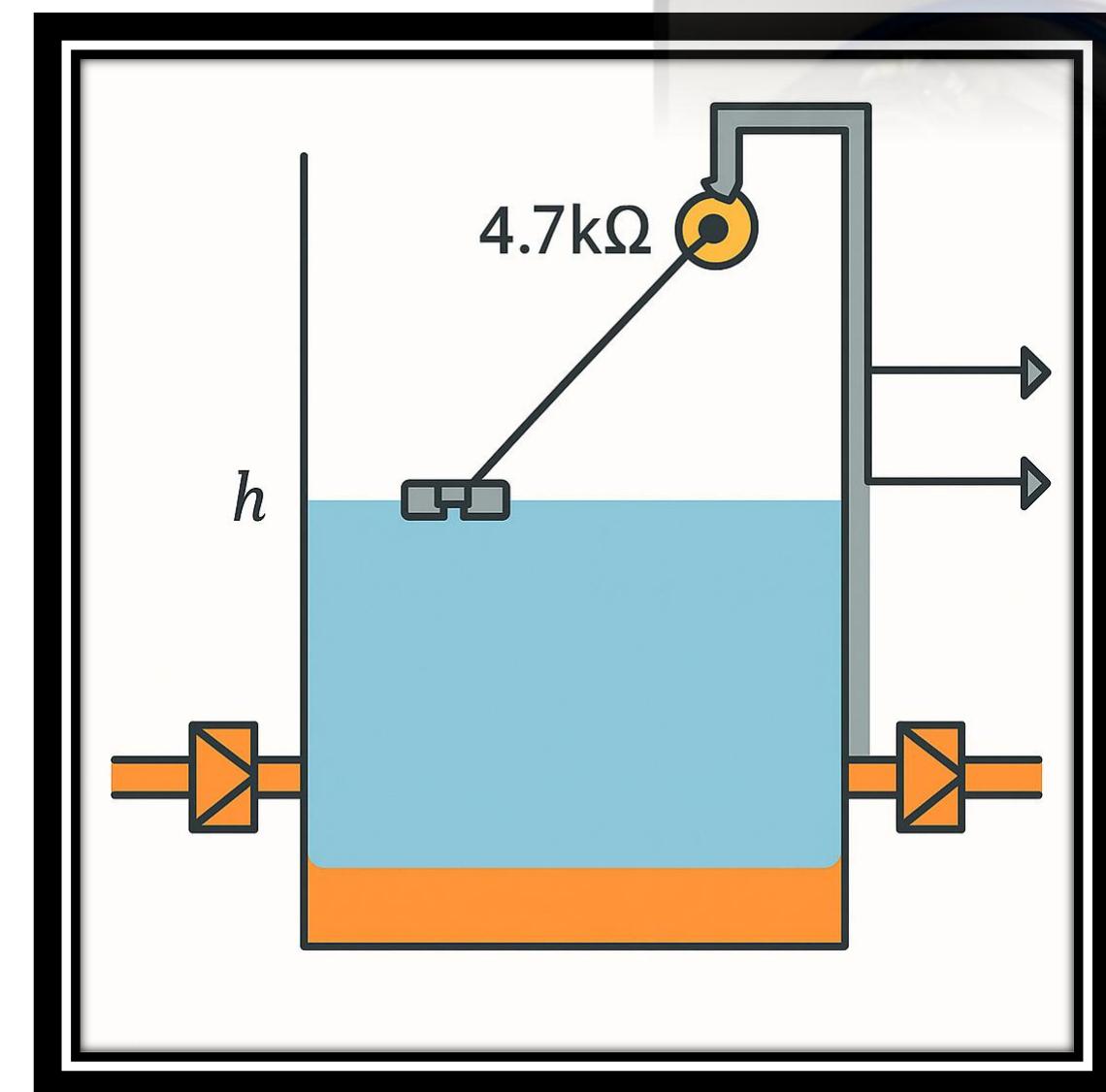
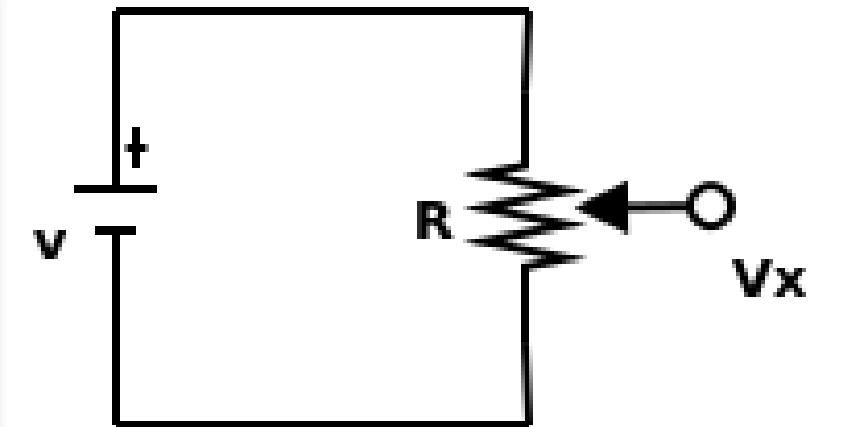
Controle de Volume: Em rádios, amplificadores e sistemas de áudio.

Ajuste de Brilho: Em lâmpadas, LEDs ou telas simples.

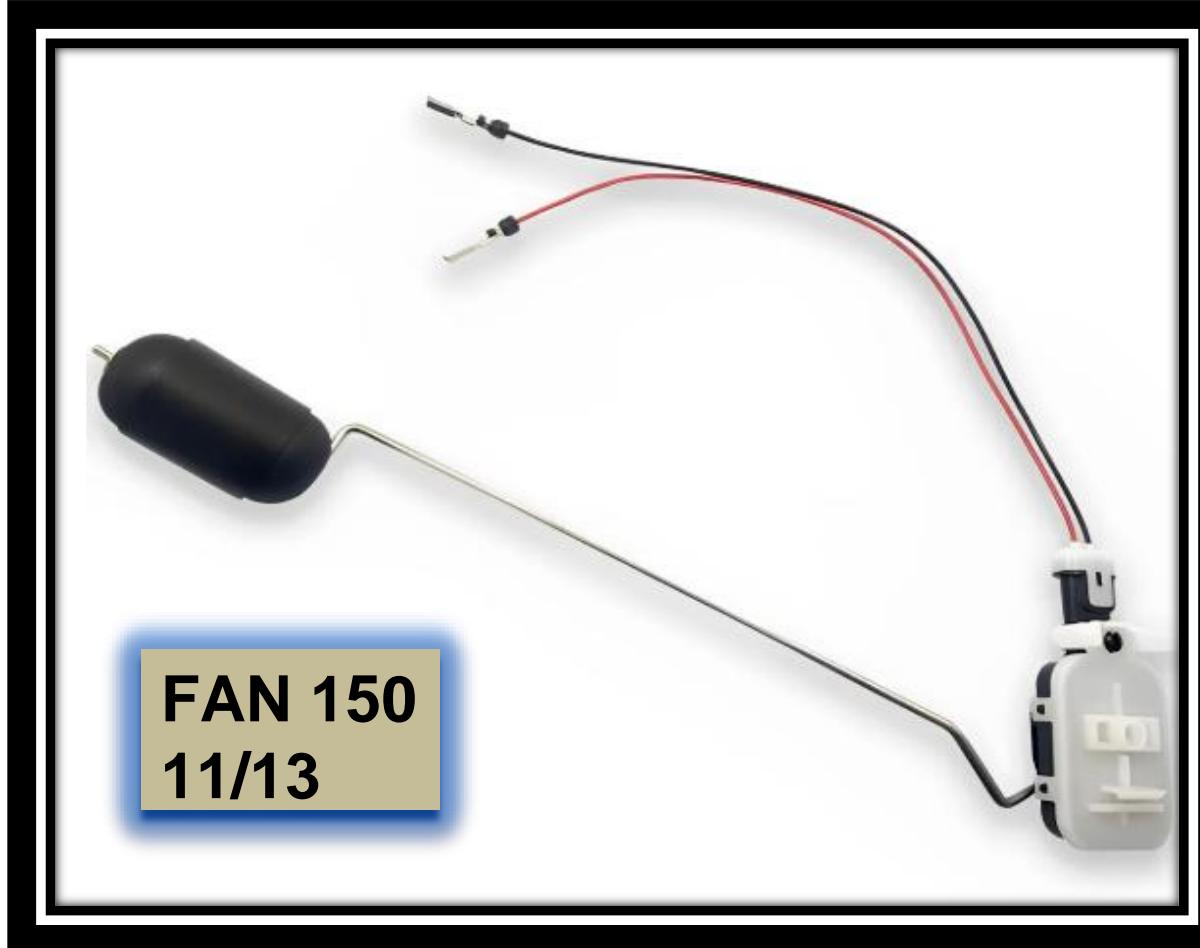
Controle de Velocidade: Em motores DC (através de um driver ou PWM).

Entrada de Usuário: Joysticks, botões giratórios em interfaces de usuário.

Medição de Posição: Em robótica ou sistemas de automação para determinar a posição angular ou linear.



Potenciômetro: Aplicações Boia Sensor de nível para veículos



FAN 150
11/13



CG 125
00/08



Corolla
2018



YBR 125 06/08



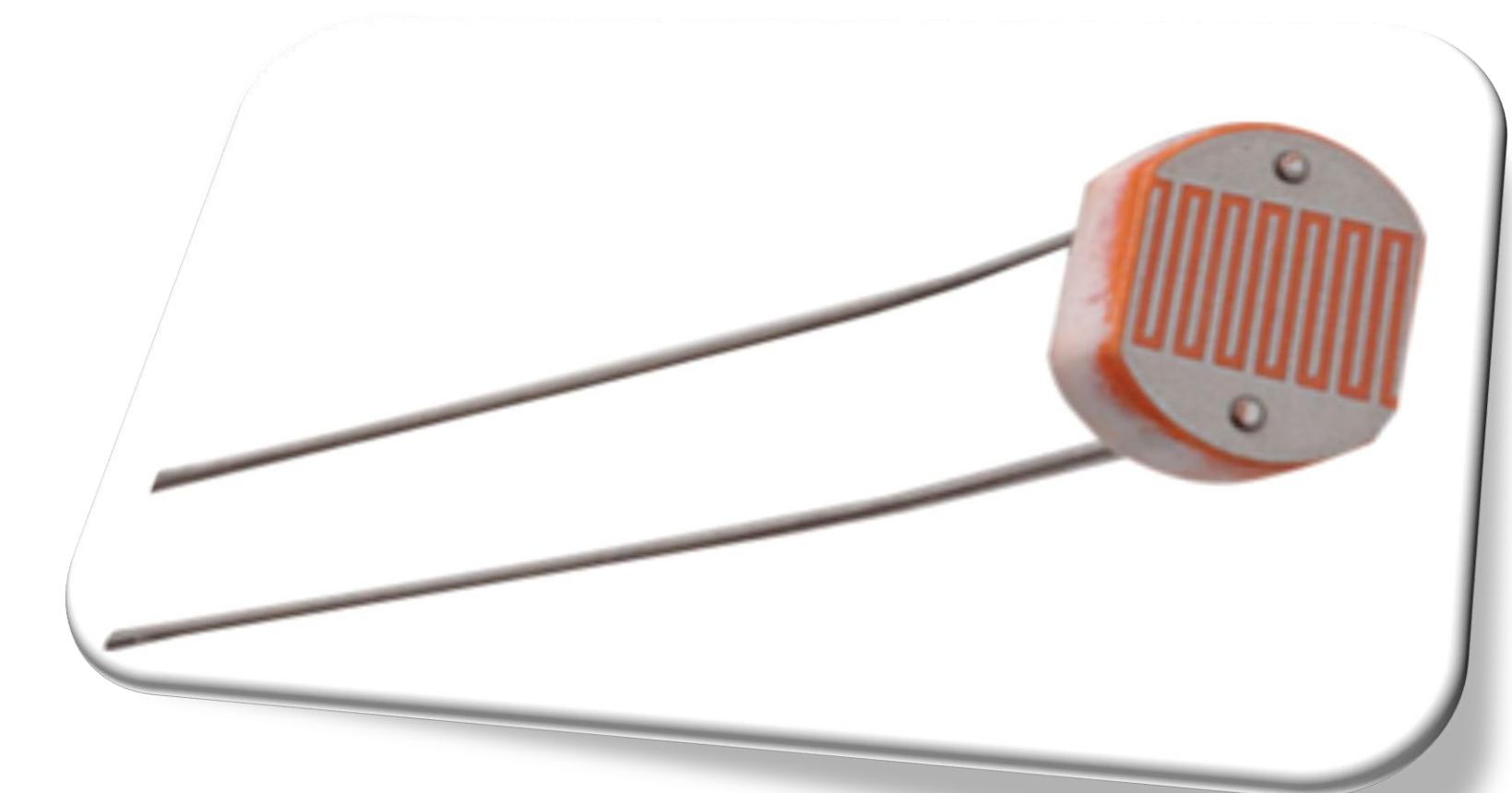
FUSCA
77/96

LDR (Light Dependent Resistor) - O que é?

LDR, ou Resistor Dependente de Luz, é um componente eletrônico cuja resistência elétrica varia **inversamente** com a intensidade da luz que incide sobre ele.

Material: Geralmente feito de sulfeto de cádmio (CdS) ou outros materiais semicondutores.

