OFICINA DE SEGUIDORES DE LINHA

SEGUIDORES DE LINHA

Presented by PETEE

O PETEE UFMG

O que é ?

O Programa de Educação Tutorial (PET) é desenvolvido por grupos de estudantes, com tutoria de um docente, organizados a partir de formações em nível de graduação nas Instituições de Ensino Superior do país orientados pelo princípio da indissociabilidade entre ensino, pesquisa e extensão e da educação tutorial.

Introdução

O que é um seguidor de linha ?

Para que serve?

Robôs seguidores de linha são máquinas capazes de percorrer um determinado trajeto através de marcações no chão.
Eles estão presentes principalmente nos ambientes industriais e possuem diversas funções, tais como, transporte de cargas de forma autônoma.

Partes para montagem

Arduino

Ponte H

Motores

Placa de sensores

Código

Controlador PID

Fonte de tensão

Motores

+ Chassi

MOTORES

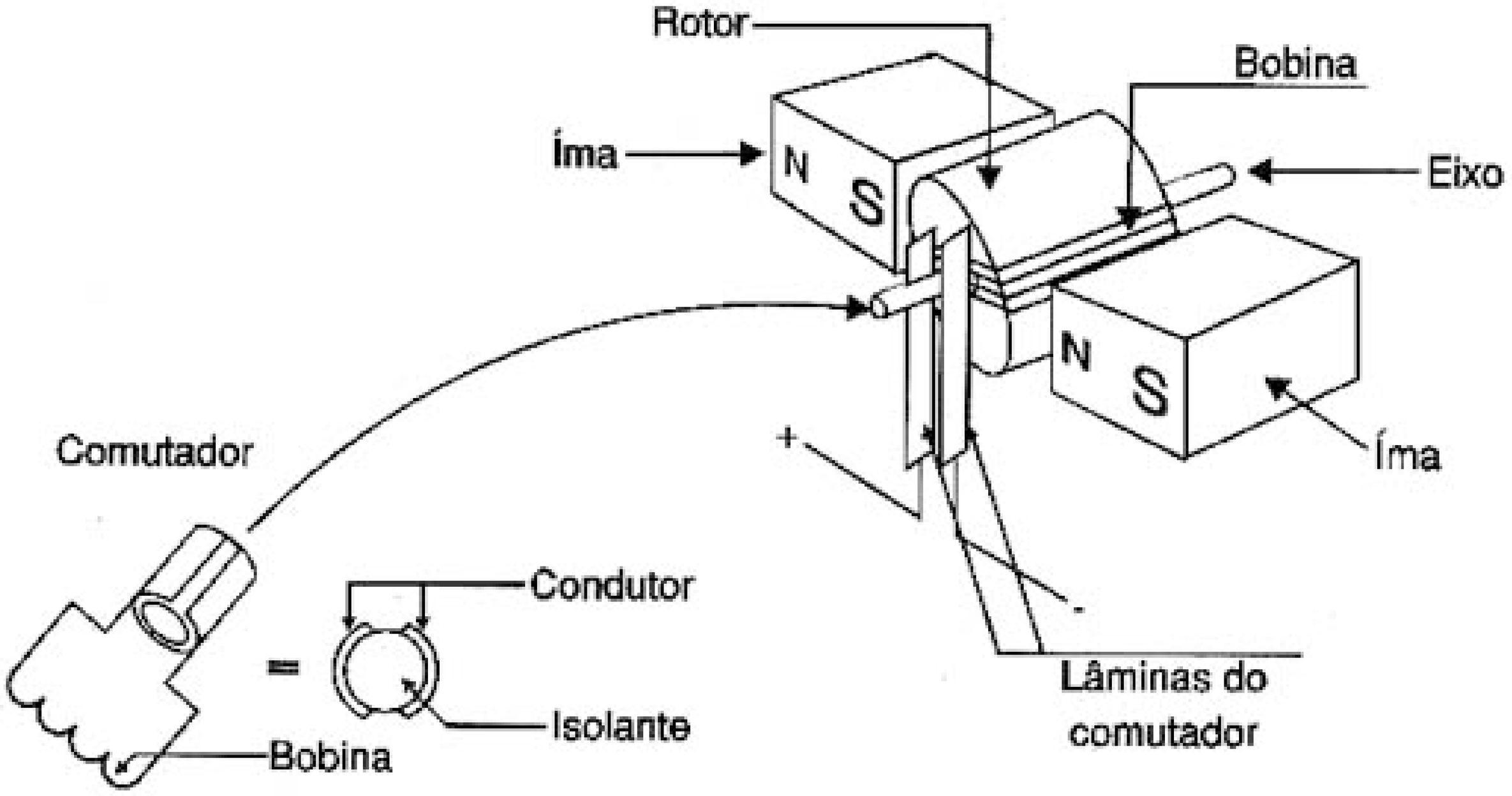
PARA QUE SERVE?

Os motores que dão mobilidade ao carrinho, são eles que movimentam o seguidor de linha pela pista.
3-6V - corrente contínua

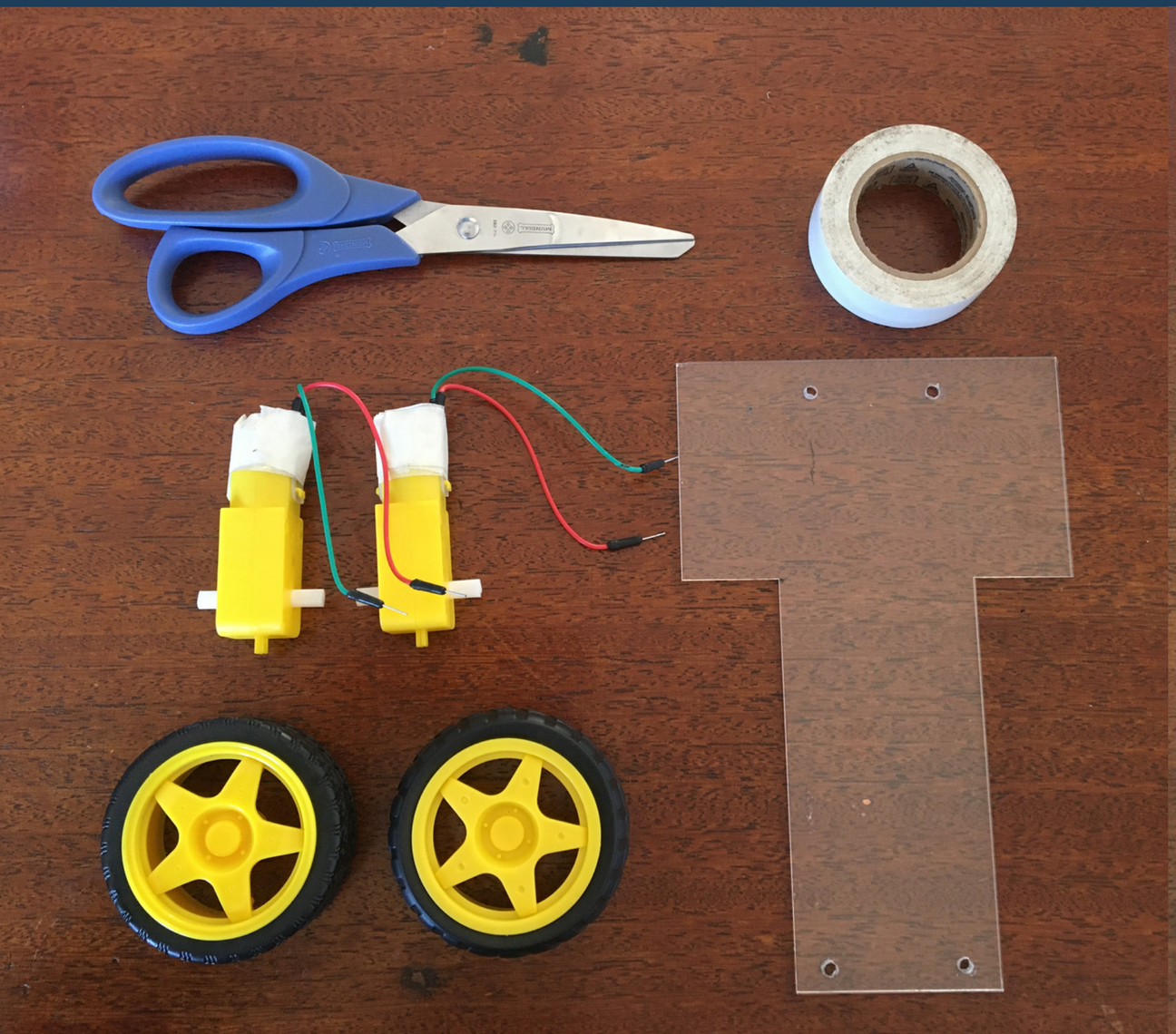
COMO FUNCIONA?

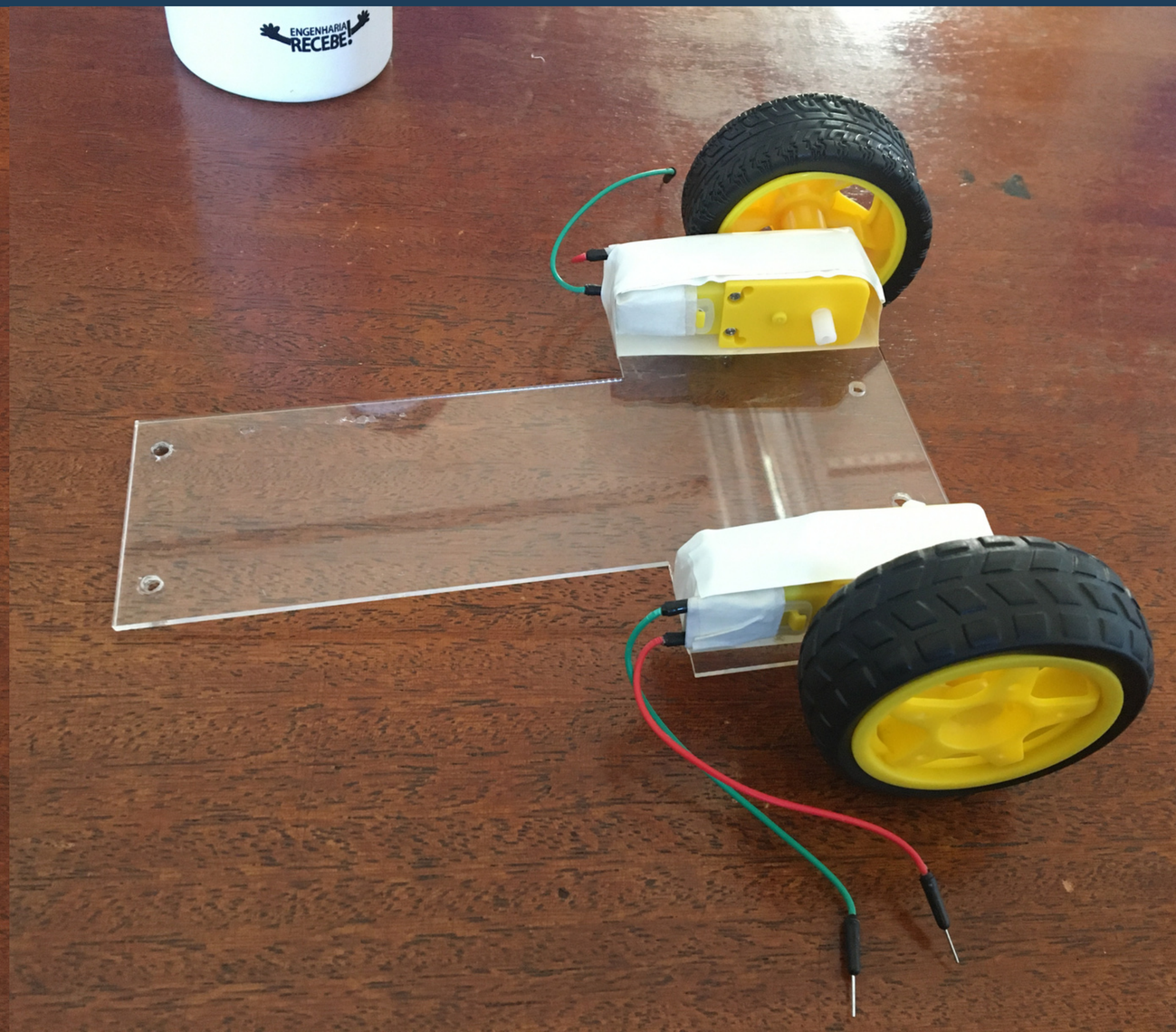
Ao ser percorrida por uma corrente, estão alinhados com o imã permanente temos a manifestação de uma força de repulsão.

Esta força de repulsão faz o conjunto móvel mudar de posição.
A tendência do rotor é dar meia volta para seu pólo Norte se aproxime do pólo Sul do imã permanente. Da mesma forma, seu pólo Sul se aproximará do pólo Norte pelo qual será atraído.
No entanto, no eixo do rotor, por onde passa a corrente que circula pela bobina, existe um comutador. A finalidade deste comutador é inverter o sentido da circulação da corrente na bobina, fazendo com que os pólo mudem.


