

Bruno de Araújo Alves
Thauan da Silva Cruz
Hugo Vinícius Leão e Silva

Desenvolvimento de um veículo controlado por smartphone para aprendizagem de algoritmos, programação e arquitetura de computadores

Brasil

2020

Bruno de Araújo Alves
Thauan da Silva Cruz
Hugo Vinícius Leão e Silva

**Desenvolvimento de um veículo controlado por
smartphone para aprendizagem de algoritmos,
programação e arquitetura de computadores**

Modelo canônico de Relatório Técnico e/ou
Científico em conformidade com as normas
ABNT apresentado à comunidade de usuários
L^AT_EX.

Instituto Federal de Educação, Ciência e Tecnologia de Goiás - IFG

Campus Anápolis

Bacharelado em Ciência da Computação

Brasil

2020

Sumário

1	INTRODUÇÃO	3
1.1	Motivação e objetivo geral	3
1.2	Componentes necessários	3
2	DESENVOLVIMENTO	7
2.1	Início	7
2.2	Ligação dos componentes	9
2.2.1	Ligação do Arduino e da fonte de alimentação à <i>protoboard</i>	9
2.2.2	Montagem do <i>buzzer</i>	10
2.2.3	Montagem do sensor de umidade e temperatura	10
2.2.4	Montagem do sensor de distância	11
2.2.5	Montagem do módulo Bluetooth	12
2.2.6	Montagem do sensor fotorresistor	13
2.2.7	Montagem dos LEDs	13
2.2.8	Montagem do ponte H e dos motores	15
2.3	Baixando e instalando o Arduino IDE	17
2.4	Baixando o repositório no GitHub	18
2.5	Programando o Arduino	19
2.6	Montando o veículo	22
2.7	Celular	24
3	DESENVOLVIMENTO DO APLICATIVO	33
3.1	Modo de desenvolvedor	33
3.2	Programando o aplicativo	33

1 Introdução

1.1 Motivação e objetivo geral

Esta documentação é o resultado da realização de um Projeto de Inovação Tecnológica por estudantes do curso de bacharelado em Ciência da Computação do Instituto Federal de Goiás (IFG) no Câmpus Anápolis.

É notório que existe uma grande dificuldade entre estudantes dos períodos iniciais de cursos de Computação no aprendizado de criação de algoritmos. Isso vem da falta de aptidão desses estudantes na criação de pensamentos estruturados para realizar tarefas.

No primeiro período de cursos de Computação também é comum encontrar disciplinas que, de maneira muito introdutória, ensinam como um computador funciona. Nesse contexto, essa disciplina aborda conceitos de instrução de computadores e de linguagem de programação e como elas se interligam. Porém, devido à dificuldade em criar algoritmos, muitos estudantes ainda não conseguem abstrair esses conceitos da maneira que se espera.

Assim, surgiu a ideia de se criar um veículo controlado remotamente por um *smartphone*. Esse veículo, em vez de ser controlado através dos controles tradicionais – frente, trás, esquerda, direita –, é controlado por uma sequência de comandos inseridos no celular. Em outras palavras, o veículo deve ser controlado por um programa de computador.

Porém, para que esse controle traga desafios semelhantes aos que um estudante de primeiro período de cursos de Computação, deve-se permitir o uso de blocos de controle como o SE e laços de repetição, além de ter suporte a variáveis. Adicionalmente, para que o veículo interaja com o ambiente, alguns sensores foram adicionais, que permitem ler valores de intensidade de luminosidade, umidade, temperatura e distância.

Com esse veículo, o estudante poderá visualizar o funcionamento de blocos básicos de alguns comandos utilizados para construir algoritmos, podendo se tornar uma útil ferramenta para facilitar o seu aprendizado. Professores de algoritmos também podem utilizá-lo para auxiliar os seus estudantes, ensinando-os a montar o veículo e mostrando como os blocos de comandos são convertidos em instruções que ele é capaz de entender. Assim, é possível ensinar conceitos iniciais de arquitetura de computadores.

1.2 Componentes necessários

Para montar o veículo são necessários, no mínimo, os seguintes equipamentos:

- Um Arduino Uno R3 juntamente com o seu cabo USB;

- Uma *protoboard* de, preferencialmente, 830 conexões;
- Um *buzzer* piezoelétrico de cinco volts;
- Um sensor de umidade e temperatura DHT22;
- Um sensor de distância HC-SR04;
- Um módulo Bluetooth HC-05;
- Um sensor fotorresistor LDR 5mm;
- Uma ponte H L298N;
- Um kit para veículo com chassis e motores elétricos;
- Uma fonte de alimentação ajustável para *protoboard*;
- Dois resistores de $220\ \Omega$;
- Dois resistores de $1\ K\Omega$;
- Um resistor de $4,7\ K\Omega$;
- Um resistor de $10\ K\Omega$;
- Dois diodos emissores de luz (LEDs);
- 45 fios (também chamados de *jumper wires*) com terminações macho-macho de cores variadas;
- Um cabo USB para conectar o *smartphone* Android ao computador.

Com exceção do kit para montagem do veículo, que está ilustrado na Figura 26a, os sensores e módulos dessa lista estão ilustrados nas Figuras XX a XX para facilitar a identificação de cada um deles.

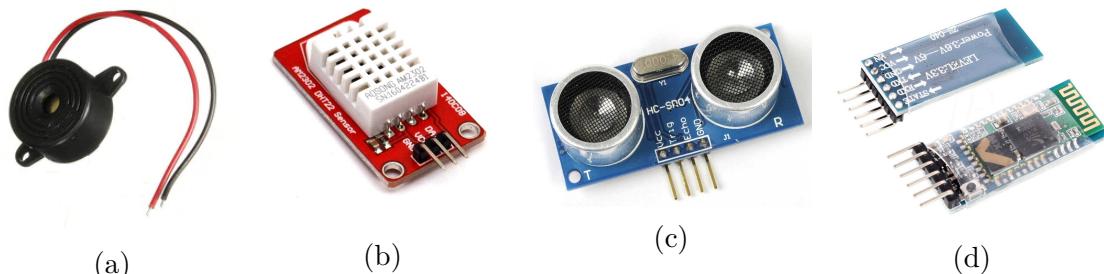


Figura 1 – *Buzzer* (a), sensor de umidade e temperatura DHT22 (b), sensor de distância HC-SR04 (c) e módulo Bluetooth HC-05 (d).

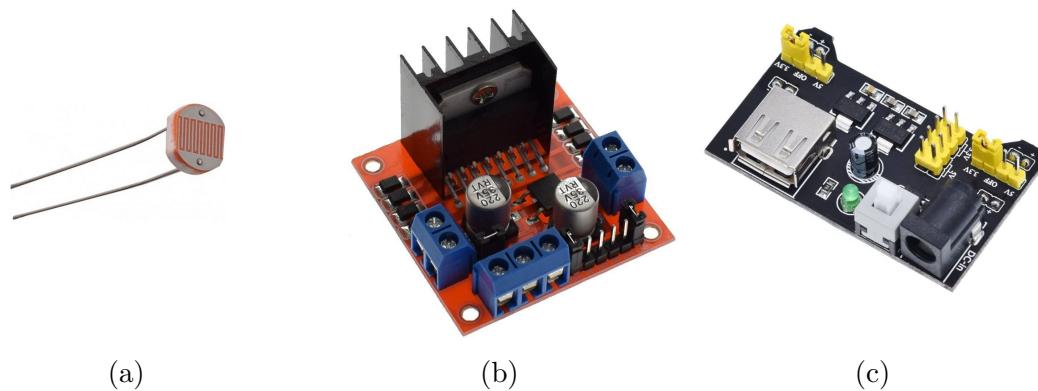


Figura 2 – Sensor de luminosidade LDR 5mm (a), ponte H L298N para a conexão de motores de corrente contínua (b) e fonte de alimentação ajustável para *protoboard* (c).

2 Desenvolvimento

2.1 Início

Este projeto utiliza um Arduino Uno R3 e uma placa de prototipação (*protoboard*) como componentes principais e, por isso, é preciso entender um pouco melhor o funcionamento deles.

A *protoboard* é onde todos os componentes se conectam e sua estrutura é basicamente dividida em três partes: 1) barramentos superiores, 2) barramentos centrais e 3) barramentos inferiores. É através desses barramentos em que são feitas as conexões entre todos os componentes que formam a parte eletrônica do veículo. Sendo assim, é importante entendermos como são as ligações desses barramentos, tendo como referência a Figura 3.

Os barramentos superiores e inferiores da *protoboard* são nomeados “+” e “-” e têm como principal função o fornecimento de corrente elétrica aos componentes ligados nela. Um ponto importante a se observar é que todas as conexões do barramento “+” superior estão ligadas entre si, assim como para barramento “-” superior, como pode ser observado o destaque em vermelho na parte superior da Figura 3 e os dois barramentos inferiores. Dessa forma, ao ligar uma fonte de alimentação à *protoboard*, basta ligar os seus terminais positivo e negativo utilizando apenas um fio para cada, como pode ser visto na Figura ???. Observe nessa figura que o terminal positivo é conectado utilizando um fio de cor vermelha e o terminal negativo, um fio de cor preta. Mantenha esse padrão na conexão de outros componentes durante a montagem desse projeto.

Os barramentos centrais, por sua vez, formam duas matrizes de conexão independentes entre si e cada conexão é nomeada por sua linha (de “A” a “J”) e por sua coluna (de “1” a “30”). Por exemplo, a conexão “D5” é aquela que está na quarta linha, quinta coluna. Diferentemente do que ocorre com os barramentos superiores e inferiores, nesses barramentos centrais, as conexões em uma coluna são ligadas entre si. Pode-se ver isso nos destiques dados em vermelho às colunas “1” e “2” na Figura 3, que ilustram que as conexões de “A1” a “E1” estão ligadas entre si, assim como de “A2” a “E2”. Ainda, vale ressaltar que as conexões de “F1” a “J1” são completamente independentes de “A1” a “E1”.

Como já citado anteriormente, o modelo utilizado do Arduino é o Uno R3, ilustrado na Figura 4. Ele possui conexões para transmissão ou recepção de sinais analógicos e digitais, além de conexões para energizar ou o Arduino ou algum componente que esteja ligado a ele, que são explicados a seguir.

Na parte superior dessa figura destacam-se conexão nomeada “GND” (ou “terra”) e as conexões de “0” a “13”, para transmitir ou receber dados. As conexões cuja identificação



Figura 3 – Ilustração de uma *protoboard*, com destaque aos barramentos.

possui o símbolo “~”, como em “~3” e “~5” permitem o envio e a recepção de sinais digitais puros ou modulados por largura de pulso (PWM, do inglês *“Pulse Width Modulation”*), enquanto nas outras conexões, apenas sinais digitais.

Na parte inferior, as conexões de “AO” a “A5” servem apenas para a recepção de sinais analógicos e as conexões na seção “Power” permitem energizar ou o Arduino ou outro componente. As duas conexões “GND” são para o terra, enquanto “3V3” aceita tensão de entrada ou saída de 3,3 volts e “5V”, uma tensão de cinco volts. Sobre essas duas conexões, é importante reforçar que, se elas forem utilizadas para energizar o Arduino, como é o caso neste projeto, elas devem estar conectadas a uma fonte ajustável para *protoboard*. Como primeira atividade, deve-se ligar o Arduino à *protoboard* como ilustrado na Figura 4, conectando o “5V” do Arduino ao barramento “+” do *protoboard* e “GND” ao “-”.

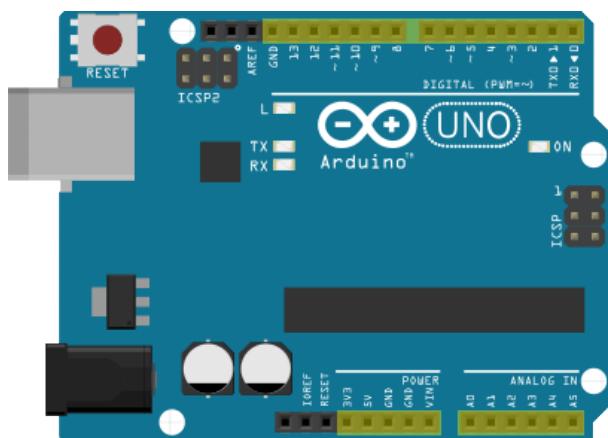


Figura 4 – Arduino UNO R3

2.2 Ligação dos componentes

2.2.1 Ligação do Arduino e da fonte de alimentação à *protoboard*

A primeira conexão a ser feita é a de alimentação, para isso é preciso conectar as portas “5V” e “GND” a *protoboard* assim como é mostrado na Figura 5. Lembre-se de seguir por toda a construção do veículo o padrão de cores para esses dois fios: vermelho para cinco volts e preto para terra.

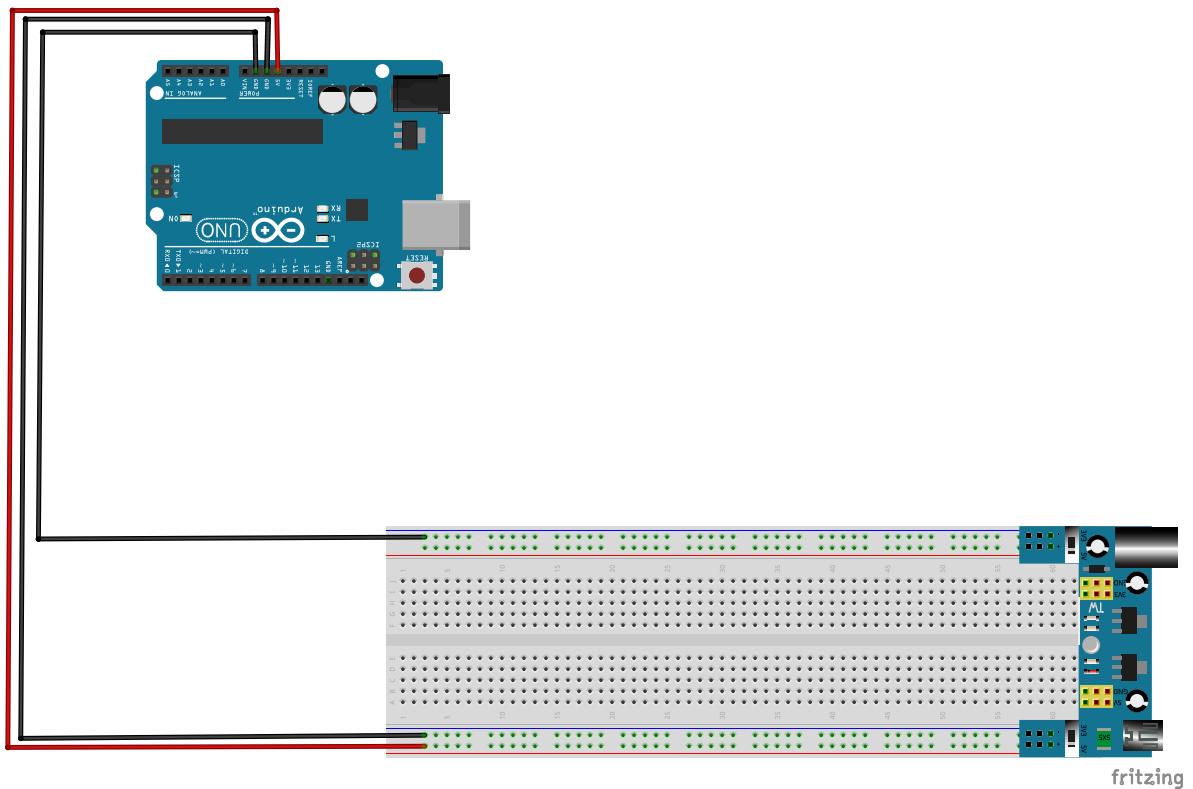


Figura 5 – Esquema de ligação do Arduino e da fonte de alimentação à *protoboard*.