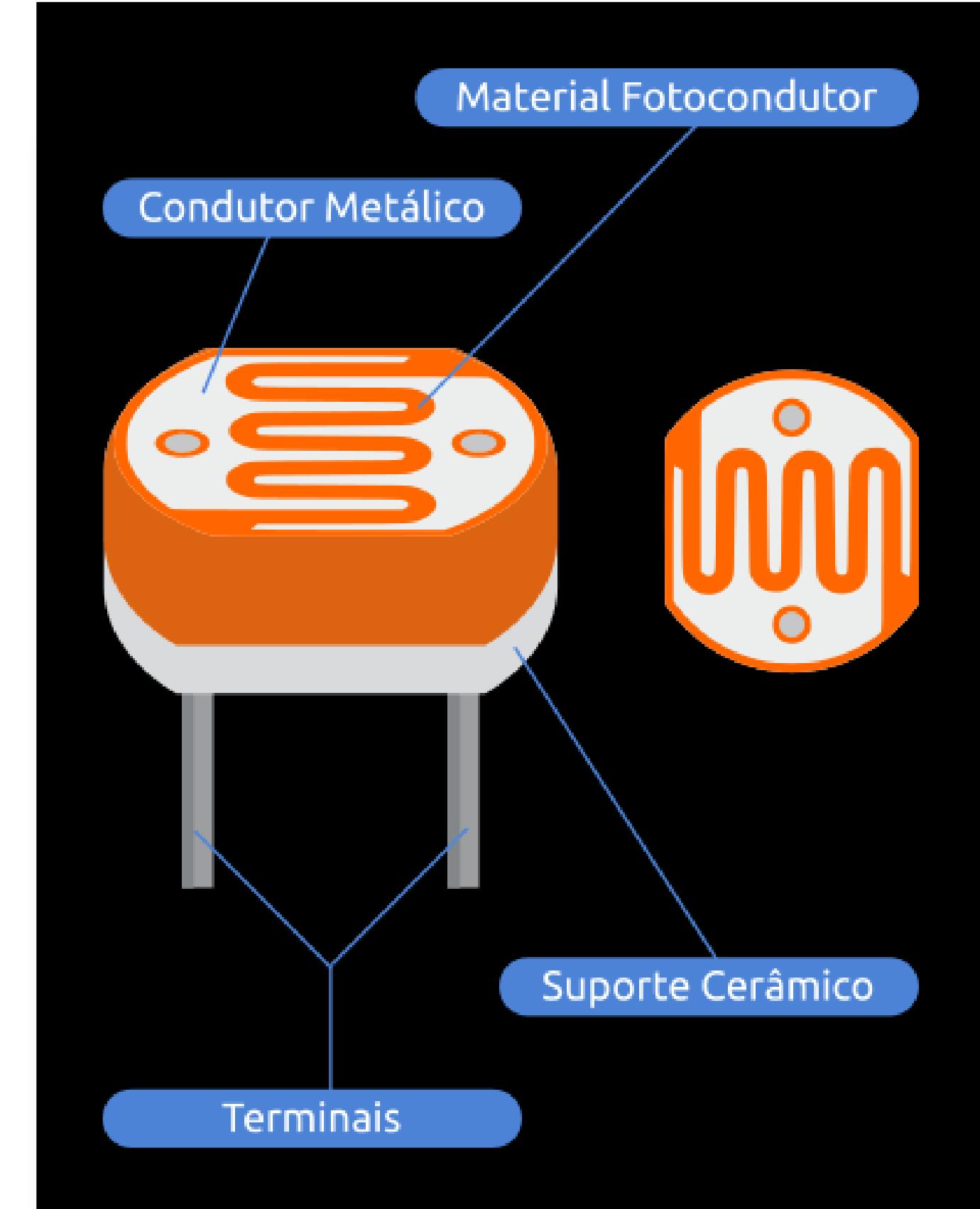
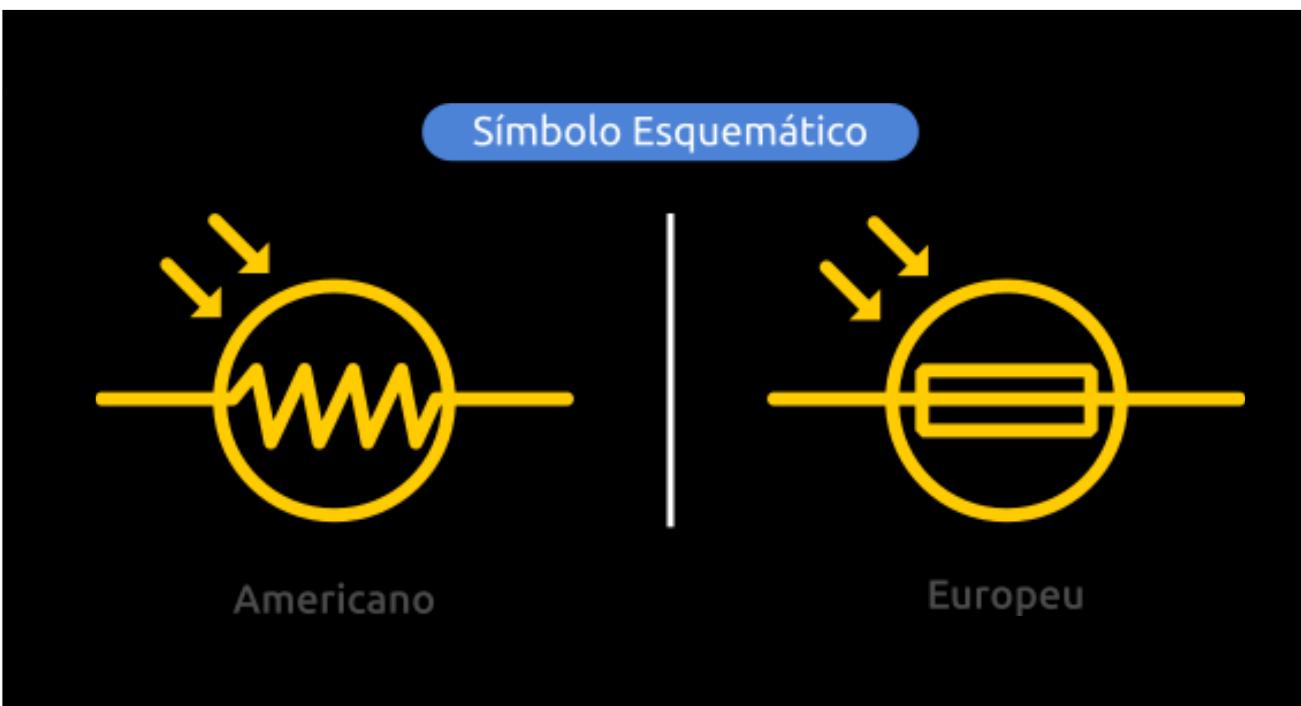
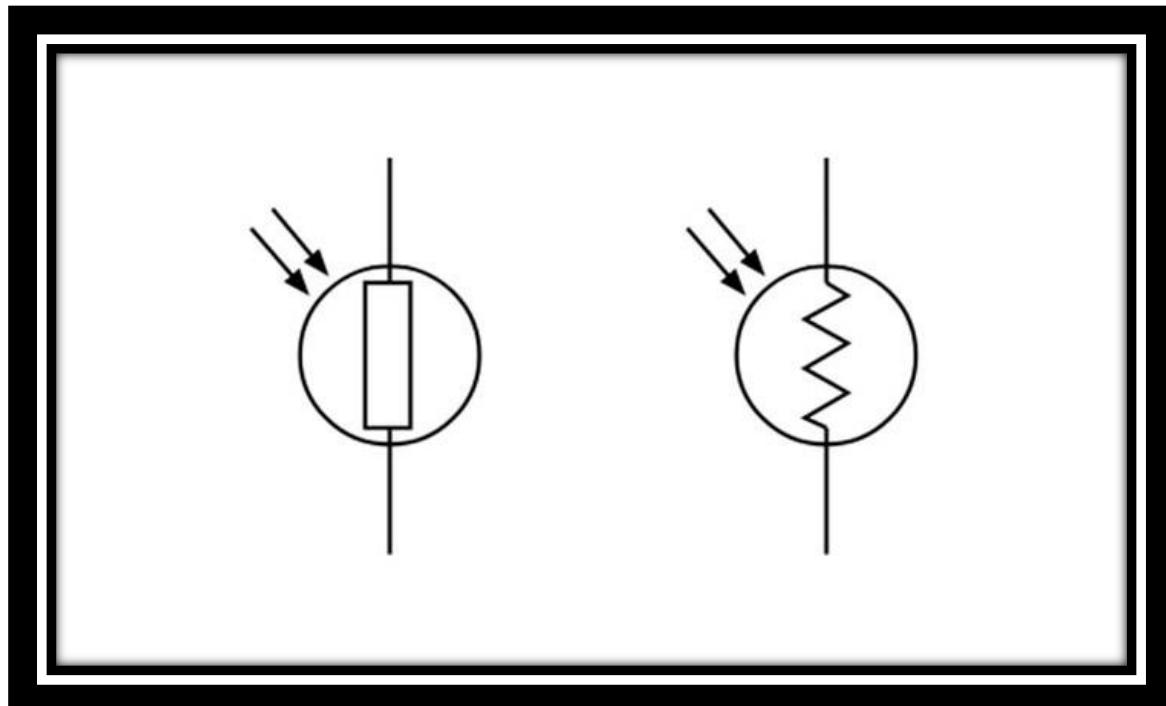
Característica Principal: Quanto maior a intensidade da luz, menor a sua resistência. No escuro, sua resistência é muito alta (**megaohms**); sob luz intensa, pode cair para algumas dezenas ou centenas de ohms.



LDR – Como funciona?

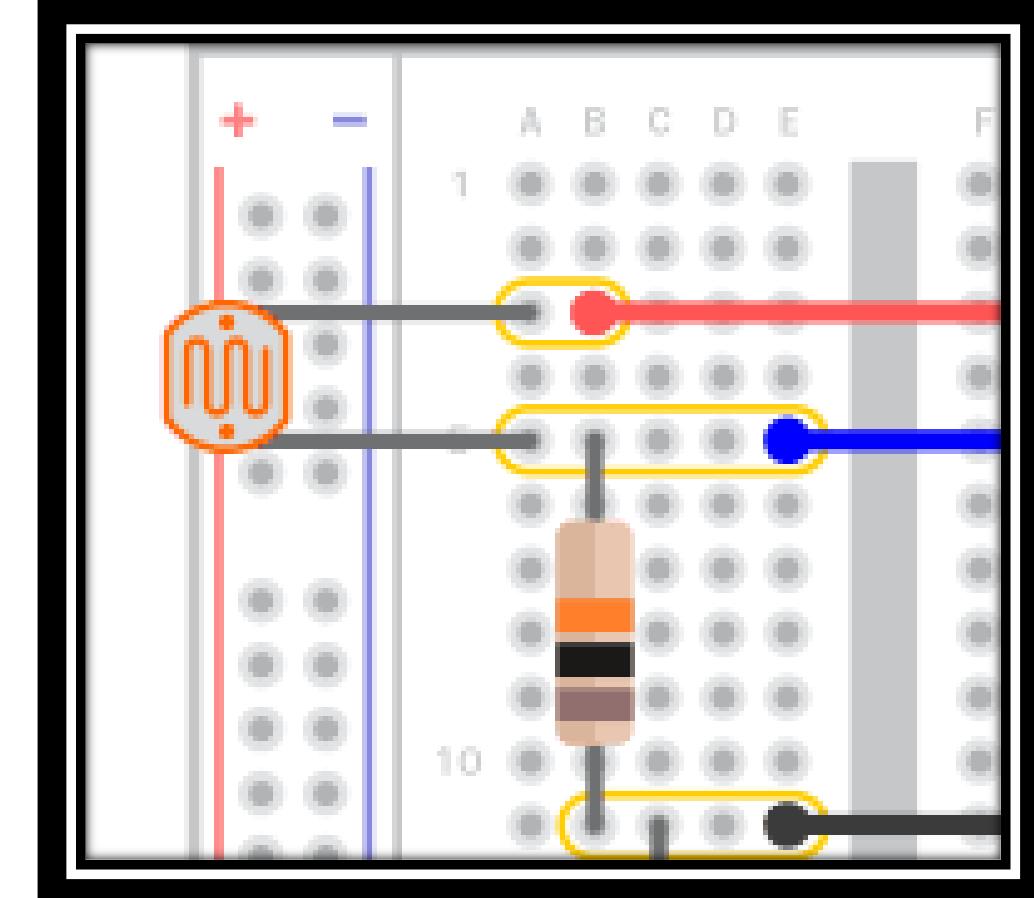
Convertendo Luz em Resistência

A luz incidente libera elétrons no material semicondutor do LDR, aumentando a sua condutividade e, consequentemente, diminuindo sua resistência.

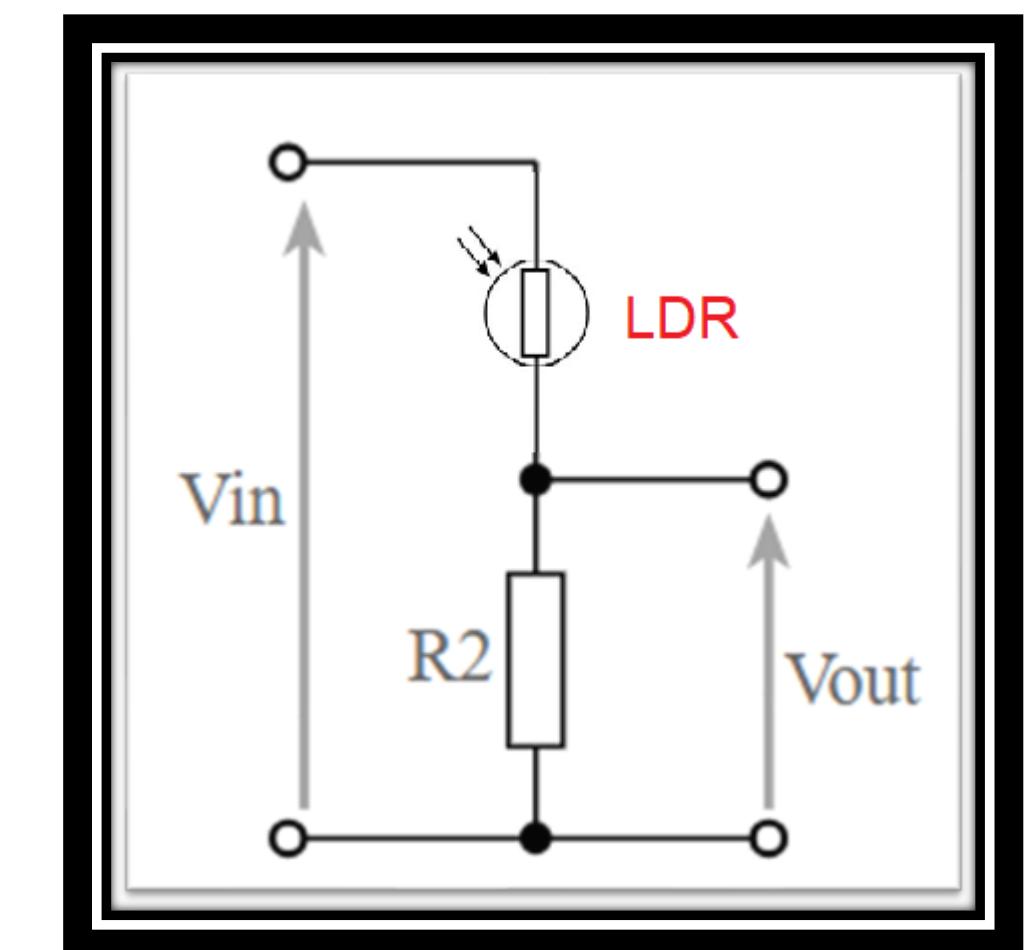


LDR – Como funciona?

Círcito Divisor de Tensão: Para ler o LDR com um microcontrolador, ele é geralmente usado em um círcito divisor de tensão, similar ao potenciômetro. A variação da resistência do LDR causa uma variação na tensão de saída do divisor, que pode ser lida por uma porta analógica.

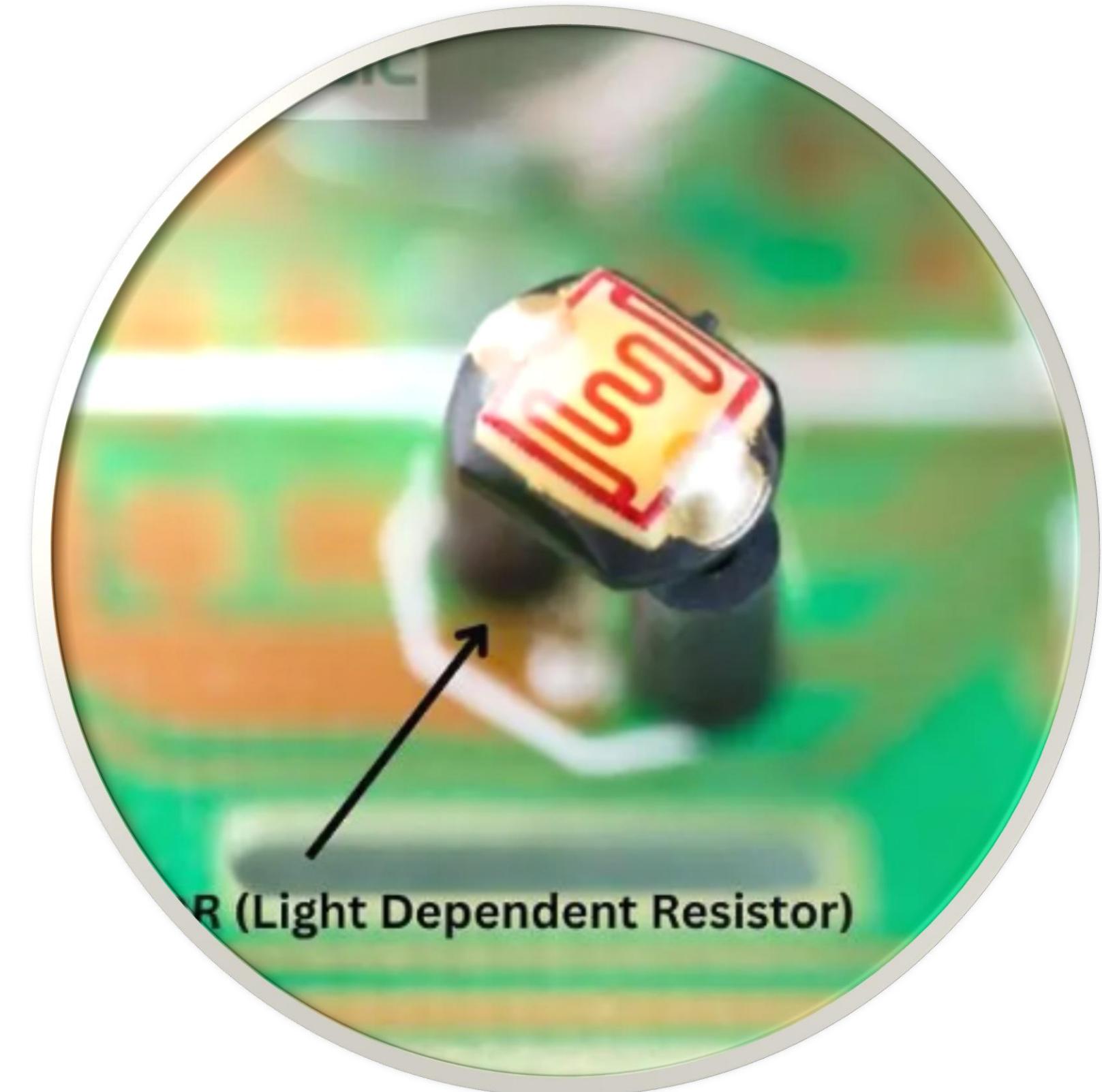


Observação: Uma característica importante do LDR é sua resposta relativamente **lenta** às mudanças de intensidade luminosa.



