Oficina de introdução à robótica

Terceira Aula

Instrutores: Pedro Rocha e João Pedro Arruda
Organização: Profa. Regiane Kawasaki e Prof. Victor Santiago

> Revisão Aula Passada

> Entrada PWM

/Sinal PWM

O que é:
O PWM é uma técnica usada para controlar a intensidade de um sinal elétrico, como uma tensão ou corrente, variando a largura dos pulsos em um sinal digital.

Como identificar:
No Arduino, você pode usar saídas PWM em alguns dos pinos digitais para gerar sinais PWM. O Arduino Uno tem vários pinos PWM, como o 3, 5, 6, 9, 10 e 11. Esses pinos são marcados com o símbolo "~" para indicar que eles são capazes de gerar sinais PWM.

/Sinal PWM

Como programar:

Para “ativar” o sinal PWM no arduino é necessário usar a função `analogWrite()` no código, o qual recebe dois argumentos, que são, a porta que será utilizada pelo componente e em seguida o ciclo de trabalho que será utilizado pela porta. As portas PWM aceitam ciclos de trabalho de 0 até 256.

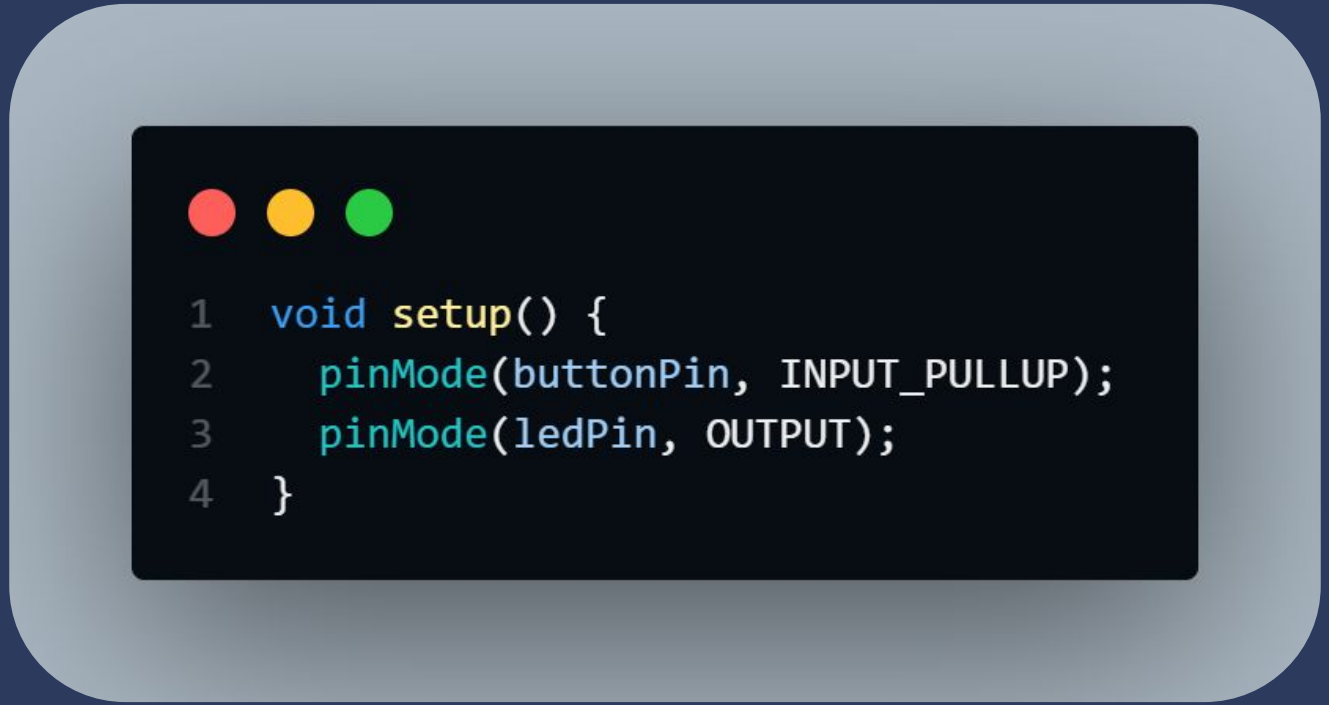
```
1  void loop() {
2    analogWrite(ledPin, intensidade);
3    delay(10);
4    intensidade = (intensidade + 1) % 256;
5  }
```