Para alimentar os vários sensores, o Arduino e os quatro motores que controlarão o veículo, uma fonte de alimentação ajustável para *protoboard* é necessária. Esse componente vai fornecer uma tensão de cinco volts para todos os componentes e deve ser conectado aos quatro barramentos horizontais da *protoboard* também como mostra a Figura 5.

Pequenas peças amarelas denominadas “*jumpers*”, como apresentadas na Figura 6a, acompanham o módulo e devem ser encaixadas nos pinos relativos para fornecer os cinco volts, como apresentado na Figura 6b.

Por fim, com todos os componentes devidamente conectados, alimente a fonte de alimentação conectando uma outra fonte de alimentação ao conector redondo do lado oposto à porta USB. Preferencialmente, esse o cabo dessa fonte deve ser longo para evitar que ele limite a movimentação do veículo. Essa fonte que estará ligada à fonte de alimentação ajustável da *protoboard* deve fornecer uma tensão entre 6,5 e 12 volts e

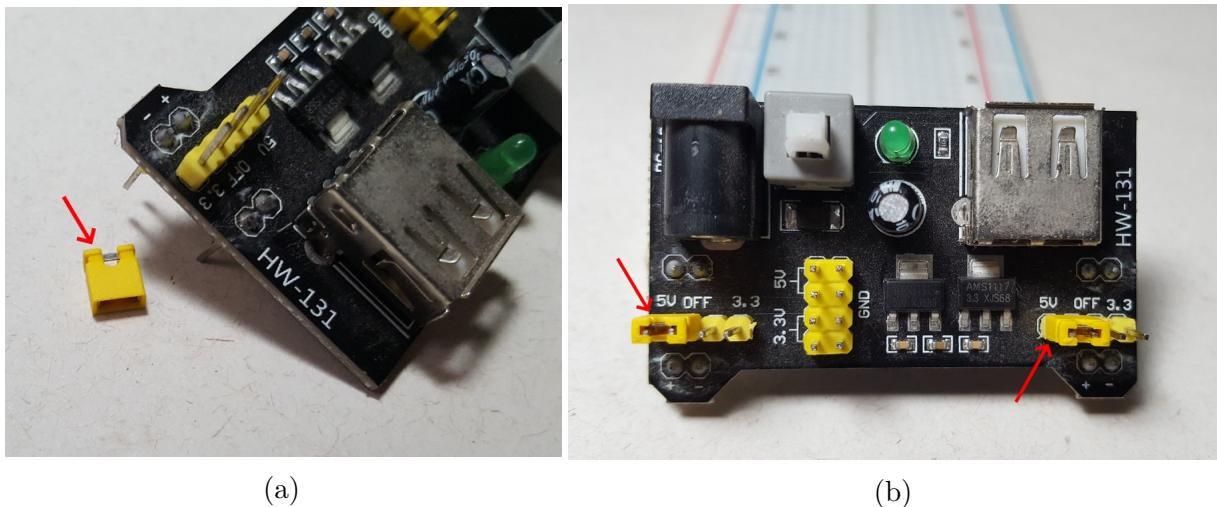


Figura 6 – Ilustração de um *jumper* à parte (a) e ele conectado nos pinos apropriados na fonte de alimentação (b).

um *ampère* de corrente elétrica contínua (DC). A Figura 5 ilustra todos os componentes ligados à *protoboard*.

2.2.2 Montagem do *buzzer*

Itens necessários:

- Um *buzzer*;
- Quatro fios com terminações macho-macho.

O *buzzer* é um dispositivo que emite som. Ele possui dois terminais, um negativo e um positivo. Repare que perto de um dos seus terminais há um sinal de “+”, significando que este é o terminal positivo. Portanto, conecte o terminal positivo na conexão “J7” da *protoboard* e, em seguida, conecte com um fio auxiliar a conexão “G7” à porta “~9” do Arduino. Feito isso, o terminal negativo do *buzzer* deve ser conectado em “J4”. Finalmente, utilizando um fio auxiliar, conecte “G4” no barramento “-” da *protoboard*.

2.2.3 Montagem do sensor de umidade e temperatura

Itens necessários:

- Um sensor de umidade e temperatura;
- Um resistor de 4,7 K Ω ;
- Cinco fios com terminações macho-macho.

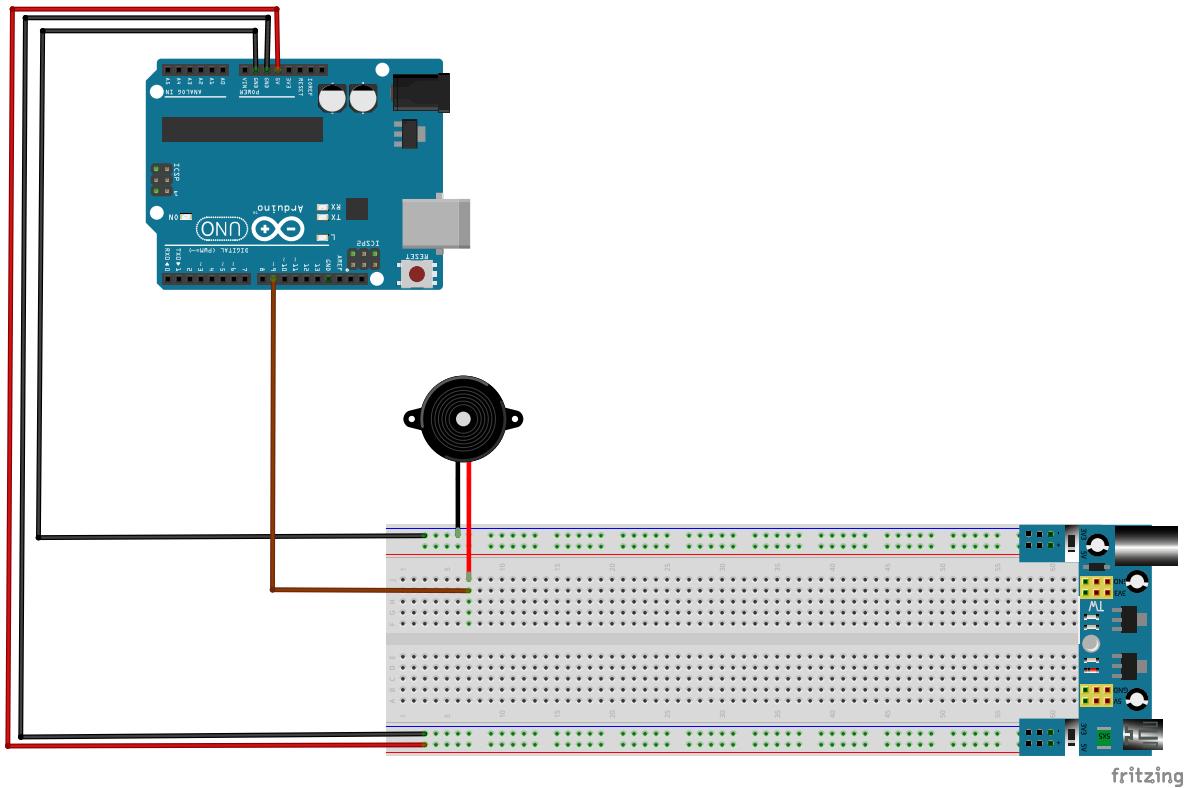


Figura 7 – Conexão do *buzzer* à *protoboard* e ao Arduino.

O sensor DHT22 possui quatro terminais de conexão. Considerando os terminais no sentido esquerda-direita, o primeiro é o terminal positivo, o segundo é o responsável pelo envio dos dados de umidade e temperatura para o Arduino e o quarto terminal é o terminal negativo. O terceiro terminal não é utilizado neste projeto.

Conecte os terminais negativos e positivos nos barramentos “+” e “-” da *protoboard* e, então, conecte o terminal três do sensor DHT22 em “J17” da *protoboard*. Conecte um resistor de 4,7 KΩ no barramento “+” da *protoboard* e em “J15”. Com um fio auxiliar conecte “J15” e “J17”. Após isso, conecte a porta “~3” do Arduino a “F17”, como ilustrado na Figura 8.

2.2.4 Montagem do sensor de distância

Itens necessários:

- Um sensor de distância HC-SR04;
- Seis fios com terminações macho-macho.

O sensor de distância HC-SR04 tem quatro terminais nomeados no próprio módulo. Da forma como é mostrado na Figura 9, os terminais no sentido esquerda-direita são “VCC”, “Trigger”, “Echo” e “GND”. Primeiramente, conecte “VCC” no barramento “+” da

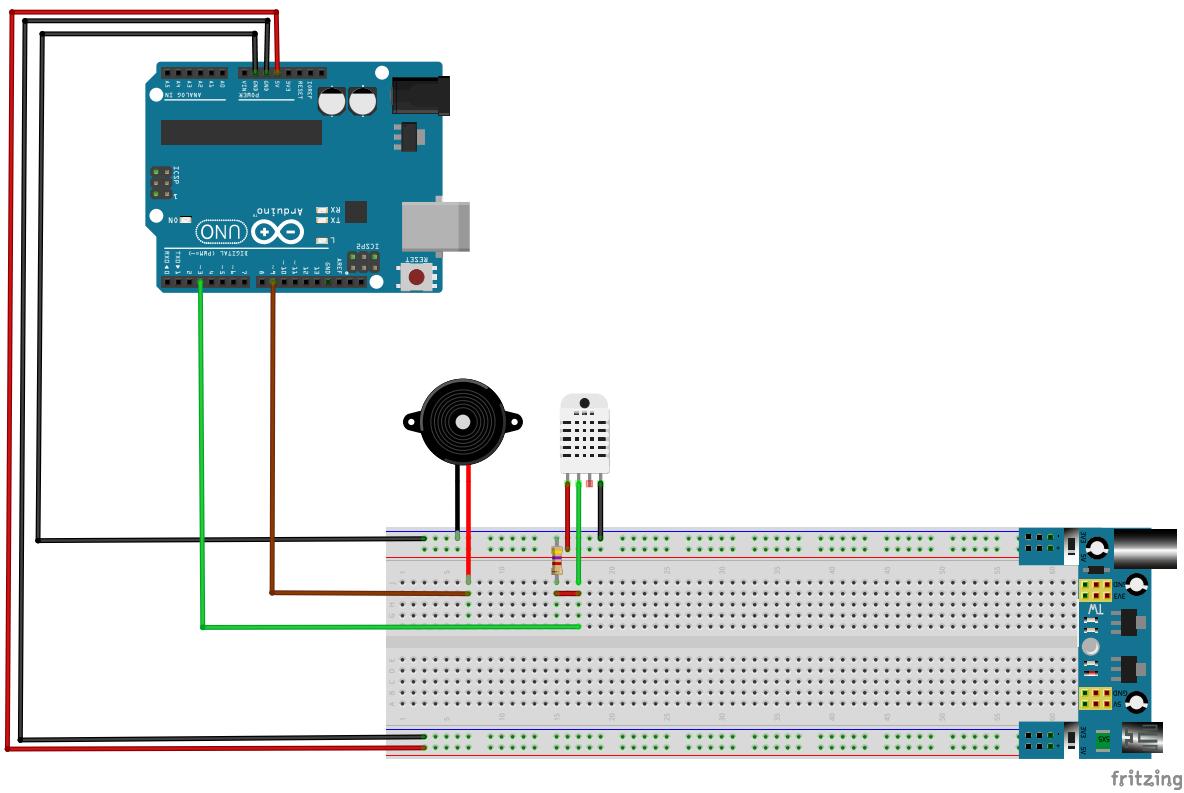


Figura 8 – Conexão do sensor de umidade e temperatura DHT22 à *protoboard* e ao Arduino.

protoboard e “GND” no barramento “-” da *protoboard*, sempre respeitando as cores dos fios. Em seguida, conecte “Trigger” e “Echo” respectivamente em “J28” e “J29” da *protoboard*. Após realizadas as conexões na *protoboard*, utilize fios auxiliares para ligar os terminais “Trigger” e “Echo” respectivamente às portas “~5” e “~6” do Arduino. A Figura 9 apresenta essas conexões.

2.2.5 Montagem do módulo Bluetooth

Itens necessários:

- Um módulo Bluetooth HC-05;
- Sete fios com terminações macho-macho;
- Dois resistores de $1\text{ K}\Omega$.

