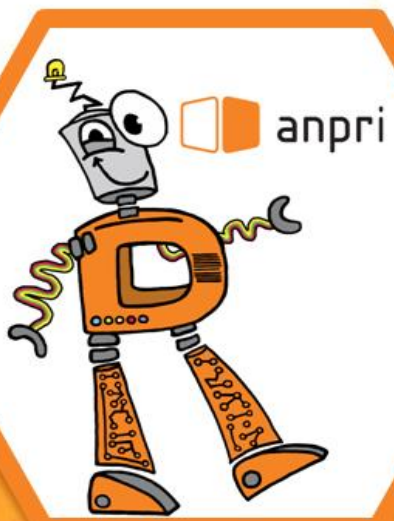
Ação: Programação com Arduinos

8ª sessão – Programação de Robôs

Maio, 2021

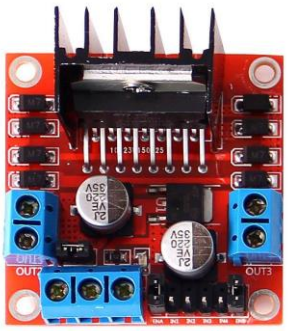


Atividade 19 – Programação de Robôs

Descrição: Nesta atividade pretende-se construir um circuito que permita controlar um robô através da utilização de um módulo Bluetooth e de um módulo L298N Motor Driver.

Módulo L298N Motor Driver

Este módulo permite controlar até 2 motores DC ou 1 motor de passo. Foi projetado para controlar cargas indutivas como relés, solenoides, motores DC e motores de passo, permitindo definir a velocidade e o sentido de rotação do motor

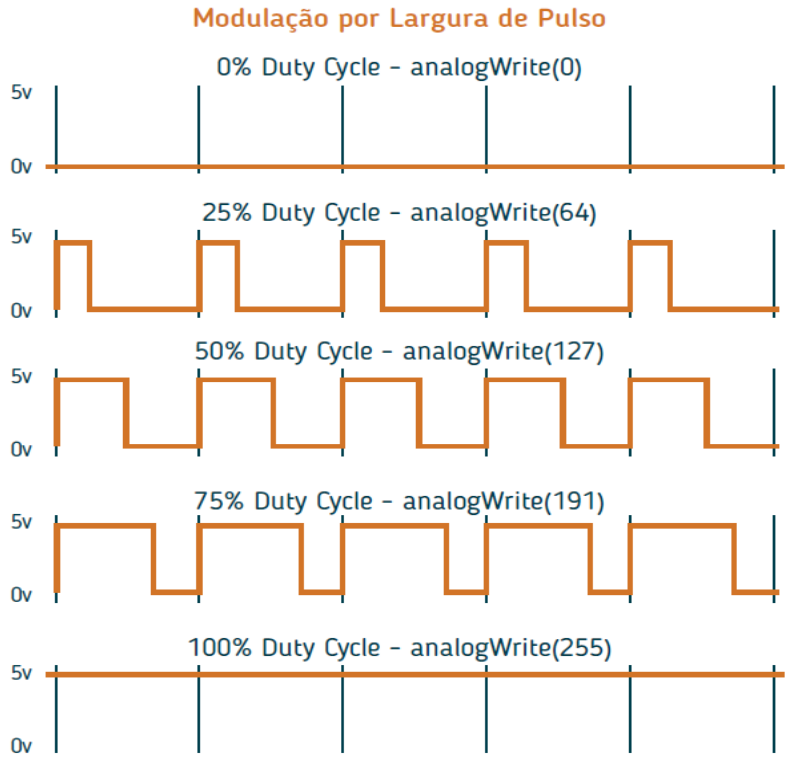


Para se obter o controle completo do motor DC, é necessário controlar a sua velocidade e sentido de rotação. Isso pode ser alcançado através da combinação de duas técnicas:

- PWM - Para controlo da velocidade
- Ponte-H - Para controlo do sentido da rotação

PWM, acrónimo de Pulse Width Modulation é uma técnica que permite por via de um sinal digital emular um resultado analógico. O controlo digital inerente ao microcontrolador, produz uma onda quadrada que permanece no tempo num de dois estados possíveis: 5V (ON) ou 0V (OFF). Regulando o tempo que o sinal permanece num estado e no outro, consegue-se modular o sinal e controlar a velocidade de motores.