> Push Button

/Botão (Push Button)

O que é:
É um dos componentes eletrônicos mais utilizados para prototipagem de projetos. Esta chave é um tipo de interruptor pulsador (conduz somente quando está pressionado).

Pull UP e Pull DOWN:
São usados em eletrônica para garantir que um circuito funcione corretamente e evite problemas indesejados. Ajudam a eliminar “ruídos elétricos” que podem causar leituras erradas no sistema.


1 void setup() {
2 pinMode(buttonPin, INPUT_PULLUP);
3 pinMode(ledPin, OUTPUT);
4 }

> Buzzer

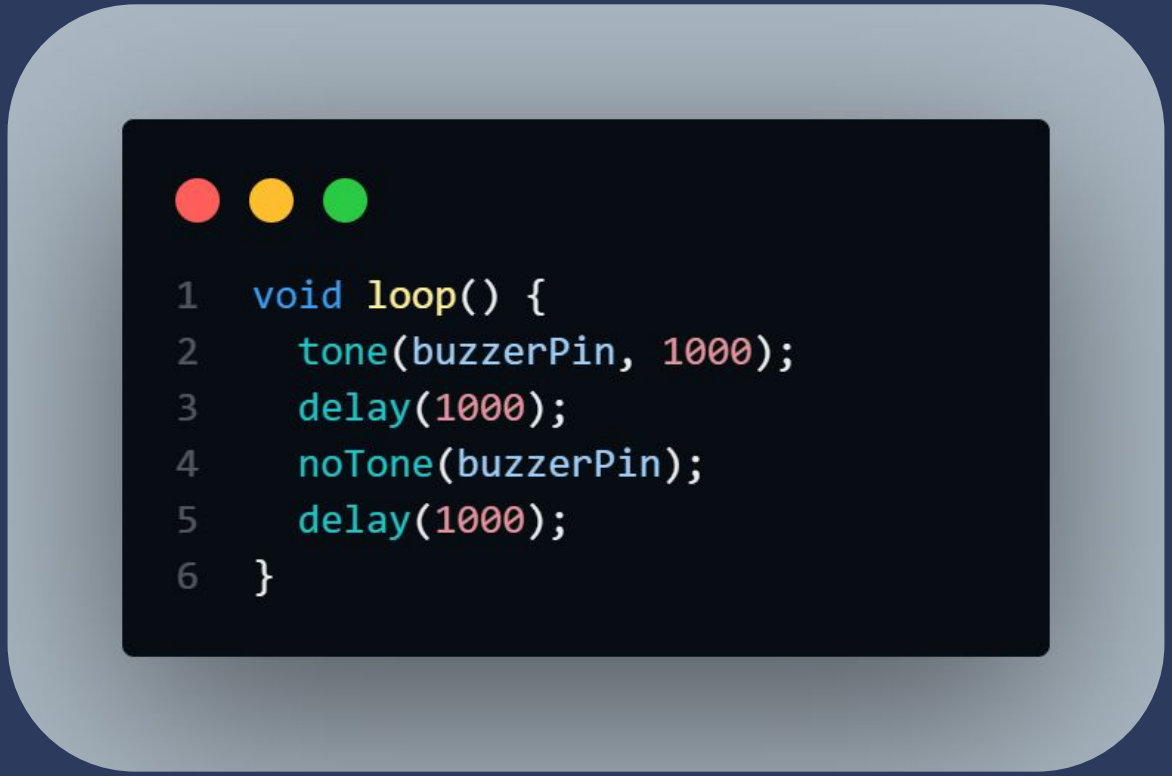
/Buzzer

O que é:
Buzzer é um dispositivo utilizado em eletrônica para produzir som ou emitir um sinal sonoro. Existem dois tipos de Buzzers, ativos e passivos. Usaremos os buzzers passivos, os quais não vêm com oscilador interno e necessitam de uma entrada PWM.