LDR – Aplicações Práticas do LDR

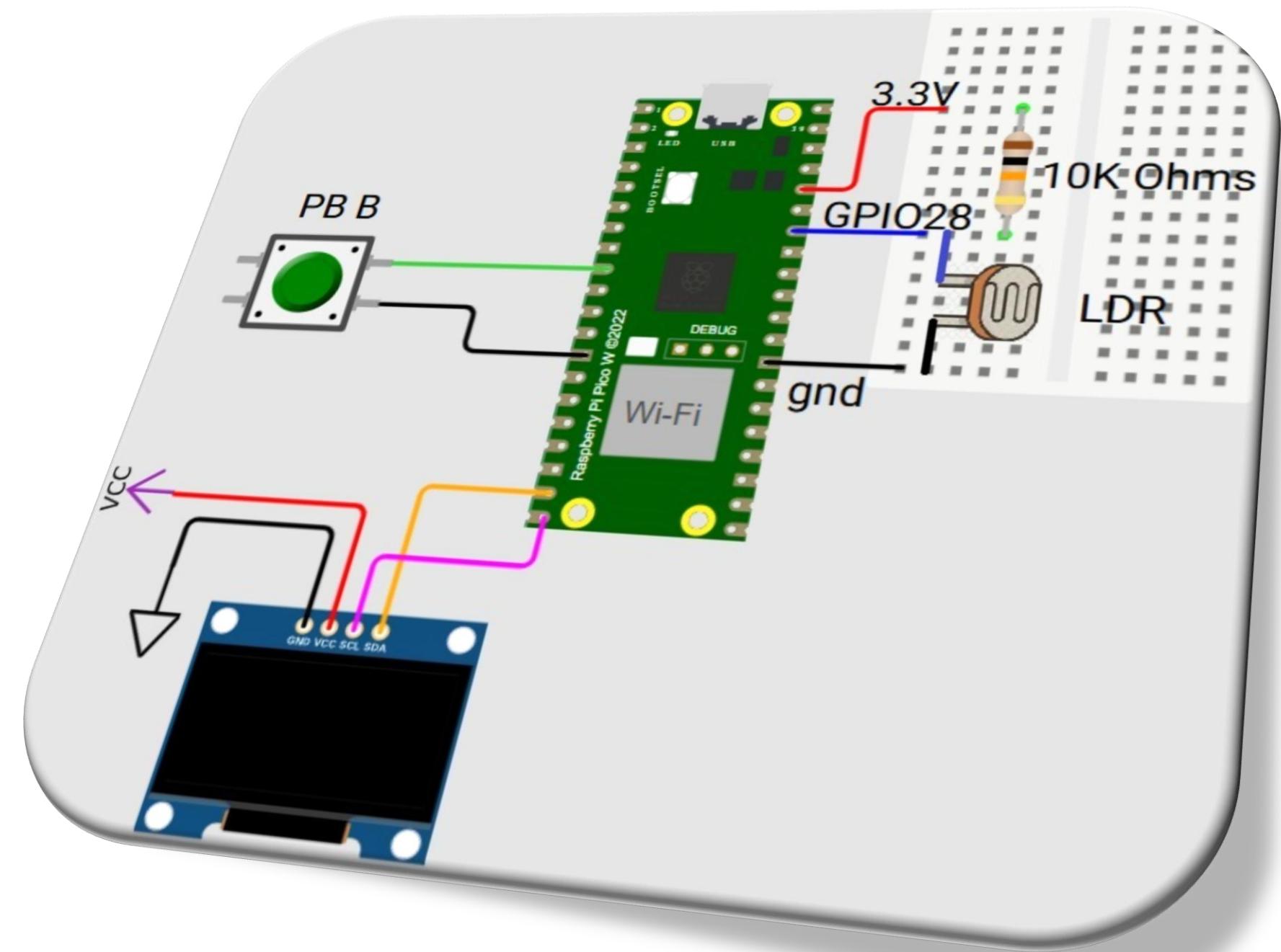
- **Acionamento Automático de Iluminação:** Ligar postes de luz ao anoitecer e desligar ao amanhecer.
- **Medidores de Luz (Fotômetros):** Em câmeras fotográficas ou instrumentos de medição de intensidade luminosa.
- **Alarmes de Segurança:** Detectar a presença de luz em ambientes escuros (ex: abertura de portas/janelas).
- **Controle de Brilho de Telas:** Ajustar automaticamente o brilho de smartphones ou TVs com base na luz ambiente.
- **Brinquedos e Robôs:** Robôs que seguem a luz ou evitam a escuridão.



https://www.pcbasic.com/blog/what_is_ldr.html

Exemplos

Exemplo de programa para uso do LDR.



Monitoramento de Iluminação com ADC no Raspberry Pi Pico W (BitDogLab)

Este projeto demonstra o uso do conversor analógico-digital (ADC) do Raspberry Pi Pico W na placa BitDogLab para monitorar níveis de iluminação ambiente em tempo real. A leitura é exibida graficamente em um display OLED SSD1306 via I2C.

Funcionalidades:

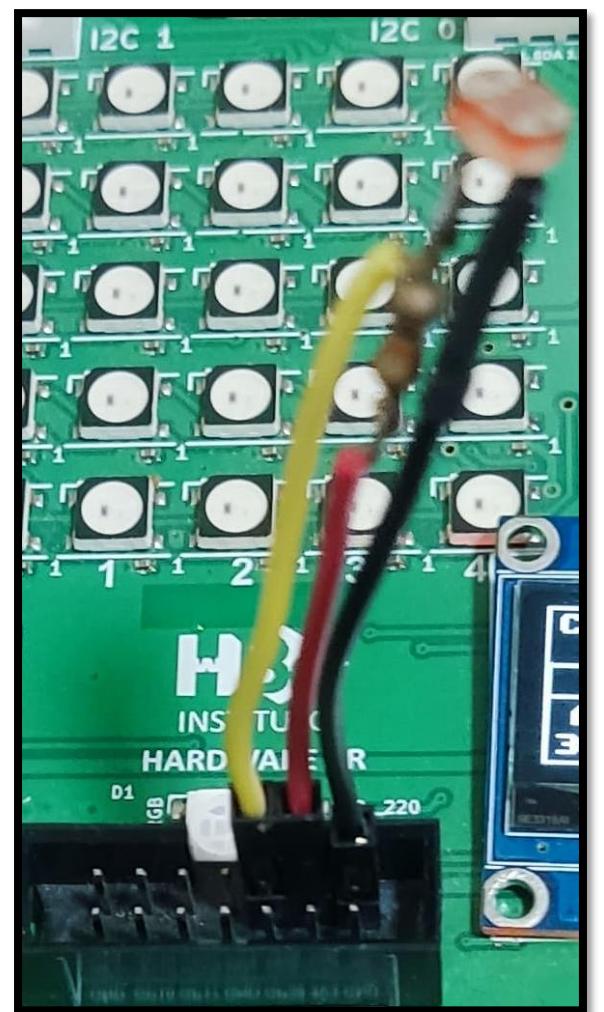
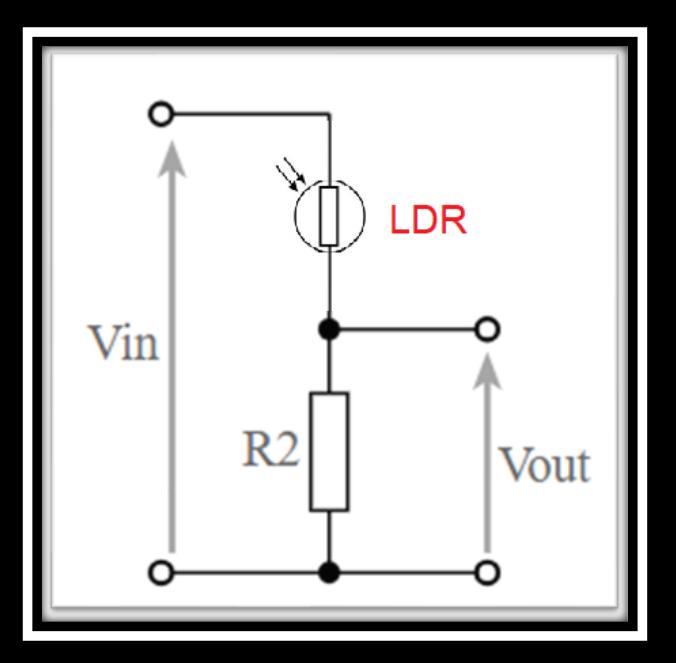
- Leitura da intensidade de luz através do ADC no pino GPIO 28.
- Conversão do valor analógico em porcentagem de luminosidade.
- Exibição das informações no display OLED (SSD1306), com interface gráfica básica.
- Suporte ao modo BOOTSEL via botão físico (GPIO 6) para reinício USB.

Sensores e Atuadores

Exemplo 1: LCD Percentual

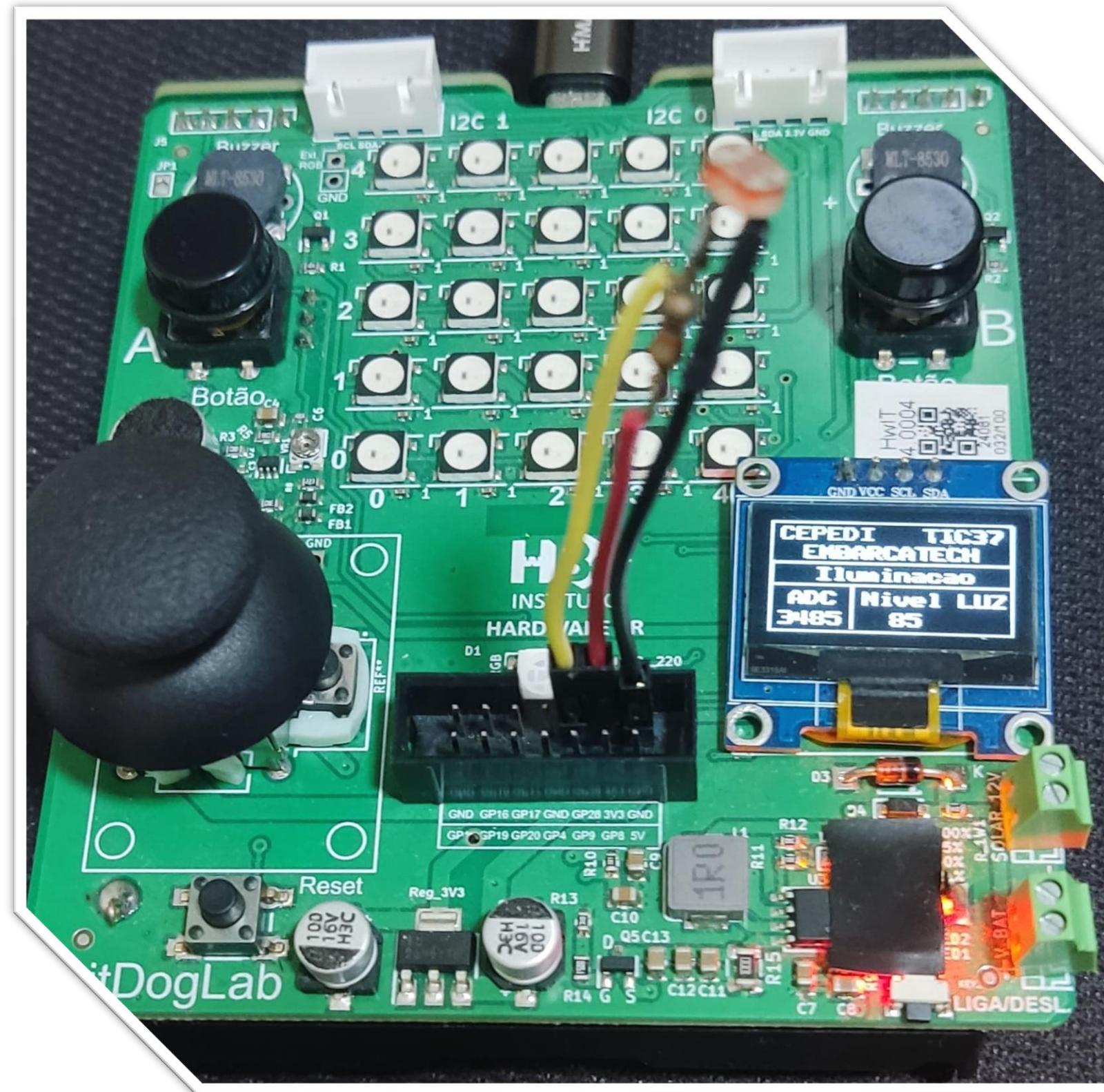
```
c Ilumina01.c x
c Ilumina01.c > botaoB
11
12 #include <stdio.h>
13 #include <stdlib.h>
14 #include "pico/stdlib.h"
15 #include "hardware/adc.h"
16 #include "hardware/i2c.h"
17 #include "lib/ssd1306.h"
18 #include "lib/font.h"
19 #define I2C_PORT i2c1
20 #define I2C_SDA 14
21 #define I2C_SCL 15
22 #define endereco 0x3C
23 #define ADC_PIN 28 // GPIO compartilhada com o microf
24
25
26 //Trecho para modo BOOTSEL com botão B
27 #include "pico/bootrom.h"
28 #define botaoB 6
29 void gpio_irq_handler(uint gpio, uint32_t events)
30 {
31     reset_usb_boot(0, 0);
32 }
33
34 int main()
35 {
36     // Para ser utilizado o modo BOOTSEL com botão B
37     gpio_init(botaoB);
38     gpio_set_dir(botaoB, GPIO_IN);
39     gpio_pull_up(botaoB);
40     gpio_set_irq_enabled_with_callback(botaoB, GPIO_IRQ_
41     //Aqui termina o trecho para modo BOOTSEL com botão
42 }
```

```
c Ilumina01.c x
c Ilumina01.c > botaoB
35 {
36     // I2C Initialisation. Using it at 400Khz.
37     i2c_init(I2C_PORT, 400 * 1000);
38
39     gpio_set_function(I2C_SDA, GPIO_FUNC_I2C); // Set the
40     gpio_set_function(I2C_SCL, GPIO_FUNC_I2C); // Set the
41     gpio_pull_up(I2C_SDA); // Pull up the data line
42     gpio_pull_up(I2C_SCL); // Pull up the clock line
43     ssd1306_t ssd; // Inicializa a estrutura do display
44     ssd1306_init(&ssd, WIDTH, HEIGHT, false, endereco, I
45     ssd1306_config(&ssd); // Configura o display
46     ssd1306_send_data(&ssd); // Envia os dados para o di
47
48
49     // Limpa o display. O display inicia com todos os pi
50     ssd1306_fill(&ssd, false);
51     ssd1306_send_data(&ssd);
52
53     adc_init();
54     adc_gpio_init(ADC_PIN); // GPIO 28 como entrada anal
55
56
57     uint16_t adc_value_x;
58     float ldr_pciento;
59     char str_x[5]; // Buffer para armazenar a string
60     char str_y[10]; // Buffer para armazenar a string
61
62     bool cor = true;
63     while (true)
64     {
```



Sensores e Atuadores

Exemplo 1: LCD Percentual

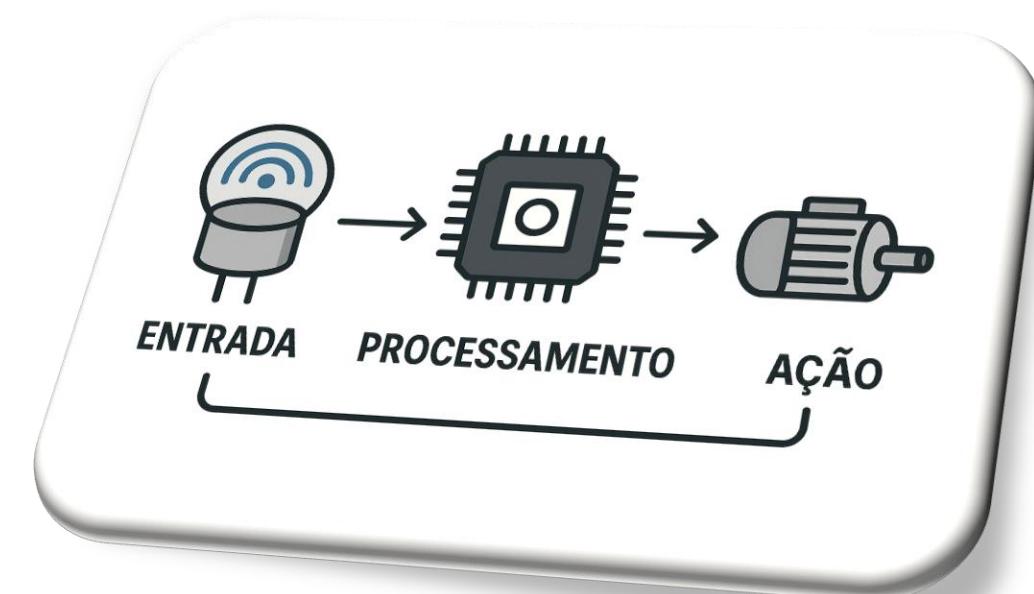


```
C Ilumina01.c X
C Ilumina01.c > botao8
35 {
66   char str_y[10]; // Buffer para armazenar a string
67
68   bool cor = true;
69   while (true)
70   {
71     adc_select_input(2); // O pino 28 como entrada analógica
72     adc_value_x = adc_read(); // Faz leitura do LRD
73     ldr_pcento = (adc_value_x * 100) / 4095; // Converte o valor lido para porcentagem
74
75     sprintf(str_x, "%d", adc_value_x); // Converte o inteiro para string
76     sprintf(str_y, "%1.0f", 100-ldr_pcento); // Converte o float para string
77
78     //cor = !cor;
79     // Atualiza o conteúdo do display com animações
80     ssd1306_fill(&ssd, !cor); // Limpa o display
81     ssd1306_rect(&ssd, 3, 3, 122, 60, cor, !cor); // Desenha um retângulo
82     ssd1306_line(&ssd, 3, 25, 123, 25, cor); // Desenha uma linha
83     ssd1306_line(&ssd, 3, 37, 123, 37, cor); // Desenha uma linha
84     ssd1306_draw_string(&ssd, "CEPEDI TIC37", 8, 6); // Desenha uma string
85     ssd1306_draw_string(&ssd, "EMBARCATECH", 20, 16); // Desenha uma string
86     ssd1306_draw_string(&ssd, " Iluminacao", 10, 28); // Desenha uma string
87     ssd1306_draw_string(&ssd, "ADC", 13, 41); // Desenha uma string
88     ssd1306_draw_string(&ssd, "Nivel LUZ", 50, 41); // Desenha uma string
89     ssd1306_line(&ssd, 44, 37, 44, 60, cor); // Desenha uma linha
90     ssd1306_draw_string(&ssd, str_x, 8, 52); // Desenha uma string
91     ssd1306_draw_string(&ssd, str_y, 75, 52); // Desenha uma string
92     ssd1306_send_data(&ssd); // Atualiza o display
93     sleep_ms(700);
94   }
95 }
```

O que é um atuador?