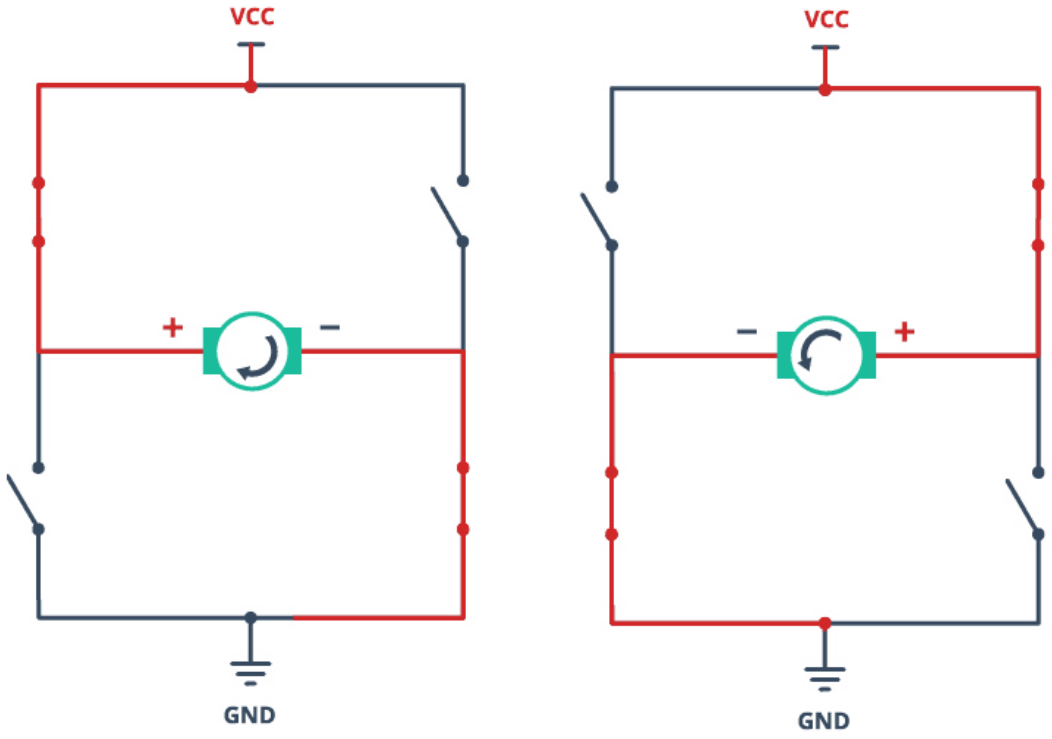
Os pinos 3, 5, 6, 9, 10 e 11 da placa Arduino Uno têm a função PWM. Nestes pinos além da função digitalWrite() que permite escrever os valores 0 ou 1, existe a possibilidade de utilizar a função analogWrite que permite definir valores numa escala de 0 a 255, de tal modo que analogWrite(255) corresponde a 100% do tempo no estado 1 (também conhecido como duty cycle 100%), já analogWrite(127) corresponde um duty cycle de 50% (ou seja num ciclo completo da onda, metade do tempo está a 0V e outra metade a 5V).



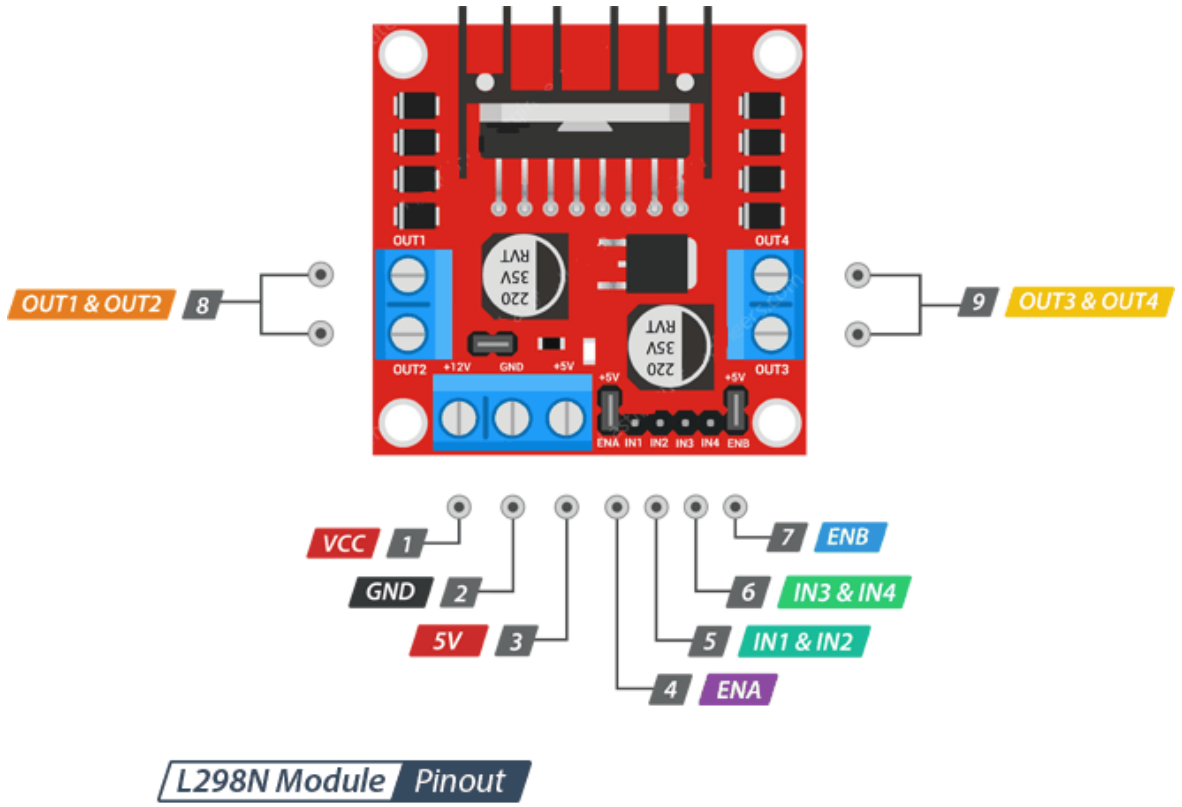
O sentido de rotação do motor DC pode ser controlado através da polaridade da tensão de entrada. Uma forma de se conseguir isso, é utilizar uma Ponte-H.

Um circuito Ponte-H é composto por quatro interruptores e o motor ao centro, em forma de H.

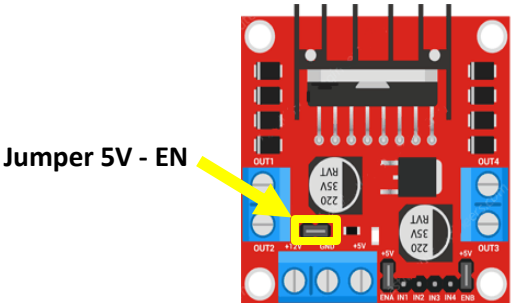
Ao fechar dois interruptores específicos ao mesmo tempo, inverte-se a polaridade da tensão aplicada ao motor e isso causa a mudança no sentido de rotação do motor.