Como usar:
Para usar um buzzer basta atribuí-lo à uma porta PWM (as que possuem sinal “~”), juntamente com um sinal GND para o aterramento .

/Buzzer

Programação:
Diferentemente dos LEDs, que possuem maneiras diferentes de funcionamento diferentes dependendo da porta que é colocado, os buzzers que usaremos não precisam fazer o uso da função `analogWrite()`, pois só funcionam com sinais PWM, e possui duas funções principais. **Tone (**responsável por atribuir o pin que se contra o buzzer e a frequência do som que ele irá emitir) e **No Tone** (responsável por parar o som que ele está emitindo).


void loop() {
tone(buzzerPin, 1000);
delay(1000);
noTone(buzzerPin);
delay(1000);
}

/Terceira Aula

/Terceira Aula

Tópicos Principais:

1. Sensores de Linha
2. Switch Case
3. Servo Motor
4. Função Random
5. Ponte H

/Sensores de Linha

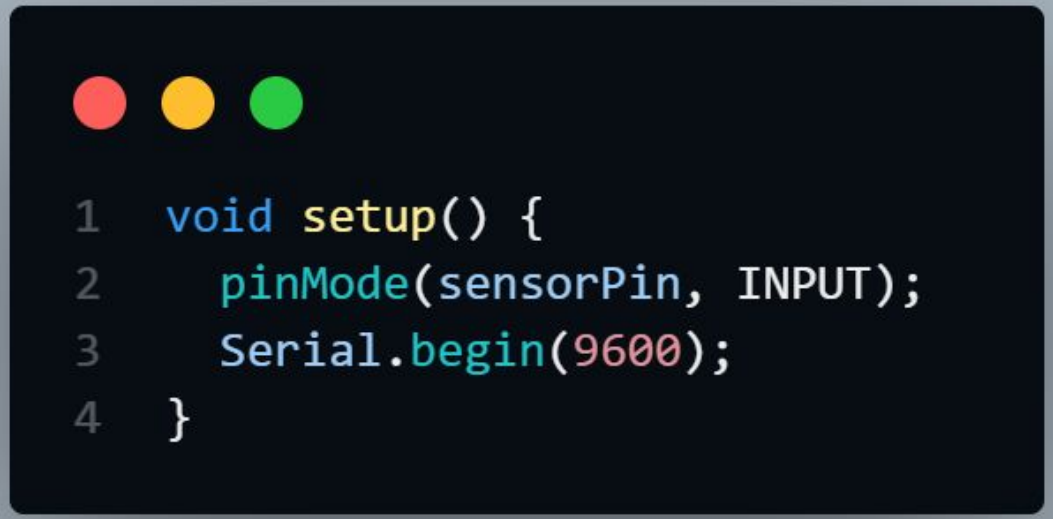
/Sensor de Linha

O que é:
sensor de linha é um dispositivo que é usado para detectar e seguir linhas em superfícies, como linhas desenhadas no chão ou faixas em robôs seguidores de linha.

Funcionamento:
Os sensores de linha são baseados na emissão de luz infravermelha e um fototransistor ou fotodiodo que detecta a luz refletida. Quando o sensor está sobre uma superfície, a luz infravermelha é refletida de volta para o fototransistor ou fotodiodo. O valor da reflexão **depende do contraste entre a linha e a superfície** circundante.