O módulo Bluetooth HC-05 tem seis terminais, sendo elas “Key”, “VCC”, “GND”, “TXD”, “RXD” e “State”. Conecte “VCC” e “GND” respectivamente aos barramentos “+” e “-” da *protoboard*. Em seguida, conecte “TXD” e “RXD” a “A11” e “A12”. Utilizando um fio auxiliar, conecte “D11” à porta “8” do Arduino. Utilizando um resistor de $1\text{ K}\Omega$, crie um divisor de tensão conectando o primeiro resistor a “C12” e “C16” e, com um fio auxiliar, conecte “A16” ao barramento “-” da *protoboard*. Ainda, conecte o outro resistor de $1\text{ K}\Omega$

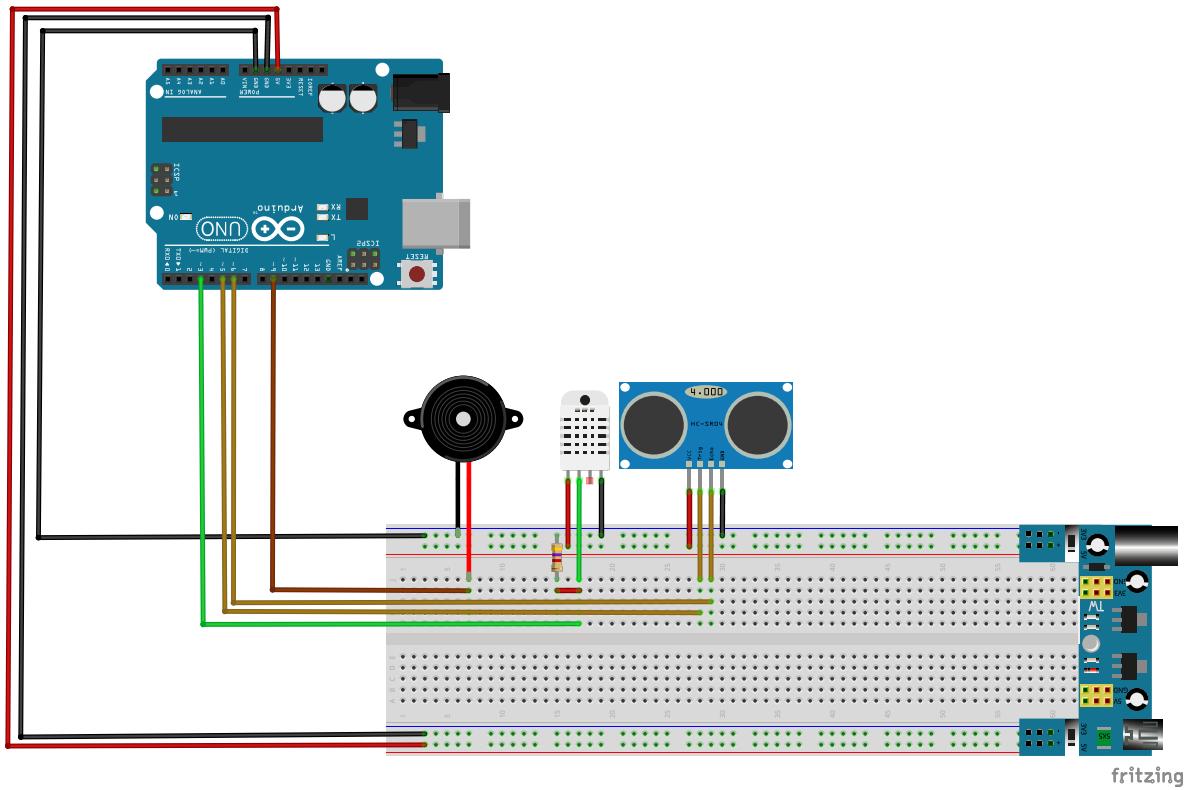


Figura 9 – Conexão do sensor de distância HC-SR04 à *protoboard* e ao Arduino.

a “B8” e “B12”. Finalmente, conecte com um fio auxiliar a “D8” e à porta “7” do Arduino. A Figura 10 mostra essas conexões.

2.2.6 Montagem do sensor fotorresistor

Itens necessários:

- Um sensor fotorresistor LDR 5mm;
- Quatro fios com terminações macho-macho;
- Um resistor de 10 KΩ.

Conecte um terminal do fotorresistor ao barramento “+” da *protoboard*. O outro terminal deve ser conectado a “A24”. Conecte o resistor de 10 KΩ conecte a “B24” e “B28” e, em seguida, utilize um fio auxiliar para conectar “A28” ao barramento “-” da *protoboard*. Utilizando outro fio auxiliar, conecte a “E24” à porta “A0” do Arduino. A Figura 11 ilustra essas conexões.

2.2.7 Montagem dos LEDs

Itens necessários:

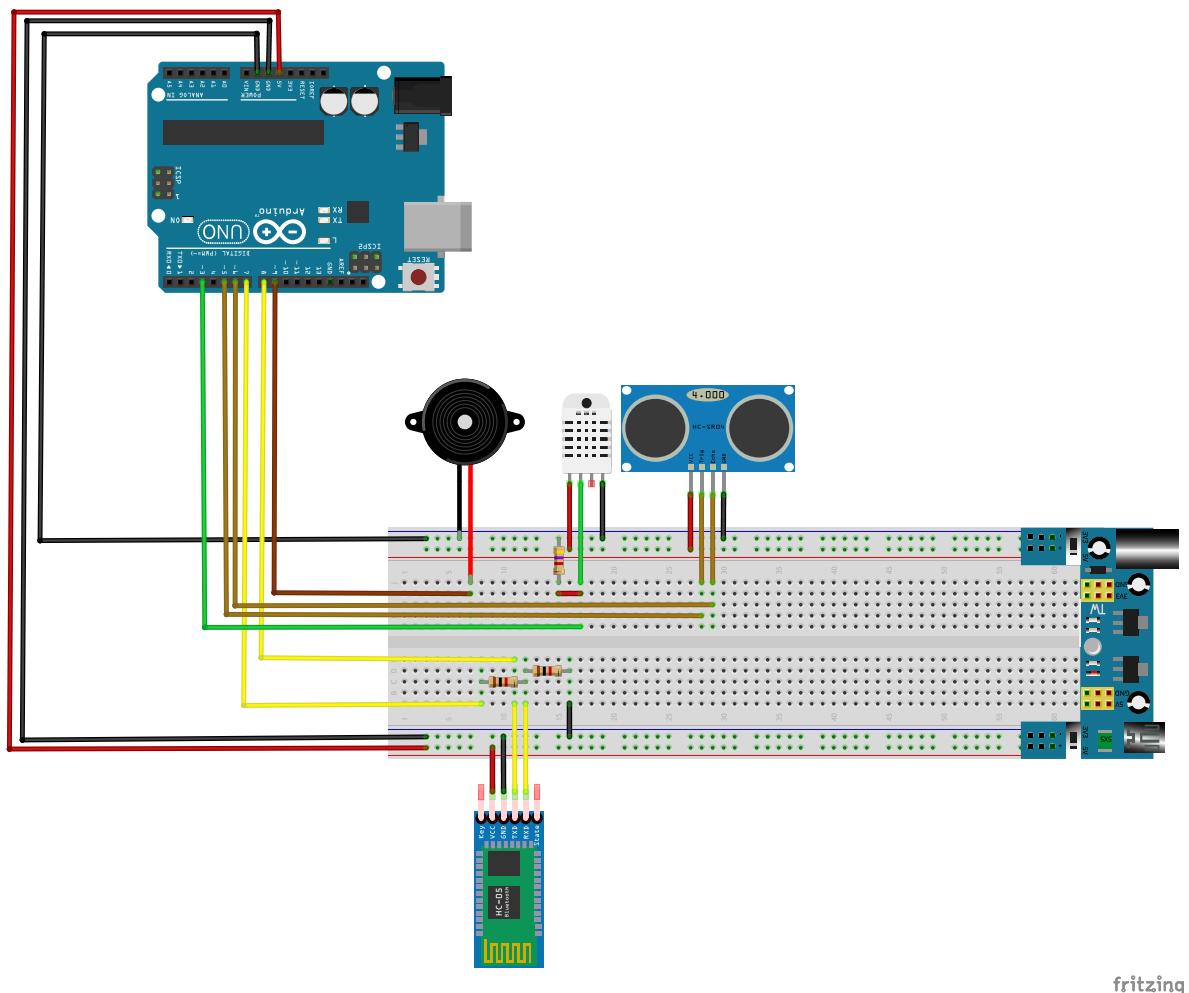


Figura 10 – Conexão do módulo Bluetooth HC-05 à *protoboard* e ao Arduino.

- Dois LEDs;
- Quatro fios com terminações macho-macho;
- Dois resistores de $220\ \Omega$.

Conecte os terminais positivos dos LEDs em “A36” e “A42”, respectivamente. Por sua vez, os terminais negativos dos LEDs devem ser conectados a “A37” e “A43”. Conecte os terminais de um resistor de $220\ \Omega$ em “B33” e “B37” e os do outro resistor em “B39” e “B43”. Utilizando um fio auxiliar, ligue “A33” ao barramento “-” da *protoboard*. Faça o mesmo com outro fio auxiliar, ligando “A39” a esse mesmo barramento. Finalmente, ligue um fio a “E36” na *protoboard* e à porta “2” no Arduino e ligue outro fio a “E42” na *protoboard* e à porta “4” no Arduino. A Figura 12 ilustra essas conexões.

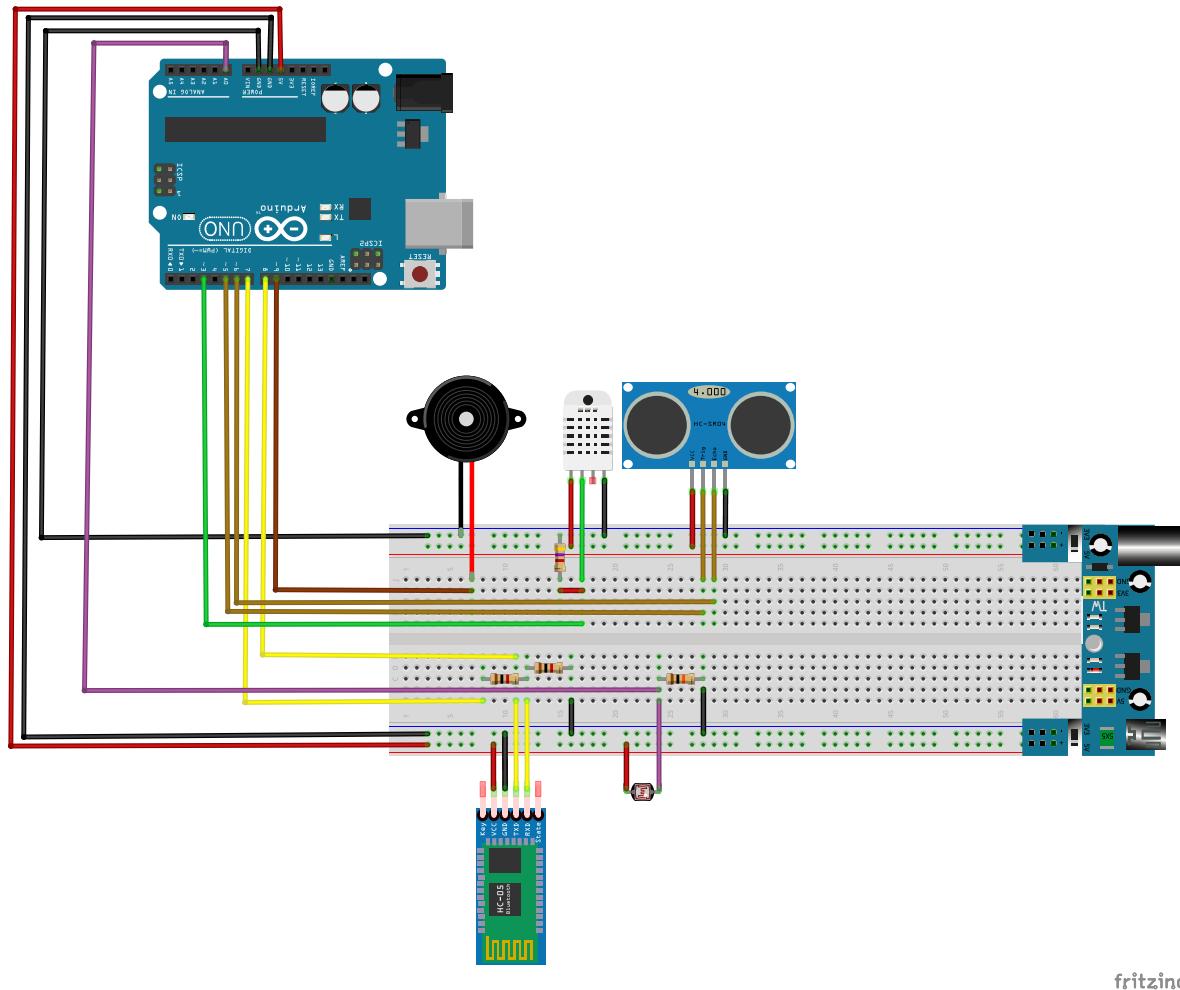


Figura 11 – Conexão do sensor fotorresistor LDR 5MM à *protoboard* e ao Arduino.

2.2.8 Montagem do ponte H e dos motores

Antes de começar este passo-a-passo é melhor conhecer um pouco mais como funciona a ponte H. Portanto, escolhemos uma página web que explica como este componente funciona e como podem ser feitas as conexões cujo *link* é <<https://www.filipeflop.com/blog/motor-dc-arduino-ponte-h-l298n/>>.

Itens necessários:

- Uma ponte H L298N;
- Quatro motores de corrente contínua de seis volts;
- 16 fios com terminações macho-macho.

Como apresentado na Figura 13a, conecte os fios aos terminais dos motores. Para esta fase é aconselhável que esses fios sejam soldados aos terminais dos motores para evitar

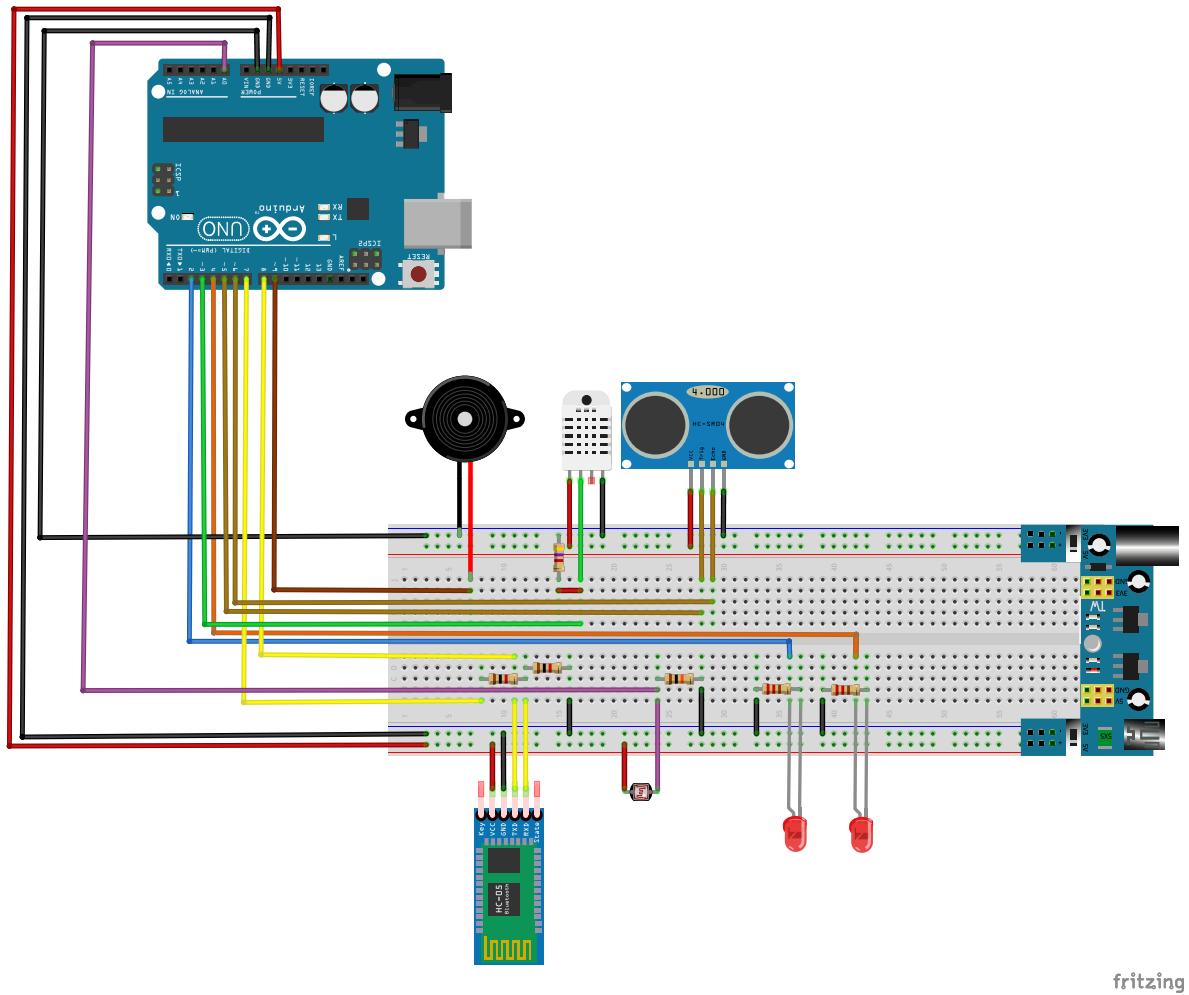


Figura 12 – Conexão dos LEDs à *protoboard* e ao Arduino.

que eles se soltem. Porém, os fios não serão conectados à ponte H neste momento como ilustrado na Figura 13b. Isso só deve ser feito durante o processo de montagem do veículo.