Um atuador é um dispositivo que converte uma forma de energia (elétrica, hidráulica, pneumática, térmica, etc.) em movimento mecânico ou força. Em outras palavras, ele é o componente que transforma um sinal de controle em uma ação física.

Se os sensores são os "**olhos**" e "**ouvidos**" de um sistema, os atuadores são os "**músculos**" e "**mãos**", responsáveis por realizar a ação física com base nos sinais recebidos.



O que é um atuador elétrico?

Atuadores Elétricos convertem energia elétrica em energia mecânica.

Podem ser:

Lineares: Convertem o movimento rotativo de um motor em movimento linear (empurrar, puxar, levantar, abaixar).

Ex: um fuso de esferas ou correia dentada acionada por um motor.

Rotativos: Produzem movimento de rotação contínua ou para um ângulo específico.

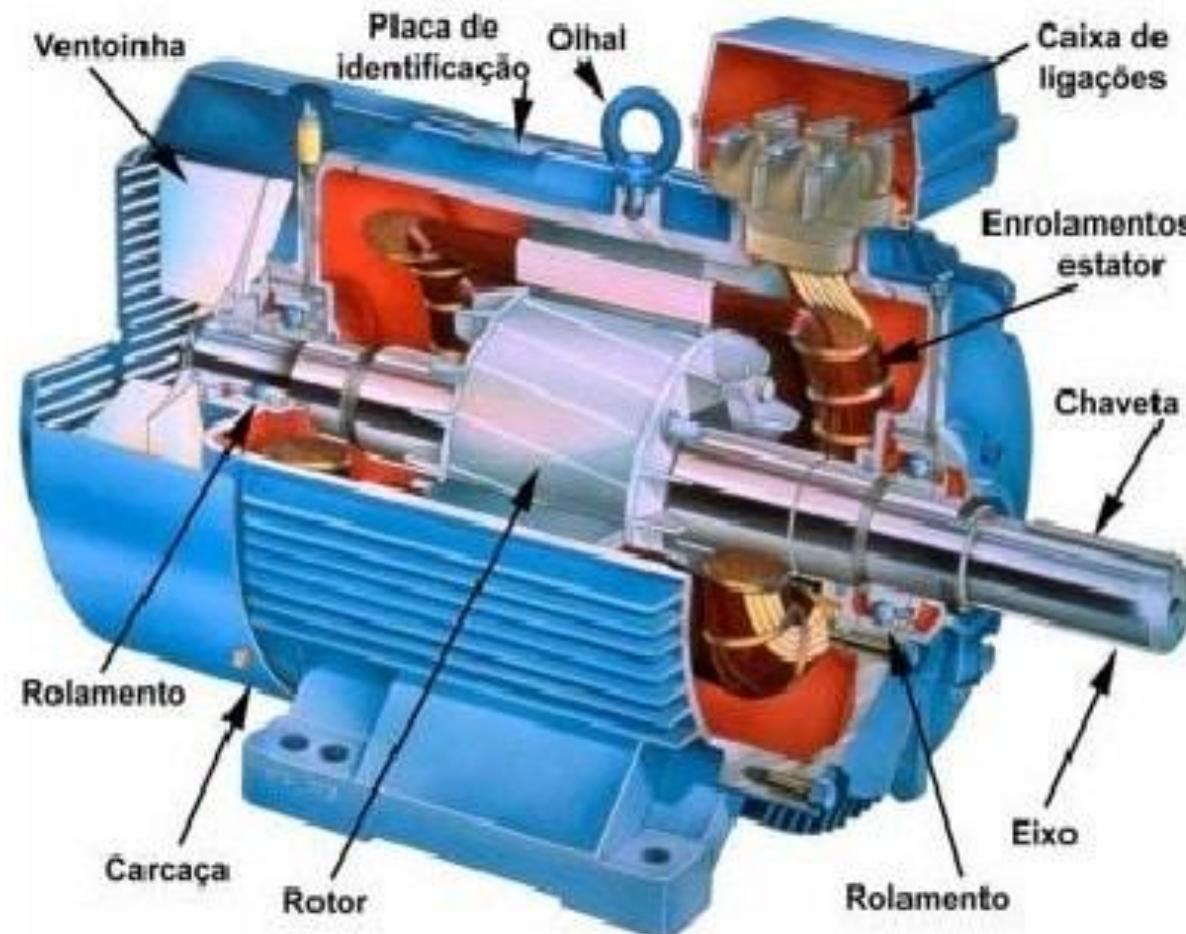
Ex: motores elétricos simples, servos, motores de passo.

Os motores elétricos são os **atuadores** mais onipresentes em nosso cotidiano e na indústria.

Um **motor elétrico** é uma máquina que transforma energia elétrica em energia mecânica de rotação (ou, em alguns casos, linear). Ele faz isso através da interação entre campos magnéticos e correntes elétricas, um princípio conhecido como eletromagnetismo.

O funcionamento de um motor elétrico baseia-se em um princípio fundamental da física: quando um condutor percorrido por uma corrente elétrica é colocado em um campo magnético, ele sofre a ação de uma força.

Motores (Corrente contínua e alternada, passo, servo motores).



<https://www.abecom.com.br/tipos-de-motor-eletroico/>



Motores elétricos: Observações.

Acionar motores diretamente dos pinos do **Raspberry Pi Pico** é uma **má ideia** e pode danificar a placa. Isso ocorre por três motivos principais:

- **Corrente**: Motores, mesmo os pequenos, demandam mais corrente do que um pino GPIO pode fornecer.
- **Voltagem**: A maioria dos motores opera com voltagens maiores do que os 3.3V dos pinos do Pico (5V, 9V, 12V, etc.) .
- **Indutância e Força Contra-Eletromotriz (Back-EMF)**: Motores são cargas indutivas. Quando a energia é desligada, a indutância do motor gera um pico de voltagem (força contra-eletromotriz) que pode ser muito maior do que a voltagem de alimentação. Se esse pico for direcionado para o microcontrolador, ele pode facilmente destruí-lo.

Motores elétricos: Drivers

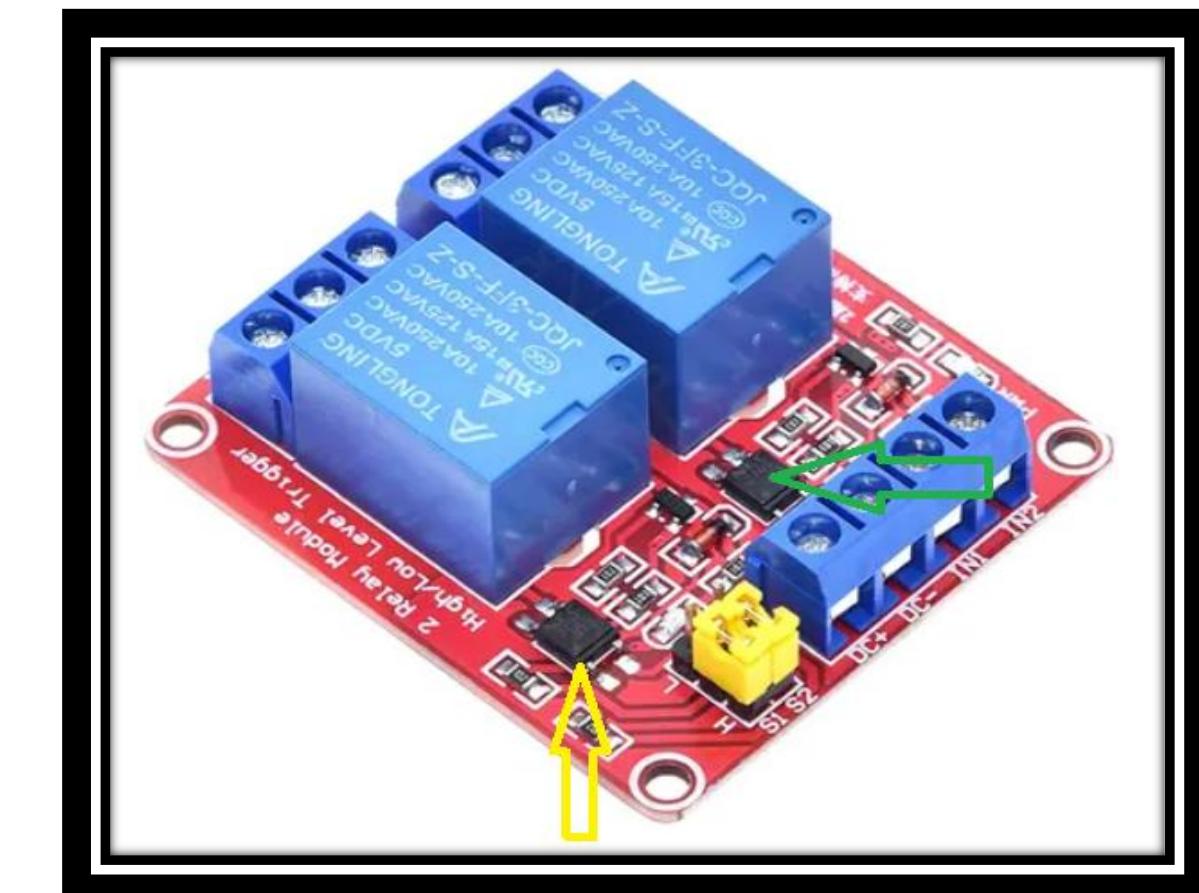
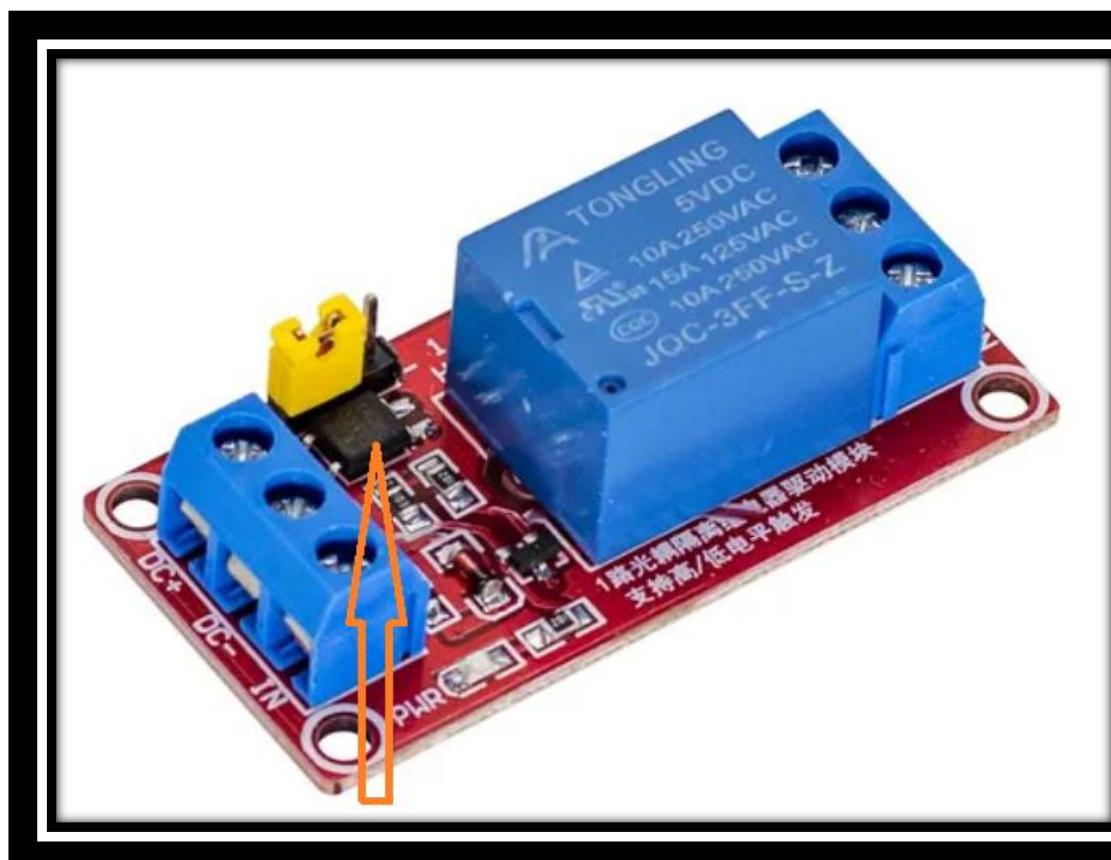
A escolha da interface (driver) dependerá do tipo de motor que você quer acionar:

- **Motores DC** – Ponte H, L298N, DRV8833, etc.
- **ServoMotores** – Sinal PWM normalmente a 50Hz e pulso de 1 a 2ms.
- **Motores de Passo** – ULN2003, A4988, DRV8825, etc.
- **Motores Brushless** – ESC (Electronic Speed Controller).

Relé Eletromecânico

Um **relé** é, em sua essência, um **interruptor eletromecânico**. Isso significa que ele usa uma pequena corrente elétrica para controlar um interruptor que, por sua vez, pode ligar ou desligar um circuito de muito maior potência.

O **relé** é um componente eletromecânico, ou seja, ele conta com uma parte mecânica de contato e o acionamento ocorre através da corrente elétrica em uma bobina.



Relé Componentes:

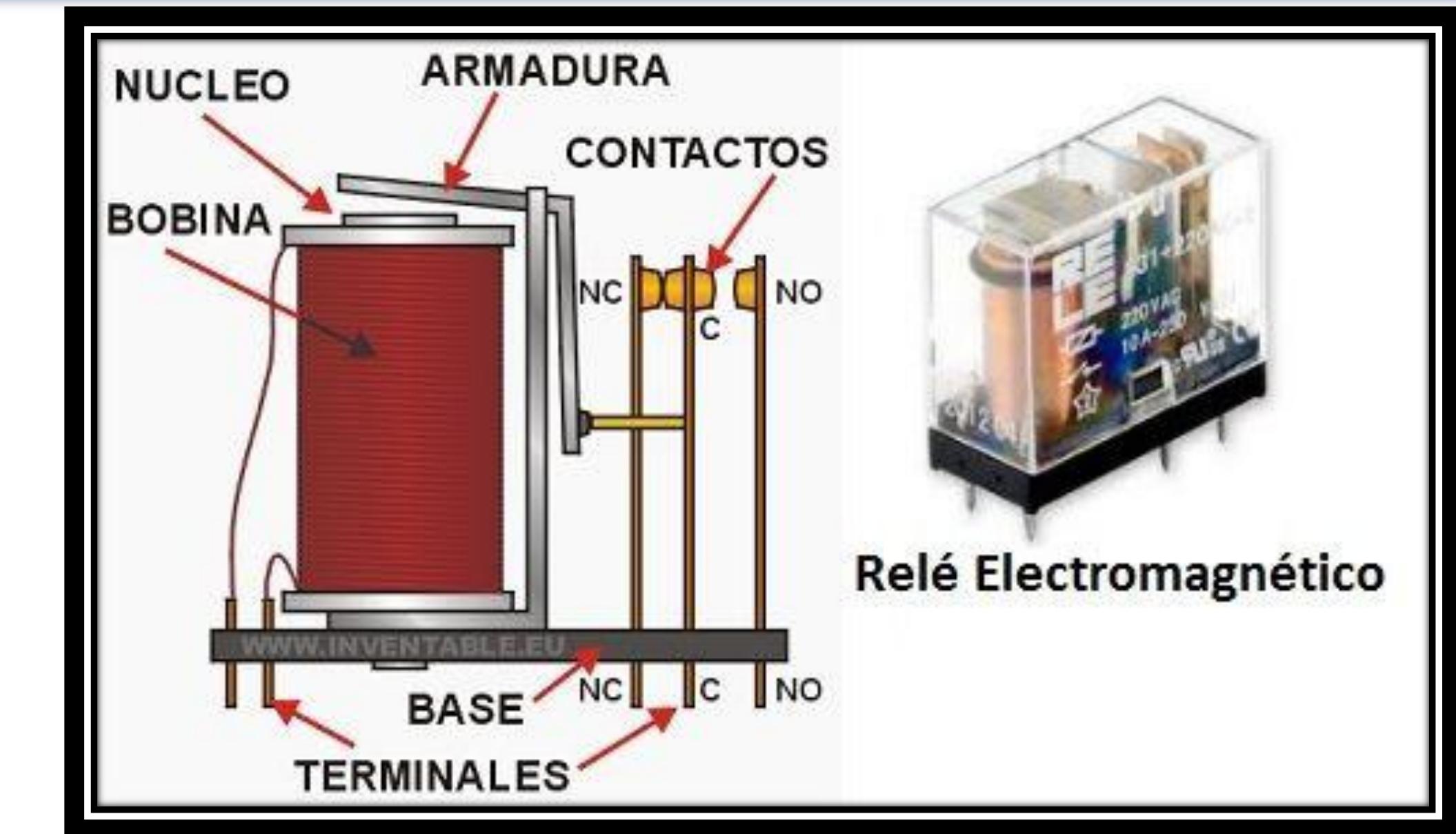
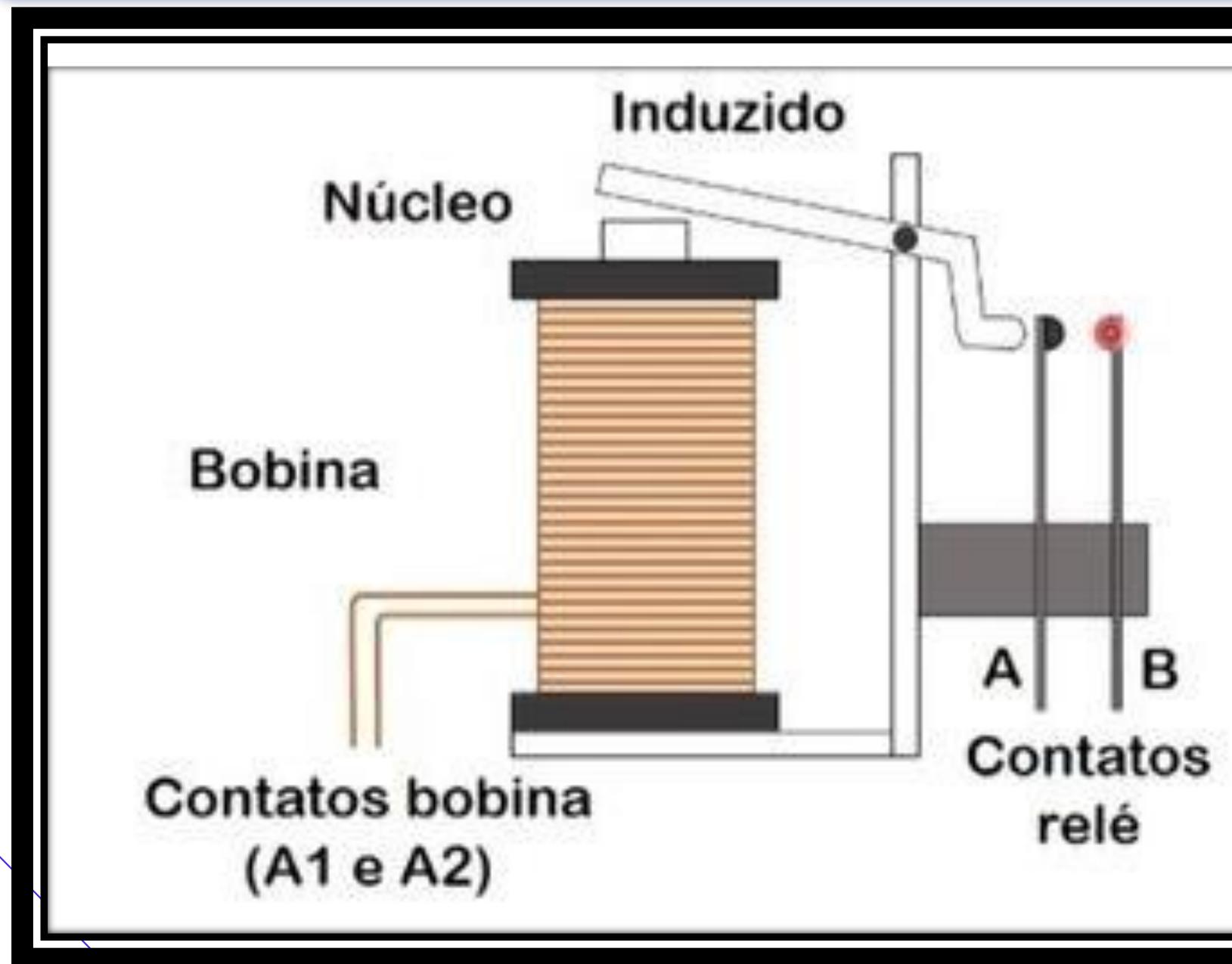
Bobina: Fio enrolado que, ao ser energizado por uma corrente de controle, gera um campo magnético.

Induzido (Armadura): Peça metálica móvel que é atraída pelo campo magnético da bobina, acionando os contatos.

Núcleo: Material ferromagnético central da bobina que intensifica o campo magnético.

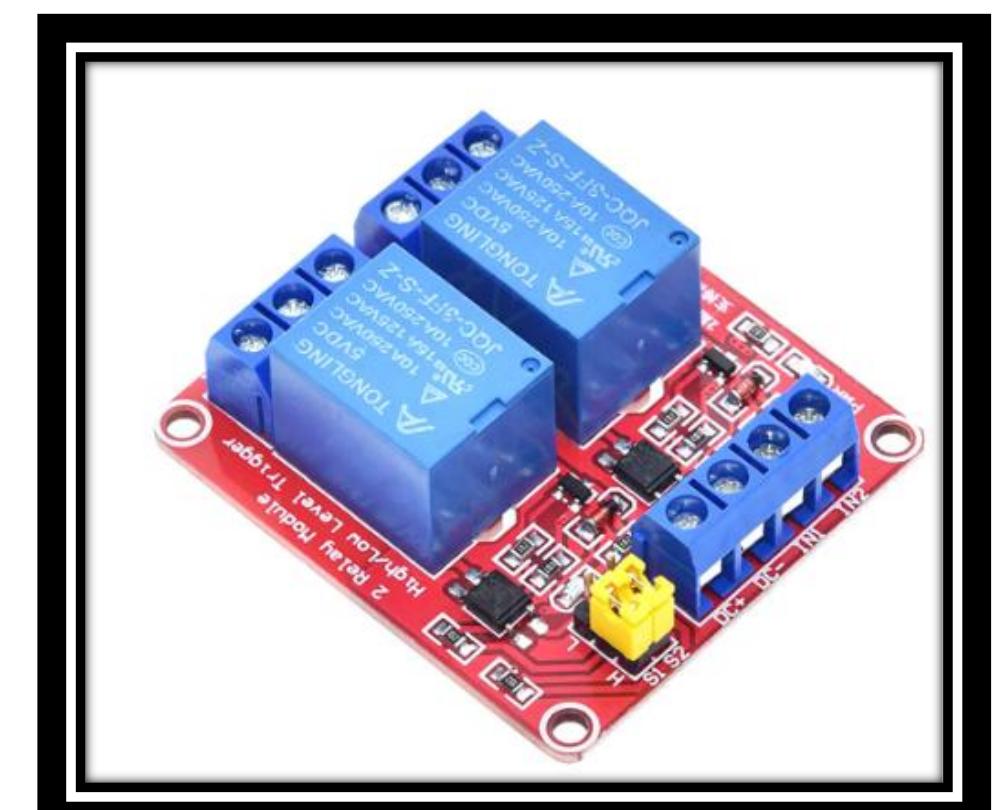
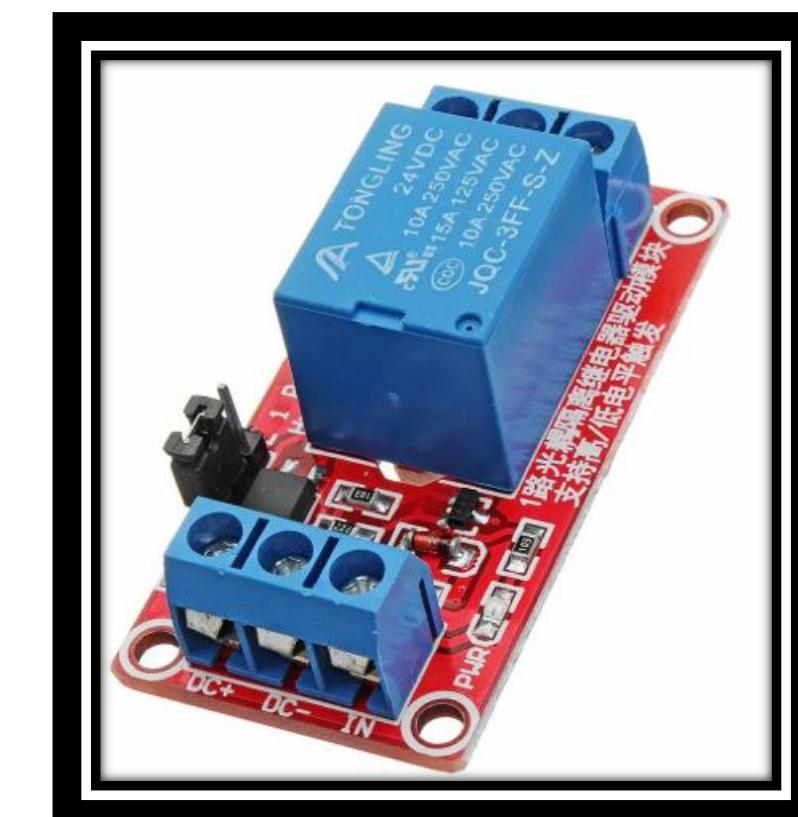
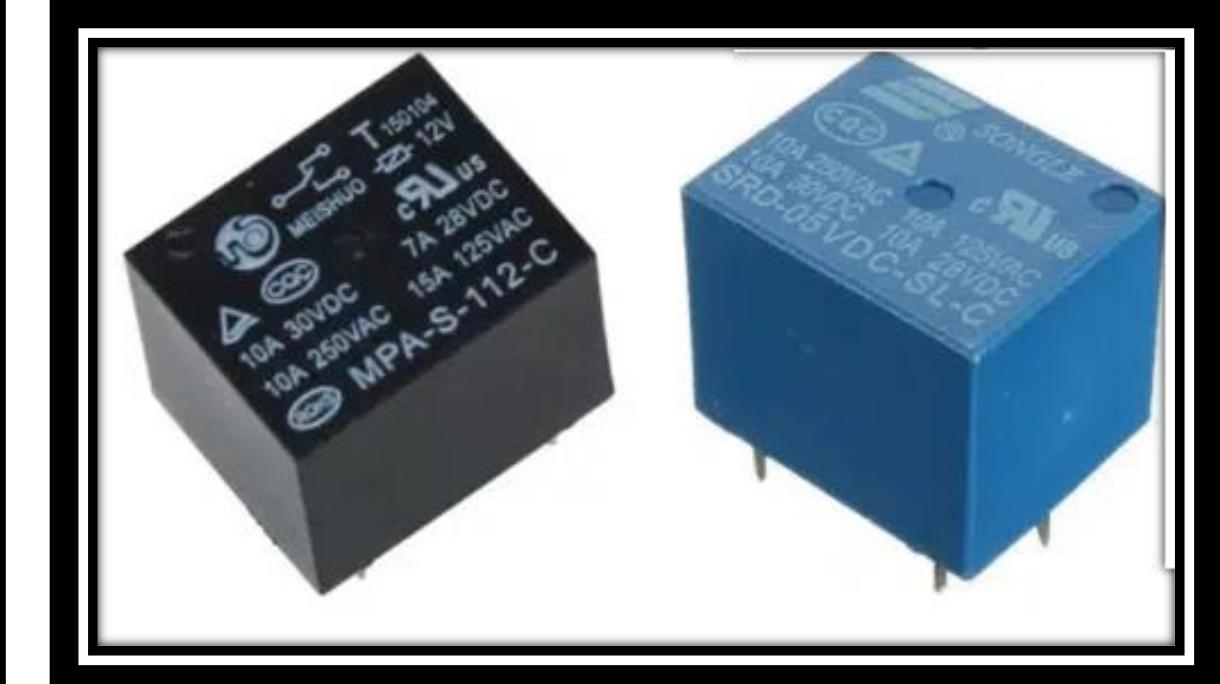
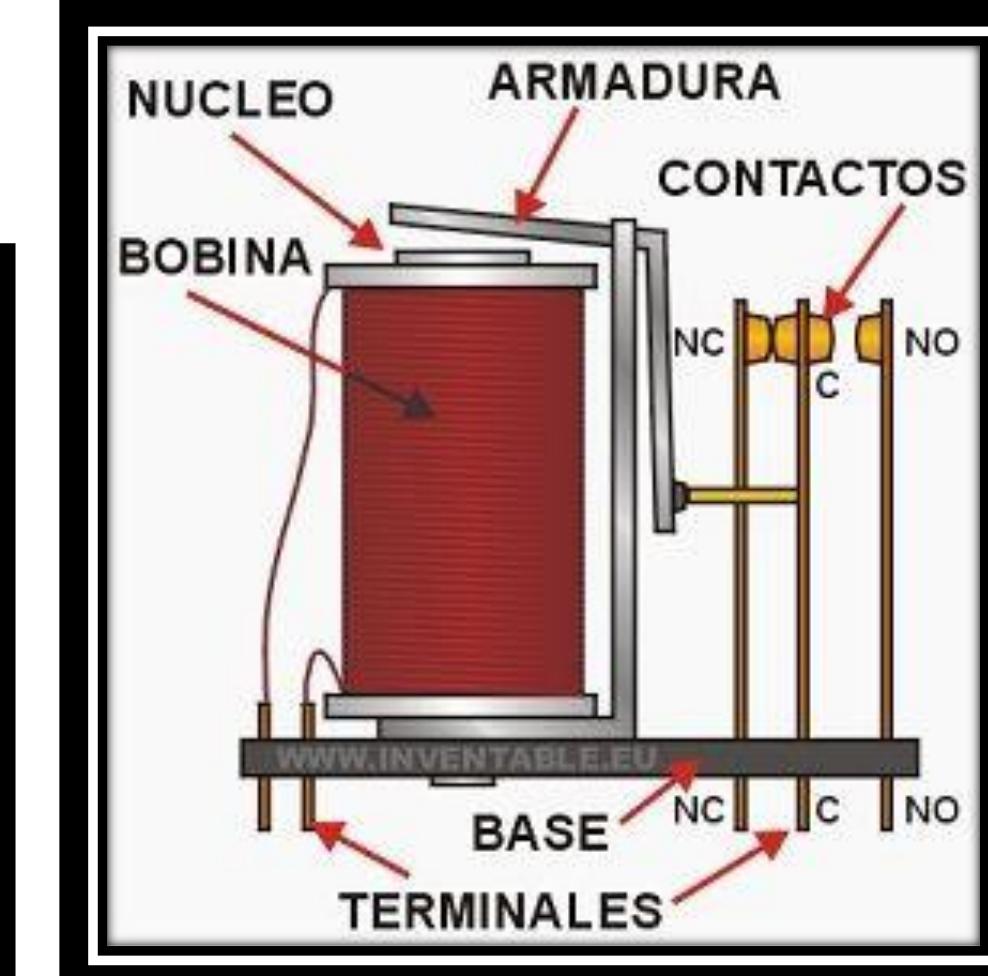
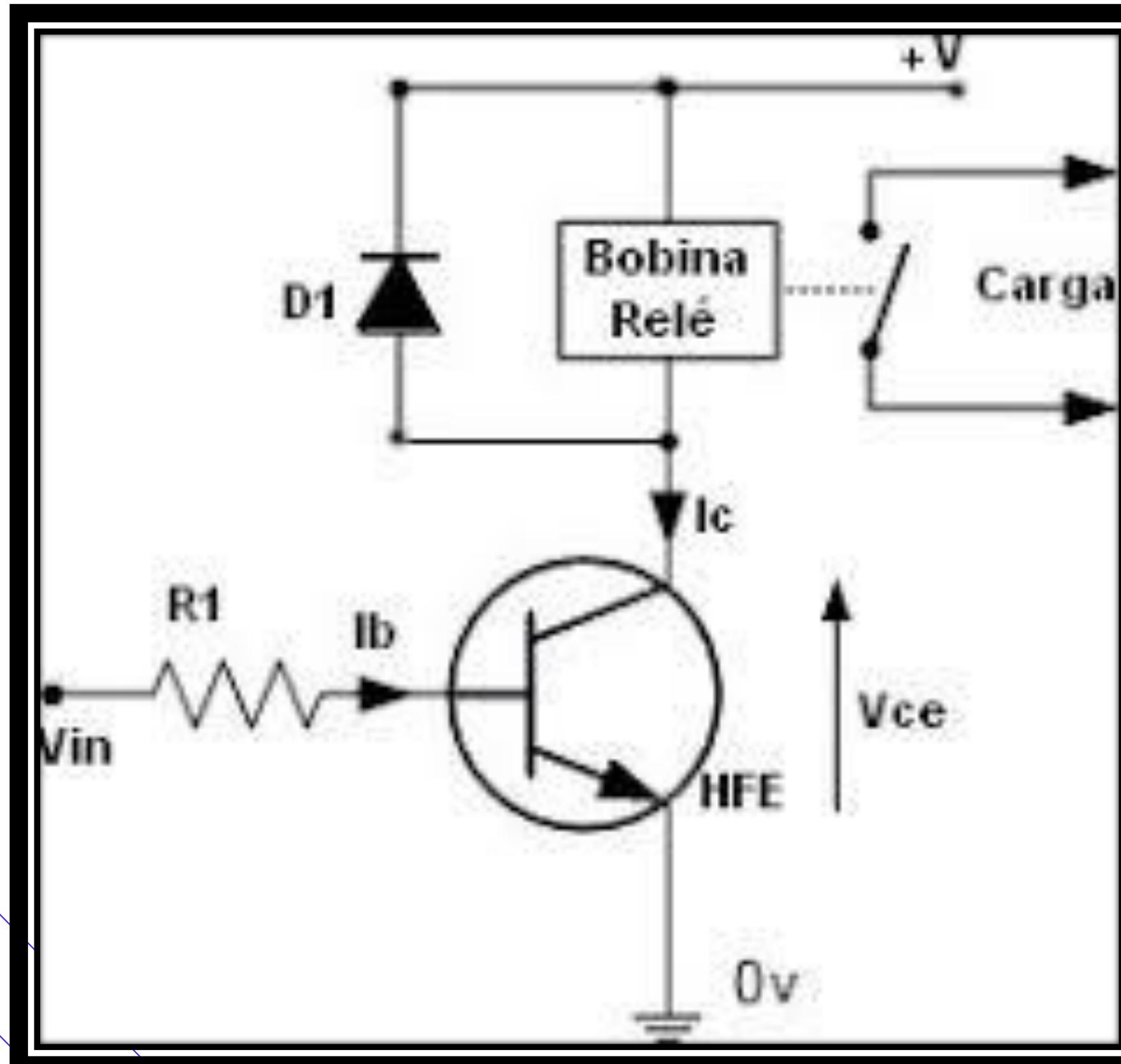
Contatos da Bobina: Terminais onde se conecta a tensão para energizar a bobina.