/Sensor de Linha

Serial Begin:
Para programar o sensor, é necessário usar a função `serial.begin()` a qual é uma função em Arduino que é usada para inicializar a comunicação serial entre o Arduino e um dispositivo externo. Essa função aceita um argumento que especifica a taxa de transmissão (baud rate) na qual os dados serão enviados e recebidos pela porta serial. O baud rate determina a velocidade com que os dados são transmitidos e recebidos, medido em bits por segundo (bps).



```
void setup() {
  pinMode(sensorPin, INPUT);
  Serial.begin(9600);
}
```

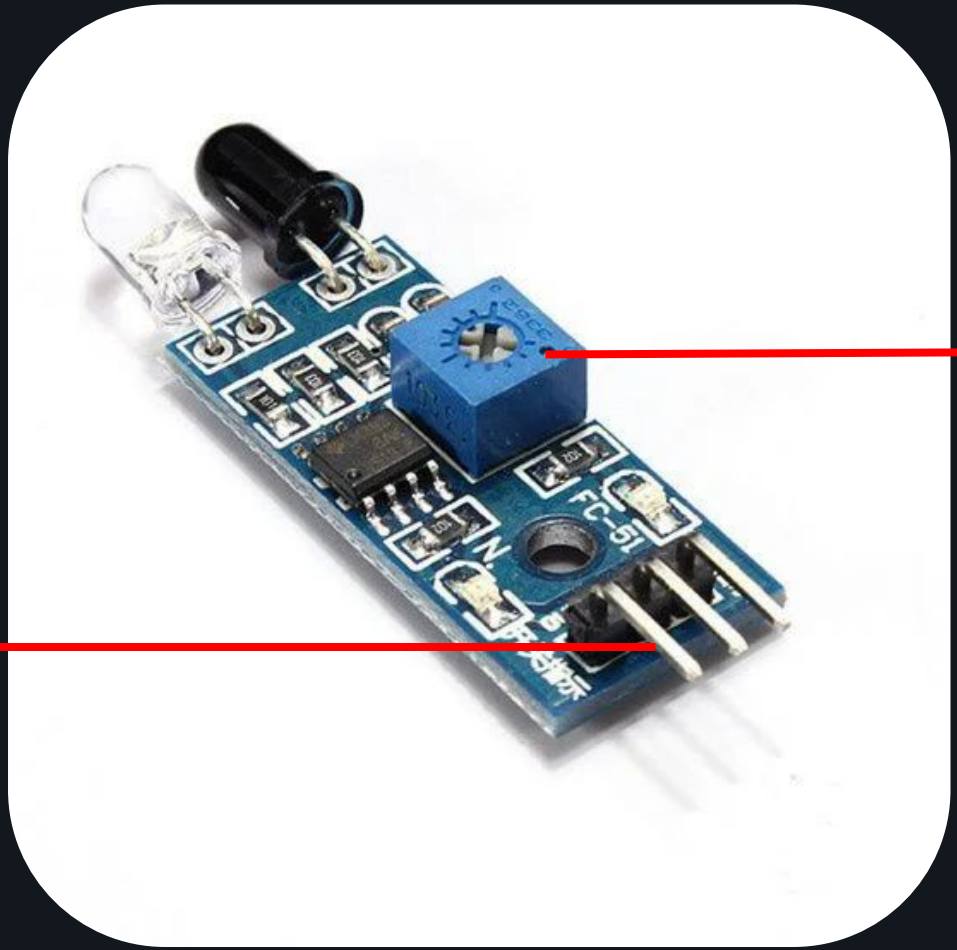
/Sensor de Linha

Calibragem do sensor:
Para calibrar o sensor, em condições boas de iluminação, basta ajustar a sensibilidade do calibrador com uma chave de fenda. Isso pode ajudar a ajustar a detecção da linha de acordo com as condições específicas do ambiente.

VCC (Voltage Common Collector):
O pino VCC é onde você fornece a tensão de alimentação ao sensor. Este pino deve ser conectado à tensão de alimentação necessária para o funcionamento do sensor. Para a maioria dos sensores, como o sensor de linha QRE1113, é comum usar uma tensão de 5V. Certifique-se de que a tensão aplicada esteja de acordo com as especificações do sensor.