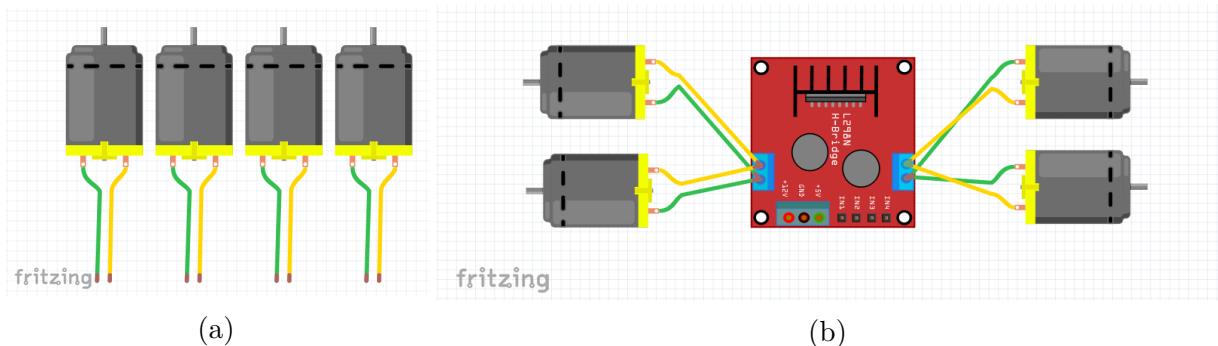


Figura 13 – Motores DC 6V com fios auxiliares soldados (a) e a conexão deles à ponte H (b).

Agora, como mostrado na Figura 14, conecte as entradas “IN1”, “IN2”, “IN3” e “IN4” da ponte H às entradas “~10”, “~11”, “12” e “13” do Arduino, respectivamente.

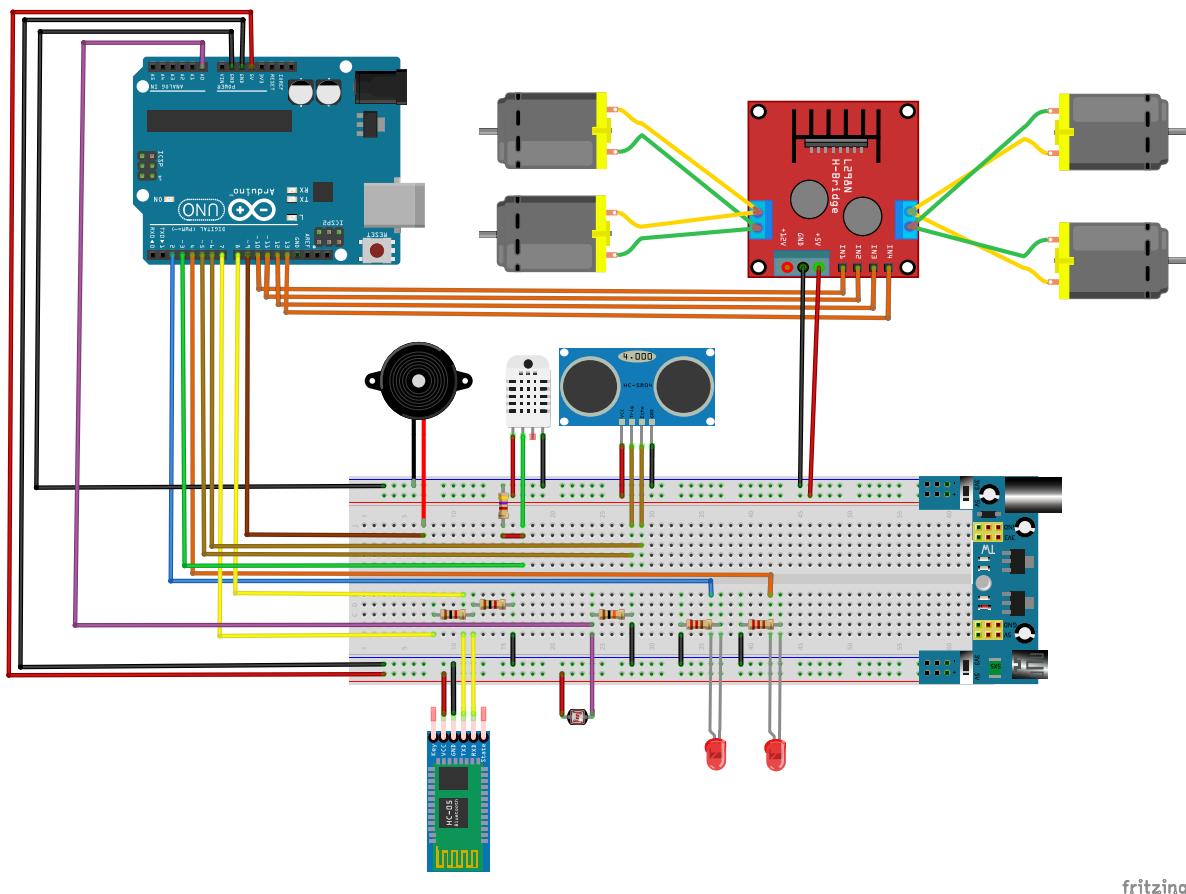


Figura 14 – Conexão da ponte H à *protoboard* e ao Arduino.

Realizada a conexão das entradas responsáveis por controlar quais motores serão ligados, é necessário conectar as entradas de energia. Para isso, conecte a entrada “+5V” da ponte H ao barramento “+” da *protoboard*. Já a entrada “GND” da ponte H deve ser conectada ao barramento “-” da *protoboard*, como ilustrado na Figura 14.

2.3 Baixando e instalando o Arduino IDE

Antes de programar o Arduino é necessário que se tenha instalado na máquina o *software* Arduino IDE em sua versão mais recente, cujo *link* é <<https://www.arduino.cc/en/main/software>>. Ao acessar esse *link*, é preciso que se navegue à seção do *site* ilustrada na Figura 15a onde é possível selecionar o sistema operacional que você utiliza em sua máquina. Considerando que o sistema operacional seja o Microsoft Windows, clique em “*Windows Installer for Windows 7 and up*”. Então, você será redirecionado para a tela de confirmação do *download* e clique em “*Just Download*”, como mostrado na Figura 15b.

Completado o *download*, que provavelmente estará na pasta “Downloads” no seu computador, clique duas vezes sobre o ícone do arquivo baixado e, assim, o instalador do Arduino IDE será iniciado. Com o instalador aberto, apenas clique nos botões “*I Agree*”,

Download the Arduino IDE

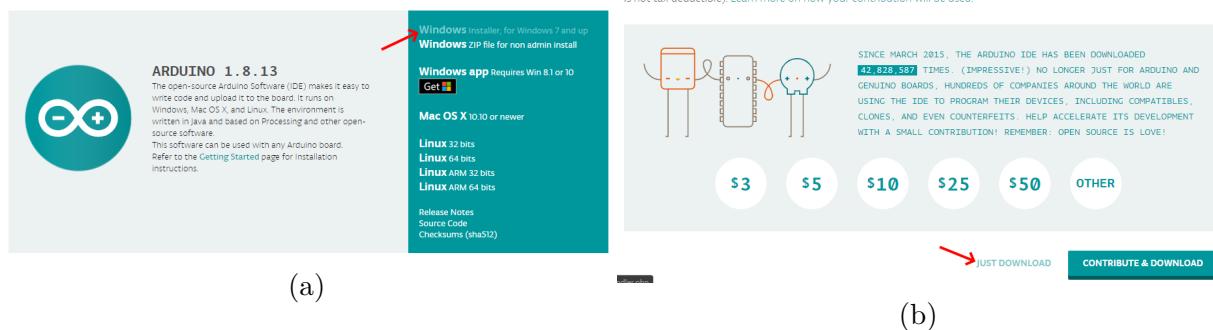


Figura 15 – Seção de seleção da versão do *download* do Arduino IDE segundo o sistema operacional (a) e tela de confirmação de *download* do Arduino IDE (b).

“*Next*”, “*Install*” e “*Done*”, como apresentado nas Figuras 16 e 17.

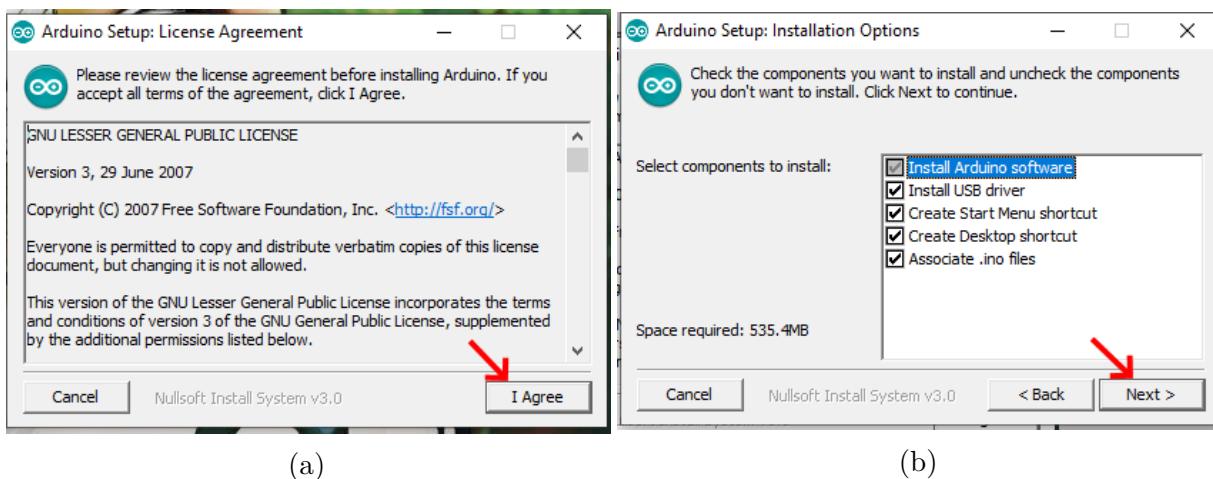


Figura 16 – Telas de instalação do Arduino IDE.

Agora, o Arduino IDE está instalado em seu computador e é possível encontrá-lo no menu iniciar do Windows.

2.4 Baixando o repositório no GitHub

Para que o veículo funcione corretamente, é necessário programá-lo antes de usá-lo. Para isso, primeiramente é preciso baixar o código disponível no repositório do GitHub no link https://github.com/devbaraus/automovel_pibit. Como apresentado na Figura 18, clique no botão “*Clone or download*” e, logo em seguida, clique em “*Download ZIP*”.

Após isso, haverá um arquivo chamado “`automovel_arduino-master.zip`” na pasta do repositório. A pasta será baixada para sua pasta “*Downloads*” do Windows. Utilizando algum descompactador de arquivos ZIP, extraia os arquivos para que possa ser utilizado nas próximas fases desta documentação. Extraído o arquivo ZIP abra a pasta “`automovel_arduino-master`” e verifique que as pastas são as mesmas apresentadas na Figura 19b.

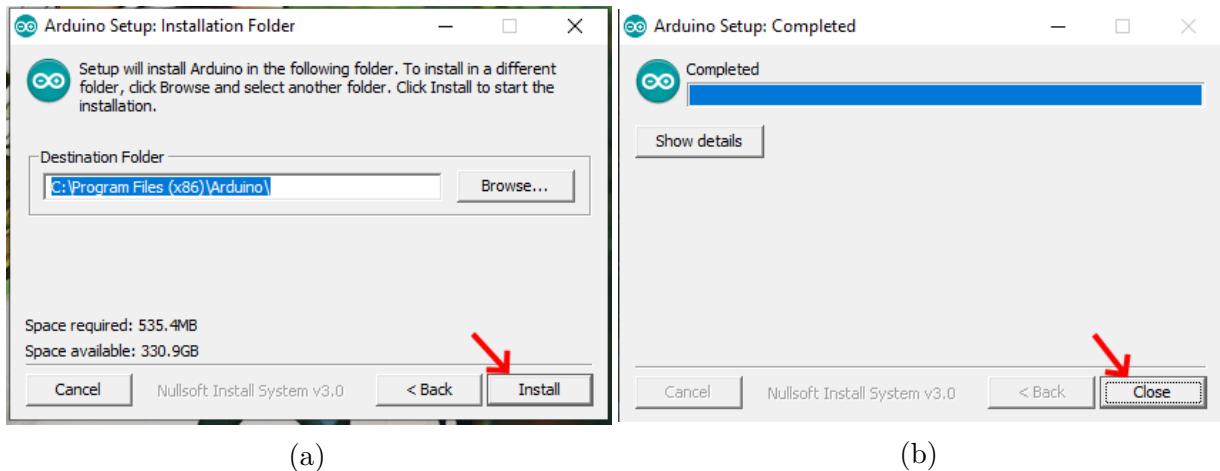


Figura 17 – Telas de instalação do Arduino IDE.