Contatos do Relé: Os interruptores físicos (NA/NF/COM) que abrem ou fecham o circuito de alta potência.



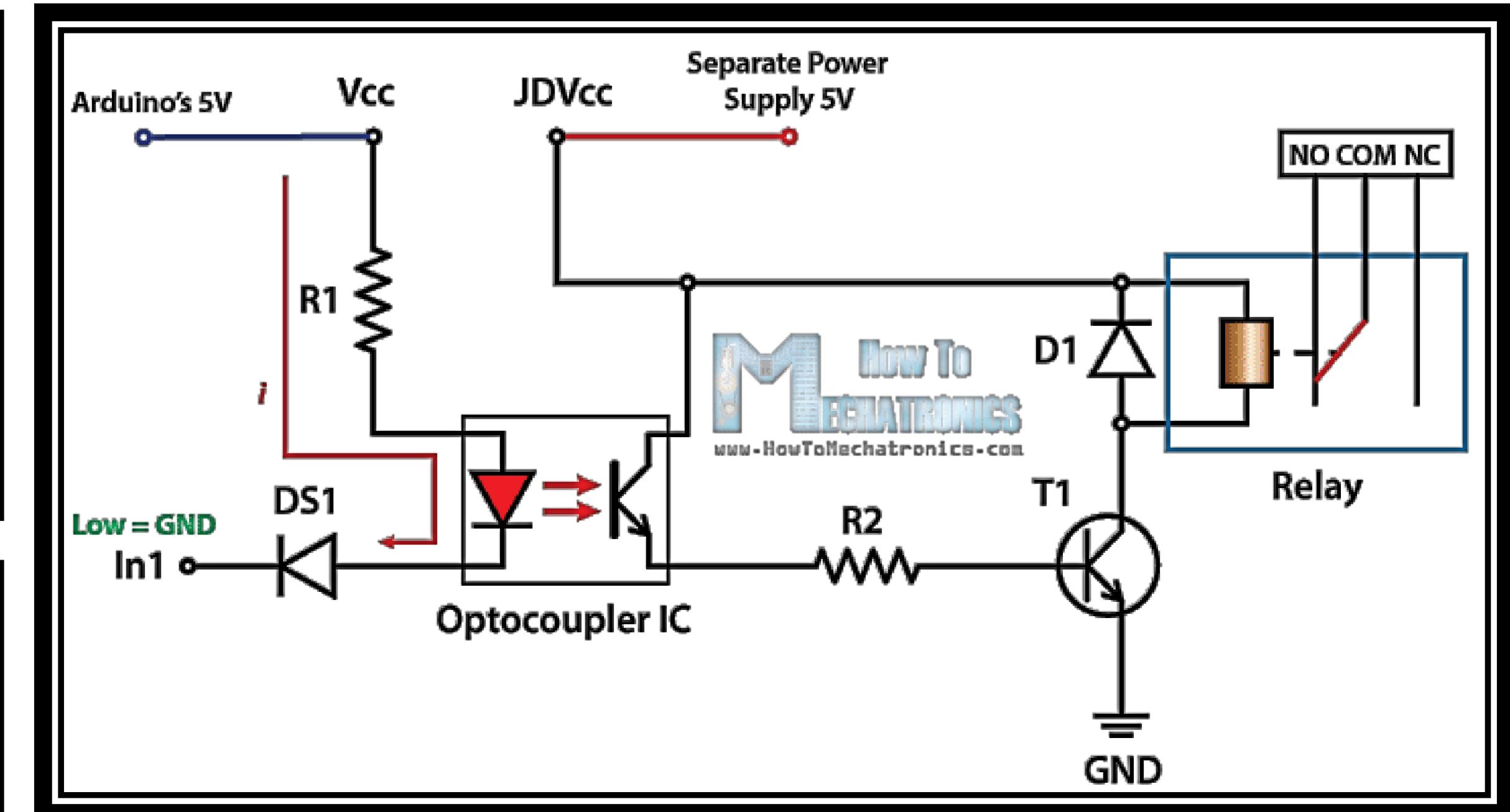
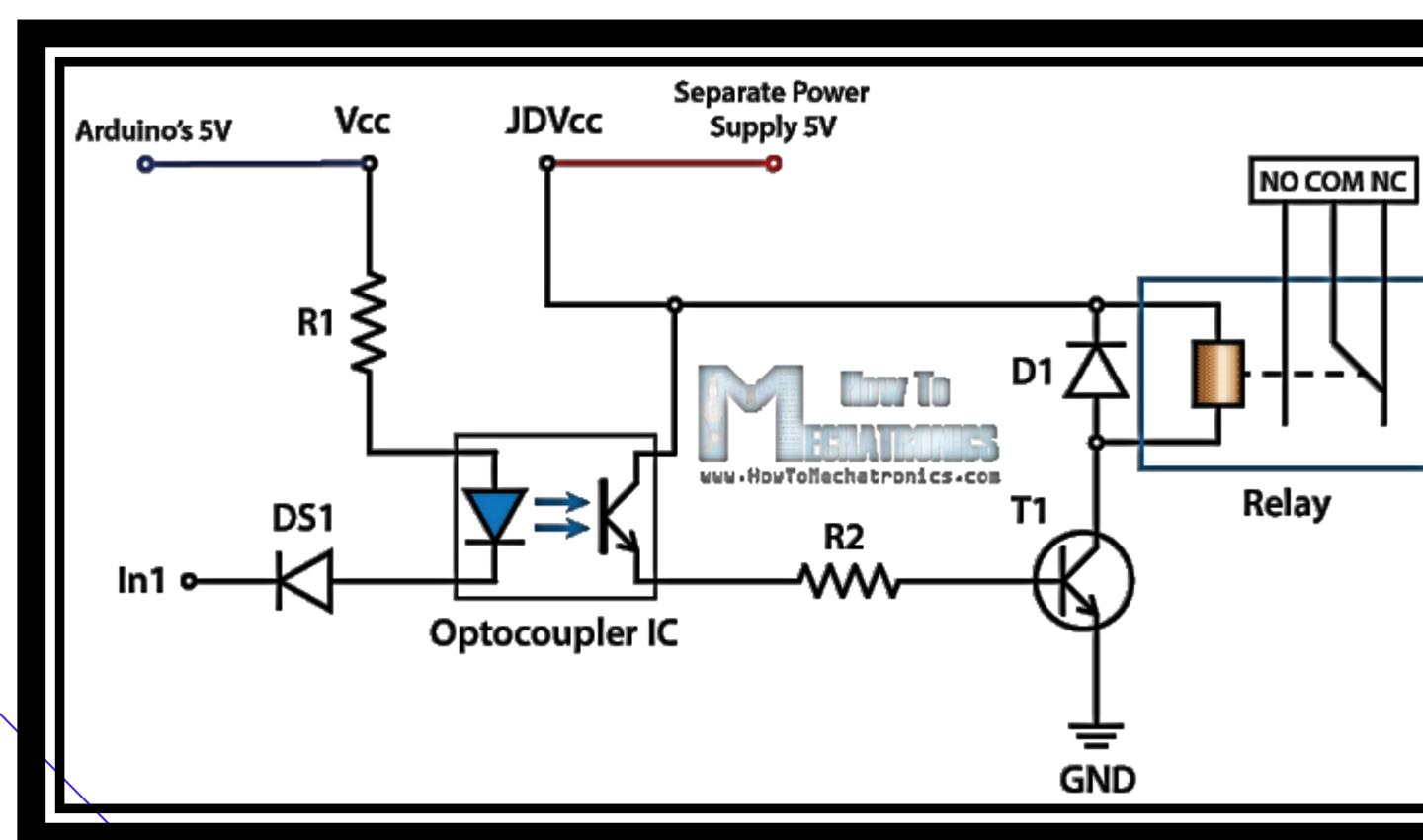
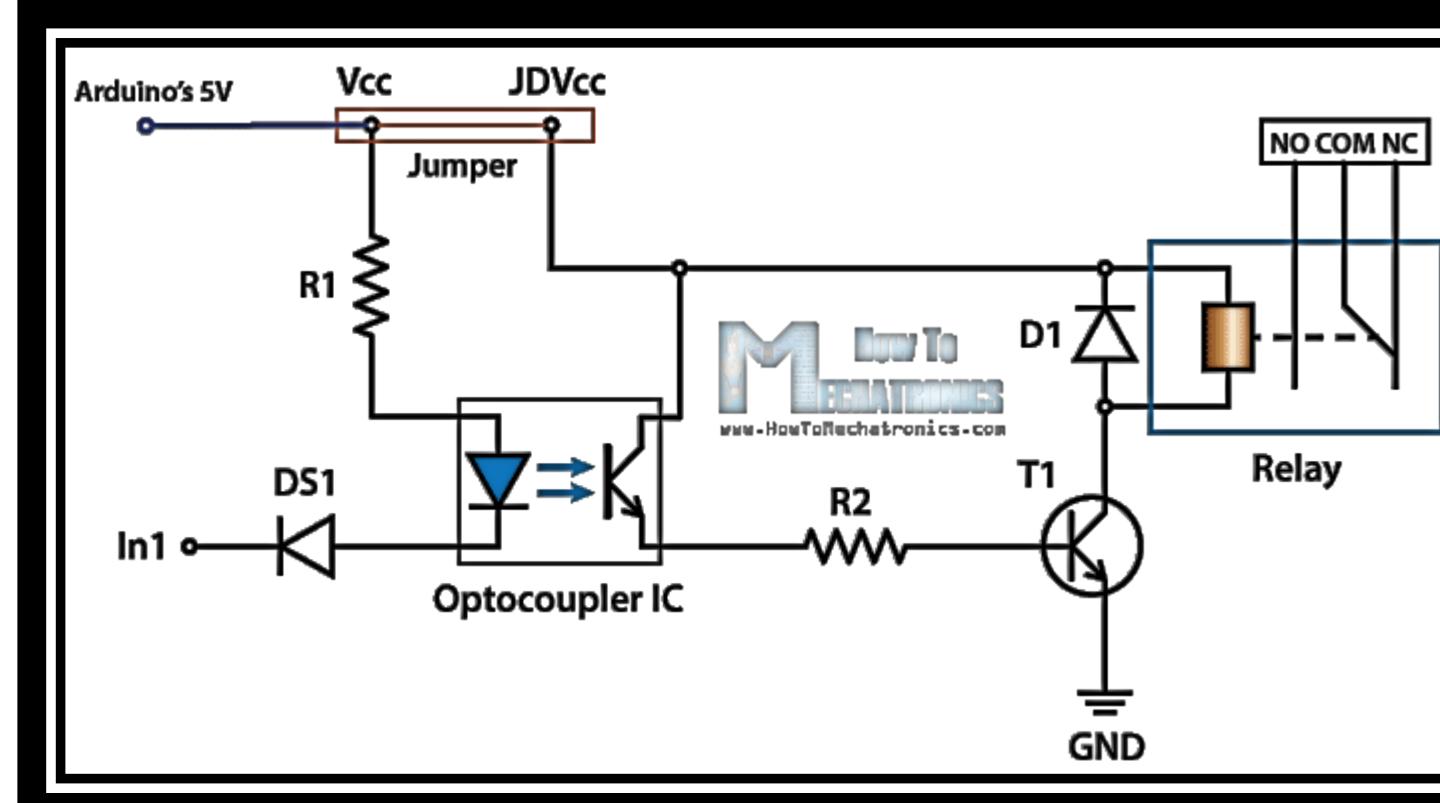
Relé na prática:

Acionamento e Módulos relés.



Relé na prática:

Acionamento e Módulos relés.



<https://howtomechatronics.com/tutorials/arduino/control-high-voltage-devices-arduino-relay-tutorial/>

Exemplos

Exemplo de programa para uso do LDR com um Relé.



