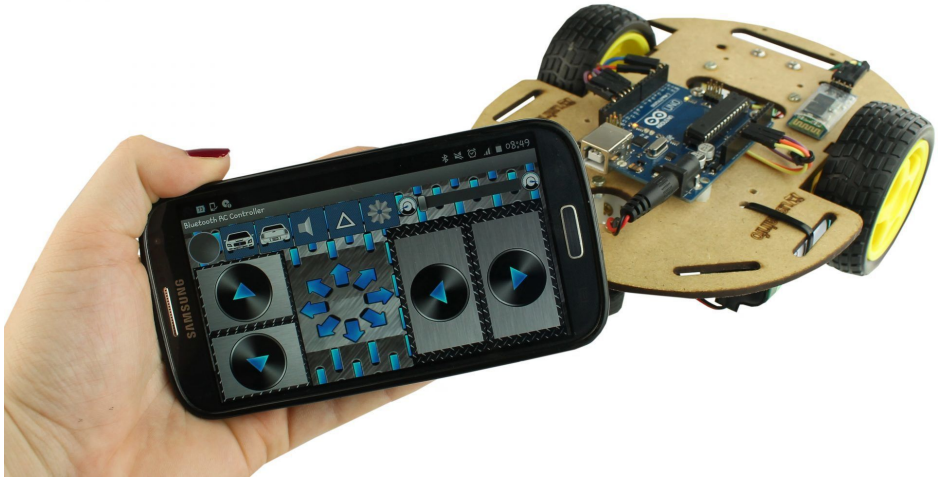
ESP32 - Carrinho de controle remoto via bluetooth

Carlos Eduardo dos Santos Pereira
Leandro Moraes Barbosa

Universidade Federal de Itajubá
PET-TEC - Projeto de Educação Tutorial
Tecnologia de Eletrônica e Computação

Objetivo

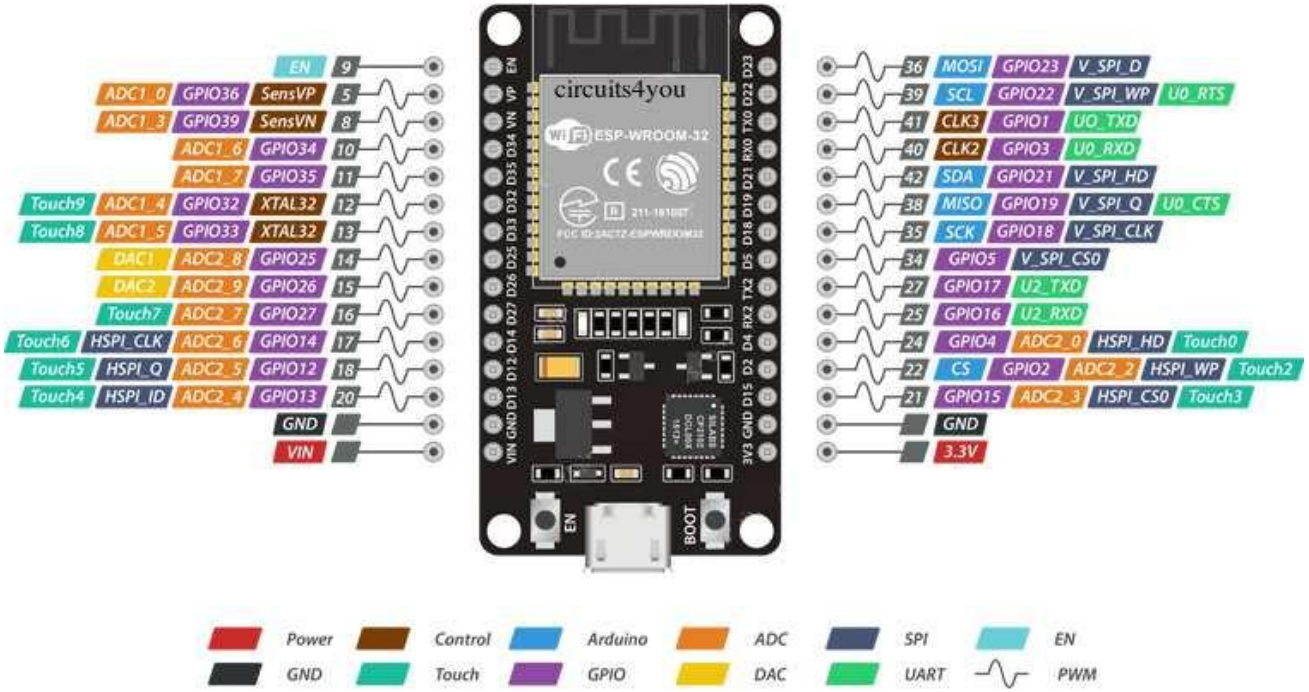
- Desenvolver um carrinho de controle remoto, controlado por celular via bluetooth, utilizando o microcontrolador ESP32, um Chassi Arduíno 2WD e um módulo Driver Ponte H.