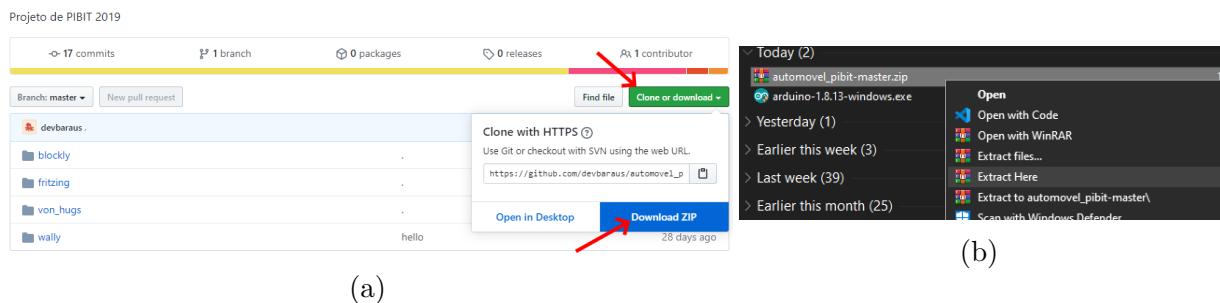


Figura 18 – Repositório do projeto no GitHub (a) e extração do arquivo ZIP do repositório (b).



Figura 19 – Pasta extraída do arquivo ZIP (a) e o seu conteúdo (b).

Agora, é preciso instalar as bibliotecas necessárias para que o código funcione corretamente. Caso contrário, o Arduino IDE não será capaz de realizar a compilação do código. Para tanto, dentro da pasta “automovel_pibit-master” entre em “von_hugs”. Lá dentro deve haver uma pasta nomeada “libraries”. Copie essa toda essa pasta para o caminho “Documentos/Arduino”. A pasta “libraries” já no local de destino deve possuir a localização e o conteúdo indicados na Figura 20b.

2.5 Programando o Arduino

Nesta parte do projeto é realizada a programação do Arduino para a comunicação dele com os módulos e sensores conectados. Considerando que os passos anteriores tenham

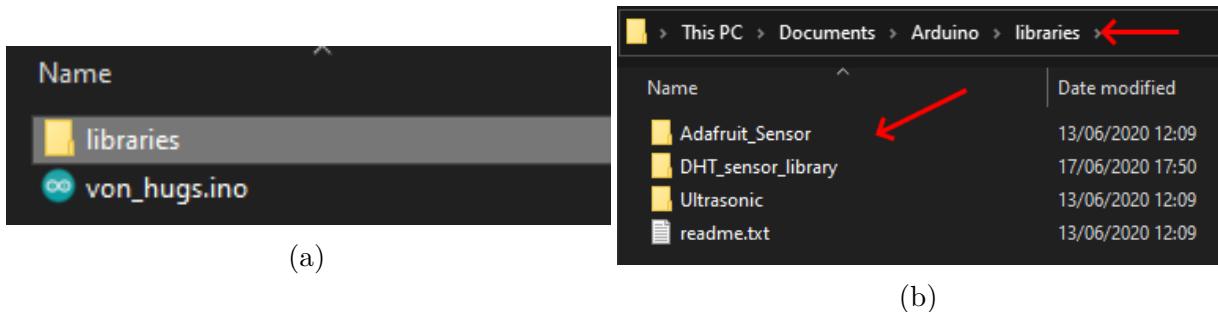


Figura 20 – Conteúdo da pasta `von_hugs` (a) e conteúdo da pasta `Documents/Arduino/libraries` após a cópia dos arquivos (b).

sido feitos corretamente, abra o arquivo no caminho “`Downloads/automovel_pibit-master/von_hugs/von_hugs.ino`” com o programa Arduino IDE. O resultado deve ser a tela apresentada na Figura 21.

```

von_hugs | Arduino 1.8.12
File Edit Sketch Tools Help
von_hugs
#include <SoftwareSerial.h>
#include "DHT.h"
#include <Ultrasonic.h>

// BLUETOOTH
// Bluetooth TX -> Arduino D8
#define BLUETOOTH_TX 8
// Bluetooth RX -> Arduino D7
#define BLUETOOTH_RX 7

SoftwareSerial bluetooth(BLUETOOTH_TX, BLUETOOTH_RX);

//MOTORES
#define PEF 10 // motor esquerdo para frente
#define PEB 11 // motor esquerdo pra tras
#define PDE 12 // motor direito para frente
#define PDB 13 // motor direito pra tras

//BUZZER
#define BUZZER 9
int freqBuzzer = 0;

//DHT - TEMPERATURA E HUMIDADE
#define DHTPIN 3 // what pin we're connected to
#define DHTTYPE DHT22 // DHT 22 (AM2302)

```

Figura 21 – Arduino IDE com o código-fonte “`von_hugs.ino`” aberto.

A partir deste momento é necessário que o Arduino Uno R3 esteja conectado ao computador. Assim que isso é feito, o Arduino IDE irá reconhecê-lo e será possível programá-lo, compilando o código-fonte e gravando o programa resultante em sua memória interna. Para tanto, clique no ícone na barra de ferramentas do Arduino IDE indicado na Figura 22a. Quando essa operação foi realizada com sucesso, deverá aparecer uma mensagem semelhante ao que está apresentada na Figura 22b.

Porém, para se ter certeza da correta programação do Arduino Uno R3, acesse o monitor serial do Arduino IDE como indicado na Figura 23a. A partir dessa ferramenta é possível enviar comandos ao Arduino Uno R3 e visualizar o seu processamento. Quando a programação foi feita corretamente, a tela do monitor serial será semelhante ao que é ilustrado na Figura 23b, com a mensagem “READY TO GO!” sendo exibida.

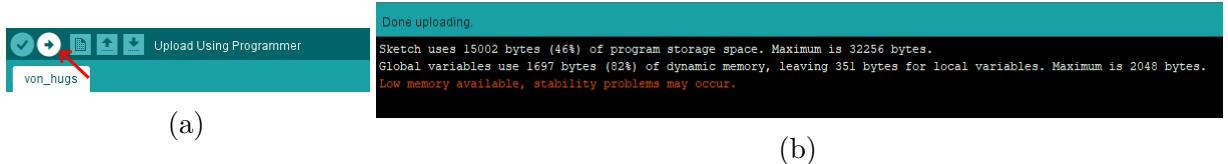


Figura 22 – Botão na barra de ferramentas do Arduino IDE para compilação do código-fonte e gravação do programa resultante (a) e mensagem de confirmação de compilação e gravação correta no Arduino Uno R3 (b).

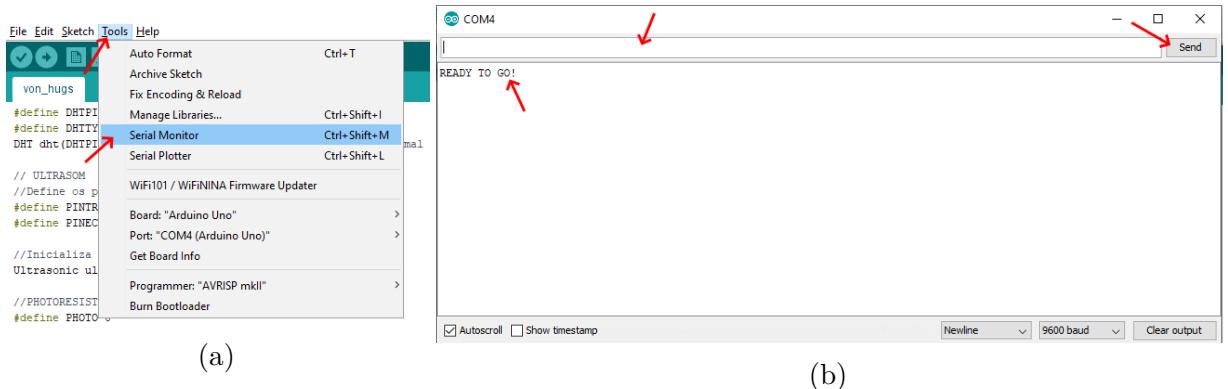


Figura 23 – Menu por onde se acessa o monitor serial no Arduino IDE (a) e captura da tela do monitor serial quando a programação do Arduino Uno R3 foi realizada corretamente (b).

Ainda, para verificar se está havendo a devida comunicação do Arduino Uno R3 com o *buzzer* conectado a ele, envie a sequência de comandos “BF0123;BT0002;HT0000;\$” sem as aspas utilizando o campo de textos e o botão indicados pelas setas vermelhas na Figura 23b. Esses comandos indicam para o Arduino Uno R3 tocar o *buzzer* com uma frequência de 123 Hz por dois segundos. Além do *beep* que será gerado, o Arduino Uno R3 também responderá via monitor serial aos comandos que lhe foram enviados para execução, como ilustrado na Figura 24.

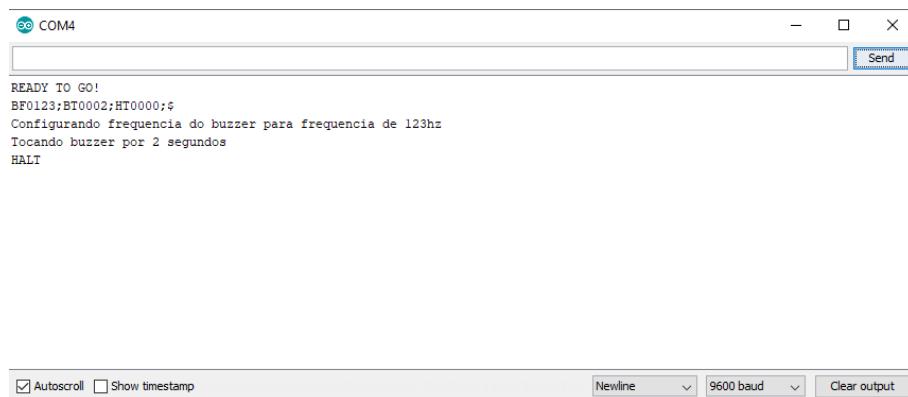


Figura 24 – Execução de comandos no Arduino, com retorno para a porta serial.

Até o momento, o Arduino Uno R3 está recebendo apenas comandos enviados via monitor serial. Porém, um dos módulos conectados a ele é um módulo Bluetooth,

permitindo que você envie comandos utilizando um *smartphone*. Para isso, é necessário que você encontre no início da função `void loop()` do arquivo “Downloads/automovel_pibit-master/von_hugs/von_hugs.ino” (entre as linhas 469 e 471) os comandos listados abaixo:

```
if(Serial.available()){
    char c = Serial.read();
    Serial.print(c);
    ...
    ...
    ...
}
```

Altere o código-fonte para que ele fique com os comandos listados abaixo, compile-o novamente e envie o programa novamente para o Arduino Uno R3 seguindo as instruções dadas no início dessa seção.

```
if(bluetooth.available()){
    char c = bluetooth.read();
    Serial.print(c);
    ...
    ...
    ...
}
```

2.6 Montando o veículo

Após montado o circuito e programado o Arduino, é chegada a hora de montar o veículo. Para o projeto, foi usado um kit de chassi com quatro motores de corrente contínua que operam com tensão entre três e seis volts e com rodas de 68 milímetros, ilustrado na Figura 25 já após o processo de montagem.

As peças que formam o kit estão ilustradas na Figura 26a, com destaque para os motores de corrente contínua já com os fios soldados na Figura 26b:

Encaixe por pressão os sensores de rotação – os discos pretos com furos no seu perímetro – ao eixo interior dos motores como ilustrado pelas setas na Figura 27a. Logo após, fixe ao chassi as oito peças plásticas que servem como suporte para os motores. De cada lado do motor, um suporte deve ser fixado. A Figura 27b, ilustra os motores já parafusados aos suportes. Para isso, devem ser usados maiores em conjunto com as porcas, ambos apresentados na Figura 26a.

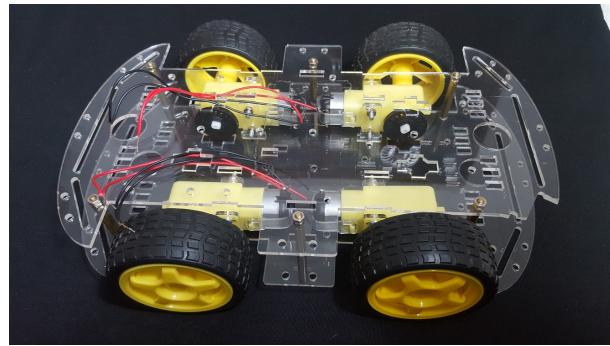


Figura 25 – Ilustração do chassis montado.

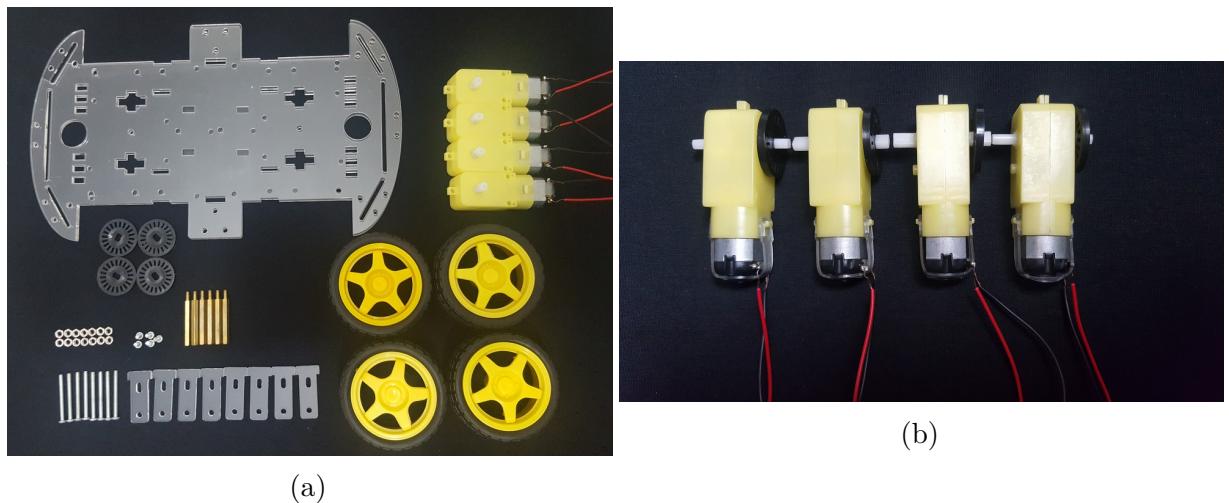


Figura 26 – Peças que formam o kit (a) e destaque para os quatro motores de corrente contínua já com os fios soldados (b).