Sensores e Atuadores

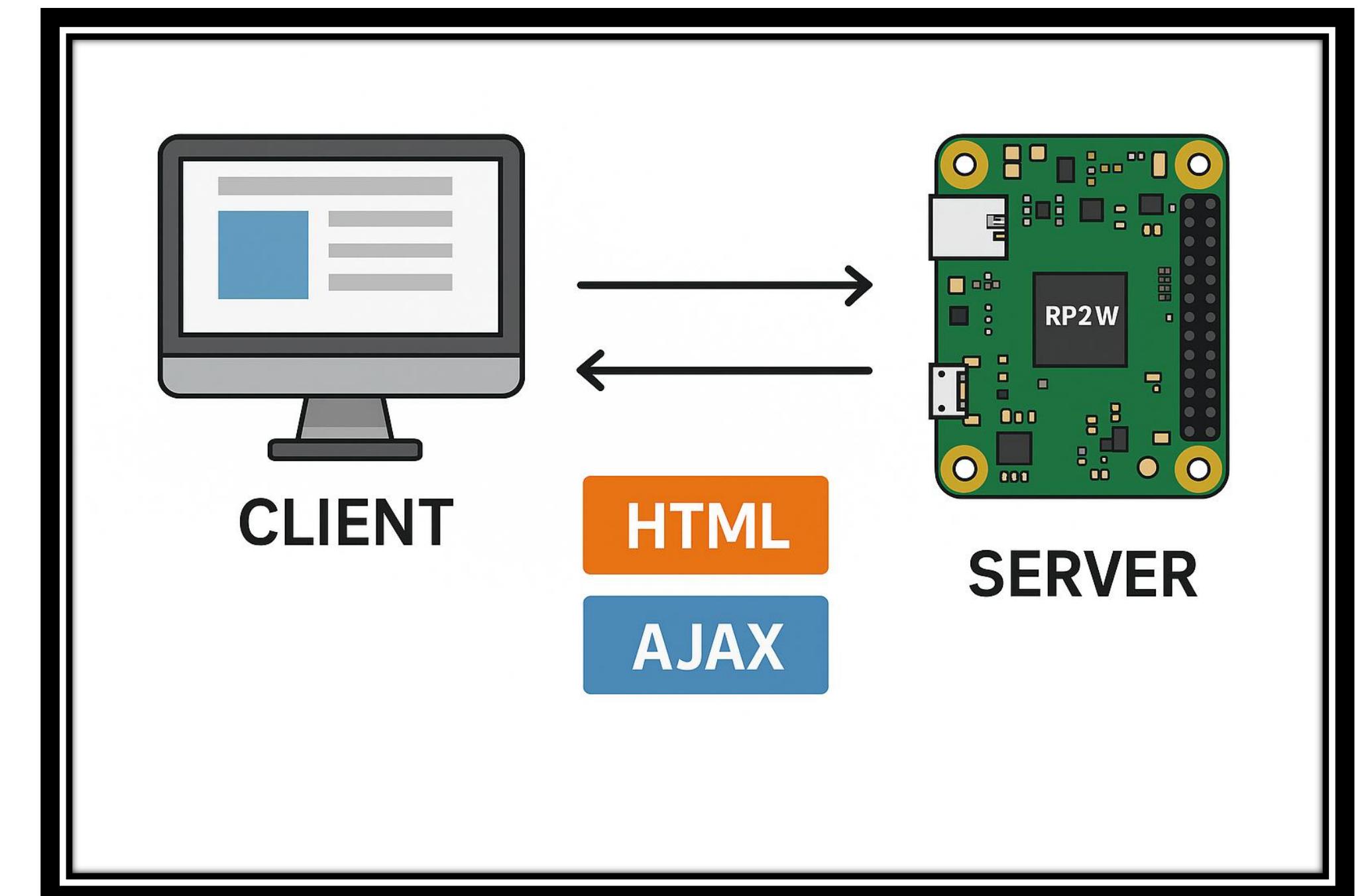
Exemplo 2: Relé Percentual

```
C Ilumina01.c > ⌂ main()
17 #include <stdio.h>
18 #include <stdlib.h>
19 #include "pico/stdlib.h"
20 #include "hardware/adc.h"
21 #include "hardware/i2c.h"
22 #include "lib/ssd1306.h"
23 #include "lib/font.h"
24
25 #define I2C_PORT i2c1
26 #define I2C_SDA 14
27 #define I2C_SCL 15
28 #define endereco 0x3C
29 #define ADC_PIN 28 // GPIO compartilhada com o microcontrolador
30
31 // Definindo o pino do relé
32 #define RELAY_PIN 8 // GPIO 8 para o relé
33
34 // Trecho para modo BOOTSEL com botão B
35 #include "pico/bootrom.h"
36 #define botaoB 6
37 void gpio_irq_handler(uint gpio, uint32_t events)
38 {
39     reset_usb_boot(0, 0);
40 }
41
42 int main()
43 {
44     // Para ser utilizado o modo BOOTSEL com botão B
45     gpio_init(botaoB);
46     gpio_set_dir(botaoB, GPIO_IN);
47     gpio_pull_up(botaoB);
48     gpio_set_irq_enabled_with_callback(botaoB, GPIO_IRQ_EDGE_RISING, &gpio_irq_handler);
49     // Aqui termina o trecho para modo BOOTSEL com botão B
50
51     // I2C Initialisation. Using it at 400Khz.
52     i2c_init(I2C_PORT, 400 * 1000);
53
54     gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
55     gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
56     gpio_pull_up(I2C_SDA);
57     gpio_pull_up(I2C_SCL);
```

```
C Ilumina01.c > ⌂ main()
42 int main()
43 {
44     gpio_pull_up(I2C_SDA);
45     gpio_pull_up(I2C_SCL);
46     ssd1306_t ssd; // Inicializa a estrutura do display
47     ssd1306_init(&ssd, WIDTH, HEIGHT, false, endereco, I2C_PORT);
48     ssd1306_config(&ssd);
49     ssd1306_send_data(&ssd);
50
51     // Limpa o display. O display inicia com todos os pixels apagados
52     ssd1306_fill(&ssd, false);
53     ssd1306_send_data(&ssd);
54
55     adc_init();
56     adc_gpio_init(ADC_PIN); // GPIO 28 como entrada analógica
57
58     // Inicializa o pino do relé
59     gpio_init(RELAY_PIN);
60     gpio_set_dir(RELAY_PIN, GPIO_OUT); // Define o pino como saída
61     gpio_put(RELAY_PIN, 0); // Garante que o relé comece desligado
62
63     uint16_t adc_value_x;
64     float ldr_pciento;
65     char str_x[5]; // Buffer para armazenar a string
66     char str_y[10]; // Buffer para armazenar a string
67
68     bool cor = true;
69     while (true)
70     {
71         adc_select_input(2); // O pino 28 como entrada analógica
72         adc_value_x = adc_read(); // Faz leitura do LRD
73         ldr_pciento = 100 - (adc_value_x * 100) / 4095; // Converte para porcentagem
74
75         // Lógica para acionar o relé
76         if (ldr_pciento > 40)
77         {
78             gpio_put(RELAY_PIN, 1); // Liga o relé
79             ssd1306_draw_string(&ssd, "RELE: ON", 75, 28);
80         }
81         else
82         {
83             gpio_put(RELAY_PIN, 0); // Desliga o relé
84             ssd1306_draw_string(&ssd, "RELE: OFF", 75, 28);
85         }
86
87         sprintf(str_x, "%d", adc_value_x); // Converte o valor para string
88         sprintf(str_y, "%1.0f", ldr_pciento); // Converte a porcentagem para string
89
90         //cor = !cor;
91         // Atualiza o conteúdo do display com animações
92         ssd1306_fill(&ssd, !cor); // Limpa o display
93         ssd1306_rect(&ssd, 3, 3, 122, 60, cor, !cor); // Desenha um retângulo
94         ssd1306_line(&ssd, 3, 25, 123, 25, cor); // Desenha uma linha
95         ssd1306_line(&ssd, 3, 37, 123, 37, cor); // Desenha uma linha
96         ssd1306_draw_string(&ssd, "CEPEDI TIC37", 8, 6);
97         ssd1306_draw_string(&ssd, "EMBARCATECH", 20, 16);
98         ssd1306_draw_string(&ssd, " Iluminacao", 10, 28);
99         ssd1306_draw_string(&ssd, "ADC", 13, 41);
100        ssd1306_draw_string(&ssd, "Nivel LUZ", 50, 41);
101        ssd1306_line(&ssd, 44, 37, 44, 60, cor); // Desenha uma linha
102        ssd1306_draw_string(&ssd, str_x, 8, 52); // Desenha a string
103        ssd1306_draw_string(&ssd, str_y, 75, 52); // Desenha a string
104        ssd1306_send_data(&ssd); // Atualiza o display
105        sleep_ms(700);
106    }
107 }
```

```
C Ilumina01.c > ⌂ main()
42 int main()
43 {
44     bool cor = true;
45     while (true)
46     {
47         adc_select_input(2); // O pino 28 como entrada analógica
48         adc_value_x = adc_read(); // Faz leitura do LRD
49         ldr_pciento = 100 - (adc_value_x * 100) / 4095; // Converte para porcentagem
50
51         // Lógica para acionar o relé
52         if (ldr_pciento > 40)
53         {
54             gpio_put(RELAY_PIN, 1); // Liga o relé
55             ssd1306_draw_string(&ssd, "RELE: ON", 75, 28);
56         }
57         else
58         {
59             gpio_put(RELAY_PIN, 0); // Desliga o relé
60             ssd1306_draw_string(&ssd, "RELE: OFF", 75, 28);
61         }
62
63         sprintf(str_x, "%d", adc_value_x); // Converte o valor para string
64         sprintf(str_y, "%1.0f", ldr_pciento); // Converte a porcentagem para string
65
66         //cor = !cor;
67         // Atualiza o conteúdo do display com animações
68         ssd1306_fill(&ssd, !cor); // Limpa o display
69         ssd1306_rect(&ssd, 3, 3, 122, 60, cor, !cor); // Desenha um retângulo
70         ssd1306_line(&ssd, 3, 25, 123, 25, cor); // Desenha uma linha
71         ssd1306_line(&ssd, 3, 37, 123, 37, cor); // Desenha uma linha
72         ssd1306_draw_string(&ssd, "CEPEDI TIC37", 8, 6);
73         ssd1306_draw_string(&ssd, "EMBARCATECH", 20, 16);
74         ssd1306_draw_string(&ssd, " Iluminacao", 10, 28);
75         ssd1306_draw_string(&ssd, "ADC", 13, 41);
76         ssd1306_draw_string(&ssd, "Nivel LUZ", 50, 41);
77         ssd1306_line(&ssd, 44, 37, 44, 60, cor); // Desenha uma linha
78         ssd1306_draw_string(&ssd, str_x, 8, 52); // Desenha a string
79         ssd1306_draw_string(&ssd, str_y, 75, 52); // Desenha a string
80         ssd1306_send_data(&ssd); // Atualiza o display
81         sleep_ms(700);
82     }
83 }
```

A arquitetura é baseada no
modelo cliente-servidor
com uso de **HTML** e **AJAX**.



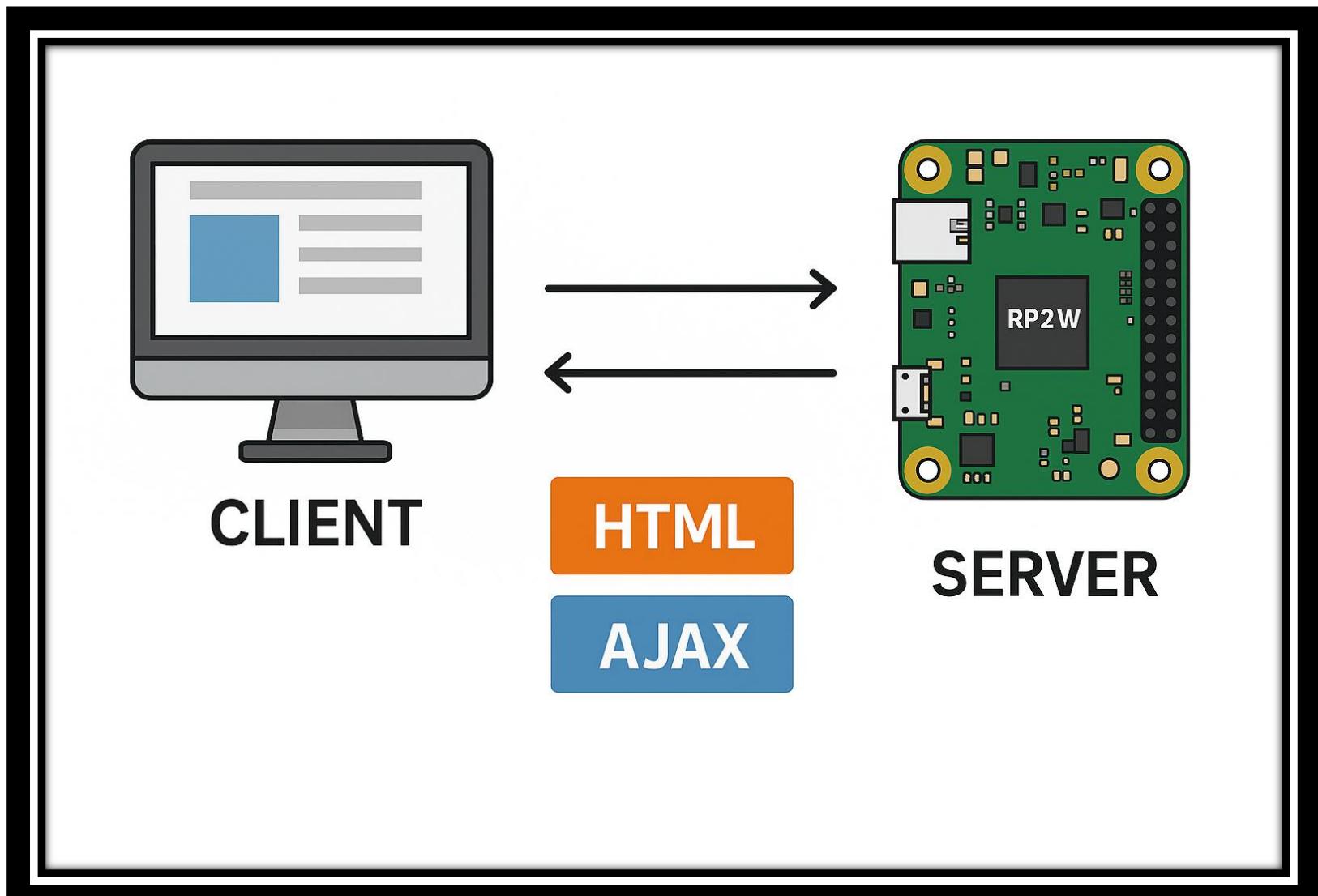
A arquitetura é baseada no **modelo cliente-servidor** com uso de **HTML** e **AJAX**.

CLIENTE

- Representa um navegador web (no computador ou celular).
- O usuário interage com uma **interface HTML** enviada pelo servidor.
- Essa interface pode incluir **botões, sliders, indicadores, gráficos**, etc.

SERVIDOR

- Atua como um **servidor HTTP** embarcado.
- Responde aos pedidos feitos pelo cliente com:
 - Páginas HTML
 - Dados em formato JSON
 - Comandos para controle de LEDs, sensores, servos, etc.



Estrutura básica de uma página HTML.

```
<!DOCTYPE html>
<html>
<head>
<title>Titulo da Pagina</title>
</head>
<body>

<h1>Este é o cabecalho nivel 1</h1>
<p>Este é um paragrafo..</p>

</body>
</html>.
```



<https://www.w3schools.com/html/>

HTML + CSS + JavaScript

O **HTML (HyperText Markup Language)** é a estrutura fundamental de uma página web, como se fosse o **esqueleto e os órgãos** do corpo humano — ele define o conteúdo e a organização dos elementos.

O **CSS (Cascading Style Sheets)** é responsável pelo **estilo e aparência visual** desses elementos, como se fosse a **roupa, os acessórios e até o penteado** — tudo aquilo que dá forma, cor e beleza ao corpo.

O **JavaScript** traz a **interatividade e os comportamentos dinâmicos** da página, funcionando como o **sistema nervoso ou o espírito do corpo** — ele permite que a página reaja a ações do usuário, tome decisões e execute tarefas.



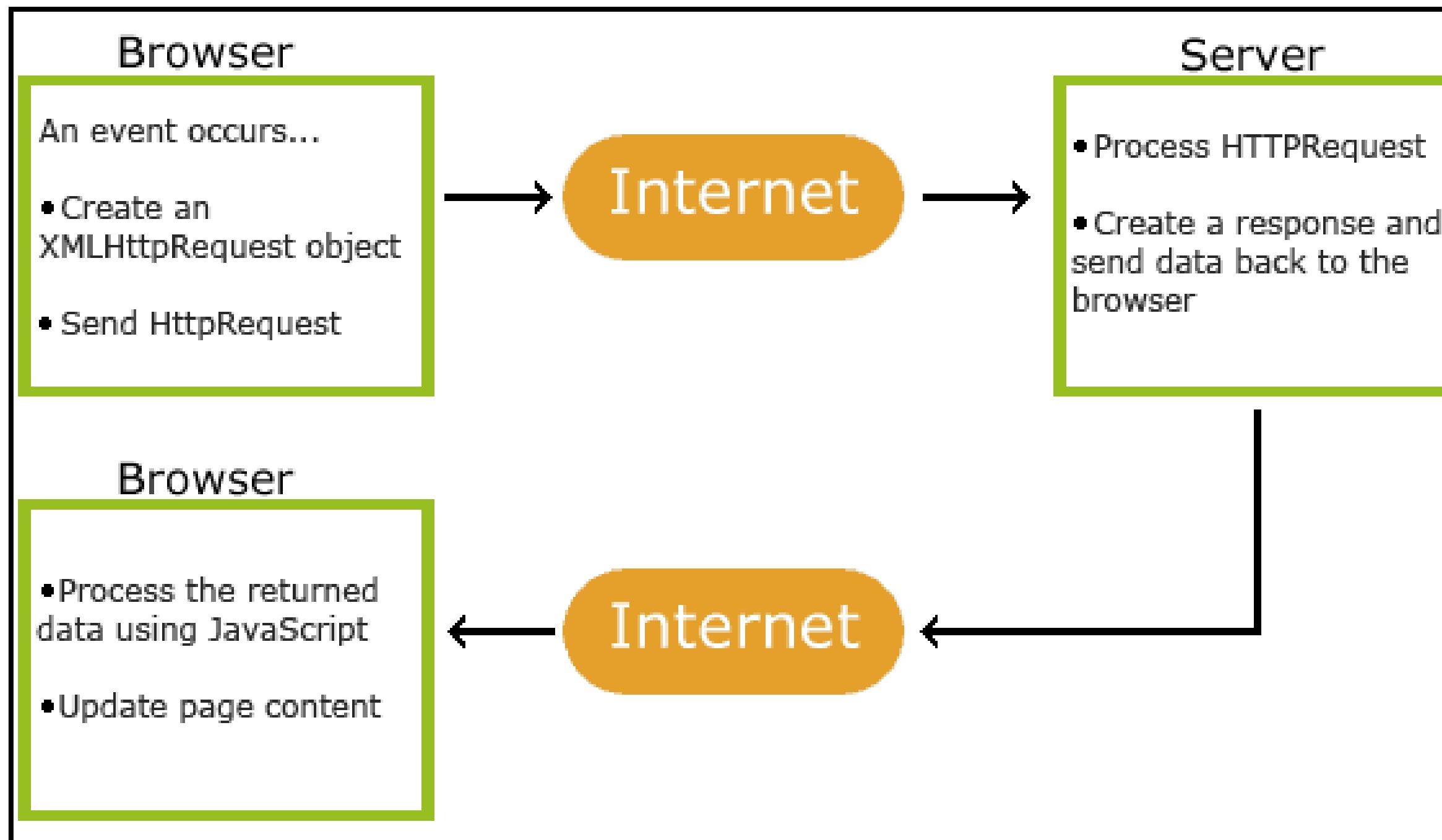
AJAX (Asynchronous JavaScript and XML) funciona como o **sistema de comunicação do corpo com o mundo exterior** — como se fosse o **sistema nervoso periférico**, que permite ao corpo receber e enviar informações **sem precisar parar ou reiniciar**. Com o AJAX, uma página pode buscar ou enviar dados para o servidor **sem recarregar toda a página**, tornando a experiência mais fluida e interativa.

JSON (JavaScript Object Notation) é o **formato da linguagem que essa comunicação utiliza**, como se fosse o **idioma falado entre o cérebro (JavaScript) e o servidor**.

Ele é leve, fácil de entender e muito usado para trocar dados estruturados de forma clara — como se fossem **mensagens organizadas que dizem: “Nome: João, Idade: 15”**.



AJAX permite que páginas da web sejam atualizadas de forma assíncrona ao trocar dados com um servidor web em segundo plano. Isso significa que é possível atualizar partes de uma página da web **sem recarregar a página inteira.**



Navegadores modernos podem usar a **Fetch API** em vez do objeto **XMLHttpRequest**. A interface da **Fetch API** permite que o navegador faça requisições **HTTP** para **servidores web**.