Rotor
Bobina
Íma
N
S
Eixo
N
S
Íma
+
-
Comutador
Condutor
=
Isolante
Bobina
Lâminas do
comutador

MONTAGEM 1

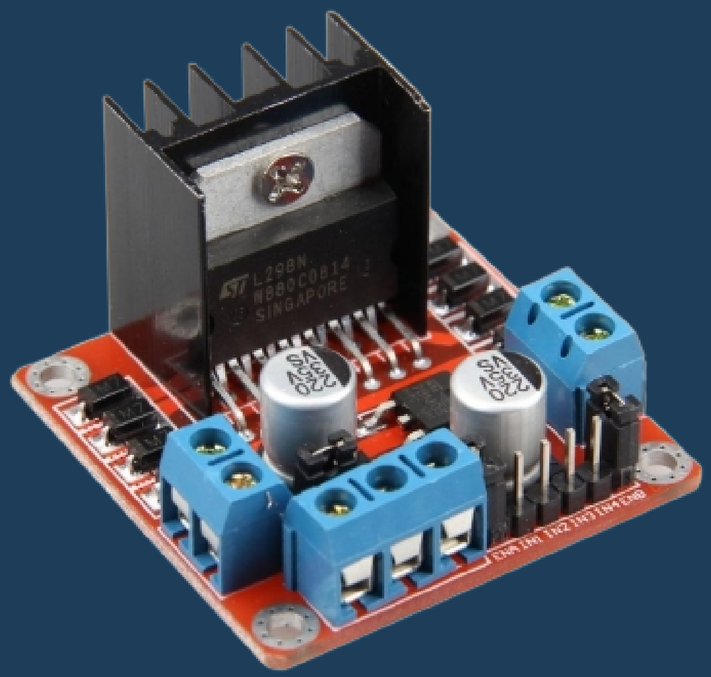




Ponte H

+ Placa de sensores

PONTE H

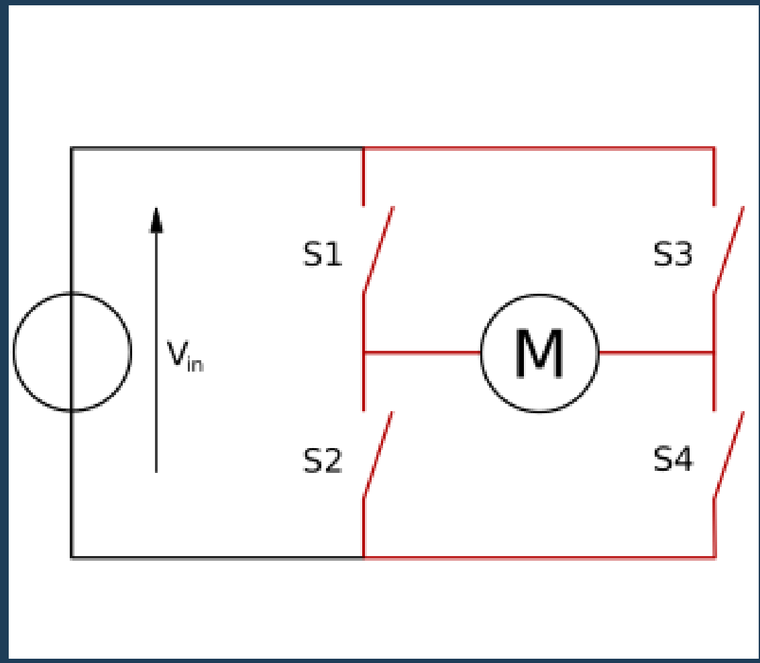


O QUE É ?

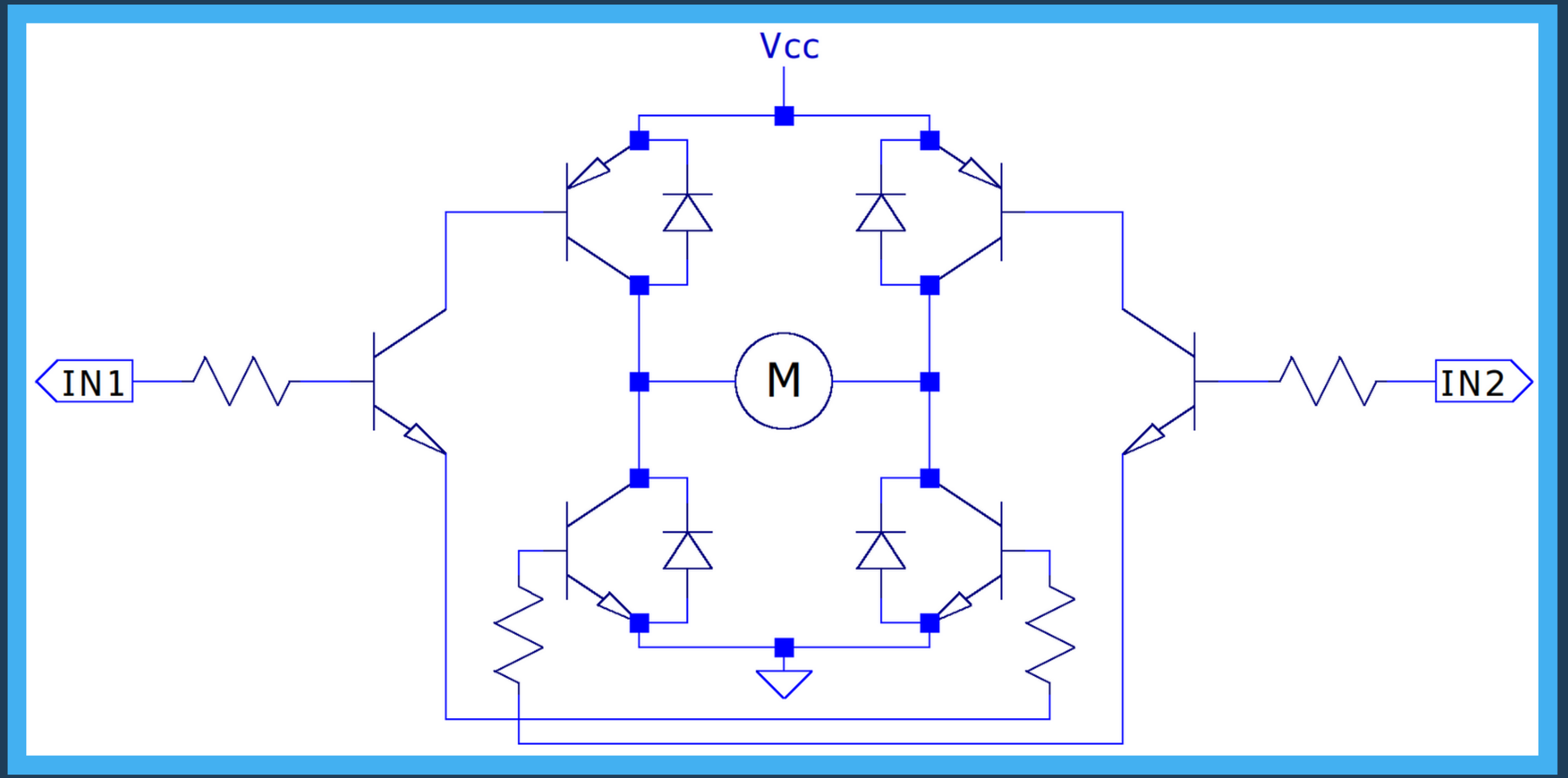
é um circuito de eletrônica de potência que pode determinar o sentido da corrente, a polaridade da tensão e a tensão em um dado sistema ou componente. Seu funcionamento dá-se pelo chaveamento de componentes eletrônicos usualmente utilizando do método de PWM para determinar além da polaridade, o módulo da tensão.

PARA QUE SERVE?

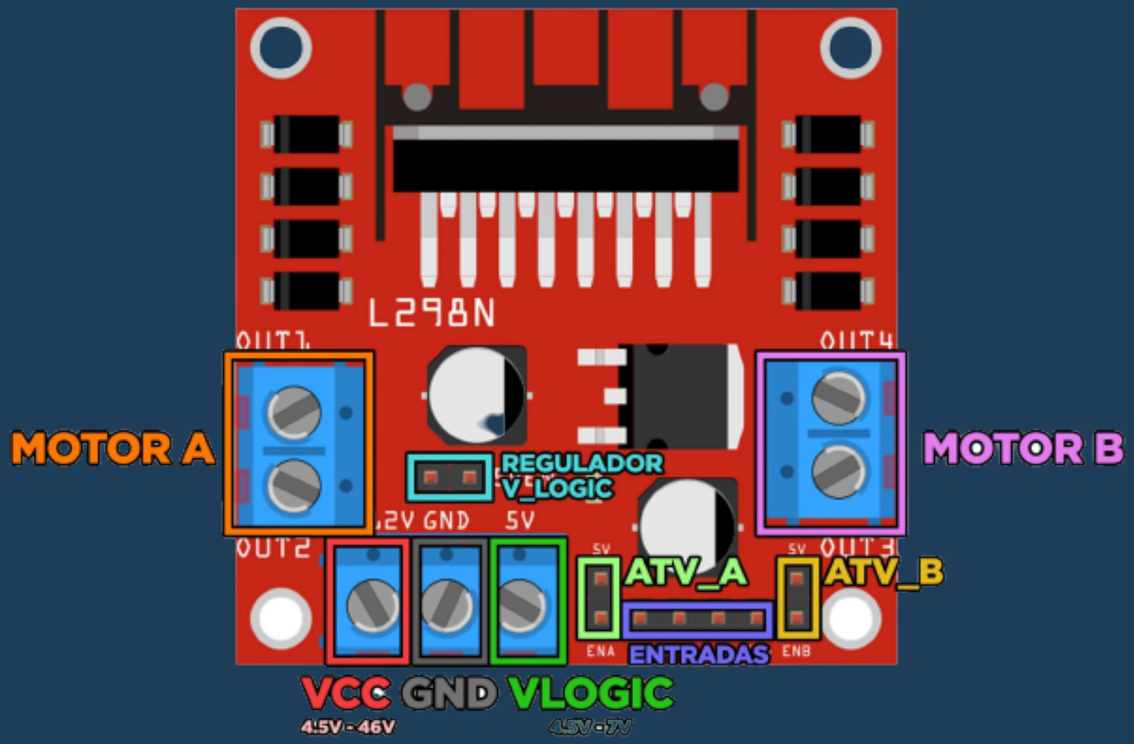
Tem como principal função o controle de velocidade e sentido de motores DC



PONTE H SIMPLIFICADA



PONTE H CONEXÕES



Vcc:

É responsável pela alimentação dos motores, com isso a tensão colocada neste borne de Vcc será copiada para os motores.

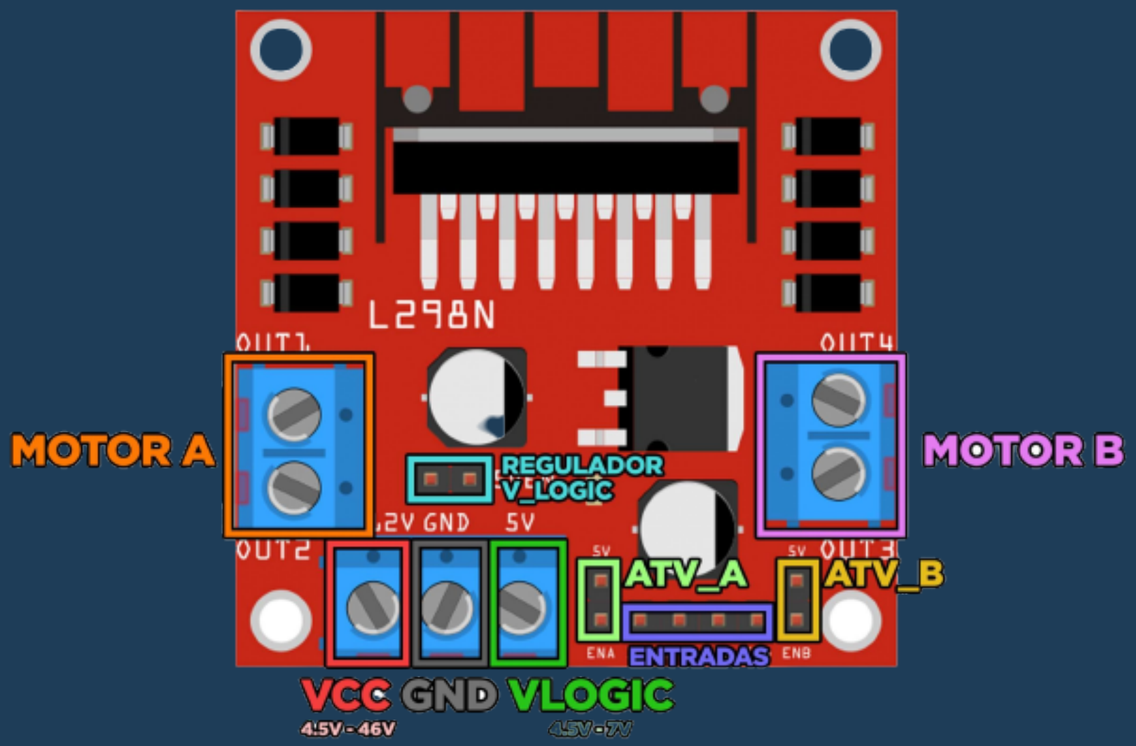
Vlogic:

Este borne é responsável pela alimentação lógica da ponte H.
Essa tensão é para uso interno do chip e pode ser aproveitada para alimentar algum circuito ou microcontrolador que use até 5V, com a limitação de corrente de aproximadamente 200mA.

Atv_A e Atv_B:

Responsáveis por desativar ou ativar os motores assim como suas velocidades. Devem ser conectados em portas PWM

PONTE H CONEXÕES



Entradas:

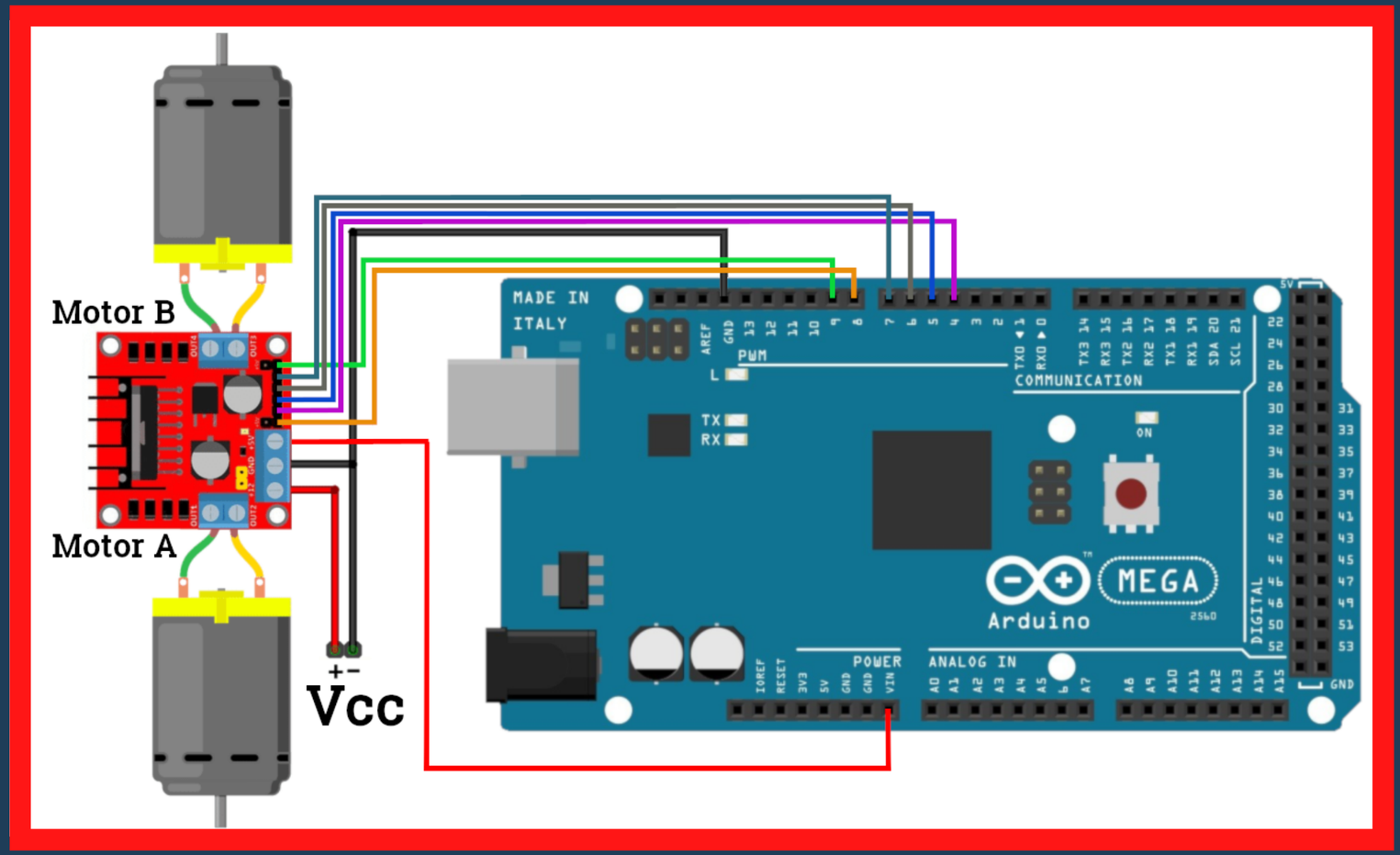
As entradas de controle da placa possuem os pinos IN1, IN2, IN3 e IN4, sendo que a entrada IN1 e IN2 são as entradas que controlam os bornes do motor A, e as entradas IN3 e IN4 controlam os bornes do Motor B. Para controlar os motores, acione os pinos da seguinte forma:

MOTOR A	IN1	IN2
Sentido direto	HIGH	LOW
Sentido reverso	LOW	HIGH
Freio	LOW	LOW
Freio	HIGH	HIGH

MOTOR B	IN3	IN4
Sentido direto	HIGH	LOW
Sentido reverso	LOW	HIGH
Freio	LOW	LOW
Freio	HIGH	HIGH

PONTE H CONEXÕES

Lembrete: verifique a polaridade dos motores



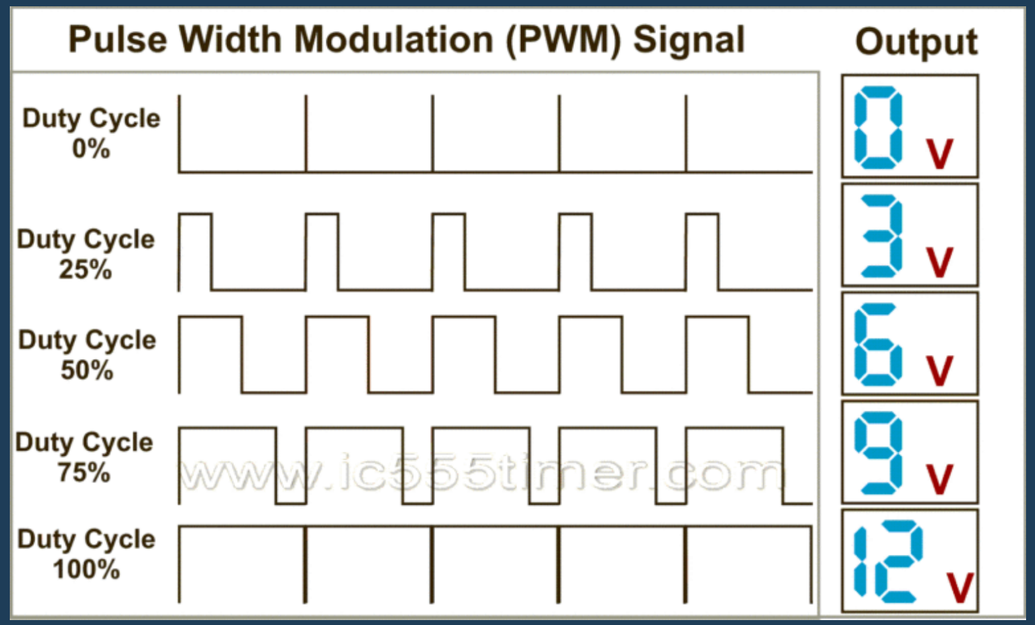
- Motor A e Motor B - Fios dos motores
- ENA e ENB- Portas PWM (8 e 9)
- IN1, IN2, IN3, IN4 - Portas digitais do Arduino (4, 5, 6 e 7)
- Vcc - Positivo da bateria
- GND - GND do Arduino e Negativo da bateria
- 5V/Vlogic - Porta Vin do Arduino

PWM

Pulse Width Modulation (Modulação de largura de pulso)

É uma técnica utilizada por sistemas digitais para variação do valor médio de uma forma de onda periódica.

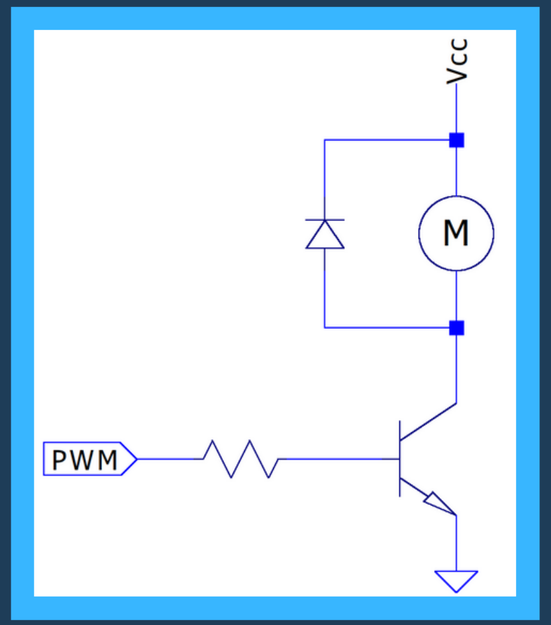
A técnica consiste em manter a frequência de uma onda quadrada fixa e variar o tempo que o sinal fica em nível lógico alto.
Esse tempo é chamado de duty cycle, ou seja, o ciclo ativo da forma de onda



As portas do arduino com ‘~’ são as portas pwm
Para controlar as portas no modo pwm se usa a função analogWrite().

PWM NO ARDUINO

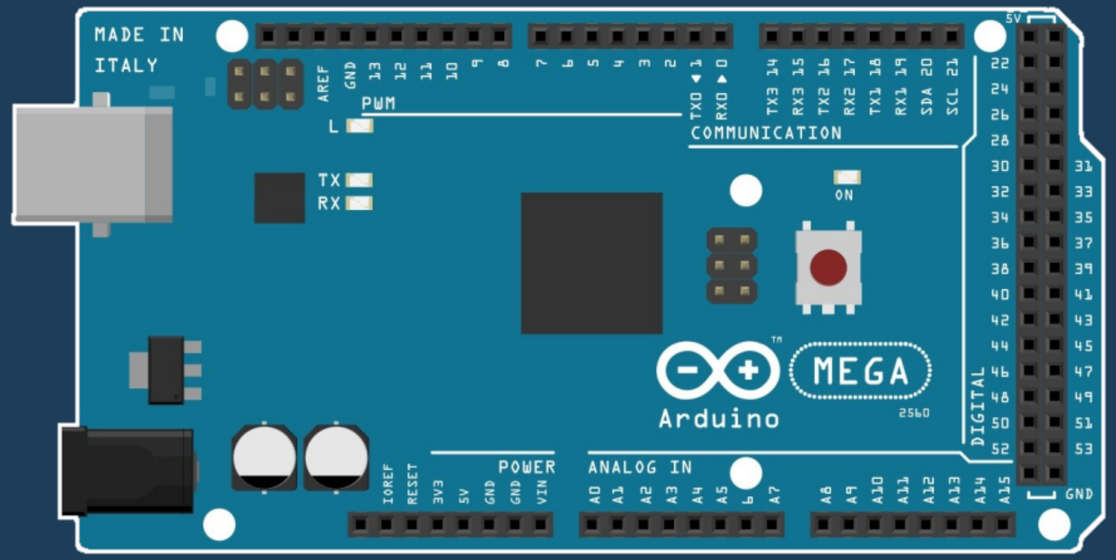
Utilizando a função analogWrite(int a) do arduino, podemos variar a largura do pulso com valores mapeados entre 0 e 255, mudando o valor do Duty Cycle.



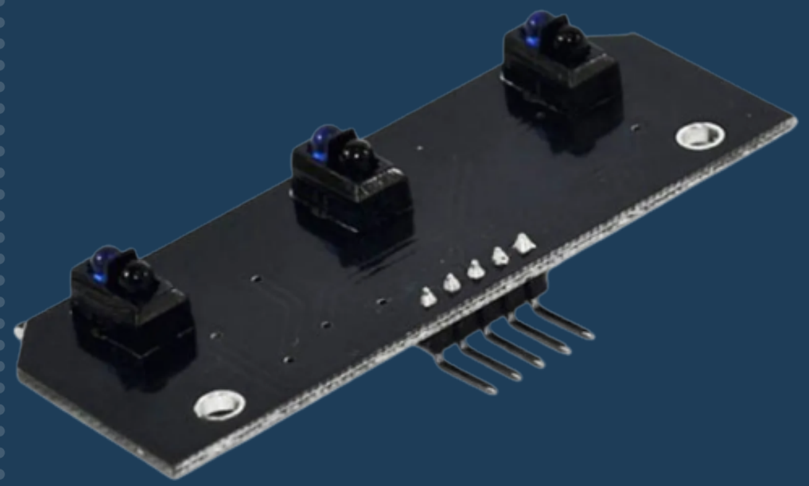
Exemplo:

Para montar uma onda de 50% Duty Cicle, usamos:

```
analogWrite(127);
```



PLACA DE SENSORES



O QUE É ?

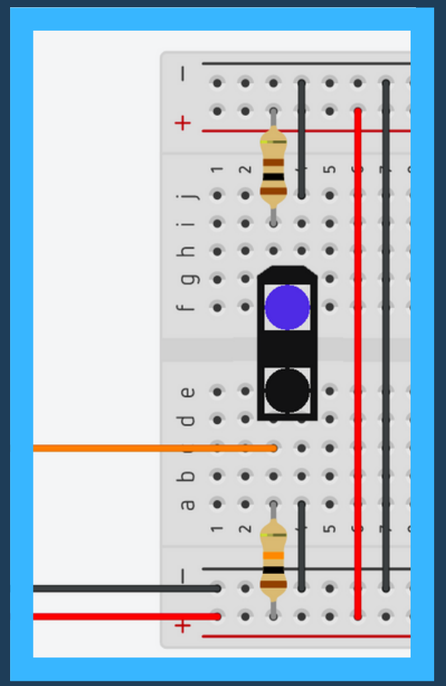
A placa de sensores é responsável por captar os estímulos da pista, ou seja, identificar a cor da pista logo abaixo dele.
Os sensores utilizados são os sensores Infravermelhos TCRT5000.
São utilizados 7 deles e soldados em uma placa para enviar os sinais captados e assim guiar as ações do seguidor.

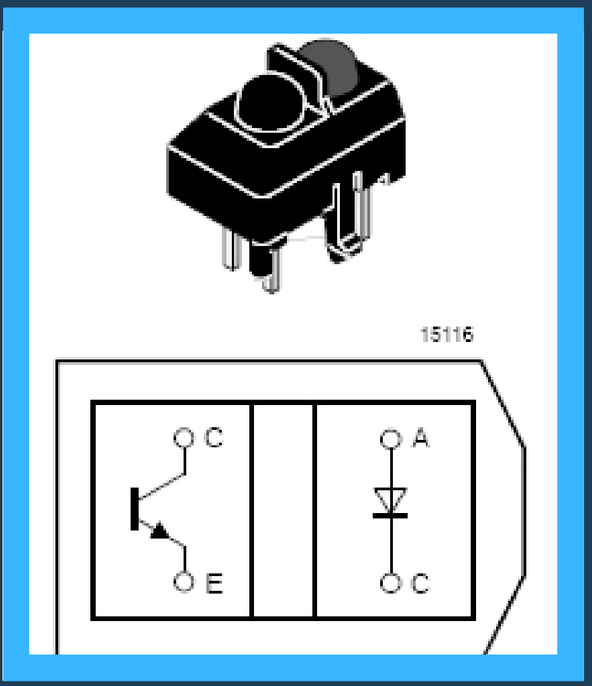
COMO FUNCIONA

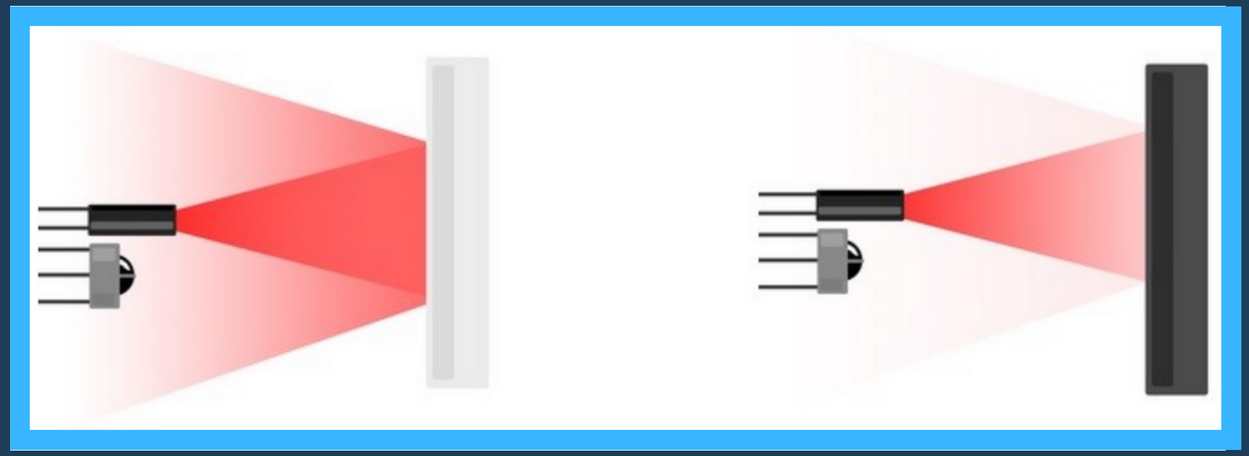
Possui dois componentes: um led que emite uma luz na faixa do infravermelho e um fototransistor que vai captar a luz refletida.
Ele vai ler 1 quando for preto e 0 quando for branco

SENSORES INFRAVERMELHOS

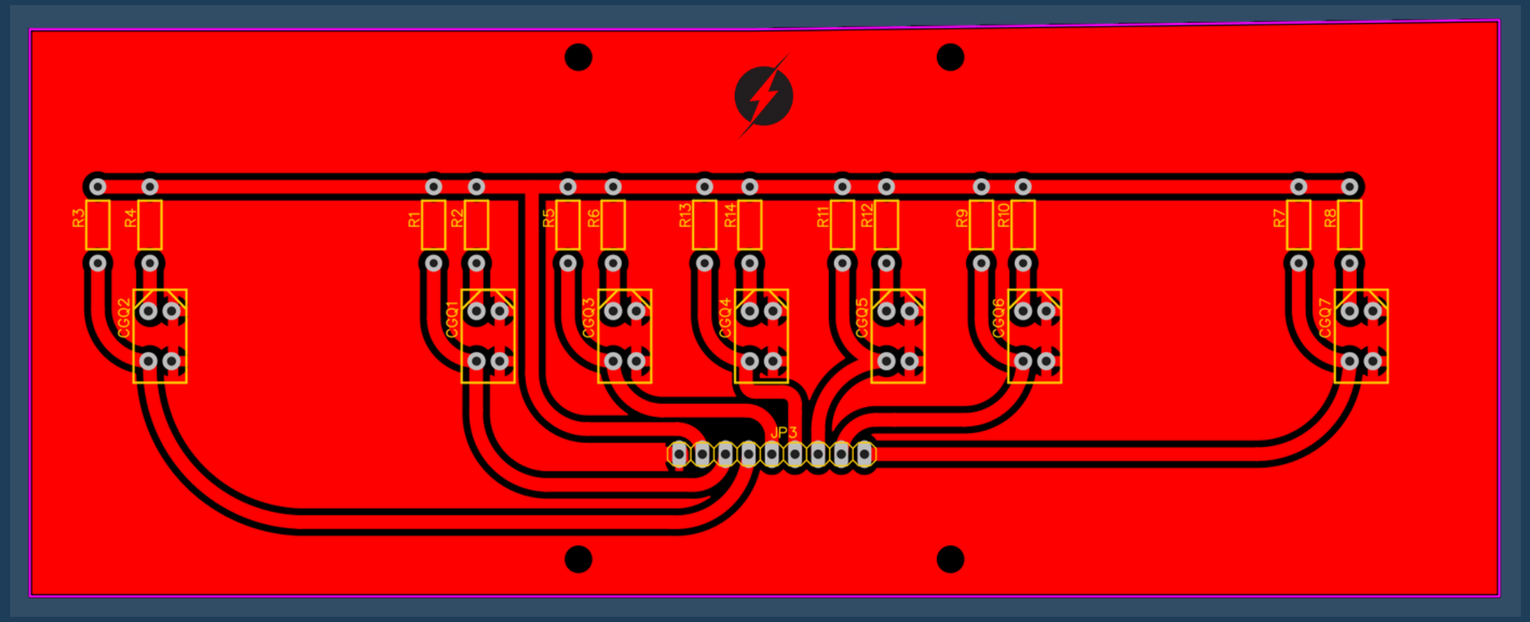
- Quando o anteparo é branco, o transistor fotosensível conduz e o sinal de saída é igual a 0V;
- Quando o anteparo é preto, o transistor não conduz, então o sinal de saída é igual a 5V.







PLACA DE SENSORES

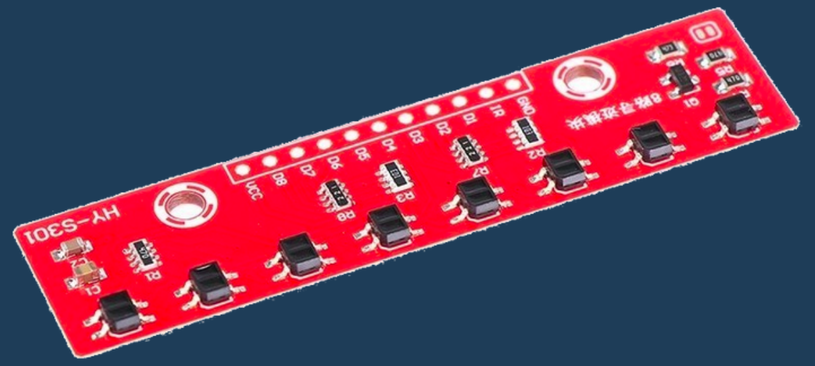


ETAPAS DE FABRICAÇÃO DA PLACA

- Impressão (Papel Fotográfico Glossy);
- Transferência térmica para placa de fenolite;
- Corrosão com Percloreto de Ferro;
- Soldagem dos componentes.

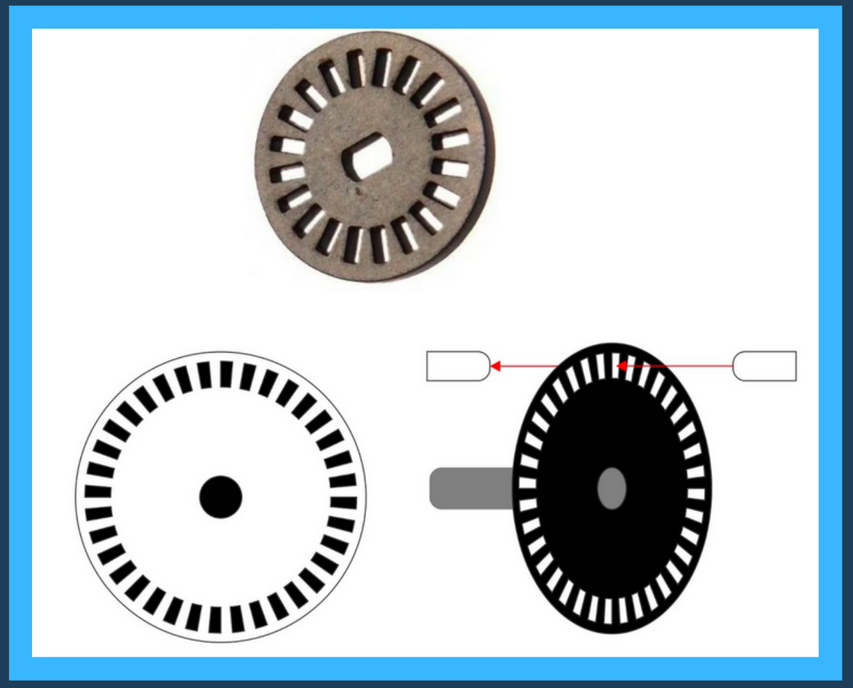
OUTROS SENSORES

SENSORES DE POSIÇÃO COMERCIAIS



- Módulo QTR Sensors

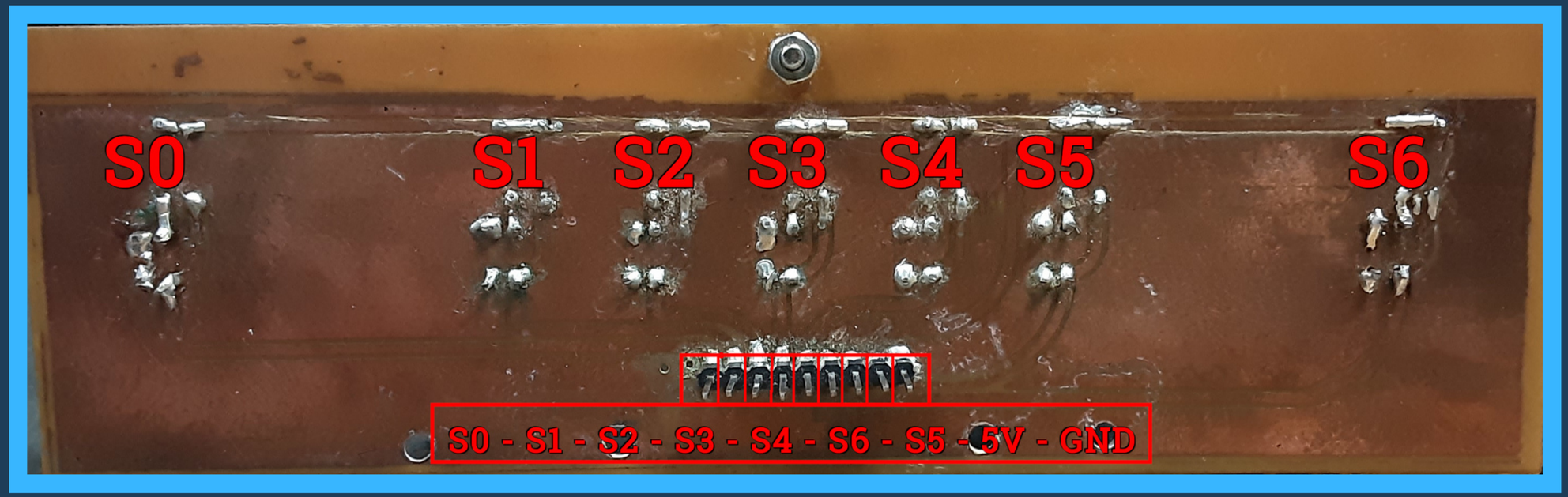
SENSOR DE VELOCIDADE



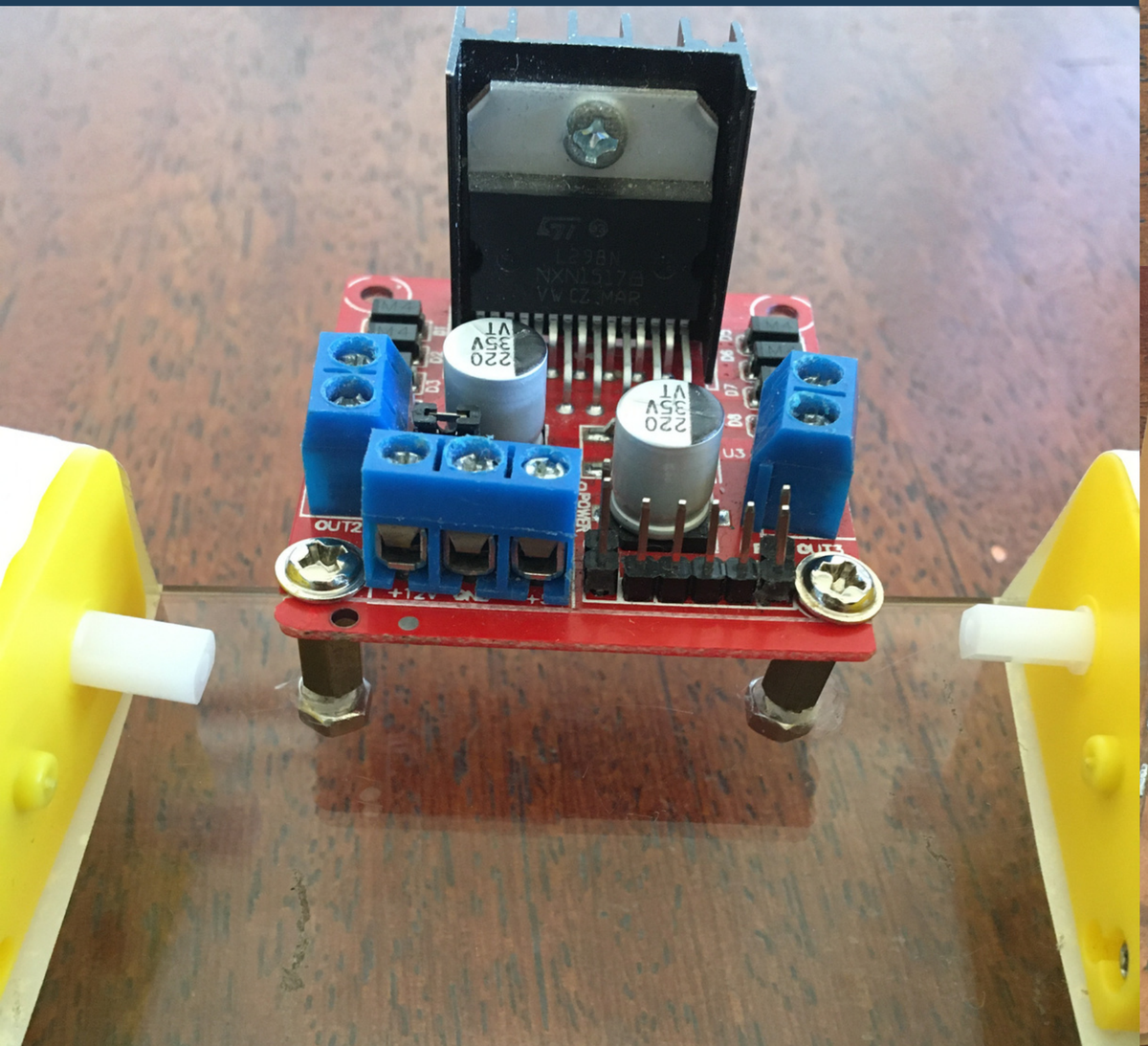
- Encoders para medir RPM;
- Módulo infravermelho.

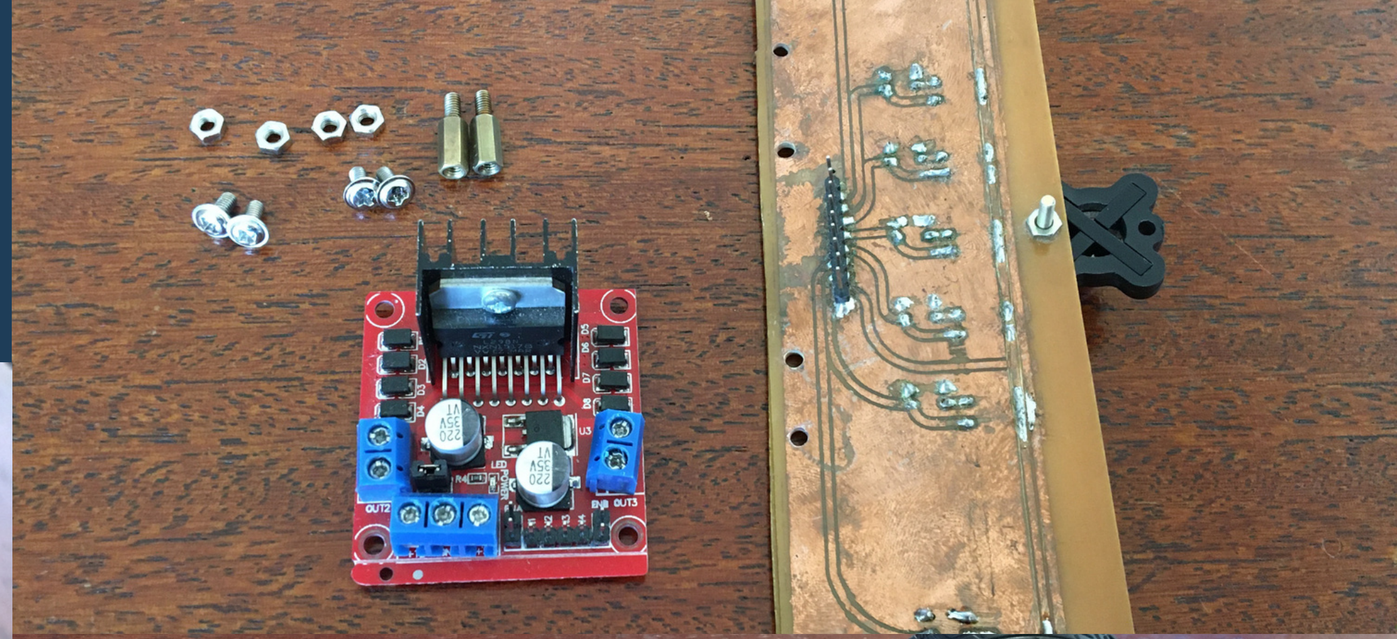
PLACA DE SENSORES
CONEXÕES

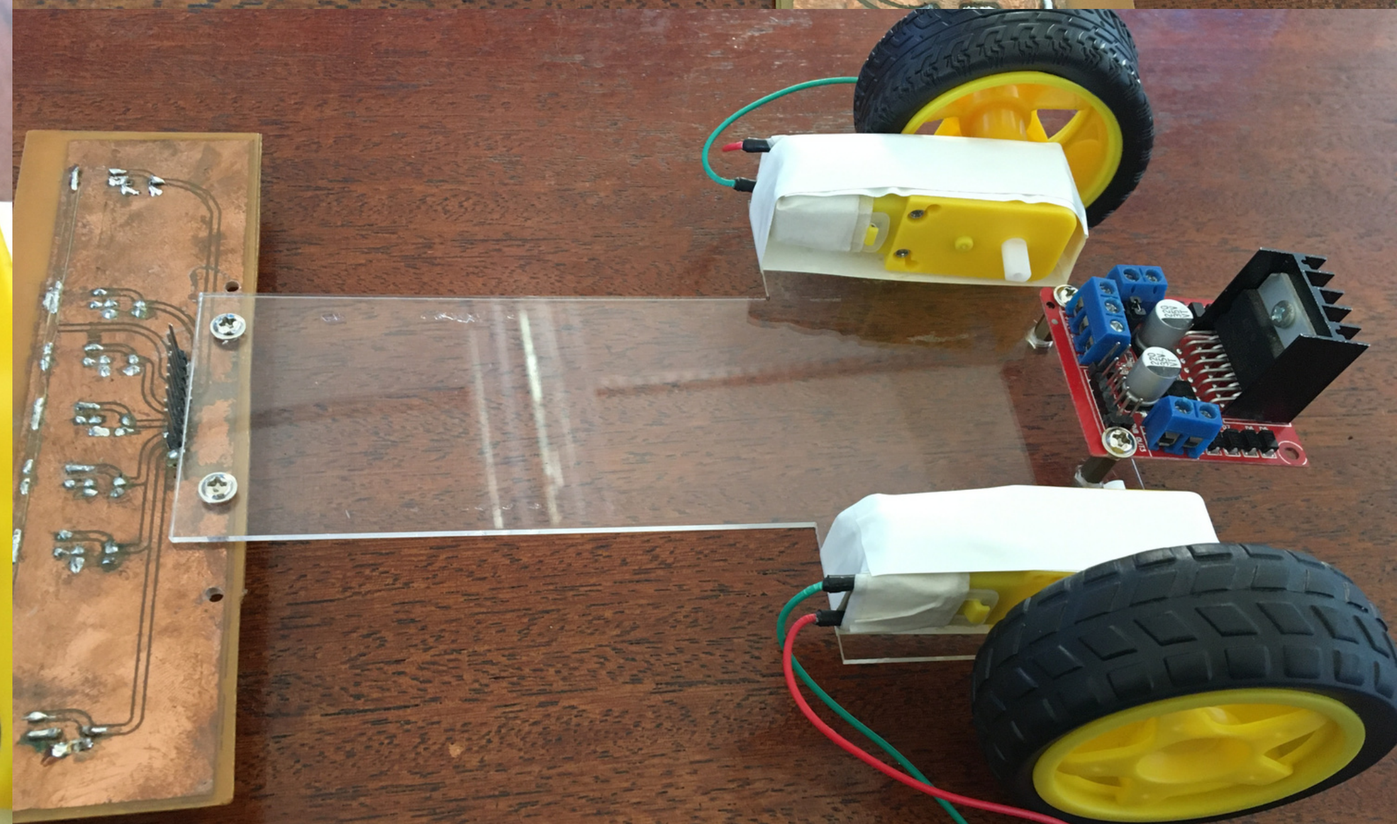
- Pinos 1-7 - Portas digitais do Arduino
  - (33, 35, 37, 39, 41, 45, 43)
- Pino 8 - 5V do Arduino
- Pino 9 - **GND** do Arduino



MONTAGEM 2



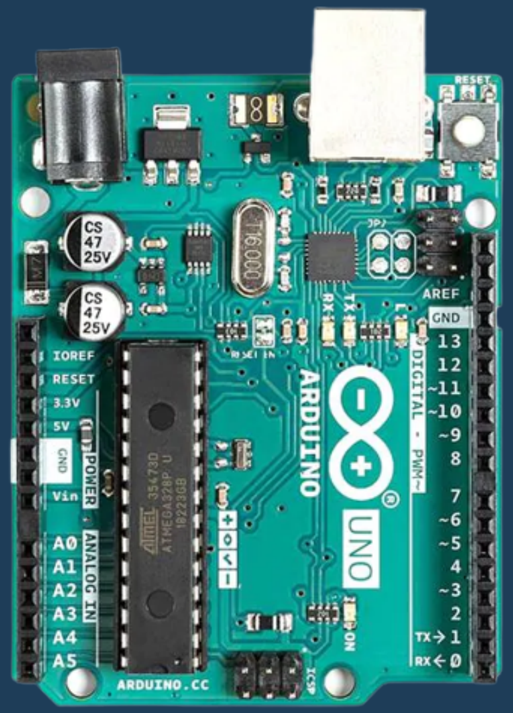




Arduino

+ Bateria

ARDUINO



O QUE É ?

Arduino é uma plataforma de prototipagem eletrônica de hardware livre e de placa única, projetada com um microcontrolador Atmel AVR com suporte de entrada/saída embutido, uma linguagem de programação padrão, a qual tem origem em Wiring, e é essencialmente C/C++.

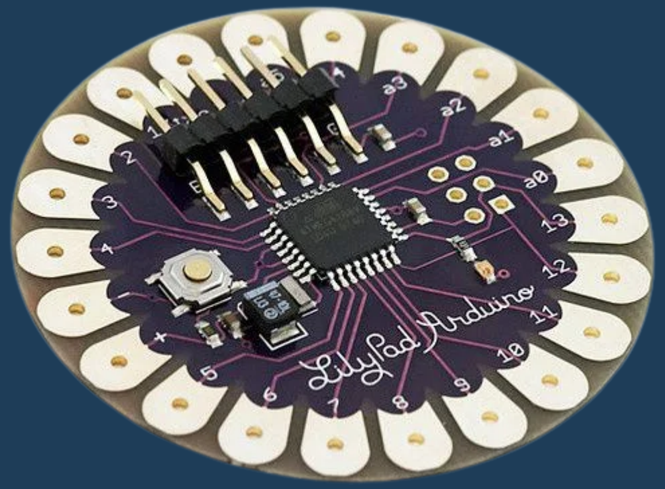
PARA QUE SERVE?

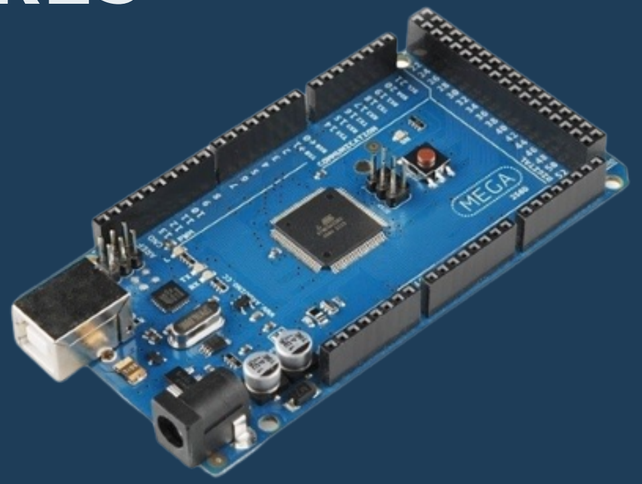
Tem a função de controlar o seguidor de linha de acordo com as entradas dos sensores

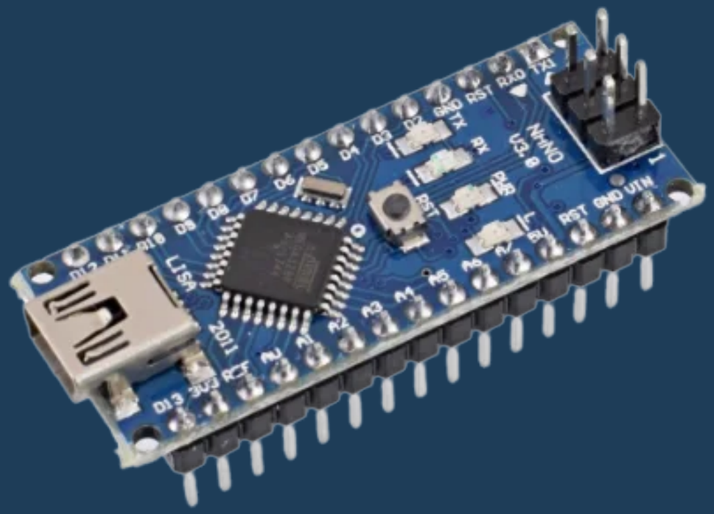
ARDUINO MEGA 2560

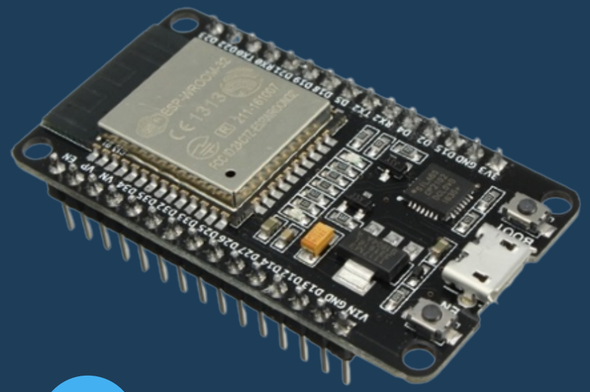
É o modelo utilizado nesta oficina devido à sua maior quantidade de portas, memória e processamento

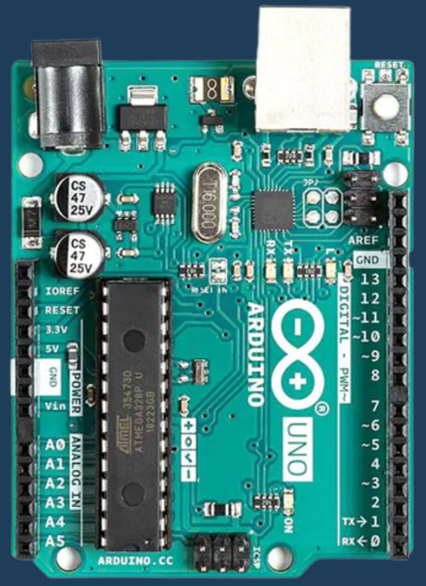
MICROCONTROLADORES

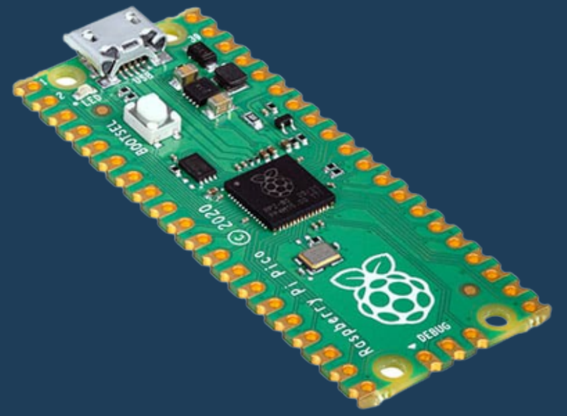












FONTE DE TENSÃO

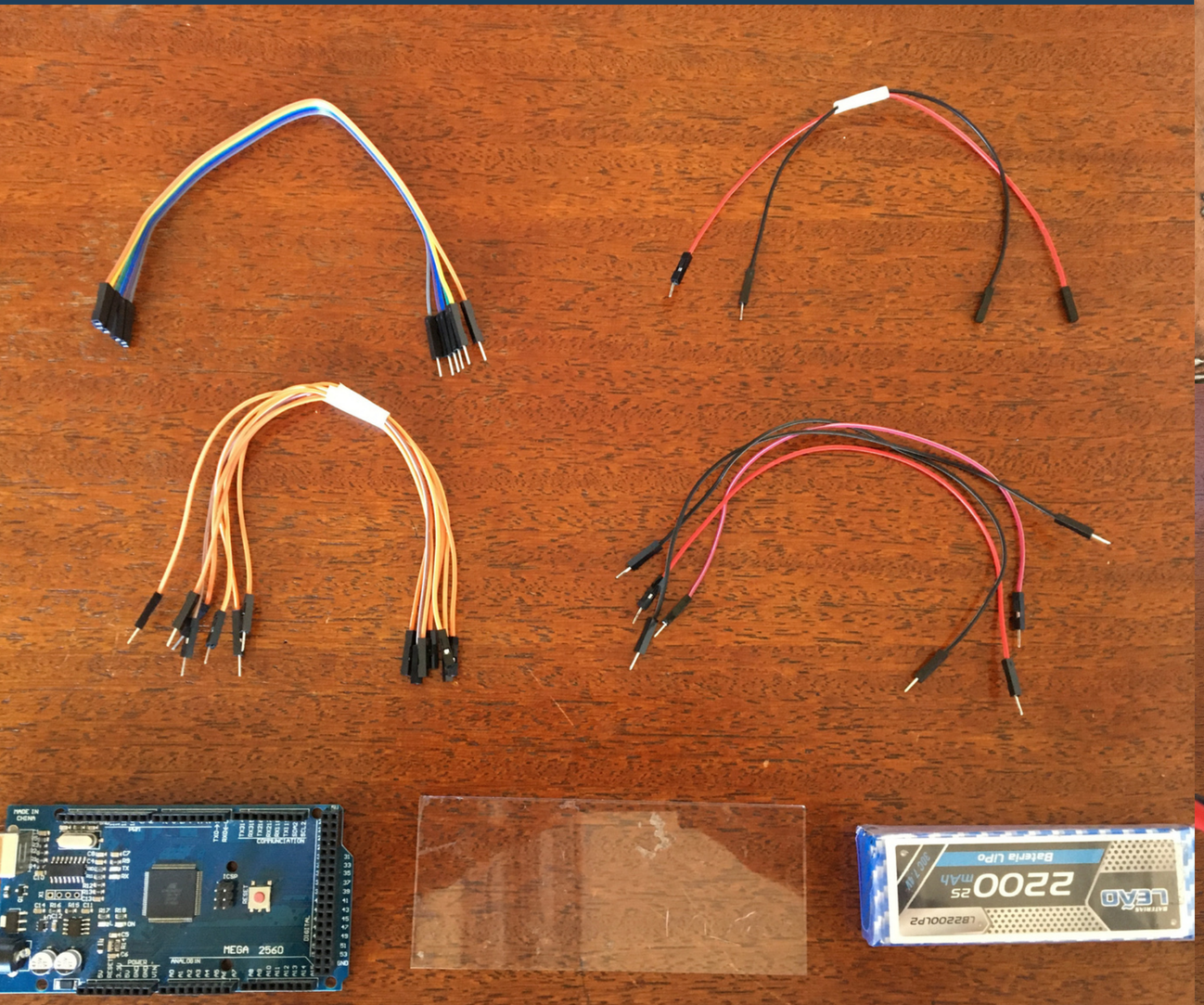
O QUE É

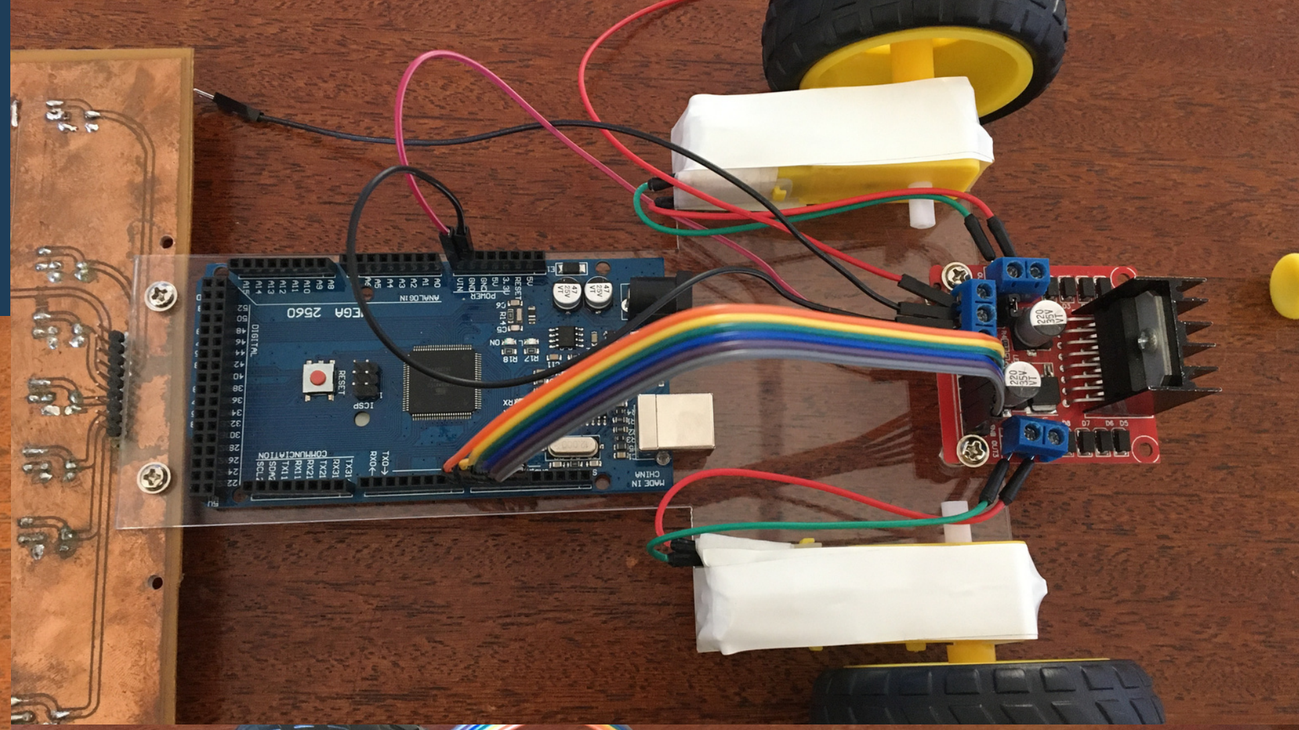
É uma bateria de lítio que possui em sua saída aproximadamente 7.4V e capacidade de 2200mAh

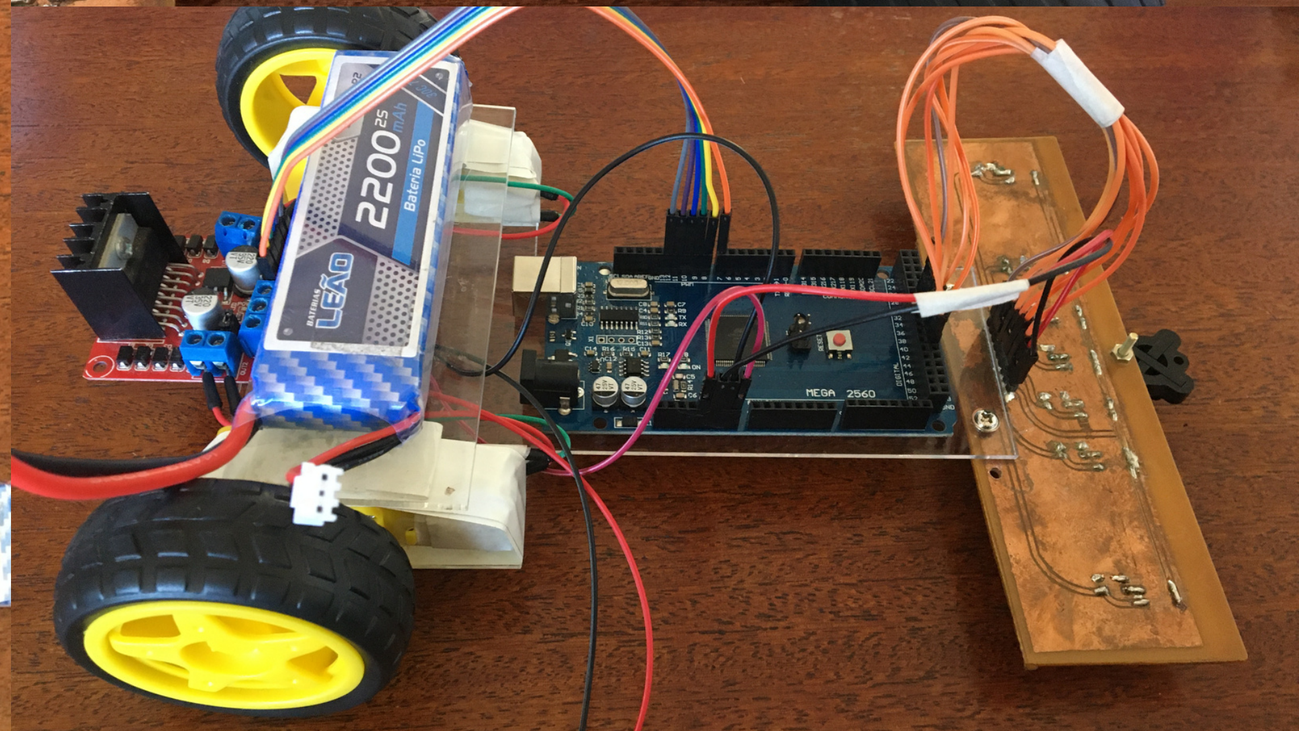
PARA QUE SERVE?

A fonte de tensão vai alimentar os circuitos do seguidor e possibilitar que eles funcionem

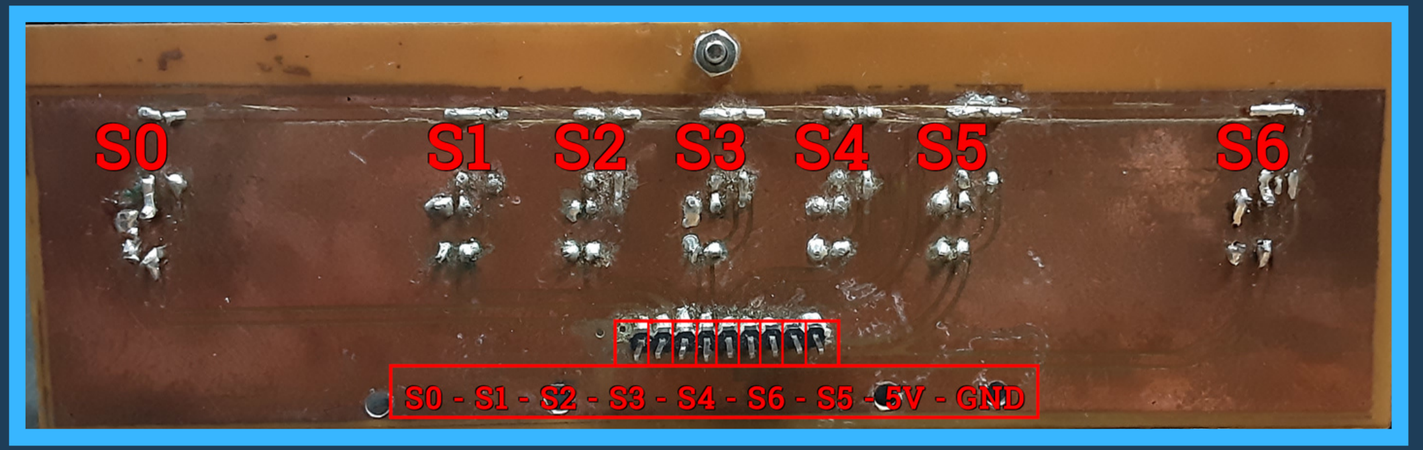
MONTAGEM 3


MEGA 2560
LEAO
2200 25 mAh
Bateria LiPo

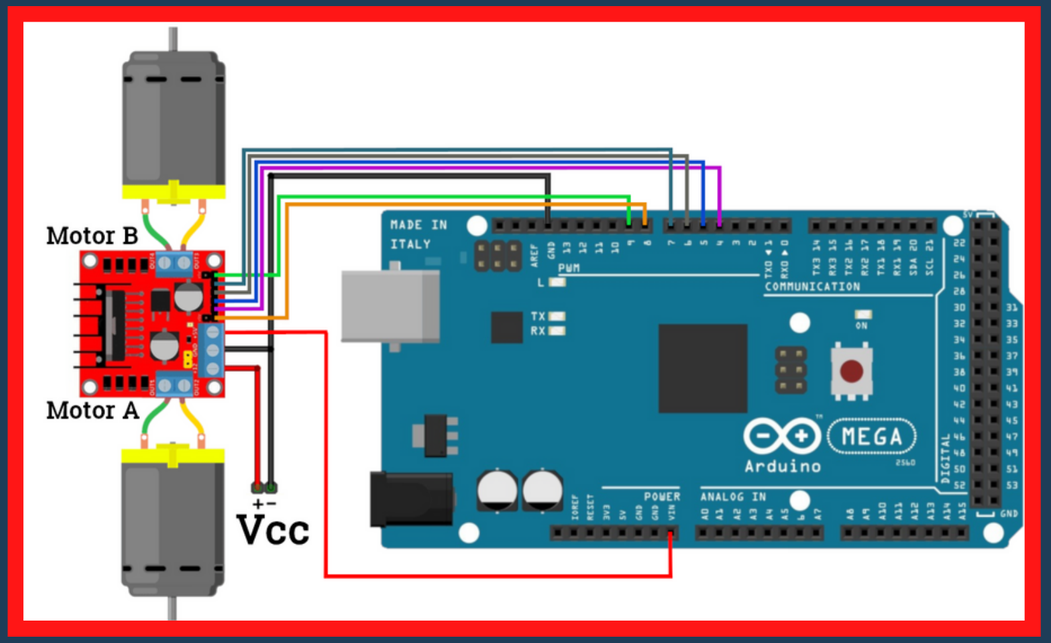

MEGA 2560


LEAO
2200 25 mAh
Bateria LiPo
MEGA 2560

- S0 - S6 - Portas digitais do Arduino
  - (33, 35, 37, 39, 41, 45, 43)
- 5V - 5V do Arduino
- GND - GND do Arduino



- Motor A e Motor B - Fios dos motores
- ENA e ENB- Portas PWM (8 e 9)
- IN1, IN2, IN3, IN4 - Portas digitais do Arduino (4, 5, 6 e 7)
- Vcc - Positivo da bateria
- GND - GND do Arduino e Negativo da bateria
- 5V/Vlogic - Porta Vin do Arduino



Código

+ Controle PID

PARTES ESSENCIAIS

1

Definição das portas e variáveis

Definir as portas de entradas dos sensores, as portas de saída para controlar a ponte H e as velociades dos motores

2

Iniciação das portas

Na função Setup as portas devem ser iniciadas através do pinMode()

3

Criar funções

Desenvolver funções para facilitar a leitura do código, as funções irão realizar diversas tarefas propostas pelo desenvolvedor como: curva 90°, curva leve, curva acentuada

4

Leitura dos sensores

O código deve constantemente ler os sensores para definir quais funções se deve chamar

Definição das portas e variáveis

```
int IN1 = 4;
int IN2 = 5;
int IN3 = 6;
int IN4 = 7;
int PWM_A = 8;
int PWM_B = 9;

int sensor0 = 33;
int sensor1 = 35;
int sensor2 = 37;
int sensor3 = 39;
int sensor4 = 41;
int sensor5 = 43;
int sensor6 = 45;

int sensor[7];
int vel_A = 200;
int vel_B = 255;
```

Iniciando as portas utilizadas

```
void setup() {
 pinMode (IN1, OUTPUT);
 pinMode (IN2, OUTPUT);
 pinMode (IN3, OUTPUT);
 pinMode (IN4, OUTPUT);
 pinMode (PWM_A, OUTPUT);
 pinMode (PWM_B, OUTPUT);
 pinMode (sensor0, INPUT);
 pinMode (sensor1, INPUT);
 pinMode (sensor2, INPUT);
 pinMode (sensor3, INPUT);
 pinMode (sensor4, INPUT);
 pinMode (sensor5, INPUT);
 pinMode (sensor6, INPUT);
}
```

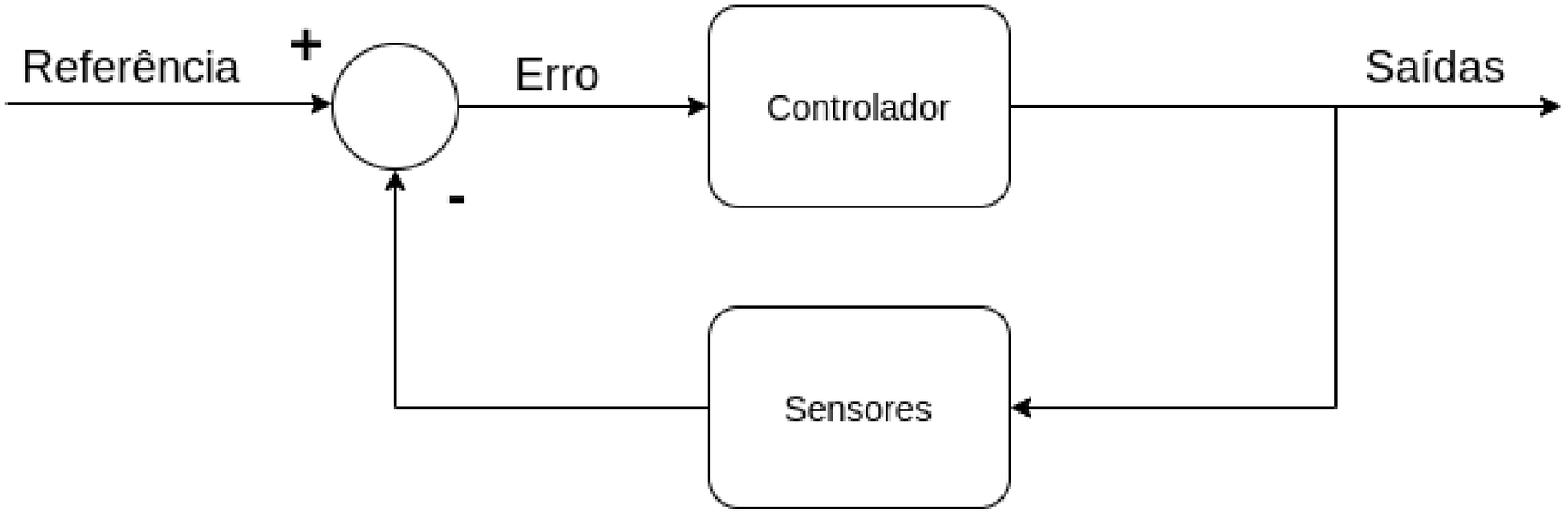
Exemplo de controle de velocidade

```
//CURVA LEVE À DIREITA
else if(sensor[0]==1 && sensor[1]==1 && sensor[2]==1 && sensor[3]==0 && sensor[4]==0 && sensor[5]==1 && sensor[6]==1){
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(PWM_A, vel_A*(0.5));
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(PWM_B, vel_B*(0.8));
}

//CURVA LEVE À ESQUERDA
else if(sensor[0]==1 && sensor[1]==1 && sensor[2]==0 && sensor[3]==0 && sensor[4]==1 && sensor[5]==1 && sensor[6]==1){
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  analogWrite(PWM_A, vel_A*(0.8));
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  analogWrite(PWM_B, vel_B*(0.5));
}
```

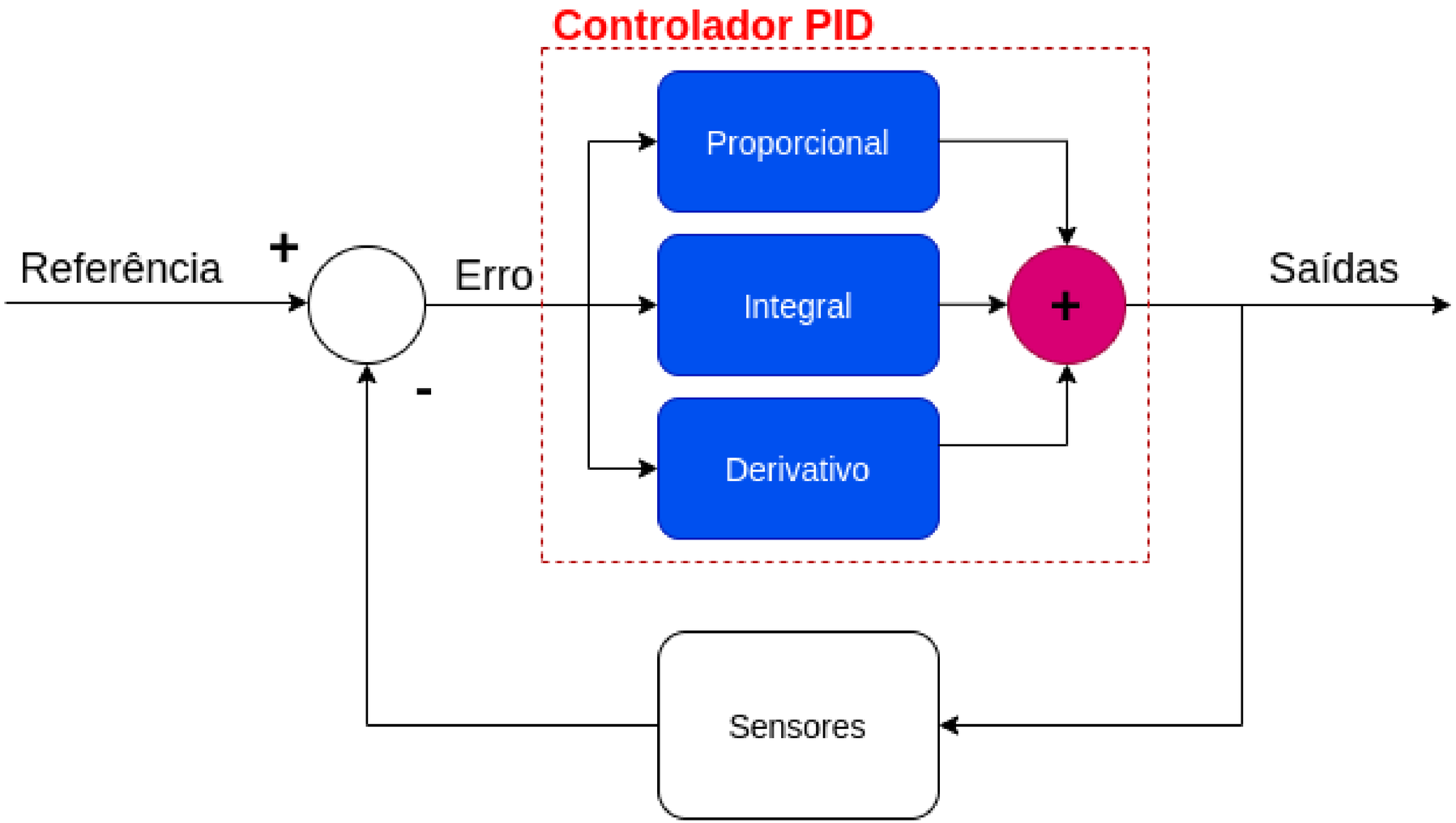
Controle PID

Definições



Controle PID

Definições

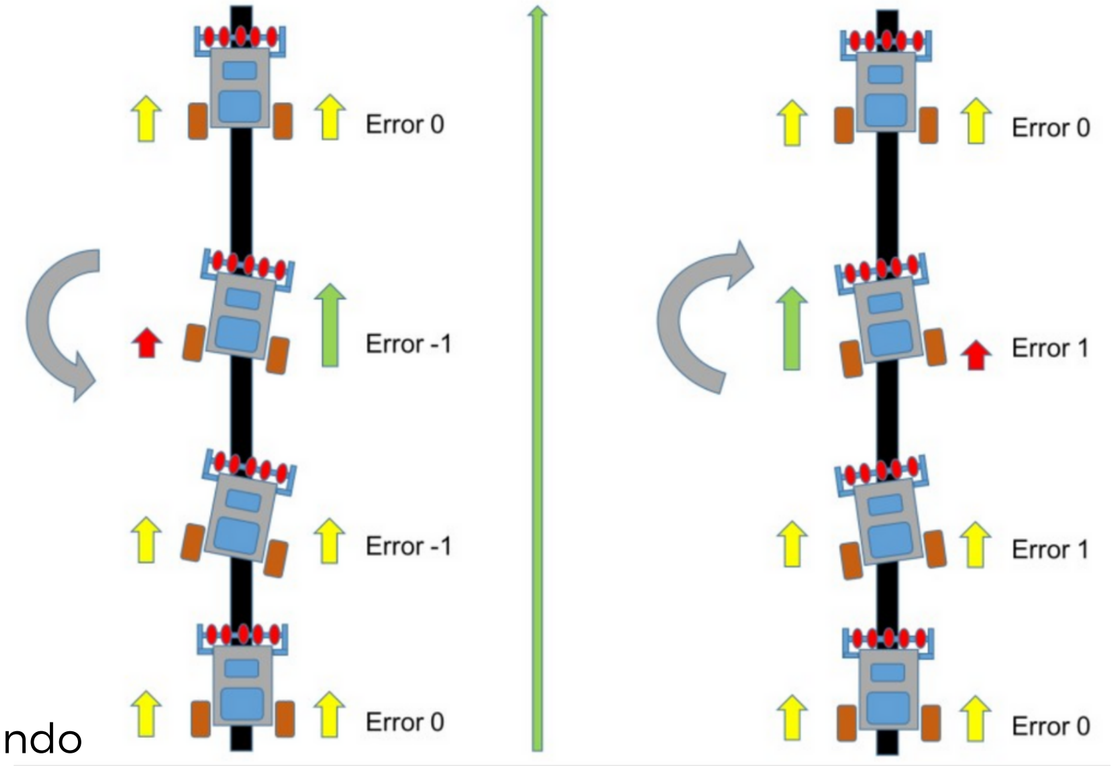


Controle PID

Calculando o Erro

A variável erro, relacionada com o estado dos sensores, será:

- 1 1 1 1 0 => erro = -4
- 1 1 1 0 0 => erro = -3
- 1 1 1 0 1 => erro = -2
- 1 1 0 0 1 => erro = -1
- 1 1 0 1 1 => erro = 0
- 1 0 0 1 1 => erro = 1
- 1 0 1 1 1 => erro = 2
- 0 0 1 1 1 => erro = 3
- 0 1 1 1 1 => erro = 4



O sistema calcula o “erro” da quantidade física em relação à referência, medindo o valor atual dessa quantidade física usando um sensor. Para voltar à referência, este “erro” deve ser minimizado, idealmente igual a zero.

Controle PID

Calculando o Erro

Precisamos então de uma função que calcule o erro

```
void calcula_erro(){
  if ((sensor[0]==1) && (sensor[1]==1) && (sensor[2]==1) && (sensor[3]==0) && (sensor[4]==1) && (sensor[5]==1) && (sensor[6]==1)){ erro = 0;}
  else if ((sensor[0]==1) && (sensor[1]==1) && (sensor[2]==1) && (sensor[3]==0) && (sensor[4]==0) && (sensor[5]==1) && (sensor[6]==1)){ erro = 1;}
  else if ((sensor[0]==1) && (sensor[1]==1) && (sensor[2]==0) && (sensor[3]==0) && (sensor[4]==1) && (sensor[5]==1) && (sensor[6]==1)){ erro = -1;}
  else if ((sensor[0]==1) && (sensor[1]==1) && (sensor[2]==1) && (sensor[3]==1) && (sensor[4]==0) && (sensor[5]==1) && (sensor[6]==1)){ erro = 2;}
  else if ((sensor[0]==1) && (sensor[1]==1) && (sensor[2]==0) && (sensor[3]==1) && (sensor[4]==1) && (sensor[5]==1) && (sensor[6]==1)){ erro = -2;}
  else if ((sensor[0]==1) && (sensor[1]==1) && (sensor[2]==1) && (sensor[3]==1) && (sensor[4]==0) && (sensor[5]==0) && (sensor[6]==1)){ erro = 3;}
  else if ((sensor[0]==1) && (sensor[1]==0) && (sensor[2]==0) && (sensor[3]==1) && (sensor[4]==1) && (sensor[5]==1) && (sensor[6]==1)){ erro = -3;}
  else if ((sensor[0]==1) && (sensor[1]==1) && (sensor[2]==1) && (sensor[3]==1) && (sensor[4]==1) && (sensor[5]==0) && (sensor[6]==1)){ erro = 4;}
  else if ((sensor[0]==1) && (sensor[1]==0) && (sensor[2]==1) && (sensor[3]==1) && (sensor[4]==1) && (sensor[5]==1) && (sensor[6]==1)){ erro = -4;}
}
```

Controle PID

Termos P, I e D

$$PID = K_P\, e(t) + K_I \int_0^t e(t)dt + K_D \frac{de(t)}{dt}$$

- Kp, Ki e Kd são coeficientes constantes;
- Resta agora calcular os termos relacionados ao erro e encontrar o valor do sinal de controle.

Controle PID

Termo Proporcional

- Este termo é proporcional ao erro;
- É responsável pela magnitude da mudança necessária na quantidade física para atingir o ponto de ajuste;
- O termo proporcional é o que determina o tempo de subida da malha de controle ou o quão rápido ele vai retornar à referência

$$P = erro$$

Controle PID

Termo Integrativo

- Este termo é a soma de todos os valores de erro anterior;
- Este valor é o responsável pela rapidez de resposta do sistema para a mudança do ponto de ajuste. O termo integral é utilizado para eliminar o erro de estado;
- Normalmente, pequenos robôs não usam o termo integral porque não estamos preocupados com erro de estado estacionário e isso pode complicar o ajuste.

$$I = I + erro$$

Controle PID

Termo Derivativo

- Este termo é a diferença entre o erro instantâneo em relação à referência e o erro a partir do instante anterior;
- Este valor é responsável por diminuir (“slow down”) a taxa de variação da quantidade física quando nos aproximamos do ponto de ajuste;
- O termo derivativo é utilizado a reduzir o “overshoot” ou o quanto o sistema “super corrige”.

D = erro - erro_anterior

Controle PID

Calculando o Sinal de Controle

```
void calcula_PID(){
  if (erro==0){
    I = 0;
  }
  P = erro;
  I = I + erro;
  if (I > 255){
    I = 255;
  }
  else if (I < -255){
    I = -255;
  }
  D = erro - erro_anterior;
  PID = (Kp*P) + (Ki*I) + (Kd*D);
  erro_anterior = erro;
}
```

$$PID = K_P\, e(t) + K_I \int_0^t e(t)dt + K_D \frac{de(t)}{dt}$$

Controle PID

Malha de Realimentação e Saídas

```
void controle_motor(){
  if (PID >= 0){
    velesq = vel_B;
    veldir = vel_A - PID;
  }
  else{
    velesq = vel_B + PID;
    veldir = vel_A
    ;
  }
  digitalWrite (IN1, HIGH);
  digitalWrite (IN2, LOW);
  digitalWrite (IN3, HIGH);
  digitalWrite (IN4, LOW);
  analogWrite (PWM_A, veldir);
  analogWrite (PWM_B, velesq);
}
```

Desafios

EXTRAS

CoRA 2022

A maior competição de robótica da UFMG!

28, 29 e 30 de Setembro de 2022

Inscrições aqui

Agradecemos

pela atenção!

Caio Teraoka
Graduando em Engenharia Elétrica
caioteraoka.petee@gmail.com

Pedro Pires
Graduando em Engenharia Elétrica
pedropires.petee@gmail.com