ESP32-DevKit

- Sucessor da ESP8266
- Processador dual-core
- frequência de 160MHz
- 34 GPIO
- Pinos sensíveis ao toque
- Conversores Digital-Analógico
- Conversores Analógico-Digital
- Os pinos podem assumir valores de 3.3V ou 0V
- WiFi e bluetooth (Clássico e Low Energy) integrados

ESP32-DevKit PinOut



Bluetooth - Características

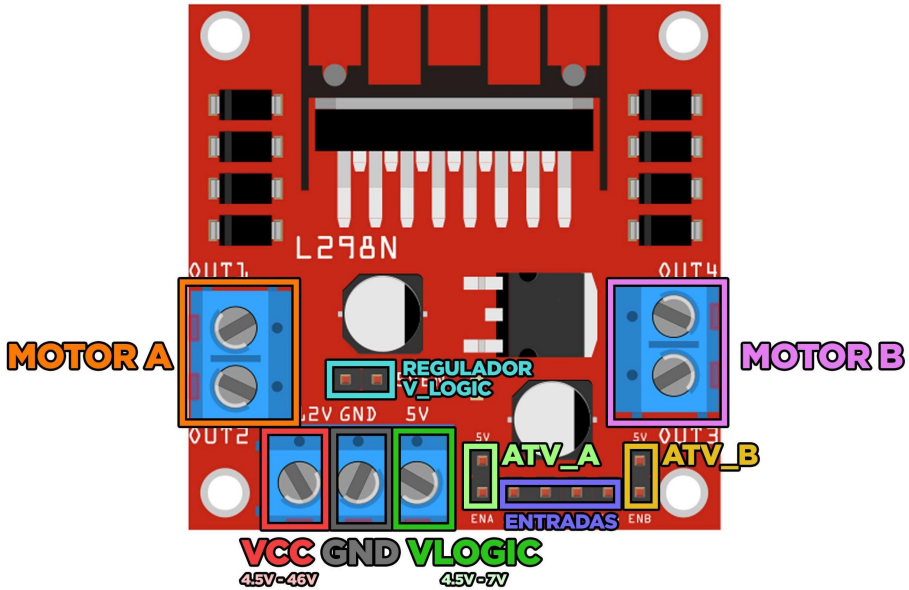
- Surgiu para suprir a demanda de conexão entre dispositivos de forma rápida e barata, sem utilização de cabos.
- Permite troca de informações com até 40 metros de distância
- Taxa de transferência de dados de até 50Mb/s (24Mb/s no caso da ESP32)
- Transmissão por rádio frequência entre 2,4 e 2,5GHz e utiliza saltos de frequência
- A ESP32 conta com o Bluetooth versão 4.2
- Fornece suporte ao IPv6 (modelo mais atual de transmissão por internet)

Bluetooth - Utilização no projeto

O bluetooth será responsável por estabelecer a conexão entre placa e celular, de modo que o celular possa enviar mensagens de forma serial (envio de um bit por vez) ao chip, que deverá reconhecer o comando recebido, por exemplo, de acelerar o carrinho.