/Sensor de Linha

OUT (Output):
O pino OUT é onde o sensor fornece a saída do sinal. A natureza dessa saída pode variar entre sensores. Pode ser uma saída analógica que fornece uma tensão proporcional à intensidade do reflexo de luz, ou uma saída digital que indica a detecção ou não detecção da linha. A leitura deste pino é a informação que você usará para determinar o status do sensor.


Calibrador
Entradas
FC-51

/Switch Case

/Switch Case

O que é:
O Switch Case é uma estrutura de controle usada em muitas linguagens de programação para realizar seleção condicional com base no valor de uma expressão. É frequentemente utilizado para simplificar o código quando você tem várias opções ou casos diferentes a serem tratados de acordo com o valor de uma variável.

Importância:
O Switch Case pode ser mais eficiente do que várias instruções "if...else" aninhadas, uma vez que a estrutura Switch Case pode ser otimizada pelo compilador. No entanto, é importante notar que essa estrutura em Arduino é mais adequado para valores discretos, como números inteiros, caracteres ou enumerações.

/Switch Case

Programação:
Primeiramente, para criar um Switch Case, é necessário escrever a palavra reservada `switch` e em seguida colocar entre parênteses o gerador de casos, após isso, é necessário abrir o escopo da função usando chaves e escrever cada caso com a palavra `case` e em seguida o possível valor do caso. Opcionalmente, pode adicionar um caso default para lidar com valores que não correspondem.

Break:
O break é usado para sair do bloco Switch Case após a execução de um caso específico. Isso impede que o programa continue a executar os casos subsequentes. Sem o "break", o programa seguiria executando todos os casos subsequentes, mesmo que o valor da expressão correspondesse a um caso anterior.

```
switch (buttonState) {
  case HIGH:
    digitalWrite(ledPin, HIGH);
    Serial.println("LED ligado");
    break;

  case LOW:
    digitalWrite(ledPin, LOW);
    Serial.println("LED desligado");
    break;
}
```

/Hora do Desafio #1

/Hora do Desafio

1. Faça o sensor identificar uma linha
2. Acenda um led e faça um sinal sonoro sempre que o sensor identificar algo

/Servo Motor

/Servo Motor

O que é:
Os servo motores são conhecidos por sua capacidade de posicionar-se com alta precisão em um ângulo específico. Isso os torna ideais para tarefas que requerem controle de posição preciso.

Funcionamento:
A maioria dos servo motores tem uma faixa limitada de movimento, geralmente de 0 a 180 graus. Isso significa que o eixo do servo pode ser movido apenas dentro desse intervalo. Para conectar um servo motor, é necessário usar uma porta PWM.

/Servo Motor

Conexões:
Servo motores geralmente funcionam com uma tensão de 4,8 a 6 volts. Eles têm três fios: um para alimentação e fornecimento de energia (VCC), um para aterramento (GND) e um para o sinal de controle que é conectado com um pino PWM.

Programação:
Para programar um servo motor, é necessário importar a biblioteca Servo.h, usar myservo.attach para definir o pino usado no setup e usar myservo.write para definir para qual posição o servo irá girar


#include <Servo.h>
Servo myservo


1 void setup() {
2 myservo.attach(9);
3 }


1 myservo.write(180);

