Projeto Objetos Inteligentes Conectados

VELOCICAR

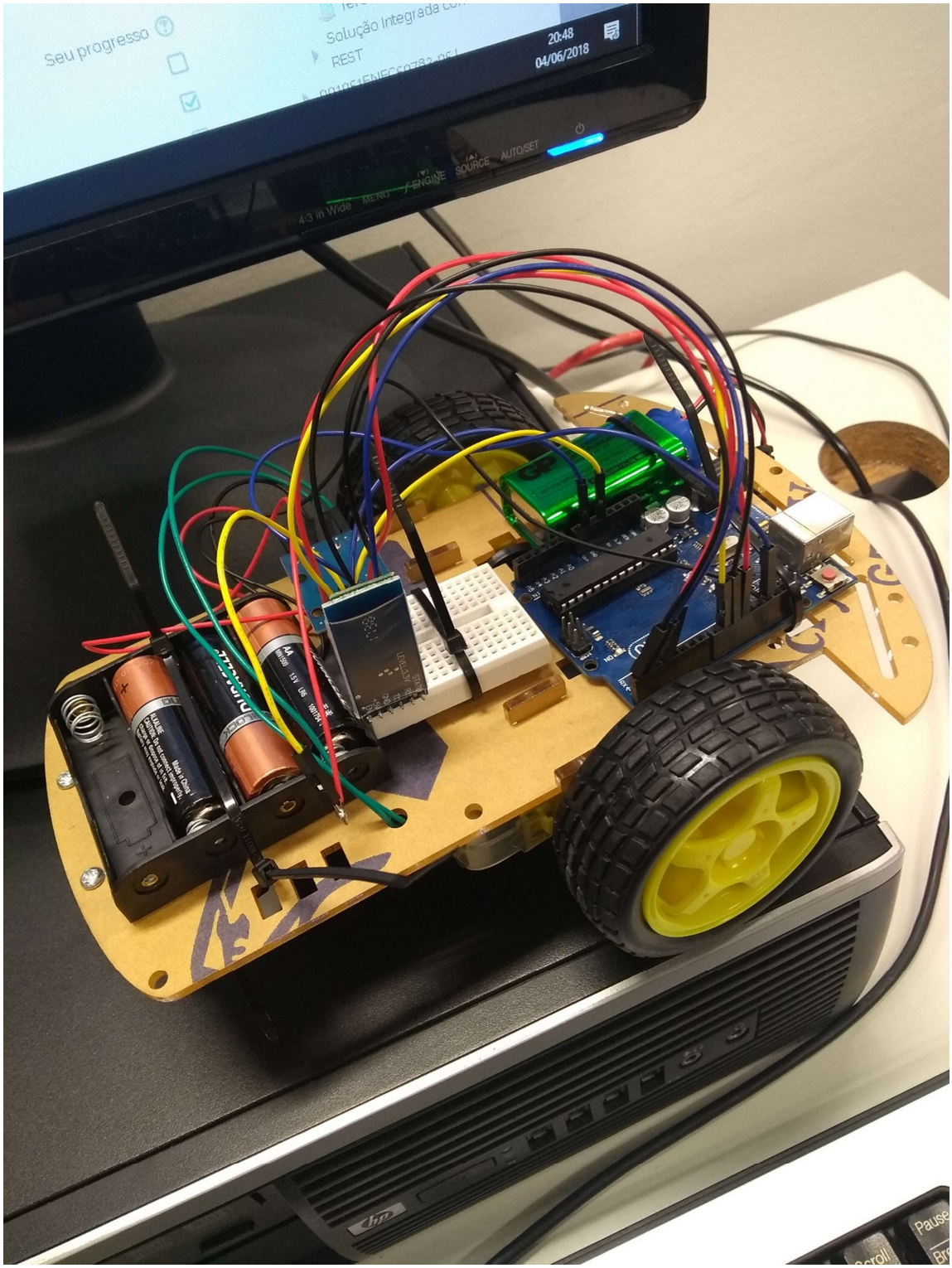
Julio Maximo Cano → 31645186

Leonardo Bacic Cezar → 31627293

Matheus Fontanetti Martins →31618571

William Kazuo → 31333036

HARDWARE

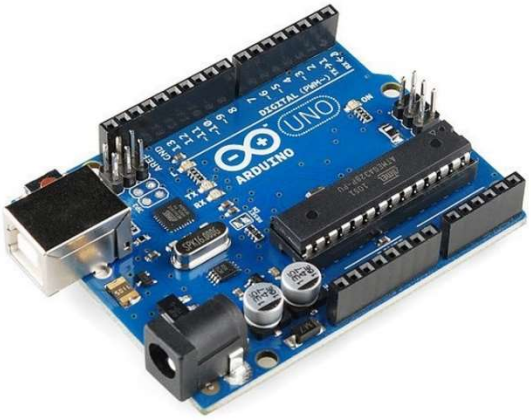

Seu progresso
Solução integrada
REST
20:48
04/06/2018
4:3 in Wide
MENU
ENGINE
SOURCE
AUTO/SET

Arduino Multi Chassis – Tração 2 Rodas

Especificações:

- 01 - Chassi em acrílico
  02 - Motores DC (3~6v)
  02 - Rodas de Borracha
  01 - Roda articulada
  02 - Discos de Encoder
  01 - Suporte para 4 Pilhas
  01 - kit de parafusos

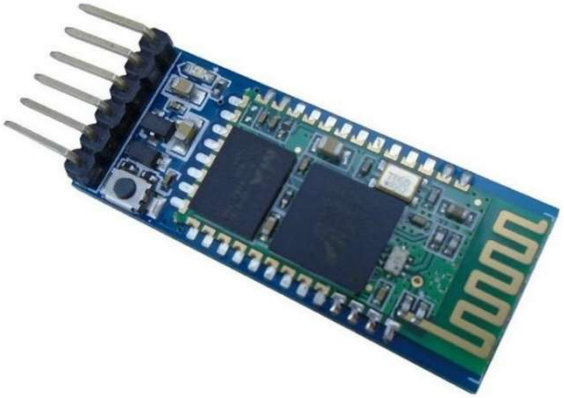
Arduino UNO



Especificações :

Micro controlador	ATmega328
Tensão de operação	5V
Tensão de alimentação (recomendada)	7-12V
Tensão de alimentação (limite)	6-20V
Entradas e saídas digitais	14 das quais 6 podem ser PWM
Entradas analógicas	6
Corrente contínua por pino de I/O	40 mA
Corrente contínua para o pino 3.3V	50 mA
Memória Flash	32 KB (ATmega328) dos quais 0.5 KB são usados pelo bootloader
Memória SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidade do Clock	16 MHz
Dimensões	68,58mm x 53,34mm
Peso	150g

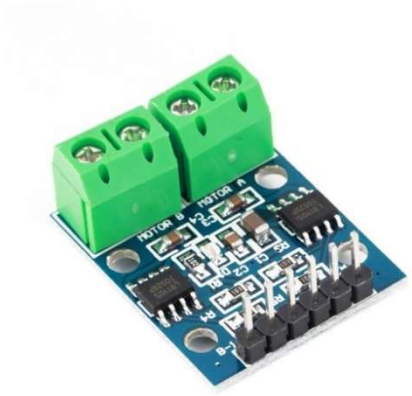
Módulo Bluetooth HC-06



Especificações :