Módulo Driver Ponte H

- **Função:** Permite manipular o sentido de rotação dos motores

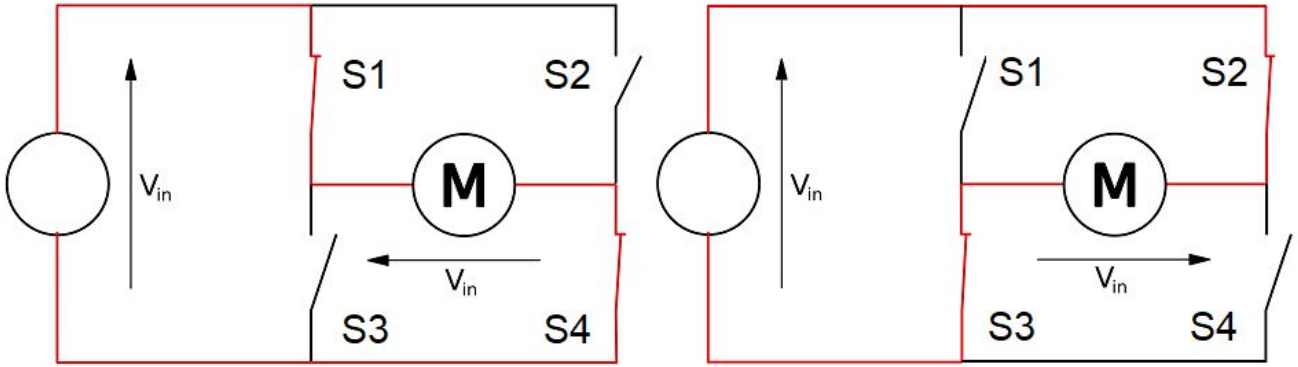


Módulo Driver Ponte H - Características

- **Tensão de Operação:** 4.5V 46V
- **Motores controlados:** Dois motores DC
- **Corrente de Operação máxima:** 2A por canal ou 4A total
- **Limites de Temperatura:** -20 a +135 °C
- **Potência Máxima:** 25W
- **Corrente lógica:** 0 a 36mA
- **Tensão lógica:** 4.5V a 7V

Módulo Driver Ponte H - Funcionamento

- Funciona a partir da combinação de 4 chaves (S1, S2, S3 e S4)
- S1 e S4 fechadas e S2 e S3 abertas: rotação do motor em um determinado sentido, horário, por exemplo (todas as combinações serão abordadas na seção de Programação)



Módulo Driver Ponte H - Utilização no projeto

Conforme indicações feitas na Figura da Ponte H (slide 7), tem-se:

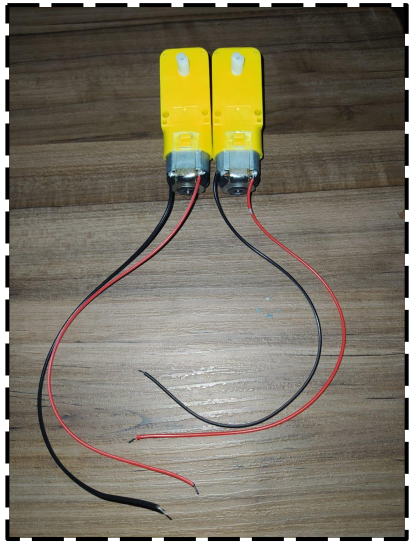
- **Motor A e Motor B:** Os motores são ligados às saídas OUT1, OUT2, OUT3 e OUT4. A posição da ligação altera o sentido de rotação dos motores.
- **VCC:** Tensão de alimentação do motor.
- **VLOGIC e Regulador VLOGIC:** o VLOGIC realiza a alimentação lógica do chip interno à placa, ele garante que a tensão seja regulada em 5V e, por meio do Regulador pode-se ativar ou desativar o VLOGIC.
- **ATV_A e ATV_B:** Ativa os motores. Assim, ao curto circuitar ATV_A, ativa-se o motor A e, ao curto circuitar ATV_B, ativa-se o motor B.
- **Entradas:** Recebe os dados do microcontrolador. As duas primeiras entradas da esquerda controlam o motor A e as outras duas controlam o motor B. Ao enviar sinais altos ou baixos controla-se a ativação dos motores.

ESP32 na IDE do Arduino

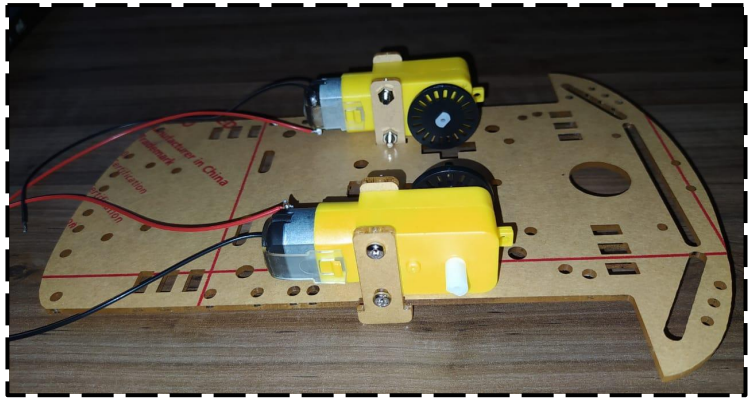
- Utilizaremos a linguagem C/C++ devido a sua vasta biblioteca para arduino, a qual é compatível com a ESP32
- Seguiremos o tutorial “Programar ESP32 com a IDE do Arduino - Tutorial completo” (2019) do autor Gustavo Teixeira, Bacharel em Ciência da Computação pela URI. O artigo encontra-se no link: https://www.usinainfo.com.br/blog/programar-esp32-com-a-ide-arduino-tutorial-completo/