/Hora do Desafio #2

/Hora do Desafio

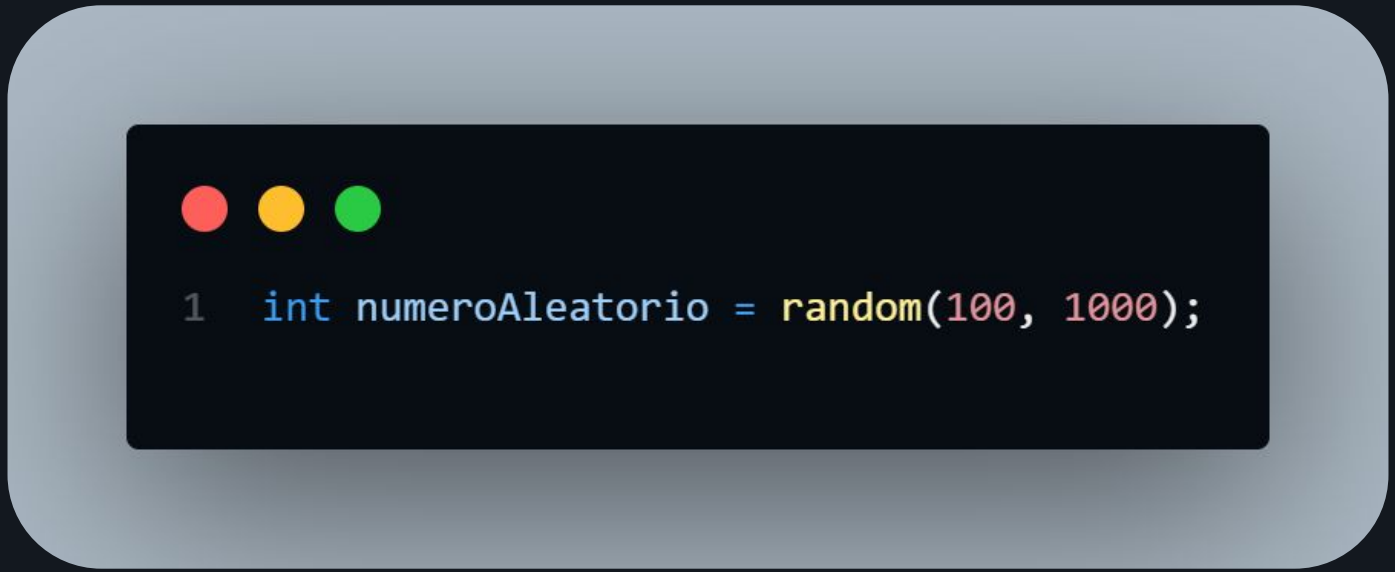
1. Faça o servo girar 45 graus
2. Faça o servo girar 90 graus
3. Faça o servo girar 180 graus
4. Faça o servo girar 360 graus

/Função Random

/Função Random

O que é:
A função random() no Arduino é uma função que gera números pseudo-aleatórios. Ela é frequentemente usada para criar variações aleatórias em programas e projetos Arduino. Aqui estão alguns pontos importantes sobre a função random():

Programação:
Para usar essa função de forma simples, basta chamar a função com a palavra reservada random e colocar como argumento o intervalo desejado


1 int numeroAleatorio = random(100, 1000);

/Ponte H

/Ponte H

O que é:
Uma ponte H é um circuito eletrônico usado para controlar a direção e a velocidade de motores elétricos, como motores de corrente contínua (DC) e motores de passo. Ela permite que os motores girem em ambas as direções e regula a velocidade por meio de pulsos PWM.