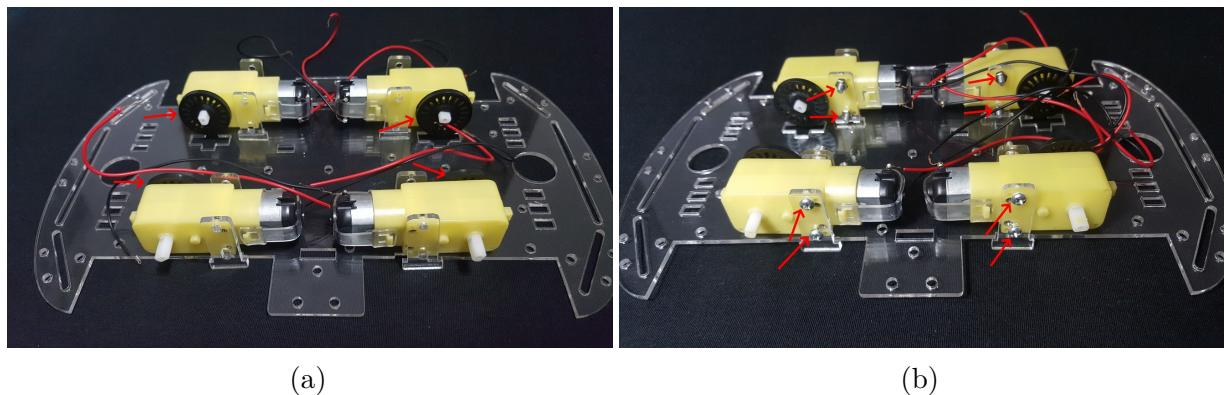


Figura 27 – Suportes encaixados ao chassis e motores com os sensores de rotação já instalados ao eixo de cada motor (a) e motores parafusados aos suportes (b).

Após isso, encaixe por pressão as quatro rodas do veículo ao eixo externo de cada motor, como mostrado na Figura 28a. Além disso, há seis parafusos menores e seis espaçadores. Parafuse cada um dos espaçadores ao chassis de forma que a rosca de cada um deles esteja apontada para cima, como mostrado pelas setas na Figura 28b.

Então, conecte (ou solde) os fios de mesma polaridade dos motores de cada lado,

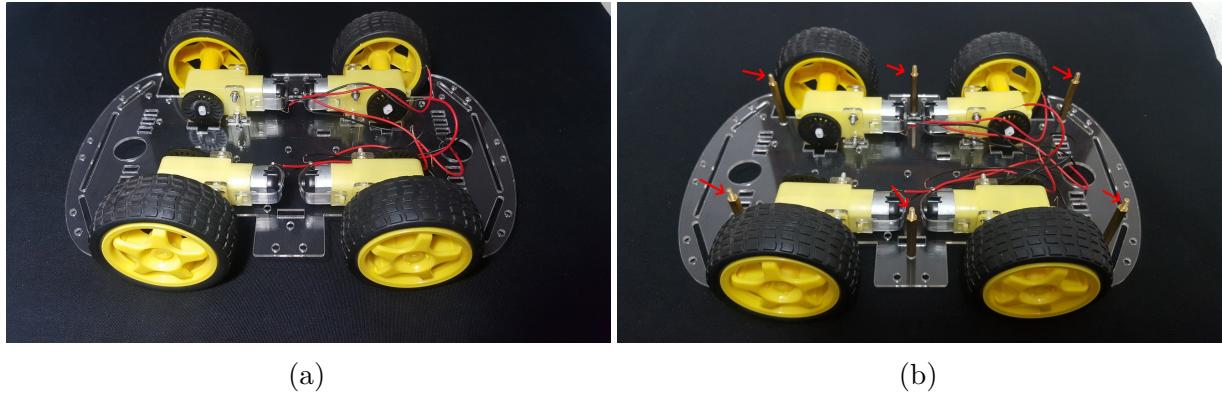


Figura 28 – Fotos que ilustram as rodas encaixadas aos eixos dos motores (a) e os espaçadores parafusados ao chassi (b).

lembra que do padrão de cores para + e -, como apresentado na Figura 28a. Feito isso, fixe a placa superior do chassi, encaixando-a aos espaçadores e parafusando-os utilizando as seis porcas restantes, como mostrado na Figura 28b. Lembre-se de passar os fios dos motores por dois orifícios em forma de cruz indicados nessa mesma figura.

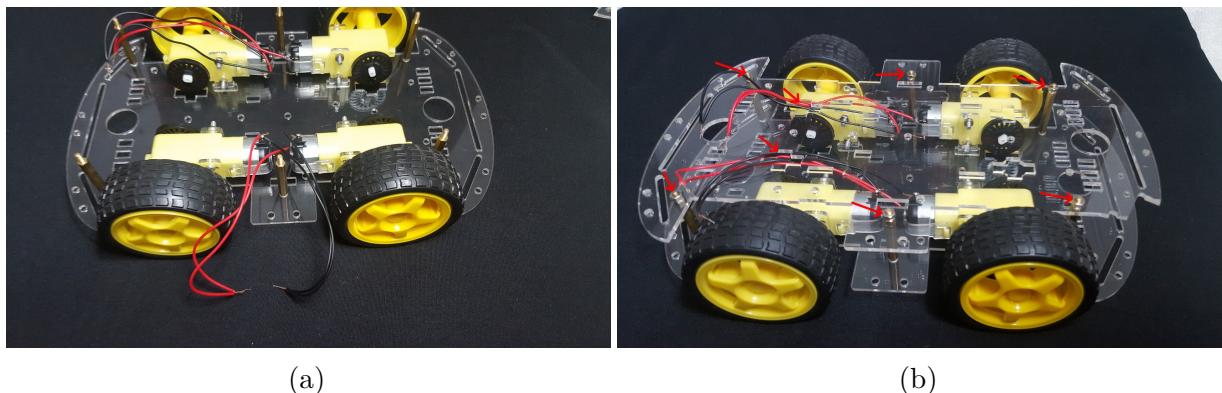


Figura 29 – Fotos que ilustram como devem ser conectados os fios dos motores (a) e como se finaliza a construção do chassi do veículo (b).

O chassi está construído. Agora pode-se ligar os motores à ponte H observando as instruções relativas à Figura 13b. Daí, cole uma fita dupla face ao lado inferior da ponte H, do Arduino Uno R3 e da *protoboard*. Então, cole-os ao chassi como mostrado na Figura 30.

Por fim, organize os sensores pelo chassi, lembrando que o sensor de distância deve ser posicionado para a frente do veículo, que é o lado em que está a ponte H.

2.7 Celular

Agora que o veículo está montado e programado, deve-se instalar no seu *smartphone* Android o aplicativo que servirá de controle remoto. Para isso, copie o arquivo no caminho “*Downloads/automovel_pibit-master/wally/app/build/outputs/apk/debug/app-debug.apk*” para o seu celular.

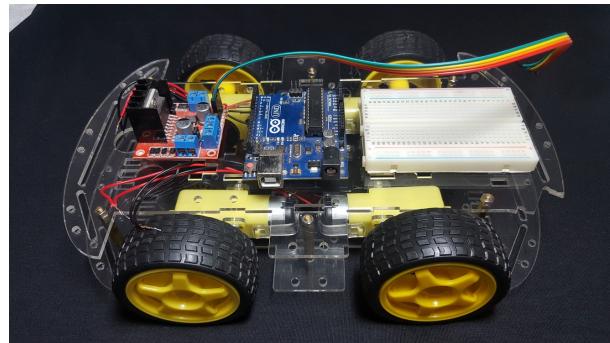


Figura 30 – Chassi com todos os componentes eletrônicos ligados.

Porém, muito provavelmente você deverá habilitar a opção de instalação de aplicativos de fontes desconhecidas. Essa opção permite ao usuário instalar aplicativos que não têm como origem a Play Store do Google. Celulares Android de fabricantes distintas podem possuir menus de configuração diferentes, porém, normalmente essa opção fica na seção de configurações de segurança do seu celular. Caso não a encontre, uma rápida pesquisa no Google pode te ajudar. A Figura 31 ilustra a seção de configuração de segurança de um celular Android.



Figura 31 – Captura da tela de configurações de segurança de um *smartphone* Android com a opção “Fontes desconhecidas” habilitada.

Após habilitada essa opção, abra o gerenciador de arquivos no seu *smartphone* e abra o arquivo recém-transmitido para o seu celular e instale o aplicativo. Porém, deve-se primeiramente realizar o pareamento do módulo Bluetooth do veículo com o *smartphone*. Para tanto, acesse as configurações de Bluetooth do seu celular e faça o procedimento

para pareá-lo com o dispositivo nomeado “HC-05”. Caso uma senha lhe seja solicitada, digite “1234”, sem as aspas. As Figuras 32a, 32b e 32c ilustram esse procedimento para um celular da Samsung.

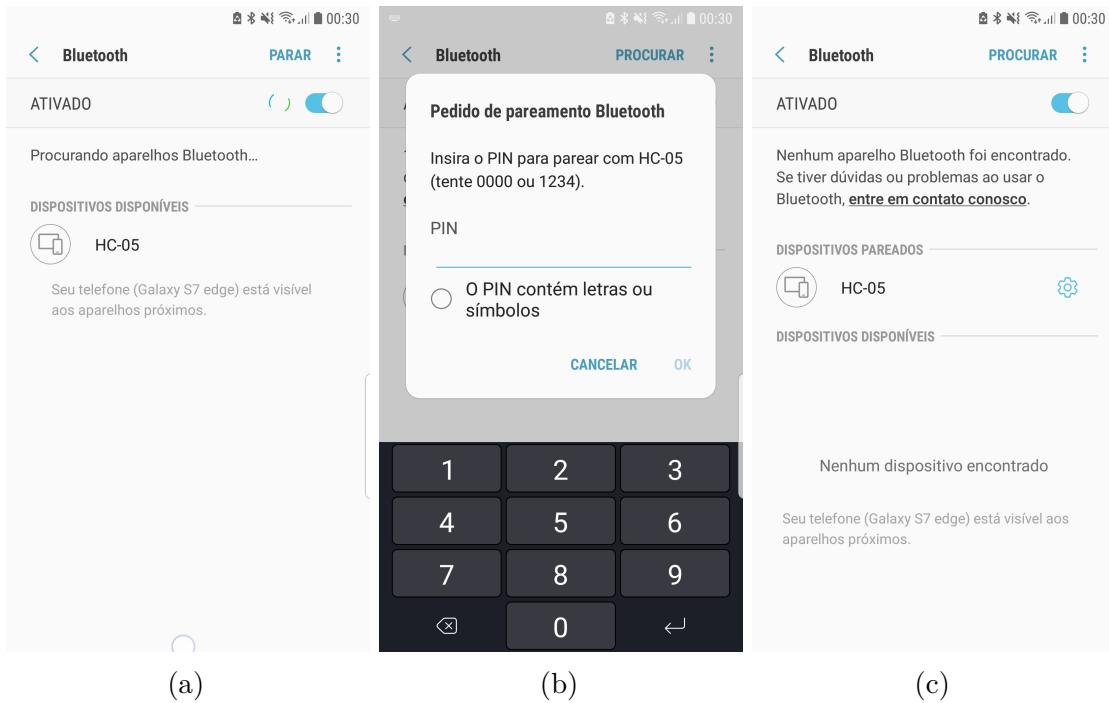


Figura 32 – Sequência de captura de telas ilustrando o módulo Bluetooth do veículo disponível para pareamento (a), o momento de pareamento com esse módulo (b) e o módulo pareado.

Agora o dispositivo já está pareado e pronto para uso. Agora, abra o aplicativo nomeado *Von Hugs* que constará na bandeja de aplicativos do seu celular. A primeira coisa a se fazer no aplicativo é selecionar qual dispositivo Bluetooth com o qual haverá a comunicação, mas não antes de conceder ao aplicativo a autorização de utilizar o Bluetooth no celular (Figura 33a). A Figura 33b ilustra a tela onde é realizada essa seleção. Nessa tela, usuário deve clicar no botão “Atualizar dispositivos pareados” e a listagem deverá constar o módulo “HC-05” para seleção.

A partir desse momento, o aplicativo se conectará ao módulo Bluetooth e estará pronto para enviar comandos para o veículo. A Figura 34 apresenta a sua tela principal, que permite ao usuário criar uma sequência de blocos dos comandos que serão executados pelo veículo.

Esses blocos de comandos são divididos em quatro categorias: “Ação”, “Lógica”, “Variáveis” e “Helpers”, ilustrados nas Figuras 35 e 36.

O aplicativo tem como objetivo facilitar a abstração dos conceitos envolvidos na programação de computadores. Por isso, o veículo é programado utilizando os blocos de comandos de maneira fácil.

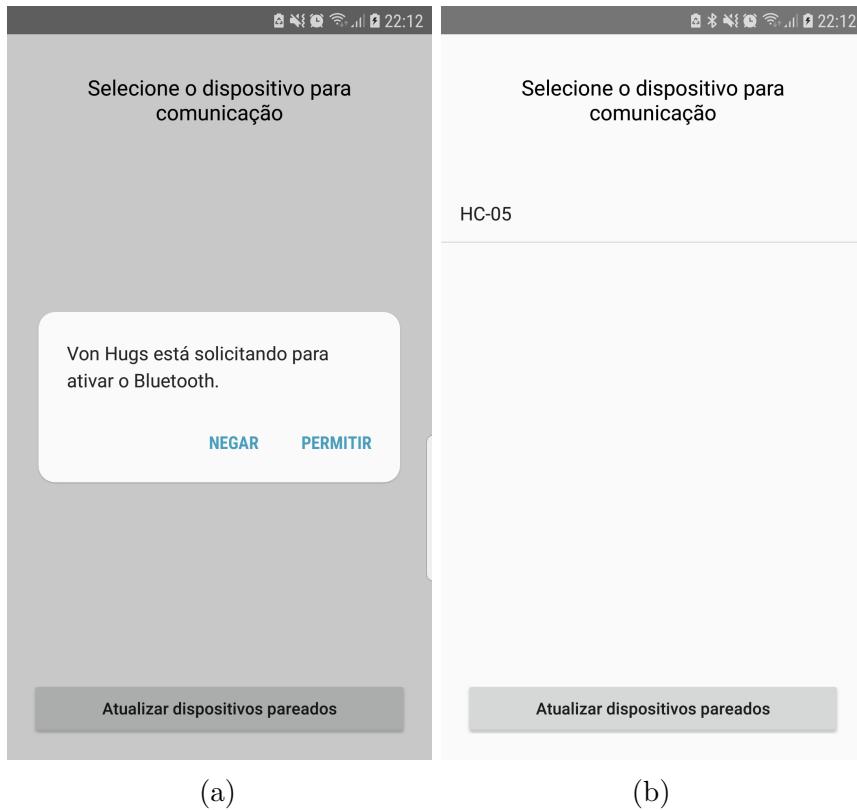


Figura 33 – Telas do aplicativo para seleção do dispositivo Bluetooth já pareado para comunicação (a) e (b).