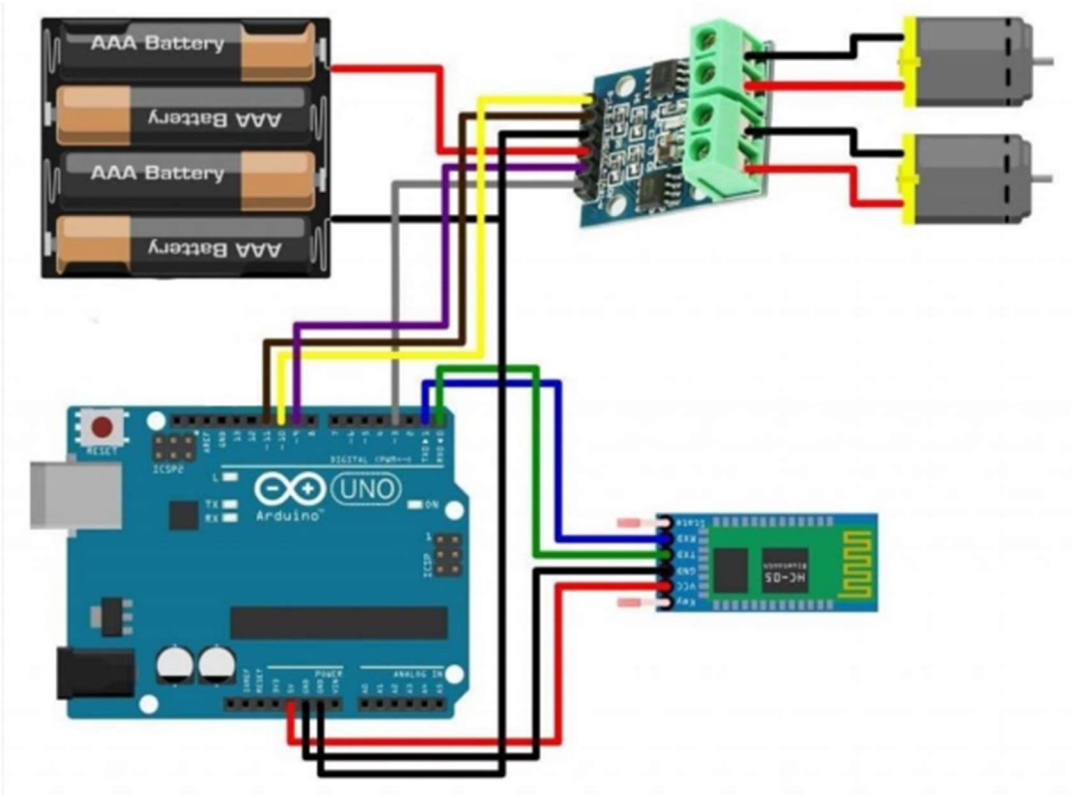
- Bluetooth: 2.0V
- Tensão de funcionamento: 3.3v~5v
- Taxa de transmissão: 2Mbps
- Frequência: 2,4 Ghz
- Nível lógico: 3.3v
- Pinos: VCC , GND , TXD , RXD;
- Perfis suportados: Escravo (slave) e Mestre (master)

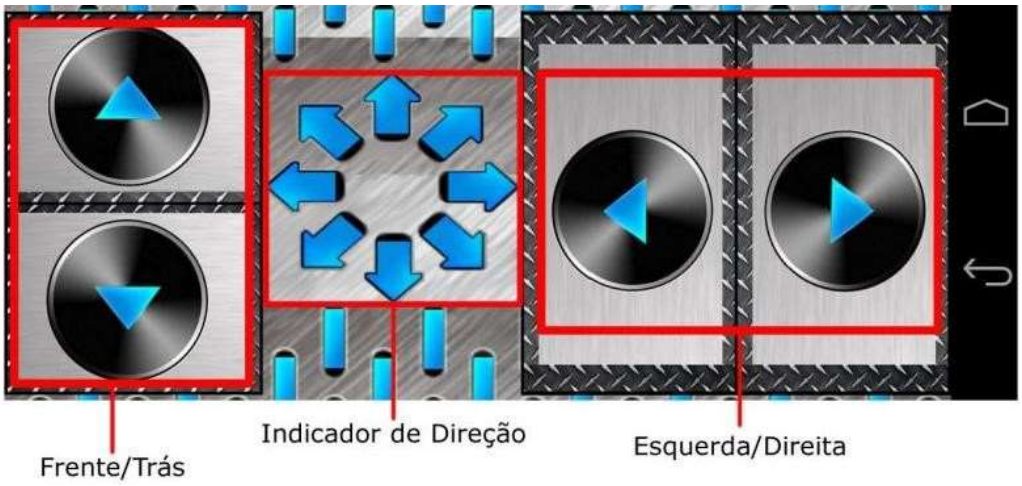
Ponte H L9110S



Especificações:

- Chip Driver: Chip dupla ponte H L9110s
- Tensão de alimentação: 2,.5V~12V
- Corrente nominal: 800mA por motor
- Pico de corrente de Saída: 1.5A~2A por porta (Somente pico, ou seja, breve)
- Tensão dos terminais de controle: 2.5~7.7 V
- Corrente dos terminais de controle: 500uA
- Temperatura de trabalho: 0ºC ~ +80ºC
- Outras características: Verificar datasheet





CÓDIGO

```
/* Carro com controle Bluetooth */
// Define os pinos de utilização do Driver L298.
const int motorA1  = 9;    // Pin  5 of L293.
```