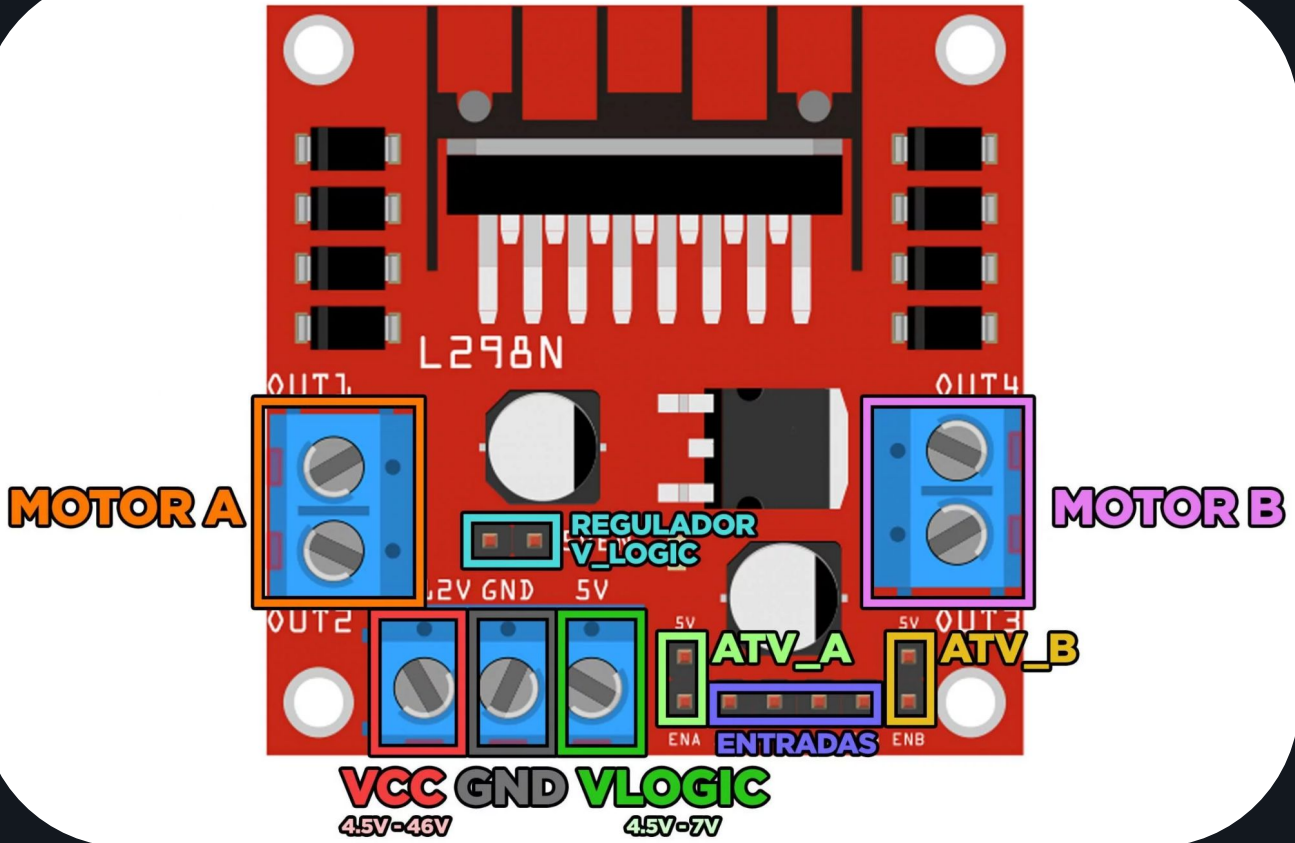
Proteção Elétrica:
A ponte H desempenha um papel crucial na proteção elétrica do Arduino. Ela isola o microcontrolador de picos de tensão e corrente gerados por motores, evitando danos. Essa proteção permite o controle seguro de motores de alta potência em projetos eletrônicos, tornando a ponte H um componente indispensável para garantir a integridade e a longevidade do Arduino.

/Ponte H

Alimentação:
A alimentação externa de uma ponte H é a fonte de energia que fornece a corrente necessária para os motores. Ela é separada da alimentação do circuito de controle (geralmente alimentado pelo Arduino) para evitar picos de corrente que possam danificar o controle, garantindo que os motores funcionem eficazmente e com segurança. Isso permite o uso de motores com maior potência do que o Arduino pode fornecer sozinho, tornando a alimentação externa essencial em muitos projetos.

/Ponte H

Conectores:
Uma ponte H pode possuir uma porção demasiada de conectores, entretanto, existem três que são os principais, que são: **Conectores para motores**, que permitem a alimentação necessária para o funcionamento dos motores, **Conector de Alimentação Externa**, que são os conectores que permitem a alimentação externa da própria ponte H, **Conectores de Controle**, que permitem a comunicação do arduino com os motores que irão ser conectados com a ponte H.


L298N
OUT1
OUT4
OUT2
OUT3
MOTOR A
MOTOR B
REGULADOR
V_LOGIC
12V GND 5V
5V
5V
ATV_A
ATV_B
ENA
ENTRADAS
ENB
VCC GND VLOGIC
4.5V - 46V
4.5V - 7V