A maneira de se operar o aplicativo é selecionando um dos blocos de comando nas categorias e arrastando-o para área em branco à direita na tela principal, como ilustrado na Figura 37.

Ainda, para se construir uma sequência de blocos de comandos – em outras palavras, um programa de computador – esses blocos podem ser encaixados uns aos outros, como mostrado na Figura 38. Como em peças de quebra-cabeças, os blocos têm uma maneira correta de se encaixar. Caso dois blocos não sejam compatíveis entre si, eles não se encaixarão. Isso é facilmente percebido pelo formato do encaixe de cada um dos blocos.

Ao tocar um bloco por alguns segundos faz aparecer opções referentes a ele, como duplicar e apagar os blocos (Figura 39a). Outra maneira para se apagar um bloco é selecionando-o e arrastando-o para a barra lateral de categorias (Figura 39b).

Na parte inferior da tela há três botões. O primeiro deles – “Atualizar” – requisita instantaneamente ao veículo os valores lidos pelos sensores de distância, temperatura, umidade e luminosidade, mostrando-os na parte superior da tela. A Figura 40a mostra um exemplo dos valores lidos pelos sensores do veículo.

O botão “Executar” envia para o Arduino Uno R3 a sequência de instruções que o veículo deve executar com base na sequência dos blocos presentes na tela principal. Assim que o botão é pressionado uma janela é aberta mostrando o código que foi enviado ao

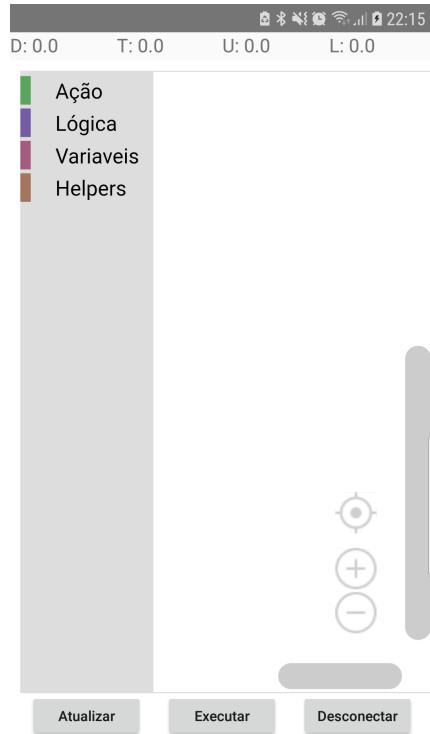


Figura 34 – Tela principal do aplicativo.

veículo, como apresentado na Figura 40b.

Por fim, o botão “Desconectar” permite a desconexão do aplicativo com o módulo Bluetooth do veículo. Dessa forma, o aplicativo volta à tela inicial de seleção de dispositivo Bluetooth para comunicação.

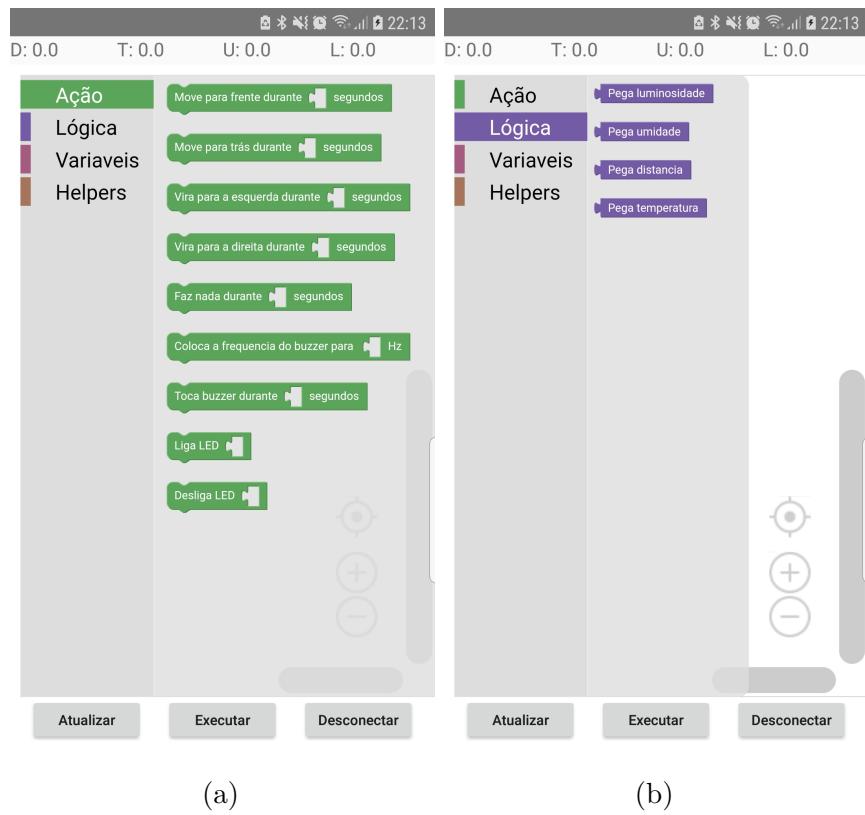


Figura 35 – Menus “Ação” (a) e “Lógica” (b) do aplicativo.

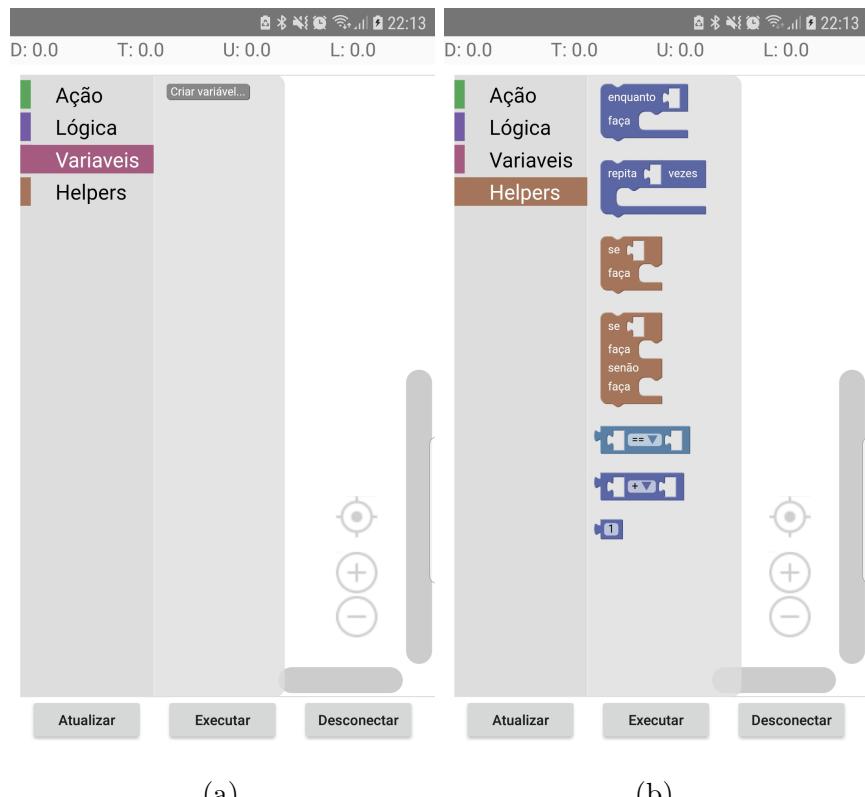


Figura 36 – Menus “Variáveis” (a) e “Helpers” (b) do aplicativo.

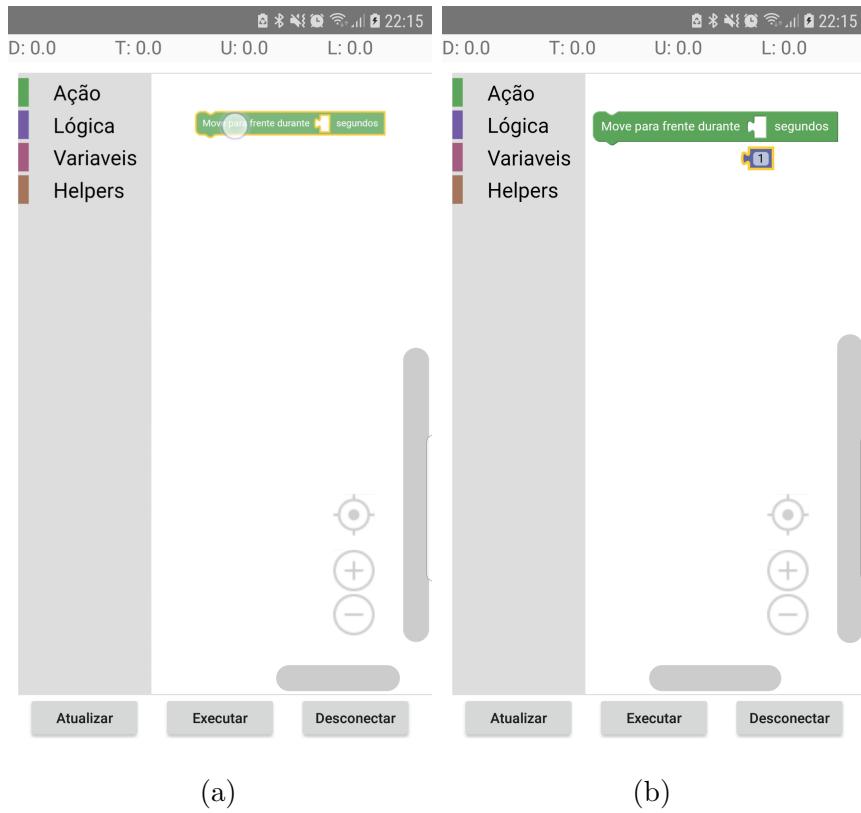


Figura 37 – Inserção do bloco para mover o veículo para frente (a) e do bloco com um valor numérico (b).

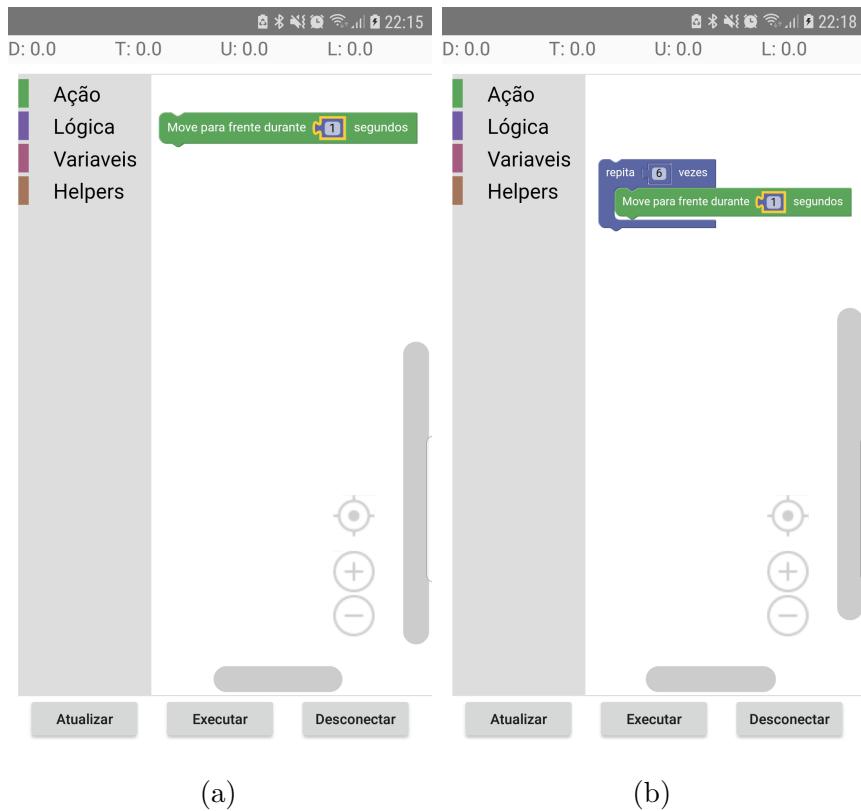


Figura 38 – Encaixe do bloco com um valor numéricico dentro do bloco para mover o veículo para frente (a) e inserção de um bloco de repetição já com o encaixe de um bloco com valor numérico e do bloco para movimentar o veículo (b).

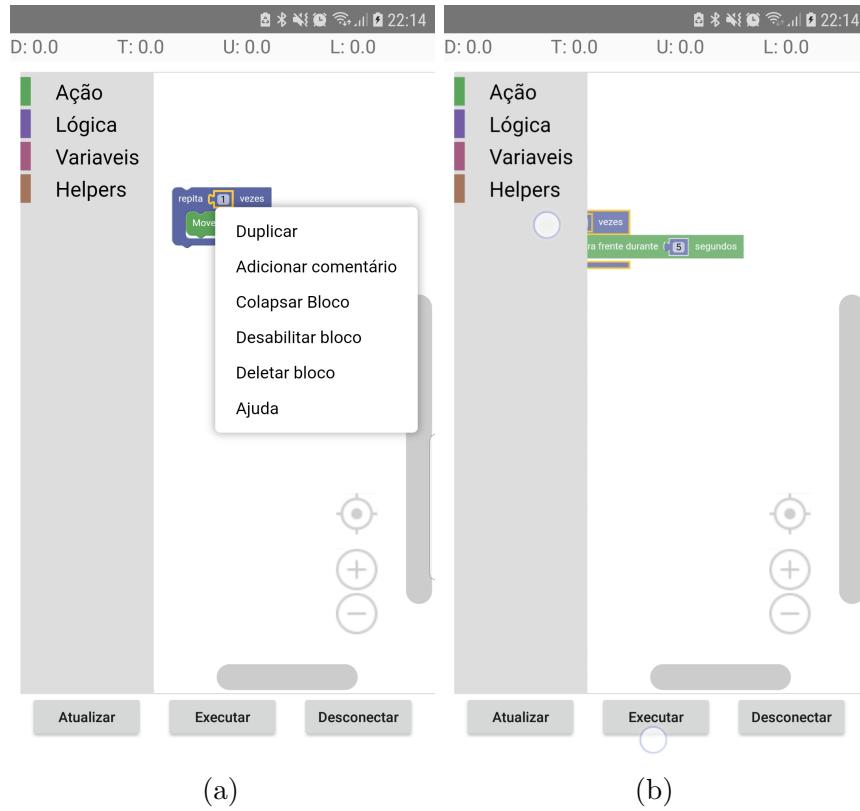


Figura 39 – Opções referentes ao bloco (a) e procedimento de exclusão de um bloco (b).