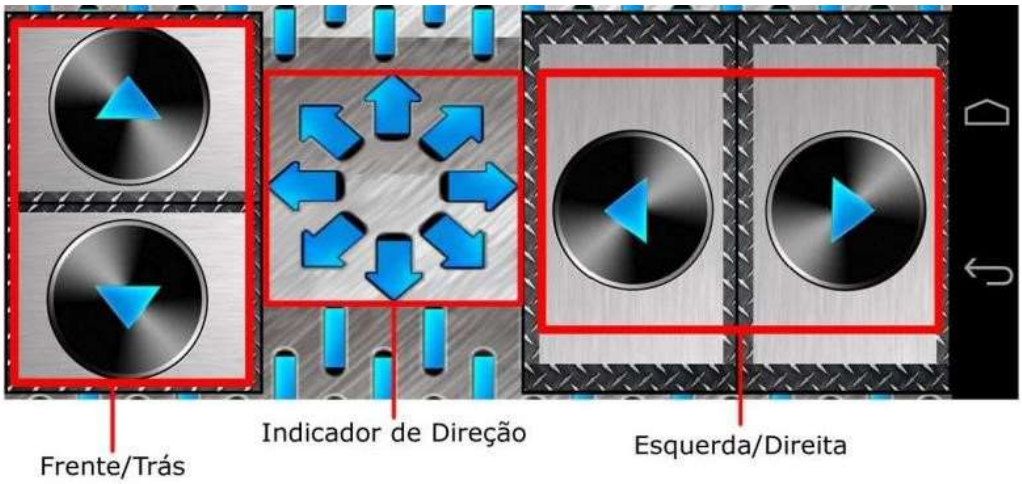
```
const int motorA2  = 3;    // Pin  6 of L293.
const int motorB1  = 11;   // Pin 10 of L293.
const int motorB2  = 10;   // Pin 9 of L293.


const int buzzer = 12 ;   // Define o Pino 13 como pino do Buzzer.


const int BTState = 2;    // Define o Pino 2 como o pino de comunicação do
Bluetooth.


// Variáveis Úteis
int i = 0;
int j = 0;
int state_rec;
int vSpeed = 200;   // Define velocidade padrão 0 < x < 255.
char state;


void setup() {
  // Inicializa as portas como entrada e saída.
  pinMode(motorA1, OUTPUT);
  pinMode(motorA2, OUTPUT);
  pinMode(motorB1, OUTPUT);
  pinMode(motorB2, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(BTState, INPUT);


  // Inicializa a comunicação serial em 9600 bits.
  Serial.begin(9600);
}


void loop() {
  // Para o carro quando a conexão com Bluetooth é perdida ou desconectada.
  if (digitalRead(BTState) == LOW) {
    state_rec = 'S';
  }
```

```
  // Salva os valores da variável 'state'
  if (Serial.available() > 0) {
    state_rec = Serial.read();
    state = state_rec;
    //   Serial.println(vSpeed);
  }


  // Altera a velocidade de acordo com valores especificados.
  if (state == '0') {
    vSpeed = 0;
  }
  else if (state == '4') {
    vSpeed = 100;
  }
  else if (state == '6') {
    vSpeed = 155;
  }
  else if (state == '7') {
    vSpeed = 180;
  }
  else if (state == '8') {
    vSpeed = 200;
  }
  else if (state == '9') {
    vSpeed = 230;
  }
  else if (state == 'q') {
    vSpeed = 255;
  }


  if (state != 'S') {
    Serial.print(state);
  }


  // Se o estado recebido for igual a 'F', o carro se movimenta para
frente.
  if (state == 'F') {
    analogWrite(motorB1, vSpeed);
    analogWrite(motorA1, vSpeed);
```

```
    analogWrite(motorA2, 0);
    analogWrite(motorB2, 0);
  }


    else if (state == 'I') {  // Se o estado recebido for igual a 'I', o
carro se movimenta para Frente Esquerda.
    analogWrite(motorA1, vSpeed);
    analogWrite(motorA2, 0);
    analogWrite(motorB1, 100);
    analogWrite(motorB2, 0);
  }


    else if (state == 'G') {   // Se o estado recebido for igual a 'G', o
carro se movimenta para Frente Direita.
    analogWrite(motorA1, 100);
    analogWrite(motorA2, 0);
    analogWrite(motorB1, vSpeed);
    analogWrite(motorB2, 0);
  }


  else if (state == 'B') { // Se o estado recebido for igual a 'B', o carro
se movimenta para trás.
    analogWrite(motorA1, 0);
    analogWrite(motorB1, 0);
    analogWrite(motorB2, vSpeed);
    analogWrite(motorA2, vSpeed);
  }


   else if (state == 'H') {  // Se o estado recebido for igual a 'H', o
carro se movimenta para Trás Esquerda.
    analogWrite(motorA1, 0);
    analogWrite(motorA2, vSpeed);
    analogWrite(motorB1, 0);
    analogWrite(motorB2, 100);
  }

  else if (state == 'J') {  // Se o estado recebido for igual a 'J', o
carro se movimenta para Trás Direita.
```

```
    analogWrite(motorA1, 0);
    analogWrite(motorA2, 100);
    analogWrite(motorB1, 0);
    analogWrite(motorB2, vSpeed);
  }


  else if (state == 'L') {   // Se o estado recebido for igual a 'L', o
carro se movimenta para esquerda.
    analogWrite(motorA1, 0);
    analogWrite(motorA2, vSpeed);
    analogWrite(motorB1, vSpeed);
    analogWrite(motorB2, 0);
  }
  else if (state == 'R') {   // Se o estado recebido for igual a 'R', o
carro se movimenta para direita.
    analogWrite(motorA1, vSpeed);
    analogWrite(motorA2, 0);
    analogWrite(motorB1, 0);
    analogWrite(motorB2, vSpeed);
  }
  else if (state == 'S') {   // Se o estado recebido for igual a 'S', o
carro permanece parado.
    analogWrite(motorA1, 0);
    analogWrite(motorA2, 0);
    analogWrite(motorB1, 0);
    analogWrite(motorB2, 0);
  }
  else if (state == 'V') { // Se o estado recebido for iguala 'V', aciona a
buzina.
    if (j == 0) {
      tone(buzzer, 1000);
      j = 1;
    }
    else if (j == 1) {
      noTone(buzzer);
      j = 0;
    }
    state = 'n';
  }
}
```