Pinout do Módulo L298N Motor Driver:



VCC O pino VCC fornece a energia para o motor com tensões de entrada entre 5 e 35V. Se o jumper 5V-EN estiver colocado, é necessário fornecer 2 volts extra acima da tensão real do motor, a fim de se obter a velocidade máxima do motor.



GND GND é o pino GROUND comum.

5V Se o jumper 5V-EN estiver colocado, funciona como saída e pode ser usado para fornecer energia ao Arduino. Se o jumper 5V-EN for removido, é necessário conectar este pino aos 5V do Arduino.

ENA **ENB** Os pinos ENA e ENB são utilizados para controlar a velocidade dos motores. Com os jumpers colocados, não é possível controlar a velocidade, esta será

máxima (255). Se os jumpers forem retirados e os pinos forem conectados a entradas digitais PWM no Arduino, então será possível controlar a velocidade dos motores.

IN1 & IN2 Os pinos IN1 e IN2 são utilizados para controlar o sentido de rotação do motor A. Se um dos pinos estiver a HIGH e o outro a LOW, o motor A irá girar. Se ambas as entradas estiverem a HIGH ou a LOW, o Motor A irá parar.

IN3 & IN4 Os pinos IN3 e IN4 são utilizados para controlar o sentido de rotação do motor B. Se um dos pinos estiver a HIGH e o outro a LOW, o motor B irá girar. Se ambas as entradas estiverem a HIGH ou a LOW, o Motor B irá parar.

OUT1 & OUT2 Os pinos OUT1 e OUT2 são conectados ao Motor A.

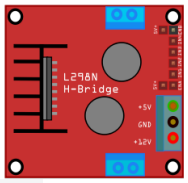
OUT3 & OUT4 Os pinos OUT3 e OUT4 são conectados ao Motor B.

A tabela seguinte apresenta os sentidos da rotação e respetivas configurações, dos pinos IN1 e IN2, relativos ao motor A. O mesmo esquema pode ser aplicado aos pinos IN3 e IN4, que controlam o Motor B.

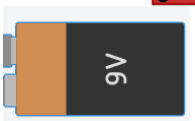
Motor	IN1	IN2
Horário	HIGH	LOW
Anti-Horário	LOW	HIGH
Parado	LOW	LOW
Parado	HIGH	HIGH

Material necessário:

- **1 L298N Motor Driver**

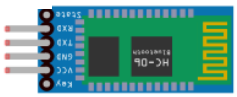


- **1 Pilha 9V**



- **1 Adaptador 9V**

- **1 Módulo Bluetooth HC-06**

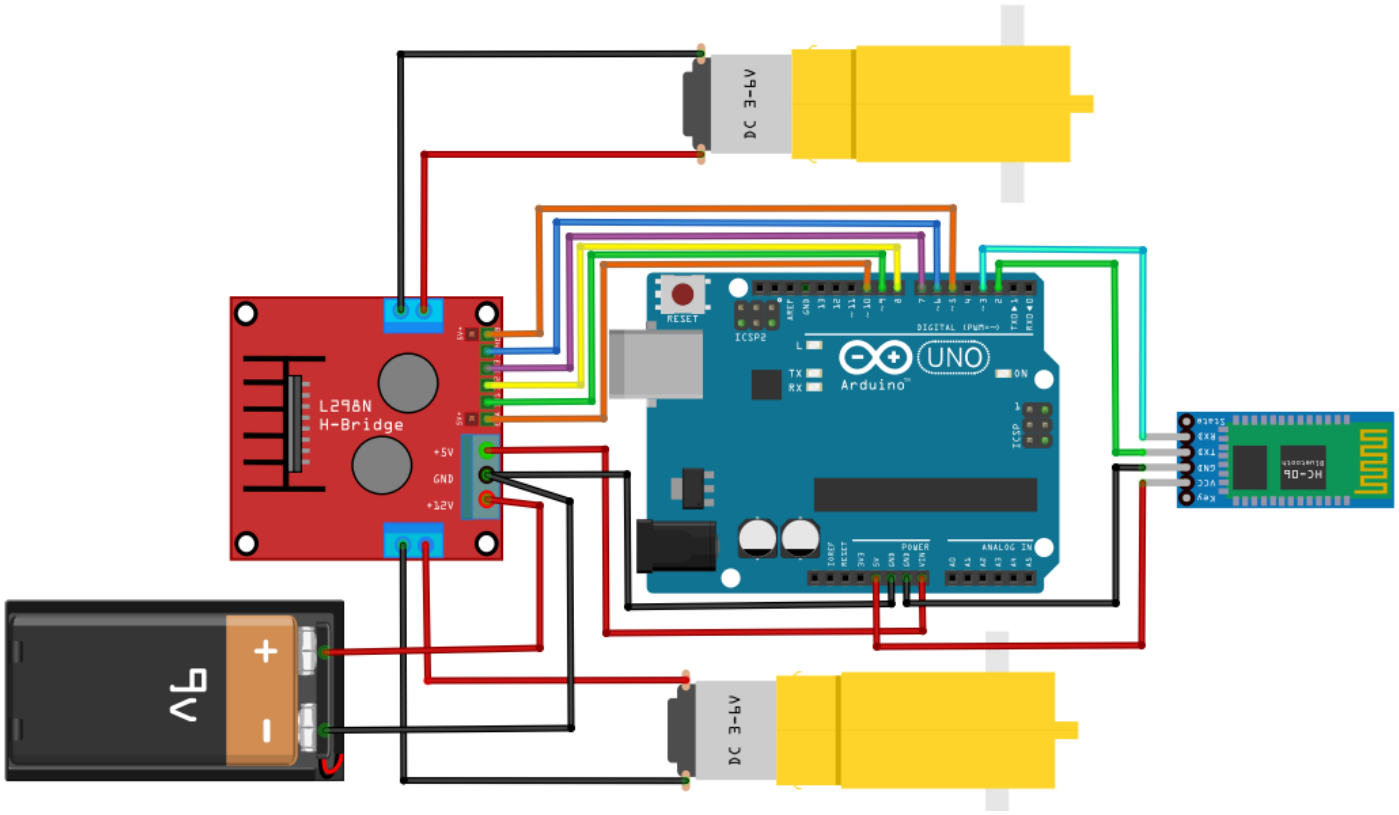


- **2 Motores DC**

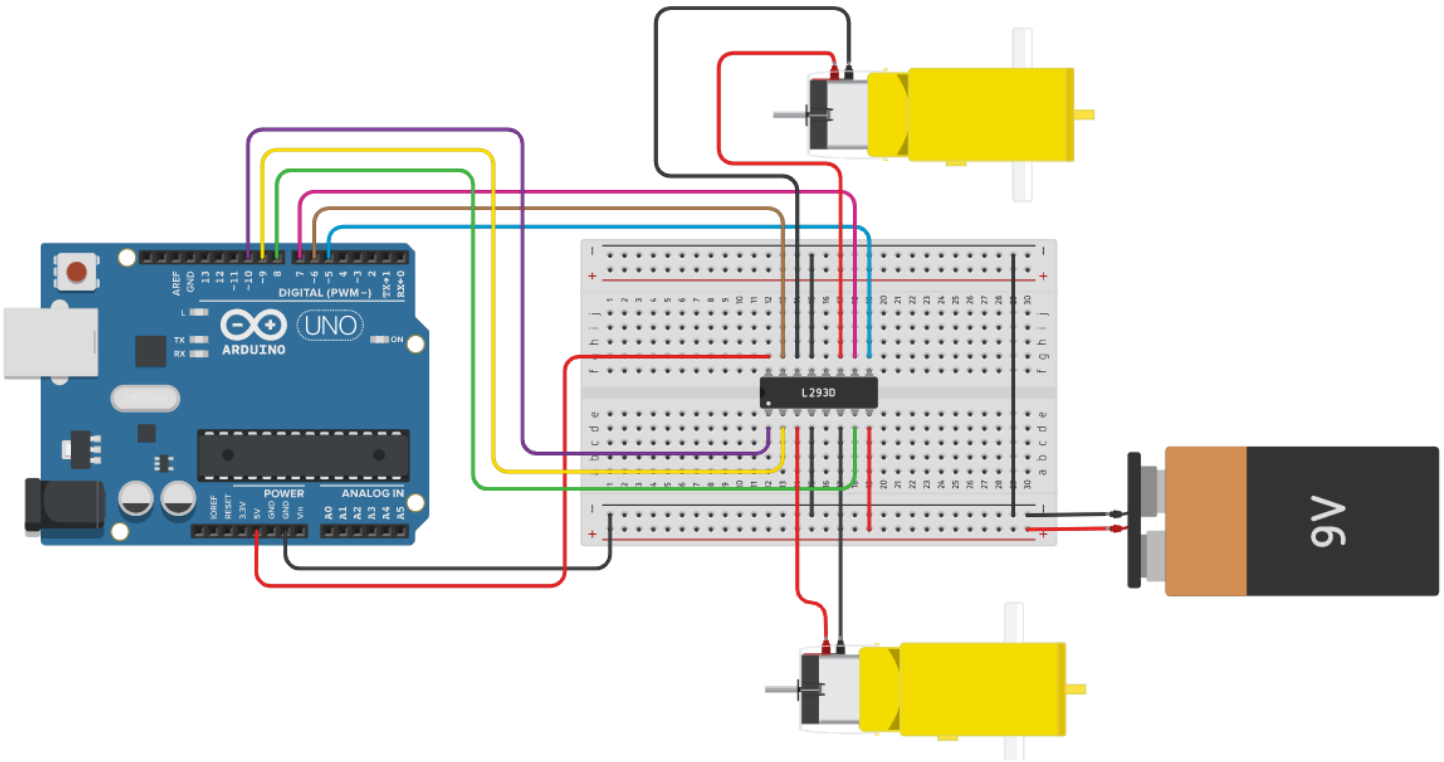
- **Fios**

- **1 circuito integrado L293D (Tinkercad)**

Esquema do circuito:



Circuito alternativo (Tinkercad):



Aplicação Móvel – Arduino Bluetooth RC Car

Para o controlo do robô iremos utilizar a aplicação Arduino Bluetooth RC Car disponível no Google Play.

Arduino Bluetooth RC Car

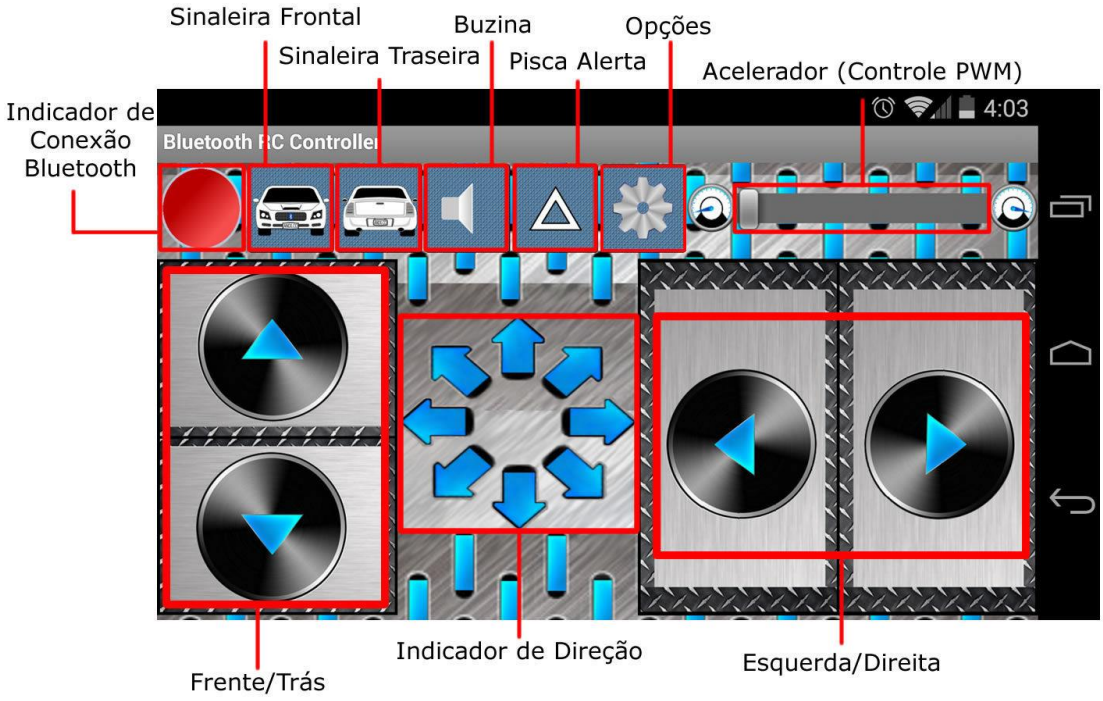
Andi.Co Educação

Todos

Esta app está disponível para todos os seus dispositivos

Pode partilhar este item com a família. Saiba mais acerca da Biblioteca da família

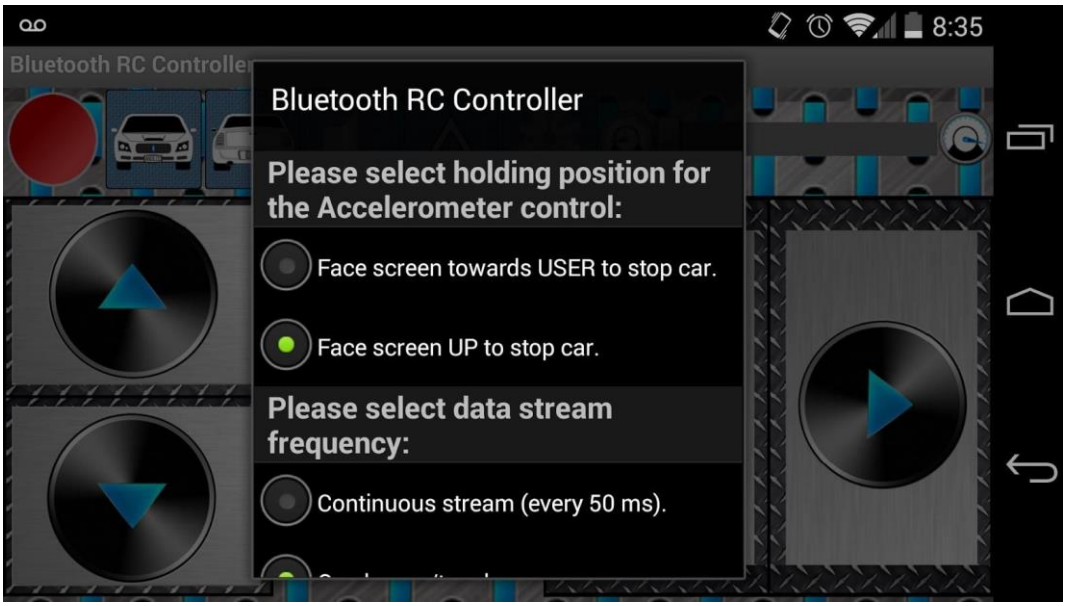
Esta aplicação permite controlar a direção, a velocidade do robô, as luzes frontais e traseiras, a buzina e os quatro piscas.

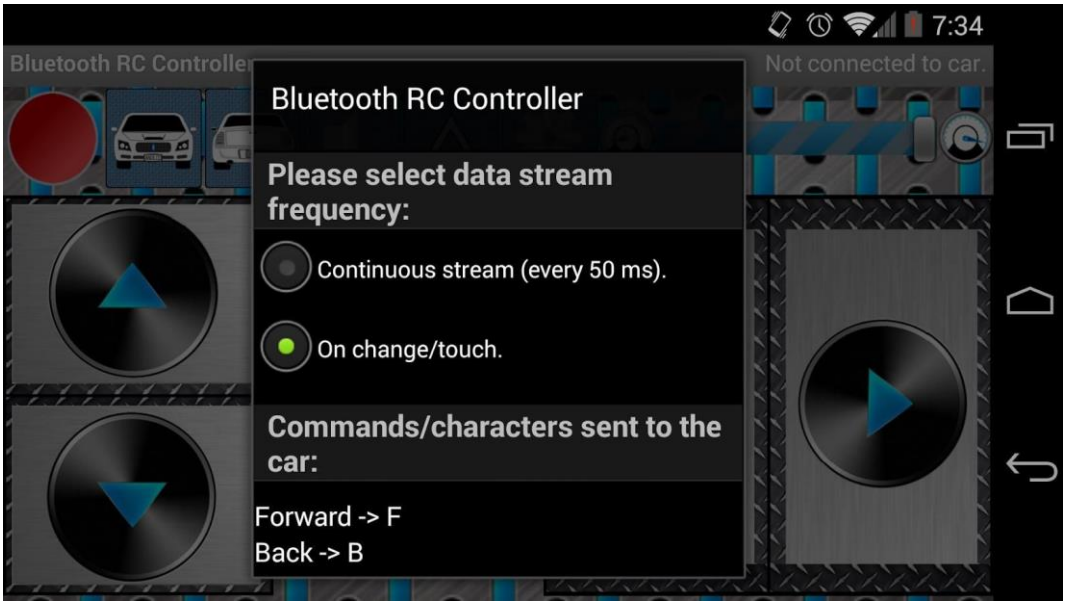


Antes de se começar a utilizar a aplicação, após a respetiva instalação, é necessário emparelhar o módulo Bluetooth com o smartphone. O código pin solicitado por defeito é 1234. Este código pode ser modificado através de comandos AT.

Depois de emparelhado, é necessário escolher o módulo Bluetooth ao qual se pretende efetuar uma conexão. Esta configuração é efetuada através da opção “Connect to Car” disponível nas opções da aplicação.

Para além disso será necessário selecionar as opções “Face screen UP to stop car” e “On change/touch” nas definições da aplicação. Depois de efetuadas estas alterações, é importante guardar as alterações através do botão “OK”.





Os caracteres associados a cada funcionalidade/ação, disponível na aplicação, estão também discriminados nas definições da aplicação.

Programação:

1º - Incluir a biblioteca SoftwareSerial:

```
#include <SoftwareSerial.h>
```

2º - Criar o objeto mySerial com a configuração da porta Rx do Arduino (2) e da porta Tx do Arduino (3).

```
SoftwareSerial mySerial(2, 3); // RX, TX
```

3º - Definir seis variáveis constantes correspondentes aos pinos digitais que estarão ligados ao módulo L298N.

```
// Motor A
#define enA 10 //Controlo da velocidade do Motor A
#define in1 9
#define in2 8
// Motor B
#define enB 5  //Controlo da velocidade do Motor B
#define in3 7
#define in4 6
```

4º - Declarar uma variável char para guardar o caracter enviado pela aplicação:

```
char opcao;
```

5º - Na função **"setup"**, escrever o código que permita configurar os pinos digitais que estarão ligados ao módulo L298N, como pinos de saída (OUTPUT):

```
pinMode(enA, OUTPUT);
pinMode(enB, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
```

6º - Na função **"setup"**, inicializar a comunicação série entre o Monitor Série e o Arduino e inicializar a comunicação série entre o Arduino e o módulo Bluetooth:

```
Serial.begin(9600);
mySerial.begin(9600);
```

7º - Na função **"loop"**, obter o caracter correspondente a cada ação do robô e atribuir-lhe uma determinada função:

```
if (mySerial.available() > 0) {
  opcao = mySerial.read();
  switch (opcao) {
    case 'F':
      moveF();
      Serial.println("Frente");
      break;
    case 'B':
      moveB();
      Serial.println("Marcha atrás");
      break;
    case 'L':
      moveL();
      Serial.println("Esquerda");
      break;
    case 'R':
      moveR();
      Serial.println("Direita");
      break;
    case 'S':
      moveStop();
      Serial.println("Parar");
      break;
  }
}
```

8º - Desenvolver as funções associadas ao controlo da velocidade e da direção do robô.

```
void controlSpeed(int velA, int velB) {
  analogWrite(enA, velA);
  analogWrite(enB, velB);
}

void moveF() {
  controlSpeed(255, 255);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}
```

Código Final:

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3); // RX, TX

// Motor A
#define enA 10 //Controlo da velocidade do Motor A
#define in1 9
#define in2 8
// Motor B
#define enB 5  //Controlo da velocidade do Motor B
#define in3 7
#define in4 6

char opcao;

void setup() {
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  Serial.begin(9600);
  mySerial.begin(9600);
}
void loop() {

  if (mySerial.available() > 0) {
    opcao = mySerial.read();
    switch (opcao) {
      case 'F':
        moveF();
        Serial.println("Frente");
        break;
      case 'B':
        moveB();
        Serial.println("Marcha atrás");
        break;
```

```
      case 'L':
        moveL();
        Serial.println("Esquerda");
        break;
      case 'R':
        moveR();
        Serial.println("Direita");
        break;
      case 'S':
        moveStop();
        Serial.println("Parar");
        break;
    }
  }
}
void moveF() {
  controlSpeed(255, 255);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}

void moveB() {
  controlSpeed(255, 255);
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}

void moveL() {
  controlSpeed(255, 255);
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}
```

```
void moveR() {
  controlSpeed(255, 255);
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}

void moveStop() {
  controlSpeed(0, 0);
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}

void controlSpeed(int velA, int velB) {
  analogWrite(enA, velA);
  analogWrite(enB, velB);
}
```

L293D Motor Driver Shield

Este módulo permite controlar:

- 4 motores DC com controlo de velocidade de 8 bits (0-255)
- 2 motores de passo (unipolar ou bipolar)
- 2 servo motores

Como o shield contém dois circuitos integrados L293D, então permite controlar individualmente até quatro motores DC, tornando-o ideal para a construção de robôs de quatro rodas. É composto por um total de 4 pontes-H e cada ponte-H pode fornecer até 0,6 A ao motor.

Fornecimento de energia

Existem três possibilidades quando se trata de fornecer energia para os motores através do shield:

- **Fonte de alimentação DC única para o Arduino e motores**:
  É necessário conectar a fonte ao conector DC do Arduino ou ao conector EXT_PWR de 2 pinos no shield. Coloque o jumper de alimentação no shield do motor. A tensão de alimentação do motor terá de ser inferior a 12 V.

- **Arduino alimentado por USB e motores através de uma fonte de alimentação DC**: Será necessário conectar o cabo USB ao Arduino e ligar a alimentação do motor ao conector EXT_PWR do shield. Remova o jumper de alimentação no shield do motor.

- **Duas fontes de alimentação DC separadas para o Arduino e motores**: Será necessário ligar a alimentação no conector DC do Arduino e conectar a alimentação do motor ao conector EXT_PWR. Remova o jumper de alimentação no shield do motor.

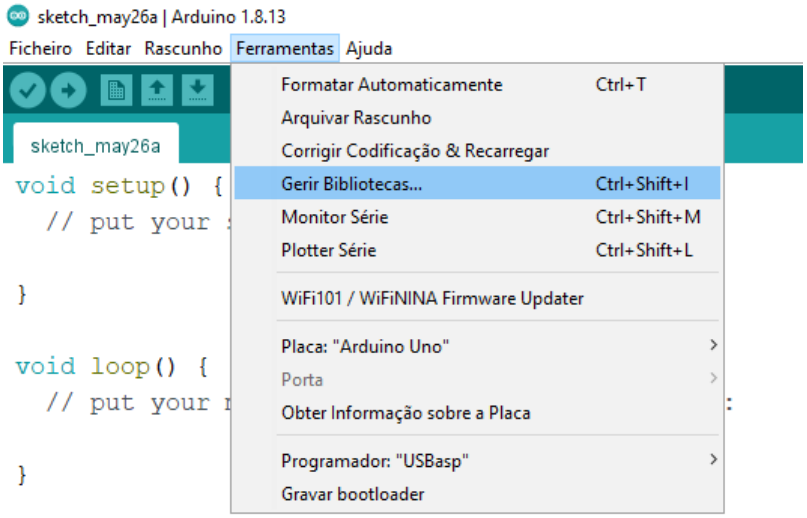
O LED integrado no shield indica que a fonte de alimentação do motor tem energia. Se não estiver aceso, os motores não funcionarão.

Instalação da Biblioteca AFMotor

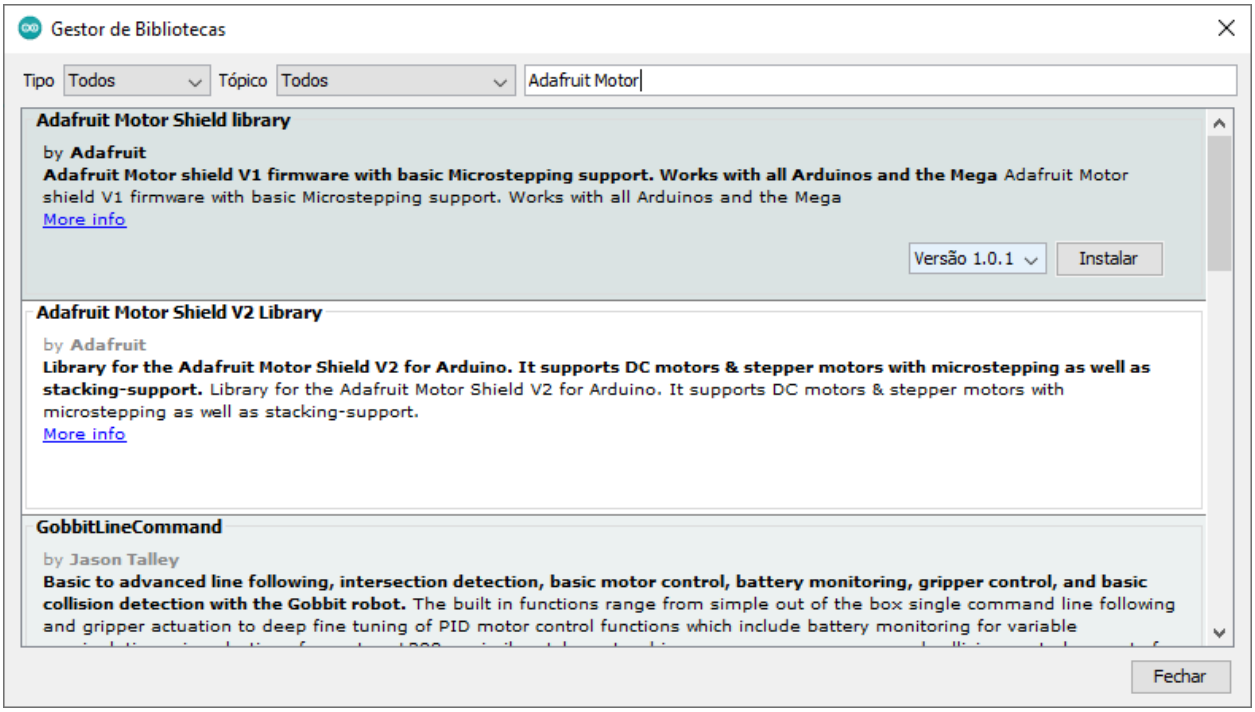
Para a comunicação com o shield, é necessário instalar a biblioteca AFMotor.h que possibilita o envio de comandos simples para o controlo dos motores.

Para instalar a biblioteca, aceda ao menu Ferramentas -> Gerir Bibliotecas...

Aguarde que o Gestor de Bibliotecas efetue o download do índice de bibliotecas e atualize a lista de bibliotecas instaladas.

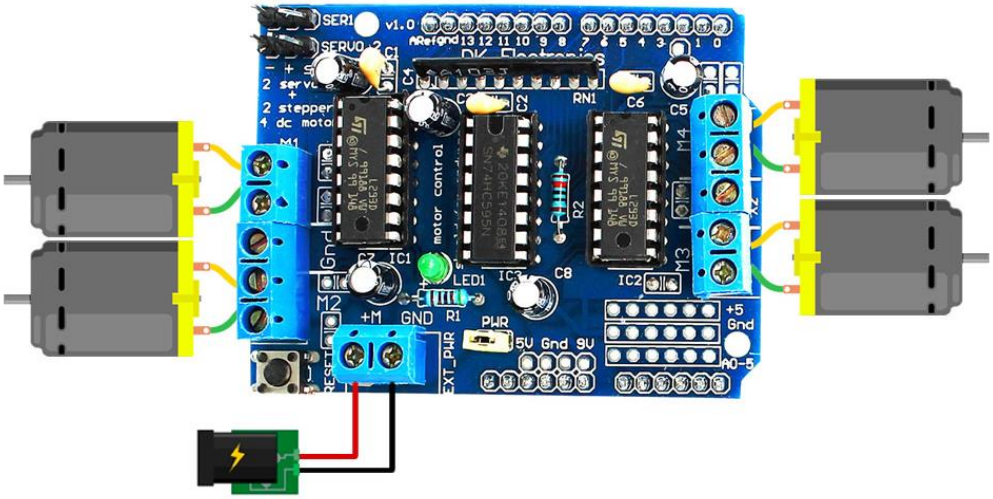


Efetue uma pesquisa por 'Adafruit Motor Shield ', selecione e instale a biblioteca Adafruit Motor Shield Adafruit V1.



Conexão dos motores DC ao shield L293D

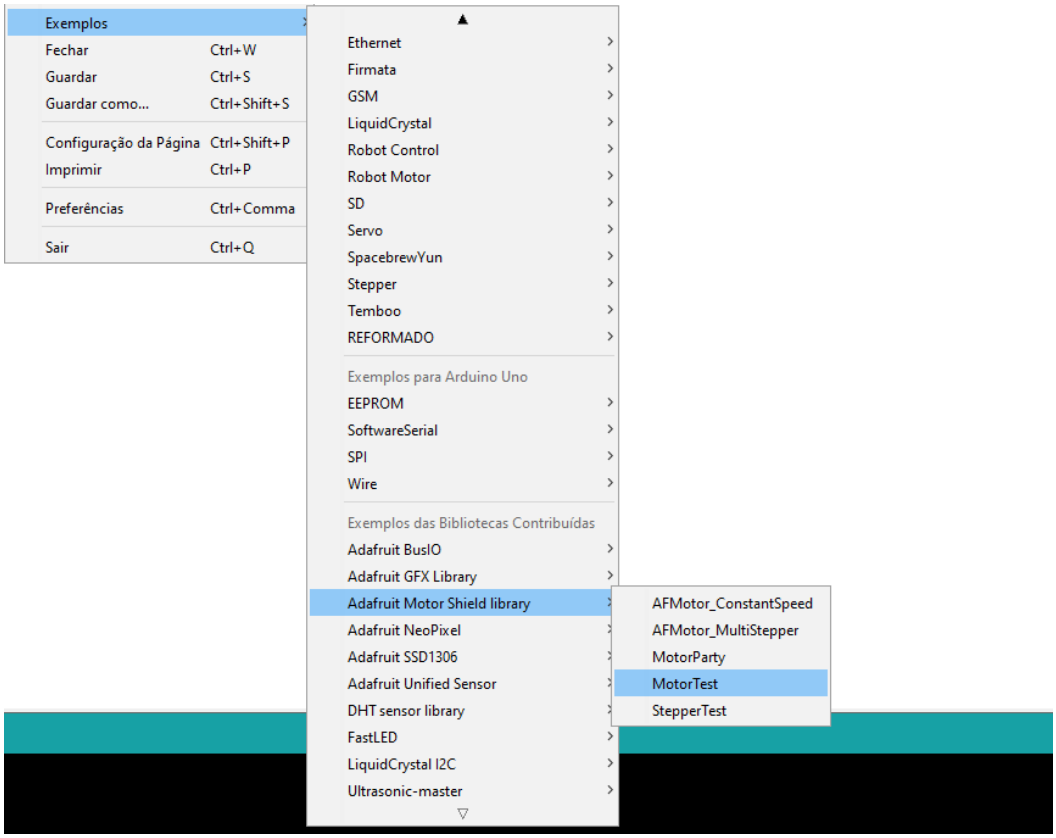
Comece por colocar o shield na parte superior do Arduino. De seguida, conecte os motores aos terminais do shield nas portas M1, M2, M3 ou M4.



Após a conexão dos motores, conecte uma fonte de alimentação aos motores, de acordo com as opções apresentadas anteriormente.

Programação

Para a iniciação à programação deste shield, sugiro a utilização do exemplo "Motor Test" incluído na biblioteca Adafruit Motor Shield.



Para a iniciação à programação deste shield, sugiro a utilização do exemplo “Motor Test” incluído na biblioteca Adafruit Motor Shield.

No início do código é necessário incluir a biblioteca AFMotor.h:

```
#include <AFMotor.h>
```

Depois é preciso criar os objetos relativos aos motores:

```
AF_DCMotor motor(4);
```

Para o movimento dos motores, é possível definir a velocidade (0-255), o sentido da rotação (FORWARD ou BACKWARD) e desligar o motor (RELEASE):

```
motor.setSpeed(200);
motor.run(FORWARD);
motor.run(BACKWARD);
motor.run(RELEASE);
```