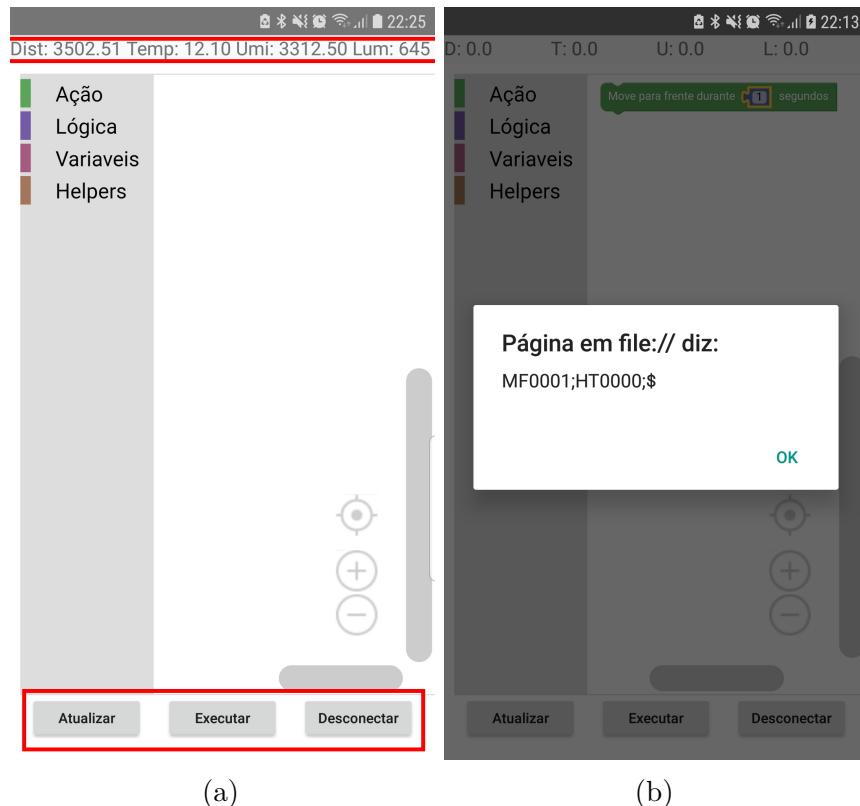


Figura 40 – Captura da tela principal do aplicativo com destaque para os valores dos sensores do veículo (a) e exemplo da sequência de comandos construída a partir dos blocos e que será executada pelo veículo (b).

3 Desenvolvimento do Aplicativo

O código-fonte de todos os programas desenvolvidos neste projeto foi disponibilizado no GitHub cuja URL foi apresentada no início da seção 2.4. O propósito disso é permitir ao aluno de computação poder estudar e realizar alterações ao funcionamento do veículo e/ou do aplicativo para *smartphones* Android. As próximas seções o instruem sobre os passos necessários para isso.

3.1 Modo de desenvolvedor

Primeiramente, é necessário habilitar no seu celular Android o “Modo de desenvolvedor”. Essa funcionalidade permite ao proprietário criar um aplicativo em seu computador e testá-lo no seu *smartphone*. Cada fabricante de *smartphone* pode adotar uma estratégia diferente para habilitar o modo de desenvolvedor. Por isso, pode ser necessária uma busca no Google para verificar como fazer isso no seu modelo de celular.

3.2 Programando o aplicativo

Para realizar alterações no aplicativo para *smartphones* Android, é necessário instalar o Android Studio, que pode ser baixado acessando a URL <<https://developer.android.com/studio>>. O código-fonte do aplicativo já está no repositório que foi baixado.

Depois de realizada a instalação do Android Studio, abra-o e logo aparecerá uma tela como a ilustrada na Figura 41. Nessa tela, selecione a opção “*Open an existing Android Studio Project*”, onde aparecerá uma outra tela para seleção do projeto, que estará no caminho `Downloads/automovel_pibit-master/wally`, como ilustrado na Figura 42.

O Android Studio carregará o projeto e procurar por problemas no código-fonte ou no projeto em geral (Figura 43). Esse processo pode demorar alguns minutos, não sendo possível fazer nada no Android Studio enquanto este processo não for concluído.

Após terminado o processo de carregamento do projeto, abra a tela “Estrutura do Projeto” usando o atalho de teclado **CTRL + ALT + SHIFT + S**. Será aberta uma nova tela e, nela, selecione a aba lateral “*Project*” e altere os campos “*Android Gradle Plugin Version*” para “3.0.1” e “*Gradle Version*” para “4.1”, como mostrado na Figura 44.

Além disso, navegue para a aba “*Modules*” e altere os campos “*Compile Sdk Version*” e “*Build Tools Version*” para “26 (API 26: Android 8.0 (Oreo))” e “26.0.3”, respectivamente, como apresentado na Figura 45.

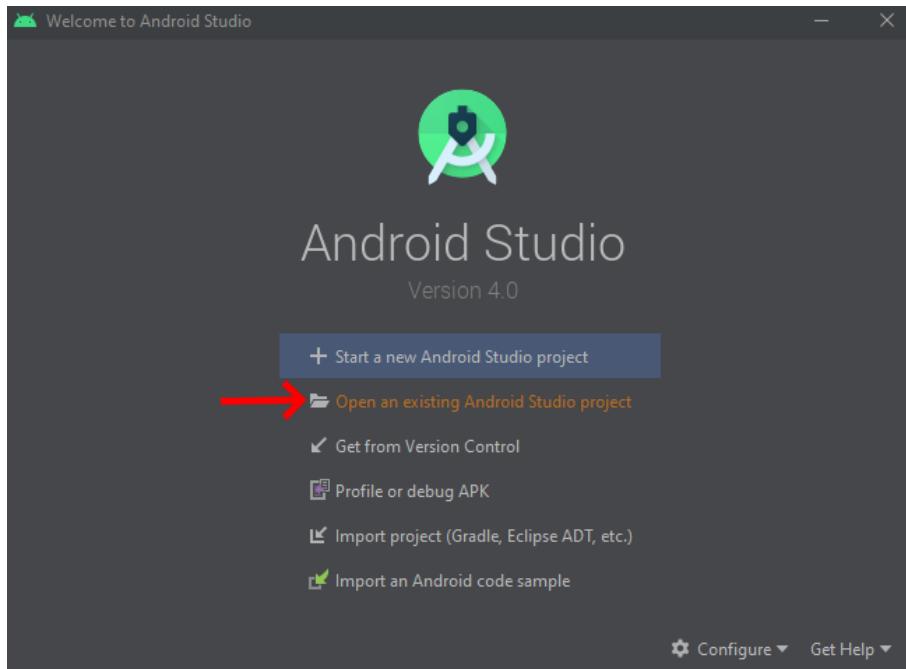


Figura 41 – Tela de boas-vindas do Android Studio.

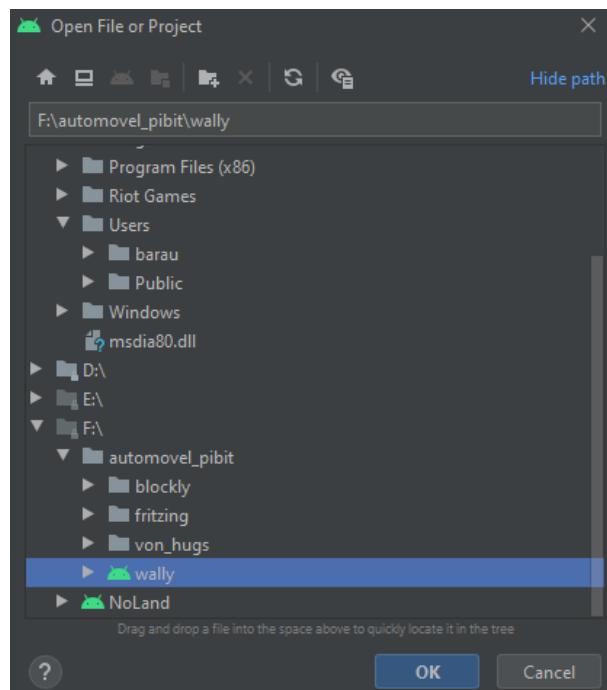


Figura 42 – Tela de seleção de projeto Android Studio já existente.

Depois disso, conecte o seu celular Android ao computador em que está instalado o Android Studio usando o cabo USB original do fabricante do seu celular ou um cabo USB de dados de qualidade. Caso tenha seguido corretamente os passos para ativar o modo desenvolvedor, o *smartphone* será reconhecido automaticamente pelo Android Studio e será mostrado na barra de ferramentas do programa, como mostrado na Figura 46.

Sendo esse o caso, é hora de compilar o aplicativo e executá-lo no celular. Para isso

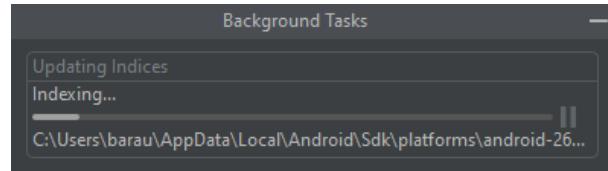


Figura 43 – Tarefa de plano de fundo do Android Studio.

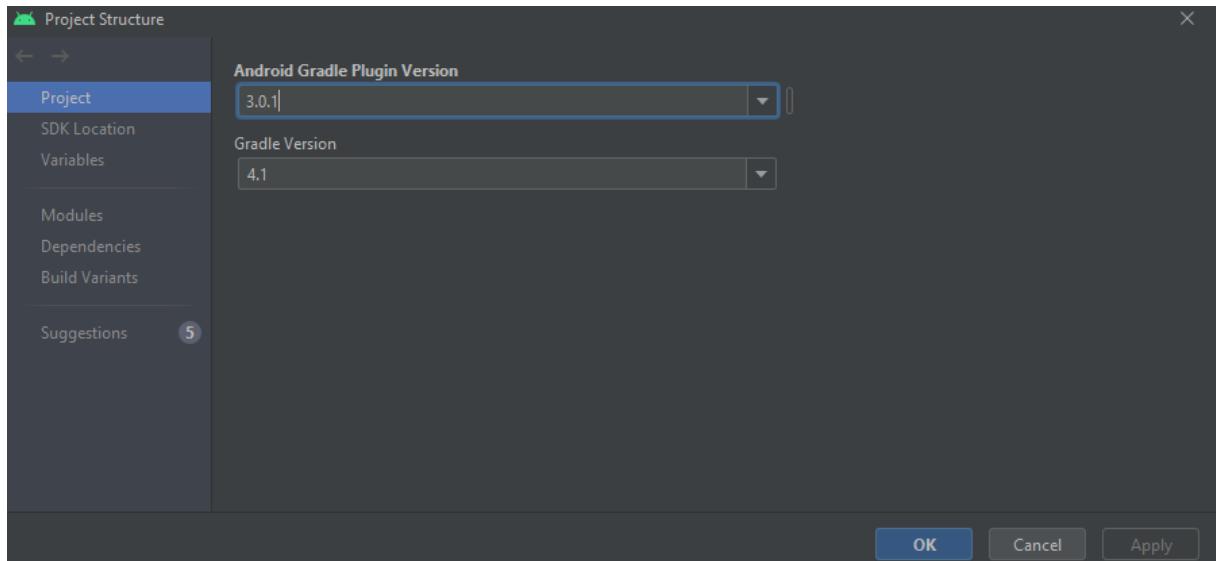


Figura 44 – Tela de versões do Gradle do projeto.

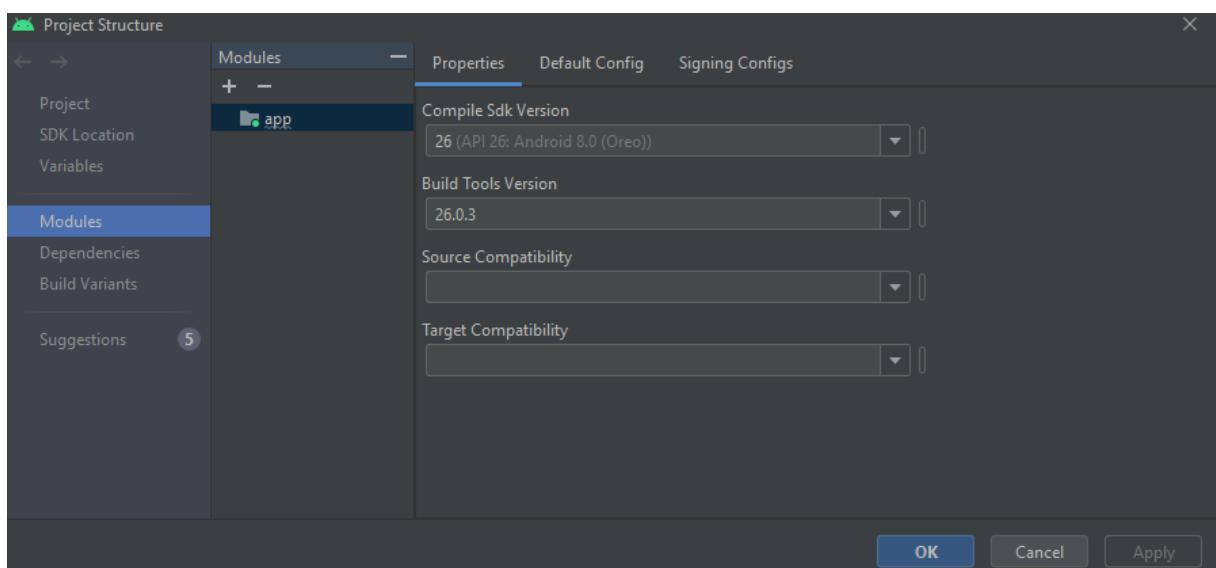


Figura 45 – Tela de versões do SDK do projeto.

clique no botão “*Play*” apresentado na barra de ferramentas do Android Studio logo à direita da caixa de seleção onde consta o nome do seu celular. O aplicativo é compilado e, caso não tenha nenhum erro, é automaticamente instalado no *smartphone* e aberto. Se, por algum motivo, não esteja habilitada a opção de instalação de aplicativos de fontes desconhecidas, você deverá fazer isso, como explicado na seção 2.7.

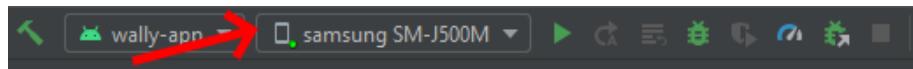


Figura 46 – Trecho da barra de ferramentas do Android Studio com destaque para o *smartphone* devidamente reconhecido pelo programa.

A partir daí, você será capaz de realizar as alterações desejadas no aplicativo Android. Não se esqueça de atualizar o código-fonte do programa do Arduino utilizando o Arduino IDE para editar o arquivo no caminho `Downloads/automovel_pibit-master/von_hugs/von_hugs.ino`.

Boa programação!