PROTOCOLOS



A pilha de protocolos Bluetooth é dividida em três partes:

• Camada de Transporte;

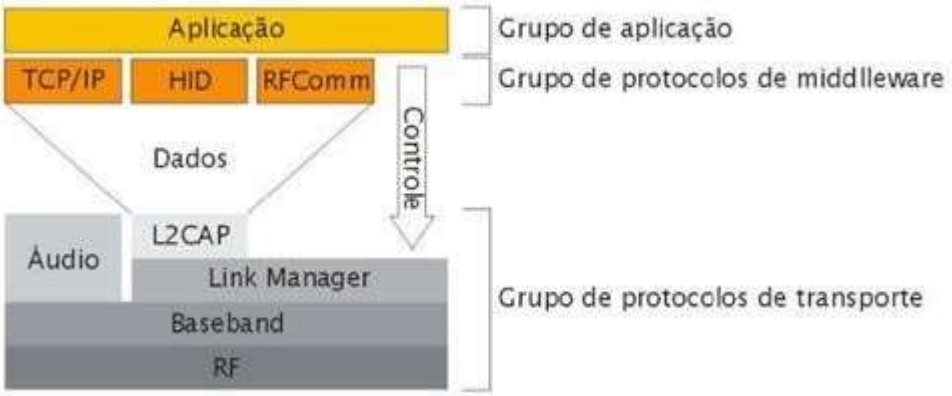
• Camada Middleware;

• Camada de Aplicação.

Os protocolos de transporte são responsáveis por localizar os dispositivos e gerenciar os links físicos e lógicos entre eles. Suportam tanto conexões síncronas quanto assíncronas e englobam as camadas de rádio frequência (RF), Baseband, Link Manager, Logical Link Control and Adaptation (L2CAP).

Os protocolos de Middleware são responsáveis por permitir a interação entre aplicações antigas e novas. Padrões como Point-toPoint Protocol (PPP), Wireless Application Protocol (WAP), Internet Protocol (IP),Trasmission Control Protocol (TCP) fazem parte desta camada.

A camada de aplicação faz referência aos aplicativos que podem usufruir da especificação bluetooth. A Figura 2 representa a pilha de protocolo bluetooth.


Aplicação
TCP/IP
HID
RFComm
Dados
Controle
Áudio
L2CAP
Link Manager
Baseband
RF
Grupo de aplicação
Grupo de protocolos de middlleware
Grupo de protocolos de transporte