Montagem do Chassi Arduino 2WD

1. Soldar os motores aos jumpers

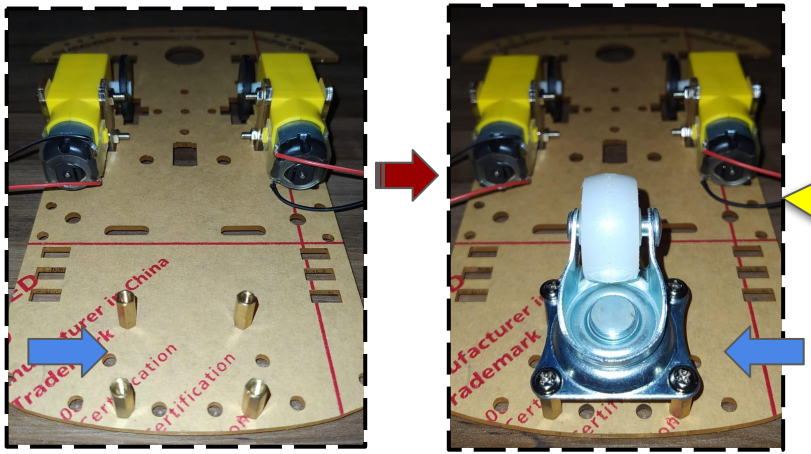


2. Fixar os motores à estrutura do carro

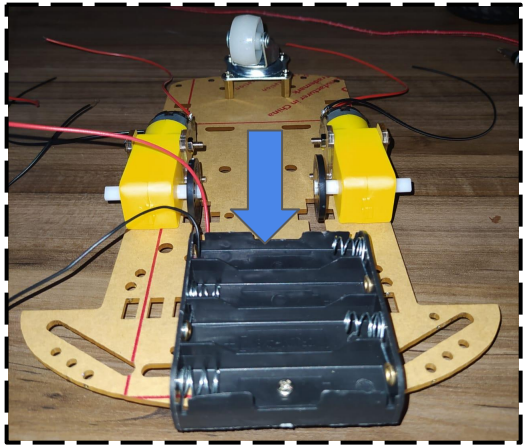


Montagem do Chassi Arduino 2WD

3. Fixar a roda de giro livre

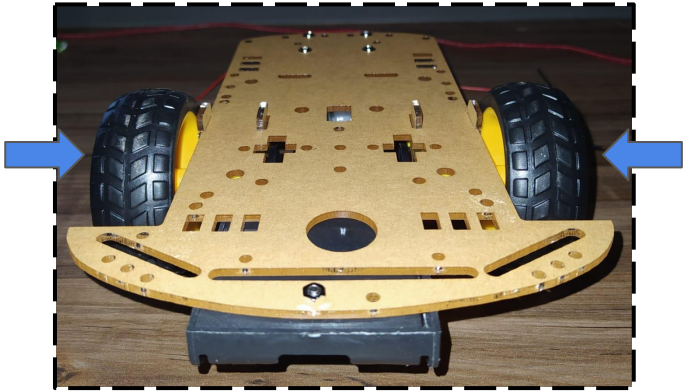


4. Fixar o primeiro suporte de pilha

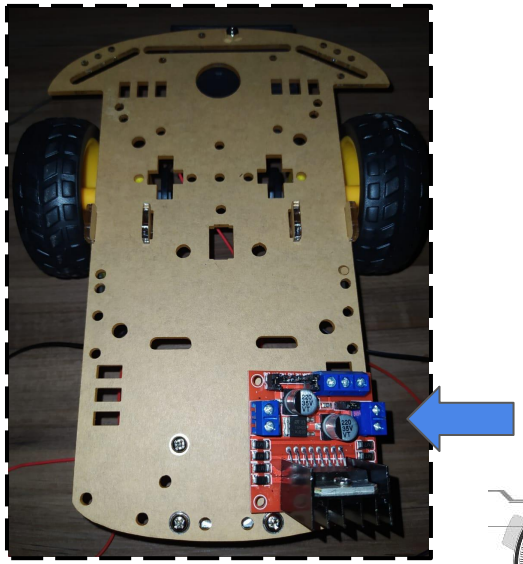


Montagem do Chassi Arduino 2WD

5. Encaixar as rodas às laterais dos motores

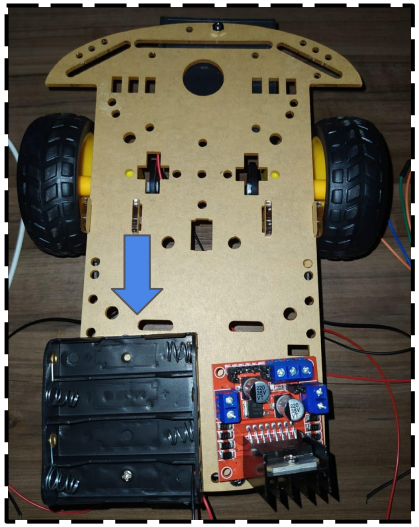


6. Parafusar o módulo Ponte H

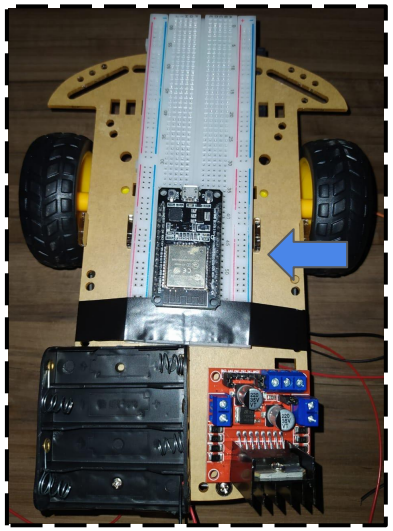


Montagem do Chassi Arduino 2WD

7. Instalar o 2º suporte para pilhas ao lado da Ponte H

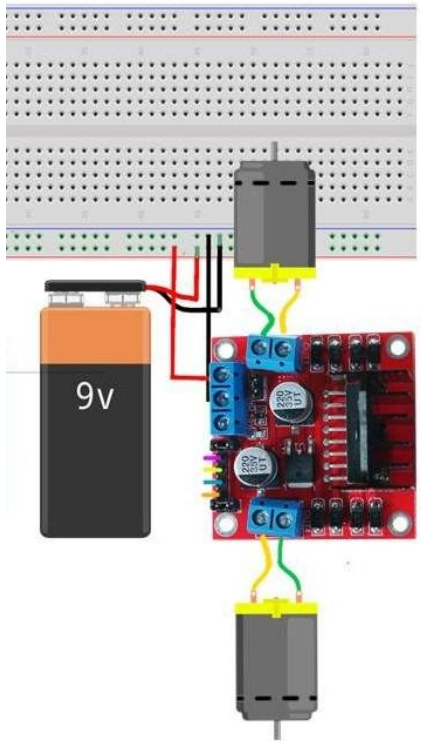


8. Finalmente, fixar a protoboard com o microcontrolador na parte superior do carro



Interligando os componentes - Motores + Ponte H

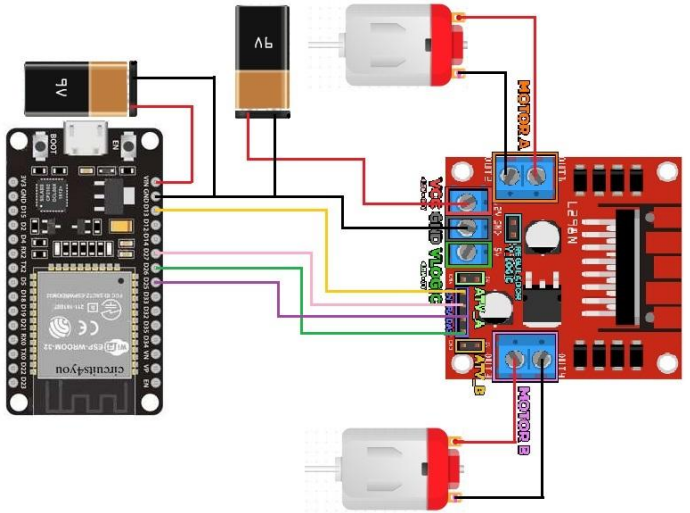
1. Conecte a bateria da parte inferior do carrinho à Ponte. **O fio vermelho deve ser ligado ao VCC e o fio preto ao GND**.
2. **Os dois motores devem ser conectados**, um ao **output** do **MOTOR A** e o outro ao **output** do **MOTOR B**.



Interligando os componentes - ESP32 + circuito

1. **A segunda fonte alimentará a placa**. Deve-se ligar o **fio vermelho ao VIN e o fio preto ao GND comum à Ponte H**.

2. **Conectar cada uma das entradas da Ponte H às saídas da placa**. Utilizaremos as correspondências seguintes.



- **GPIO13 - IN1**
- **GPIO27 - IN2**
- **GPIO25 - IN3**
- **GPIO26 - IN4**

Programação - Como controlar a placa?

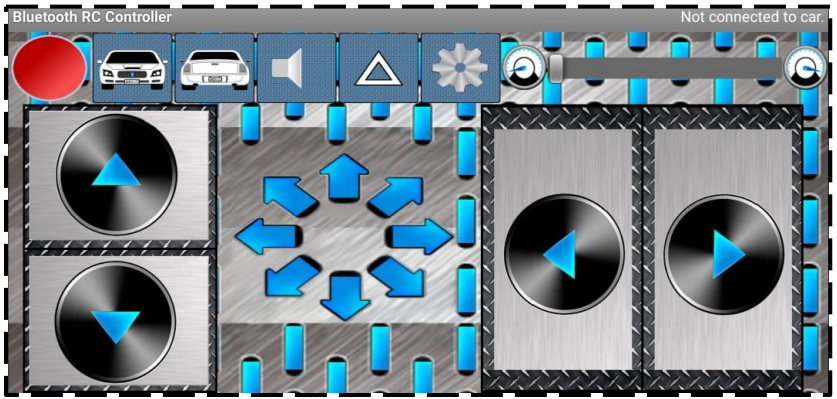
Já sabemos que utilizaremos a comunicação via bluetooth, mas como realizaremos a conexão, de fato?

Para enviar dados do celular para o microcontrolador utilizaremos um software gratuito já consolidado, chamado ***Bluetooth RC Car***:

https://play.google.com/store/apps/details?id=braulio.calle.bluetoothRCcontroller&hl=pt_BR&gl=US

Programação - Bluetooth RC Car

- Aplicativo de fácil utilização que possui interface pronta com todos controles do carro. Neste projeto utilizaremos apenas os direcionais



Bluetooth RC Car - Funcionamento

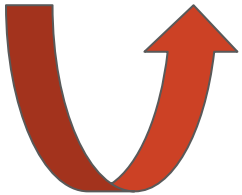
Ao conectar-se ao carro, a **aplicação envia um caractere para cada interação do usuário**, na tabela ao estão todas interações que serão utilizadas neste projeto e seus caracteres correspondentes

Comando	Caractere
Frente	F
Trás	B
Esquerda	L
Direita	R
Frente e esquerda	G
Frente e direita	I
Trás e esquerda	H
Trás e direita	J

Programação - Código

1. Realizaremos uma **comunicação bluetooth serial**, portanto, deve-se **incluir a biblioteca** correspondente e instanciar um objeto do tipo bluetooth serial
2. **criaremos 4 constantes**, as quais cada uma corresponderá à uma saída
3. A variável **"comando"** do tipo caractere é necessária para armazenar os valores dos comandos recebidos

```
#include "BluetoothSerial.h"
#define pin1 13
#define pin2 27
#define pin3 26
#define pin4 25

BluetoothSerial SerialBT;
char comando;
```



Programação - Código

1. Dentro da função **setup()**, deve-se **inicializar o serial**
2. **Iniciar o serial bluetooth** com o nome **"carrinho"**, que será o nome do dispositivo bluetooth
3. **Definir cada um dos pinos como saída**
4. **Gerar um atraso**, para que não haja perda de dados

```
void setup() {
  Serial.begin(9600);
  SerialBT.begin("Carrinho");

  pinMode(pin1, OUTPUT);
  pinMode(pin2, OUTPUT);
  pinMode(pin3, OUTPUT);
  pinMode(pin4, OUTPUT);

  Serial.println("Fim Setup");
  delay(2000);
}
```

Programação - Código

1. O objetivo da função **loop()** é **identificar a ocorrência de uma ação**. Sempre que o dispositivo estiver disponível, deve-se armazenar os dados enviados por ele na variável comando, para na sequência **decidir quais tratamentos deverão ser realizados**.

```
void loop() {

  if (SerialBT.available()) {
    comando = SerialBT.read();
  }

  [...]
}
```

Programação - Lógica de funcionamento

- O carrinho possui **9 estados**
- Deve-se identificar cada estado por seu caractere identificador e decidir quais pinos deverão ser ativados ou desativados para se obter a movimentação desejada

Pino Ativo	Movimento
Pin1	Motor 1 em sentido horário
Pin2	Motor 1 em sentido anti-horário
Pin3	Motor 2 em sentido horário
Pin4	Motor 2 em sentido anti-horário

Lógica de funcionamento - Movimentação do carro

- **Frente:** acionar os **dois motores** com o sentido de rotação **horário**.
- **Trás:** acionar os **dois motores** com o sentido de rotação **anti-horário**.
- **Esquerda:** acionar **apenas o segundo motor** no sentido **horário.**
- **Direita:** acionar **apenas o primeiro motor** no sentido **horário.**
- **Frente e esquerda:** acionar o **motor 1** no sentido **anti-horário** e o **motor 2** no **sentido horário**.
- **Frente e direita:** acionar o **motor 1** no sentido **horário** e o **motor 2** no sentido **anti-horário**.
- **Trás e esquerda:** acionar apenas o **motor 2** no sentido **anti-horário**.
- **Trás e direita:** acionar apenas o **motor 1** no sentido **anti-horário**.
- **Parar:** todos os pinos são **desativados**.

Lógica de funcionamento - Movimentação do carro

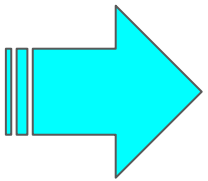
Movimento	Pin1	Pin2	Pin3	Pin4
Frente	HIGH	LOW	HIGH	LOW
Trás	LOW	HIGH	LOW	HIGH
Esquerda	LOW	LOW	HIGH	LOW
Direita	HIGH	LOW	LOW	LOW
Frente e esquerda	LOW	HIGH	HIGH	LOW
Frente e direita	HIGH	LOW	LOW	HIGH
Trás e esquerda	LOW	LOW	LOW	HIGH
Trás e direita	LOW	HIGH	LOW	LOW

Programação - Movimentação do carro

Basta **associar cada movimento ao seu caractere e sua combinação lógica**. Utilizando a estrutura **switch/case**, faremos a correspondência entre cada caractere recebido e a sua respectiva sequência de comandos

```
void loop() {

    [...]
    switch (comando) {
        case 'F': { //move frente
            digitalWrite(pin1, HIGH);
            digitalWrite(pin2, LOW);

            digitalWrite(pin3, HIGH);
            digitalWrite(pin4, LOW);
            break;
          }
```

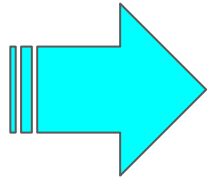
```
case 'I': {//frente direita

            digitalWrite(pin1, HIGH);
            digitalWrite(pin2, LOW);

            digitalWrite(pin3, LOW);
            digitalWrite(pin4, HIGH);
            break;
          }
```



Programação - Movimentação do carro

```
case 'G': {//frente esquerda

            digitalWrite(pin1, LOW);
            digitalWrite(pin2, HIGH);

            digitalWrite(pin3, HIGH);
            digitalWrite(pin4, LOW);
            break;
          }
        case 'R': {//direita

            digitalWrite(pin1, HIGH);
            digitalWrite(pin2, LOW);

            digitalWrite(pin3, LOW);
            digitalWrite(pin4, LOW);
            break;
          }
```

```
case 'L': {//esquerda

            digitalWrite(pin1, LOW);
            digitalWrite(pin2, LOW);

            digitalWrite(pin3, HIGH);
            digitalWrite(pin4, LOW);
            break;
          }
        case 'B': {// ré

            digitalWrite(pin1, LOW);
            digitalWrite(pin2, HIGH);

            digitalWrite(pin3, LOW);
            digitalWrite(pin4, HIGH);
            break;
          }
```

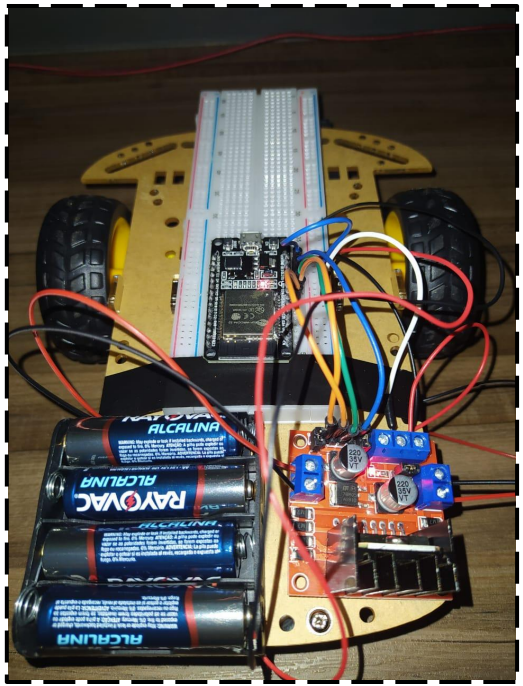


Programação - Movimentação do carro

```
case 'H': {// ré esquerda

            digitalWrite(pin1, LOW);
            digitalWrite(pin2, LOW);

            digitalWrite(pin3, LOW);
            digitalWrite(pin4, HIGH);
            break;
          }
        case 'J': {//ré direita
            digitalWrite(pin1, LOW);
            digitalWrite(pin2, HIGH);

            digitalWrite(pin3, LOW);
            digitalWrite(pin4, LOW);
            break;
          }
```

```
default: {
            digitalWrite(pin1, LOW);
            digitalWrite(pin2, LOW);

            digitalWrite(pin3, LOW);
            digitalWrite(pin4, LOW);
            break;
          }
      }
}
```

Finalizado!



Referências

- Gustavo Teixeira. **"PROGRAMAR ESP32 COM A IDE ARDUINO – TUTORIAL COMPLETO"**. https://www.usinainfo.com.br. Disponível em: https://www.usinainfo.com.br/blog/programar-esp32-com-a-ide-arduino-tutorial-completo/ . Acesso em: 07 de Dezembro de 2020.

- Equipe Wiki IFSC. **"Diagrama de um circuito Ponte H"**. https://wiki.ifsc.edu.br/, 2020. Disponível em: https://wiki.ifsc.edu.br/mediawiki/index.php/AULA_7_-_Microcontroladores_-_Eng . Acesso em: 07 de Dezembro de 2020.

- Gustavo Nery. **"Guia definitivo de uso da Ponte H L298N"**. https://blog.eletrogate.com/, 2020. Disponível em: https://blog.eletrogate.com/guia-definitivo-de-uso-da-ponte-h-l298n/ . Acesso em: 07 de Dezembro de 2020.

- Matheus Gebert. **"COMO INVERTER MOTOR DC COM PONTE H L298N?"**. https://www.usinainfo.com.br/, 2020. Disponível em: https://www.usinainfo.com.br/blog/como-inverter-motor-dc-com-ponte-h-l298n/ . Acesso em: 07 de Dezembro de 2020.

Referências

- Equipe Circuits4you. **"ESP32 DevKit ESP32-WROOM GPIO Pinout"**. https://circuits4you.com/, 2018. Disponível em: https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/ . Acesso em: 07 de Dezembro de 2020.

- Blynk Inc. **"Developers: Everything you need to build an amazing IoT project".** https://blynk.io, 2020. Disponível em: https://blynk.io/en/developers . Acesso em: 07 de Dezembro de 2020.

- Andi.co. **"Arduino Bluetooth RC Car"**. https://play.google.com, 2020. Disponível em: https://play.google.com/store/apps/details?id=braulio.calle.bluetoothRCcontroller&hl=pt_BR&gl=US . Acesso em: 07 de Dezembro de 2020.

- ST.**"DUAL FULL-BRIDGE DRIVER"**. https://www.st.com, 2020. Disponível em: https://www.st.com/resource/en/datasheet/l298.pdf . Acesso em: 07 de Dezembro de 2020.

Referências

- Blog da Curto. **"Conhecendo a ESP32"**. www.curtocircuito.com.br, 2018. Disponível em:<https://www.curtocircuito.com.br/blog/conhecendo-esp32> . Acesso em: 07 de dez. de 2020.

- LOPES, Guilherme José; OLIVEIRA, Valter de Lima. **“Braço Articulado Controlado Remotamente Via Bluetooth”**. 2013. 72 f. TCC (Graduação) - Curso de Engenharia da Elétrica/Eletrônica, Universidade do Vale do Paraíba, São Jose dos Campos, 2013. Disponível em: Acesso em: 08 de dez. de 2020.

- Emerson Alecrim; ALECRIM, Emerson. **“Tecnologia Bluetooth: o que é e como funciona?”**. www.infowester.com, 2018. Disponível em: <https://www.infowester.com/bluetooth.php#bluetooth-4-2> Acesso em: 08 de dez. de 2020.