Resumindo esta analogia:

- **HTML** → Esqueleto do corpo (estrutura)
- **CSS** → Roupa e aparência (estilo)
- **JavaScript** → Espírito ou sistema nervoso central (ações e reações)
- **AJAX** → Sistema nervoso periférico (comunicação com o ambiente)
- **JSON** → Idioma claro e estruturado usado para enviar e receber essas informações

Servidor com RP2 W

Exemplo 3: Servidor com Ajax

```
C LigaLedAjax.c X
C LigaLedAjax.c > ...
1 #include "pico/cyw43_arch.h"
2 #include "pico/stdlib.h"
3 #include "lwip/tcp.h"
4 #include "hardware/i2c.h"
5 #include "hardware/adc.h"
6 #include "hardware/gpio.h"
7 #include <stdio.h>
8 #include <string.h>
9 #include <stdlib.h>
10 #include "ssd1306.h"
11 #include "font.h"
12
13 #define LED_PIN 12
14 #define BOTAO_A 5
15 #define BOTAO_JOY 22
16 #define JOYSTICK_X 26
17 #define JOYSTICK_Y 27
18
19 #define WIFI_SSID "Seu SSID"
20 #define WIFI_PASS "Sua Senha"
21
22 #define I2C_PORT_DISP i2c1
23 #define I2C_SDA_DISP 14
24 #define I2C_SCL_DISP 15
25 #define endereco 0x3C
26
27 const char HTML_BODY[] =
28 "<!DOCTYPE html><html><head>
29 <style>"
```

```
</style>
<script>
"function sendCommand(cmd) { fetch('/led/' + cmd); }"
"function atualizar() {
"  fetch('/estado').then(res => res.json()).then(data => {
"    document.getElementById('estado').innerText = data.led ? 'Ligado' : 'Desligado';
"    document.getElementById('x_valor').innerText = data.x;
"    document.getElementById('y_valor').innerText = data.y;
"    document.getElementById('botao').innerText = data.botao ? 'Pressionado' : 'Solto';
"    document.getElementById('joy').innerText = data.joy ? 'Pressionado' : 'Solto';
"    document.getElementById('bolinha_a').style.background = data.botao ? '#2126F3' : '#ccc';
"    document.getElementById('bolinha_joy').style.background = data.joy ? '#4C7F50' : '#ccc';
"    document.getElementById('barra_x').style.width = Math.round(data.x / 4095 * 100) + '%';
"    document.getElementById('barra_y').style.width = Math.round(data.y / 4095 * 100) + '%';
"  });
"}"
"setInterval(atualizar, 1000);"
"</script></head><body>"
```

Essa parte do código corresponde ao bloco **JavaScript** que fica dentro das tags **<script>...</script>**.
Ele é responsável por enviar comandos ao servidor e atualizar dinamicamente os elementos da página com os dados recebidos via **AJAX**.

<https://github.com/wiltonlacerda/EmbarcaTechResU3Ex03.git>

Servidor com RP2 W

Exemplo 3: Servidor com Ajax

```
<!DOCTYPE html>
<html>
<head>
<meta charset='UTF-8'>
<title>Controle do LED</title>
<style>
body { font-family: sans-serif; text-align: center; padding: 10px; margin: 0; background: #f9f9f9; }
.botao { font-size: 20px; padding: 10px 30px; margin: 10px; border: none; border-radius: 8px; }
.on { background: #4CAF50; color: white; }
.off { background: #f44336; color: white; }
.barra { width: 30%; background: #ddd; border-radius: 6px; overflow: hidden; margin: 0 auto 15px auto; height: 20px; }
.preenchimento { height: 100%; transition: width 0.3s ease; }
#barra_x { background: #2196F3; }
#barra_y { background: rgb(177, 96, 153); }
.label { font-weight: bold; margin-bottom: 5px; display: block; }
.bolinha { width: 20px; height: 20px; border-radius: 50%; display: inline-block; margin-left: 10px; background: #ccc; transition: background 0.3s ease; }
@media (max-width: 600px) {
    .botao { width: 80%; font-size: 18px; }
}
</style>
<script>
function sendCommand(cmd) {
    fetch('/led/' + cmd);
}
function atualizar() {
    fetch('/estado')
        .then(res => res.json())
        .then(data => {
            document.getElementById('estado').innerText = data.led ? 'Ligado' : 'Desligado';
            document.getElementById('x_valor').innerText = data.x;
            document.getElementById('y_valor').innerText = data.y;
            document.getElementById('botao').innerText = data.botao ? 'Pressionado' : 'Solto';
            document.getElementById('joy').innerText = data.joy ? 'Pressionado' : 'Solto';
            document.getElementById('bolinha_a').style.background = data.botao ? '#2126F3' : '#ccc';
            document.getElementById('bolinha_joy').style.background = data.joy ? '#4C7F50' : '#ccc';
            document.getElementById('barra_x').style.width = Math.round(data.x / 4095 * 100) + '%';
            document.getElementById('barra_y').style.width = Math.round(data.y / 4095 * 100) + '%';
        });
}
</script>
```

```
setInterval(atualizar, 1000);
</script>
</head>
<body>

<h1>Controle do LED</h1>

<p>Estado do LED: <span id='estado'>--</span></p>

<p class='label'>Joystick X: <span id='x_valor'>--</span></p>
<div class='barra'><div id='barra_x' class='preenchimento'></div></div>

<p class='label'>Joystick Y: <span id='y_valor'>--</span></p>
<div class='barra'><div id='barra_y' class='preenchimento'></div></div>

<p class='label'>Botão A: <span id='botao'>--</span> <span id='bolinha_a' class='bolinha'></span></p>
<p class='label'>Botão do Joystick: <span id='joy'>--</span> <span id='bolinha_joy' class='bolinha'></span></p>

<button class='botao' on' onclick="sendCommand('on')">Ligar</button>
<button class='botao off' onclick="sendCommand('off')">Desligar</button>

<hr style='margin-top: 20px;'>
<p style='font-size: 15px; color: #336699; font-style: italic; max-width: 90%; margin: 10px auto; '>
    Utilização da BitDogLab para exemplificar a comunicação via rede Wi-Fi utilizando o protocolo HTML com JavaScript
</p>

</body>
</html>
```

Servidor com RP2 W

Exemplo 3: Servidor com Ajax

Result Size: 586 x 724 [Get your own website](#)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset='UTF-8'>
  <title>Controle do LED</title>
  <style>
    body { font-family: sans-serif; text-align: center; padding: 10px; margin: 0; background: #f9f9f9; }
    .botao { font-size: 20px; padding: 10px 30px; margin: 10px; border: none; border-radius: 8px; }
    .on { background: #4CAF50; color: white; }
    .off { background: #f44336; color: white; }
    .barra { width: 30%; background: #ddd; border-radius: 6px; overflow: hidden; margin: 0 auto 15px auto; height: 20px; }
    .preenchimento { height: 100%; transition: width 0.3s ease; }
    #barra_x { background: #2196F3; }
    #barra_y { background: rgb(177, 96, 153); }
    .label { font-weight: bold; margin-bottom: 5px; display: block; }
    .bolinha { width: 20px; height: 20px; border-radius: 50%; display: inline-block; margin-left: 10px; background: #ccc; transition: background 0.3s ease; }
    @media (max-width: 600px) {
      .botao { width: 80%; font-size: 18px; }
    }
  </style>
  <script>
    function sendCommand(cmd) {
      fetch('/led/' + cmd);
    }
    function atualizar() {
      ...
    }
  </script>

```

Controle do LED

Estado do LED: --

Joystick X: --

Joystick Y: --

Botão A: --

Botão do Joystick: --

Ligar

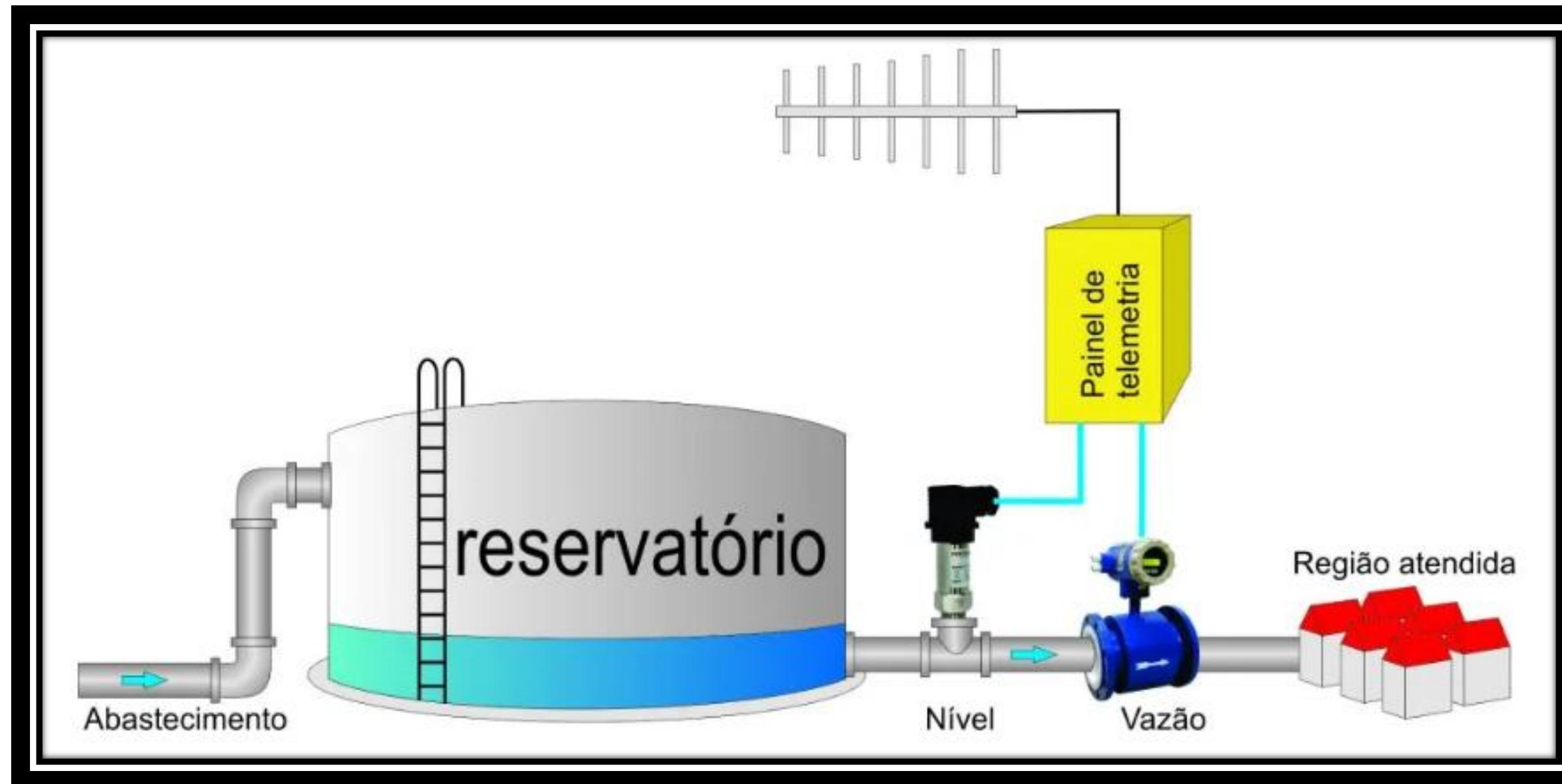
Desligar

Utilização da BitDogLab para exemplificar a comunicação via rede Wi-Fi utilizando o protocolo HTML com JavaScript

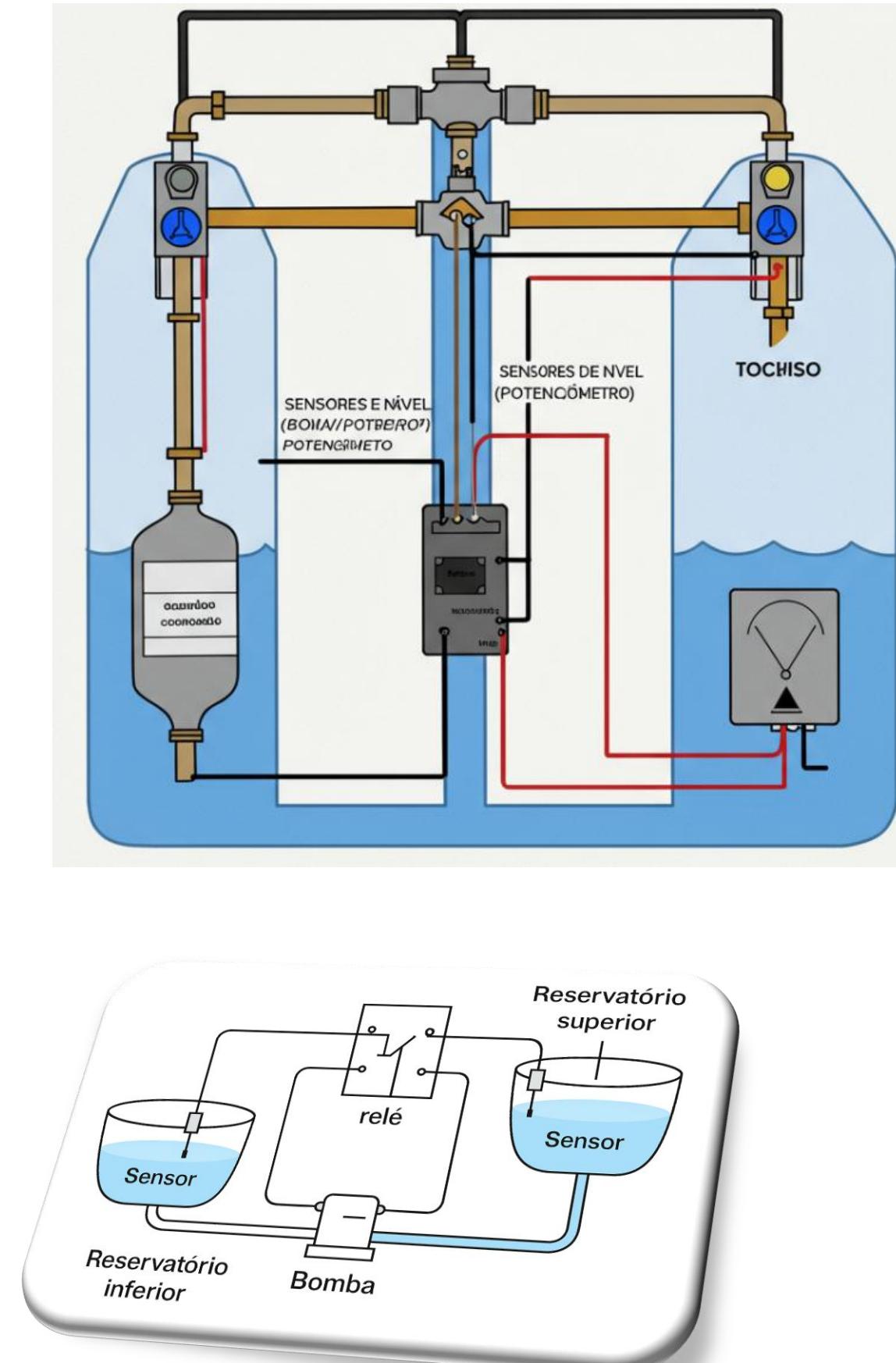
Tarefa



Controle de processos



<https://alfacomp.net/2020/12/22/projeto-de-automacao-e-telemetria-de-um-reservatorio-de-agua-tratada/>



Sensores e Atuadores

Tarefa:

Controle de Nível com Interface Web



Enunciado

Consolidar os conhecimentos adquiridos sobre diversos temas estudados até a presente data por meio do desenvolvimento de um sistema embarcado funcional e acessível.

Descrição do Projeto

Os **mentores** serão os responsáveis por formar os grupos de residentes. Cada mentor selecionará residentes para compor **dois grupos distintos**. Dessa forma, cada polo trabalhará com um total de **quatro grupos**.

Cada grupo terá a tarefa de desenvolver e implementar um **sistema embarcado para controle de nível de líquido**. A medição do nível do reservatório será feita por uma **boia acoplada a um potenciômetro**, fornecendo uma leitura analógica a um microcontrolador.

Funcionalidades do Sistema

O sistema permitirá a **configuração remota dos limites mínimo e máximo de nível** por meio de uma página HTML. Essa página será servida pelo próprio **RP2040 via conexão Wi-Fi**.

A lógica de controle do sistema se baseará nos seguintes critérios:

- Se o nível de água estiver **abaixo do valor mínimo configurado**, uma bomba d'água será acionada automaticamente.
- Quando o nível de água **ultrapassar o valor máximo**, a bomba será desligada.

Interface do Usuário (HTML)

A interface HTML deverá exibir o **nível atual** lido pelo sistema e o **estado da bomba** (ligada ou desligada). Além disso, a interface permitirá ao usuário **configurar novos limites de nível mínimo e máximo em tempo real**, sem a necessidade de reiniciar o sistema.

Interfaces e Recursos da Plataforma BitdogLab

Além da interface web, é fundamental utilizar os **recursos visuais e interativos da plataforma BitDogLab** para fornecer informações locais ao usuário.

- O **display LCD** deve exibir dados relevantes como o nível atual de água, os limites configurados, o estado da bomba (ligada/desligada) e o status da conexão Wi-Fi.



Obrigado!

Executores:



Coordenação:



Iniciativa:

