

Motores

Corrente

- A corrente máxima que o Arduino Uno gerencia é de cerca de 200mA por placa e no máximo 40mA por porta lógica (recomendando-se 20mA por segurança)

Corrente

- Motores de corrente contínua podem ser fabricados para trabalhar com correntes pequenas, MAS, podem consumir bastante mais corrente se forem ‘travados’
- Desta forma recomenda-se não ligar motores diretamente às portas do Arduino
 - Usar circuitos ‘*driver*’

Corrente

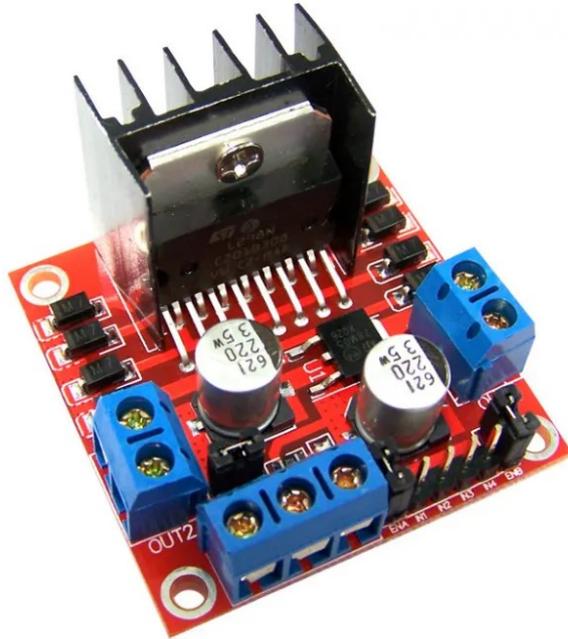


Tipicamente
160mA



Picos de consumo de 650mA. Quando ligado(s) na placa do Arduino pode deixá-lo instável

Circuitos *drivers*

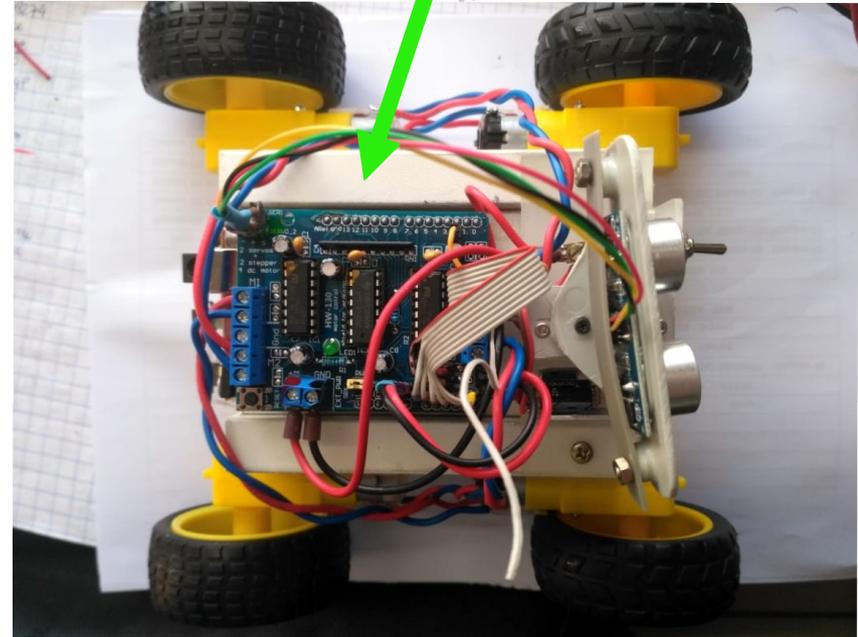
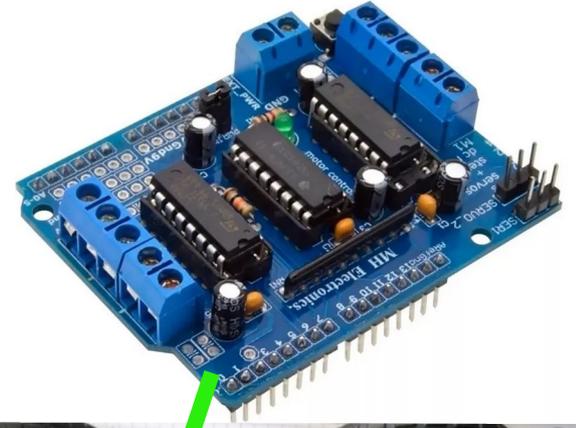
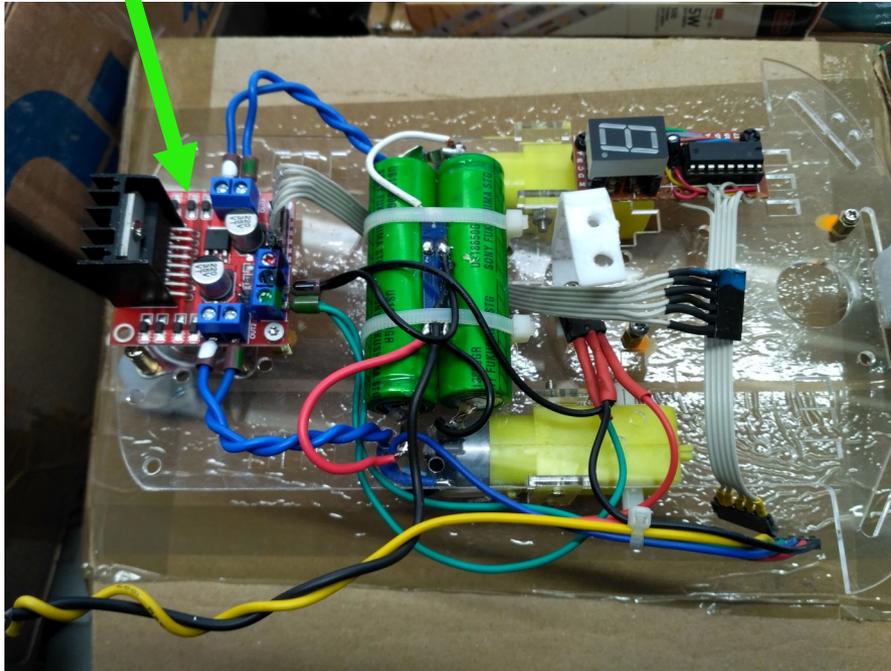
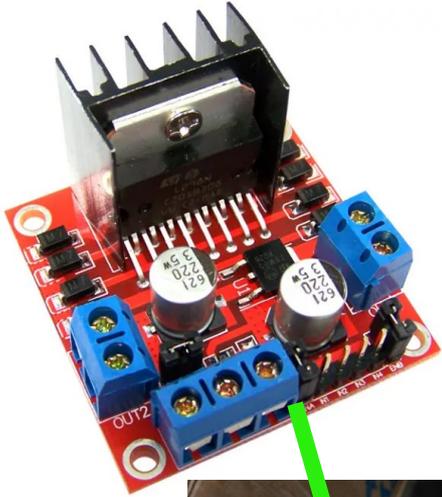


LD298, 2 motores DC ou um de passo de 4 fios



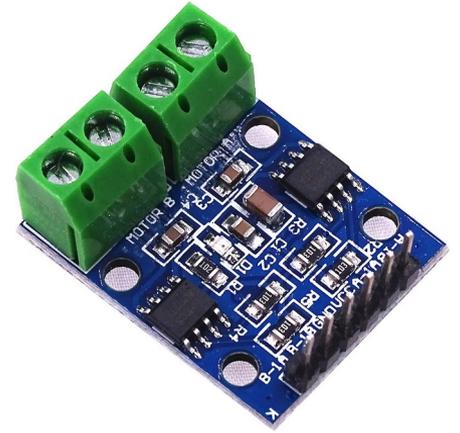
LD293, 4 motores DC ou 2 de passo de 4 fios; também dá acesso a algumas portas do Arduino e tem ligação para 2 servo motores (mas, usando alimentação do Arduino)

Circuitos drivers

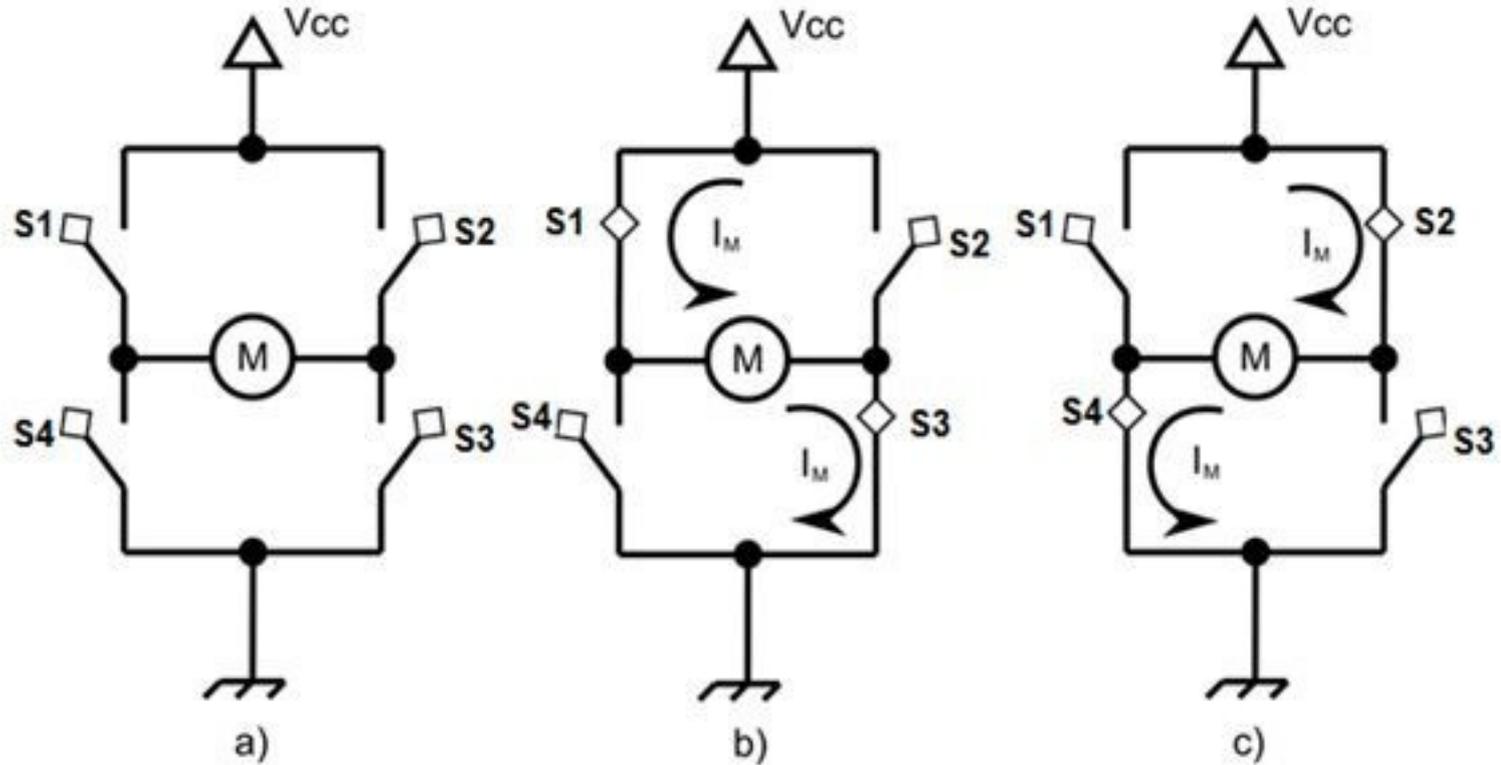


Ponte H

- Motores de corrente contínua giram em sentido oposto se a corrente tiver sentido invertido
- A ponte 'H' permite inverter o fluxo da corrente em um circuito
 - Os *drivers* para motores vistos usam esta técnica



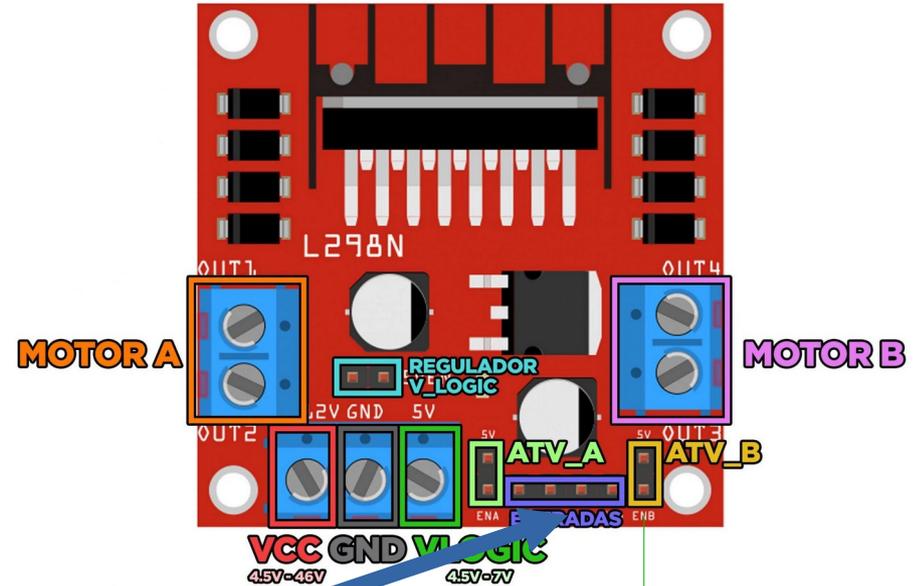
Ponte H



Nos módulos L298

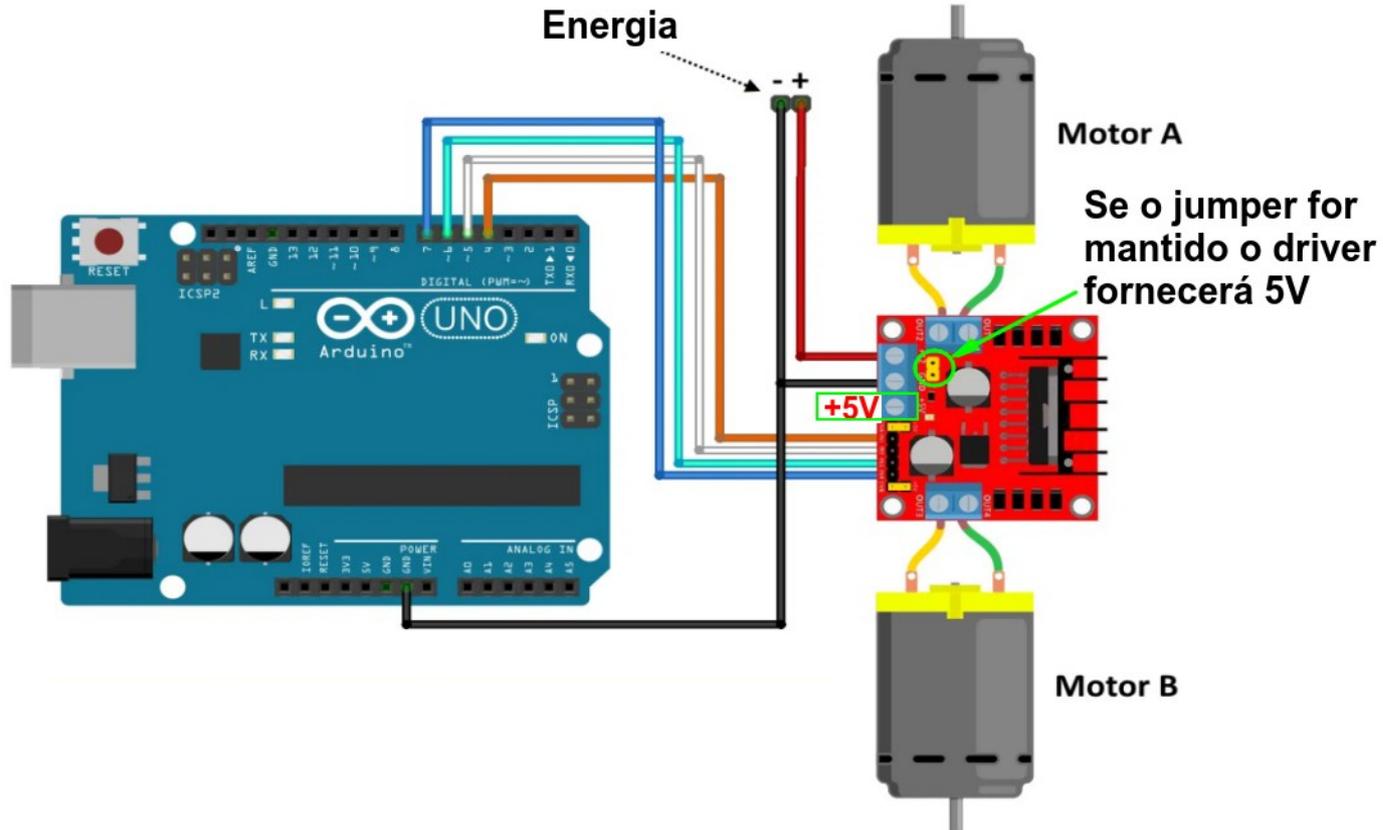
MOTOR	IN1	IN2
HORÁRIO	5v	GND
ANTI-HORÁRIO	GND	5v
PONTO MORTO	GND	GND
FREIO	5v	5v

In1 / In2 (motor A) ou
In3 / In4 (motor B)



EnA, EnB habilitam os
motores ou PWM

Nos módulos L298



Controle

motorL298

```
1 //Direção do motor A
2 #define IN1 4
3 #define IN2 5
4 //Direção do Motor B
5 #define IN3 6
6 #define IN4 7
7
8 void setup()
9 {
10     pinMode(IN1, OUTPUT);
11     pinMode(IN2, OUTPUT);
12     pinMode(IN3, OUTPUT);
13     pinMode(IN4, OUTPUT);
14 }
15
```

Define as portas utilizadas para controle da direção de giro dos motores



Define as portas utilizadas para controle da direção de giro dos motores como sendo saídas



Controle

```
16 void motorAhorario(){
17     digitalWrite(IN1, HIGH);
18     digitalWrite(IN2, LOW);
19 }
20
21 void motorBhorario(){
22     digitalWrite(IN3, HIGH);
23     digitalWrite(IN4, LOW);
24 }
25
26 void motorAantiHorario(){
27     digitalWrite(IN1, LOW);
28     digitalWrite(IN2, HIGH);
29 }
30
31 void motorBantiHorario(){
32     digitalWrite(IN3, LOW);
33     digitalWrite(IN4, HIGH);
34 }
35
36 void paraMotorA(){
37     digitalWrite(IN1, HIGH);
38     digitalWrite(IN2, HIGH);
39 }
40
41 void paraMotorB(){
42     digitalWrite(IN3, HIGH);
43     digitalWrite(IN4, HIGH);
44 }
45
```

FUNÇÕES

Utilizadas para controlar os níveis lógicos nas portas de controle, e, com isso, o sentido do giro dos motores

```
46 void loop()
47 {
48     motorAhorario();
49     motorBhorario();
50     delay(3000);
51     paraMotorB();
52     delay(100);
53     motorBantiHorario();
54     delay(3000);
55     paraMotorA();
56     paraMotorB();
57     delay(3000);
58     motorAantiHorario();
59     motorBhorario();
60     delay(3000);
61     paraMotorA();
62     paraMotorB();
63     delay(5000);
64 }
```

Controle

Chamada das funções

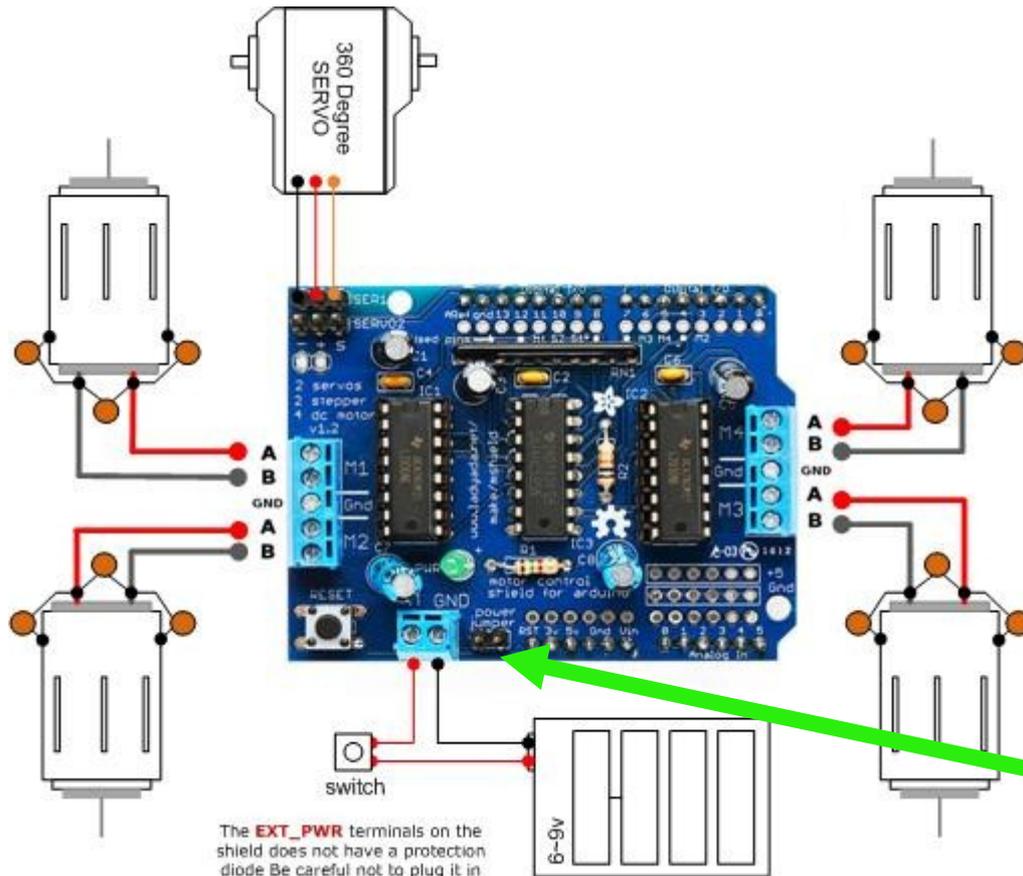
←

Como não foi definida a velocidade (os jumpers EnA e ENB não forma retirados da placa L298), a velocidade será sempre máxima

Controle de velocidade

- RETIRAR os jumpers EnA e EnB
- Ligar os controles EnA e EnB ao Arduino em portas PWM (ex.: 9 e 10)
- Definir as portas como sendo de saída
- Para controle de velocidade, usar:
 - `AnalogWrite(9, 100);` por exemplo
neste caso a porta 9 receberá o valor 100 (pode ir de 0 a 255)

Nos módulos L293



The **EXT_PWR** terminals on the shield does not have a protection diode Be careful not to plug it in backwards or you will destroy the motor shield and/or your Arduino

Se o jumper estiver ligado a energia recebida será conectada ao 'Vin' do Arduino

Ligações FIXAS

Sinal	Físico	Físico	Sinal	Uso	
			18	Digital 19 / SCL	
			17	Digital 18 / SDA	
			16	AREF	
			15	GND	
Sem ligação	8		14	Digital 13 / LED	livre
IOREF	7		13	Digital 12 /	HC595 DIR_LATCH
Reset	6		12	Digital 11 / PWM	PWM2A / DC Motor #1 / Stepper #1
VCC + 3,3Volts	5		11	Digital 10 / PWM	PWM1B / Servo1
VCC + 5Volts	4		10	Digital 09 / PWM	PWM1A / Servo2
GND	3		9	Digital 08	HC595 DIR_SER
GND	2				
+Vin, 7 a 12VDC	1		8	Digital 07	HC595 DIR_EN
Digital 14 / Analógica 00	6		7	Digital 06 / PWM	PWM0A / DC Motor #4 / Stepper #2
Digital 15 / Analógica 01	5		6	Digital 05 / PWM	PWM0B / DC Motor #3 / Stepper #2
Digital 16 / Analógica 02	4		5	Digital 04	HC595 DIR_CLK
Digital 17 / Analógica 03	3		4	Digital 03 / PWM	PWM2B / DC Motor #2 / Stepper #1
SDA / Digital 18 / Analógica 04	2		3	Digital 02	livre
SCL / Digital 19 / Analógica 05	1		2	Digital 01 / Tx	
			1	Digital 00 / Rx	

Controle

motorL293BibliotecaAFRun

```
1 #include <AFMotor.h>
2
3 AF_DCMotor motor1(1); //Ligado na posição 1
4
5 void setup() {
6   motor1.setSpeed(100); //Velocidade do motor (PWM)
7   motor1.run(RELEASE); //Para o motor
8 }
9
10 void loop() {
11   motor1.run(FORWARD); // Liga o Motor
12   delay(2000);
13   motor1.setSpeed(200);
14   delay(2000);
15   motor1.run(RELEASE); // Desliga o Motor
16   delay(50);
17   motor1.run(BACKWARD); //Inverte a rotação
18   delay(2000);
19   motor1.run(RELEASE);
20   delay(3000);
21 }
```

Biblioteca AFMotor

Basta declarar uma variável e informar qual motor será controlado por ela (1 a 4), e, depois qual a função desejada:

SetSpeed – ajusta velocidade (0–255)

run com

RELEASE – para o motor

FORWARD – sentido frente

BACKWARD – sentido inverso

Com sensores de linha

simples2sensores

```
1 //Leitura analógica - ajustar os valores antes
2 #define MotorA1 7
3 #define MotorA2 6
4 #define MotorB1 9
5 #define MotorB2 8
6 #define PWMmotor1 5
7 #define PWMmotor2 10
8
9 int pwmMotor1=120; // velocidade motor 1
10 int pwmMotor2=150; // velocidade motor 2
11 int limiarLeitura = 36;
12
13 void setup() {
14     pinMode(MotorA1,OUTPUT);
15     pinMode(MotorA2,OUTPUT);
16     pinMode(MotorB1,OUTPUT);
17     pinMode(MotorB2,OUTPUT);
18     pinMode(PWMmotor1,OUTPUT);
19     pinMode(PWMmotor2,OUTPUT);
20     pinMode(A0, INPUT); // para leitura analógica do sensor 1
21     pinMode(A1, INPUT); // para leitura analógica do sensor 2
22 }
```

Sensores IR conectados nas portas A0 e A1; as demais ligações vem do 'shield' L293

```

24 void loop() {
25   int SensorEsquerdo = analogRead(A0);
26   int SensorDireito = analogRead(A1);
27   if(SensorDireito<limiarLeitura && SensorEsquerdo<limiarLeitura) //Em frente
28   {
29     digitalWrite(MotorA1, HIGH);
30     digitalWrite(MotorA2, LOW);
31     digitalWrite(MotorB1, HIGH);
32     digitalWrite(MotorB2, LOW);
33     analogWrite(PWMmotor1, pwmMotor1);
34     analogWrite(PWMmotor2, pwmMotor1);
35   }
36   else if(SensorDireito>limiarLeitura && SensorEsquerdo<limiarLeitura) //Esquerda
37   {
38     digitalWrite(MotorA1, LOW);
39     digitalWrite(MotorA2, HIGH);
40     digitalWrite(MotorB1, HIGH);
41     digitalWrite(MotorB2, LOW);
42     analogWrite(PWMmotor1, pwmMotor2);
43     analogWrite(PWMmotor2, pwmMotor2);
44   }
45   else if(SensorDireito<limiarLeitura && SensorEsquerdo>limiarLeitura-1) //Direita
46   {
47     digitalWrite(MotorA1, HIGH);
48     digitalWrite(MotorA2, LOW);
49     digitalWrite(MotorB1, LOW);
50     digitalWrite(MotorB2, HIGH);
51     analogWrite(PWMmotor1, pwmMotor2);
52     analogWrite(PWMmotor2, pwmMotor2);
53   }
54   else if(SensorDireito>limiarLeitura && SensorEsquerdo>limiarLeitura) //Voltar
55   {
56     digitalWrite(MotorA1, LOW);
57     digitalWrite(MotorA2, LOW);
58     digitalWrite(MotorB1, LOW);
59     digitalWrite(MotorB2, LOW);
60     delay(10000);
61   }
62 }

```

Comparações simples dos valores lidos, os quais dependem de haver ou não reflexão de luz (testar os valores para SEU sensor, ajustando a sensibilidade, e, se necessário, alterar no código o valor de 'limiarLeitura' – que neste exemplo tem o valor 36).