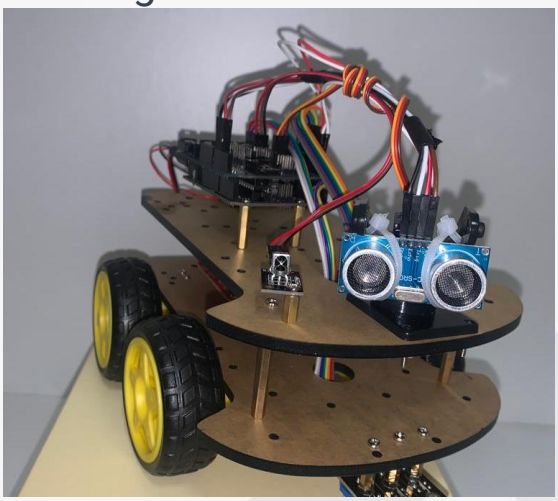
UNIVERSIDADE
LUSÓFONA

SeguidorDeLinha-EV3



Ana Cristina Luísa - a22205855

Betânia Sila – a22102238

Felipe Silva – a22103965

Wisley Costa – a22103352

Arquitetura Avançada de Computadores | Licenciatura em Engenharia Informática | 16/01/2023

www.ulusofona.pt

Índice

Introdução

Este trabalho teve por base os conhecimentos adquiridos na disciplina de Arquiteturas Avançadas de Computadores no referente a Arduino Uno R3 compreendendo a parte de hardware e software. O Arduíno é uma ferramenta de prototipagem rápida utilizando uma plataforma microcontroladora que tem boas capacidades de processamento e facilidade de integração no projeto a desenvolver.

O Arduino permite a ligação de sensores necessários a criar automatismos em toda a gestão. Este, em termos de software, permite ao utilizador o usufruto de uma aplicação móvel que permite interagir com o sistema. O microcontrolador do Arduino foi programado com a linguagem C + +, de modo a ter lógica e capacidade de tomar decisões.

A figura (Figura 1) seguinte é demonstrativa de uma placa de Arduino que foi utilizada no projeto.

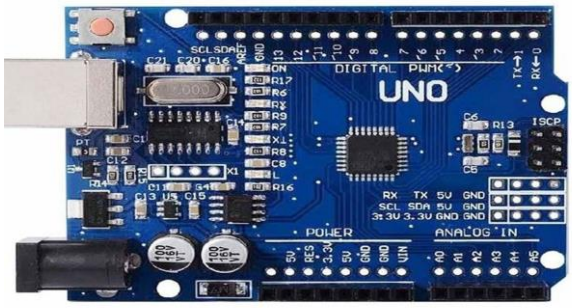


Figura 1 – Placa de Arduino Uno R3

Um carro Arduino é um veículo robótico que pode ser controlado usando um microcontrolador Arduino e vários sensores e atuadores. Este projeto envolve o uso de programação e eletrônica para criar um carro que se pode mover, evitar obstáculos e seguir uma linha. O código usado neste projeto é um exemplo do que se pode realizar, usando as bibliotecas IRremote e Servo, bem como vários sensores e atuadores para controlar o movimento e o comportamento do carro.

O sistema vai, através de um conjunto de sensores descritos em fases posteriores no projeto, monitorizar várias variáveis como a emissão de alertas. Além deste primeiro objetivo, foi ainda acrescentado ao projeto um controle remoto que se comunica sem fios com o Arduino e deixa o utilizador parar, colocar a andar e mover o mesmo.

Material

1. Abordagem a realizada

Utilizou-se o IDE do Arduino desenvolvido na linguagem de programação C + +, para ser usado de forma a facilitar a programação e construção do carro utilizando sensores, leds, display e controle remoto.

2. Especificação técnica

No presente trabalho usaram-se alguns componentes para a implementação, ou seja, para a construção do carro como são as descritas seguidamente.

1. 4 pcs Gear Motor :

   Um motor de engrenagem é um dispositivo que permite a baixa potência dos motores a impulsionar uma grande quantidade de força em um objeto que opere com baixa velocidade.

2. 2 pcs High Quality Type

   Para conexão via USB.

3. 4 pcs Motor Fixed Pieces

   Suporte de fixação dos motores.

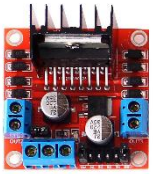
4. 1 pcs Universal Wheel:

A roda é uma das seis máquinas simples com vastas aplicações no transporte e em máquinas mecânicas, caracterizada pelo movimento rotativo no seu interior.

5. 2 pcs Robot Chassis:

Os chassis são essenciais para criação de agentes robóticos em formato de carros, braços, tanque, entre outros…

6. 1 pcs L298N Motor Driver Board:

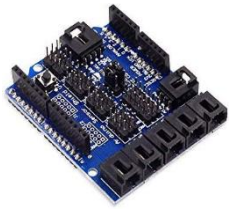


É uma placa de controle de motor que é normalmente usada para controlar a velocidade e a direção dos motores.

7. 1 pcs UNO r3:

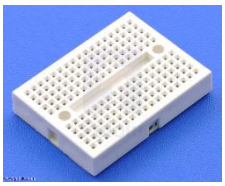
Arduino UNO R3 é uma placa de desenvolvimento baseada no microcontrolador ATmega328 (datasheet).

8. 1 pcs Sensor Extension Plate V4:



A Placa de Expansão para Arduino ou Arduino Sensor Shield V4.0 permite conectar vários módulos como sensores, servos, relés, botões, potenciómetros apenas através do encaixe deste shield na sua placa microcontroladora.

9. 1 pcs 170 Point Breadboard



É uma breadboard sem solda (plug-in) com 170 pontos de ligação

10. 1 pcs SG90 Servo:

É ideal para a utilização em aeromodelismo, controlando de forma precisa os movimentos.

11. 1 pcs Ultrasonic Sensor Module

Um sensor ultrassónico é um dispositivo eletrônico que mede a distância de um objeto alvo emitindo ondas sonoras ultra sônicas e converte o som refletido em um sinal elétrico.

12. 3 pcs TCRT5000 Tracing Module

O Módulo de Sensor Infravermelho (TCRT5000) é um Módulo de Rastreamento de Linha. É um módulo infravermelho de distância mais usado para detecção de objetos e obstáculos e aplicações de robôs como um carro robô seguidor de linha, decodificador de linha de produtos e muito mais.

13. 1 pcs Infrared Receiving Sensor Module

Módulo sensor de recepção infravermelho para bloquear outras faixas de luz que não seja a do próprio emissor, evitando que as iluminações do ambiente venham causar alguma interferência.

14. 1 pcs Remote

É um aparelho utilizado para realizar uma operação remota a um dispositivo eletrônico (Carro).

15. 1 pcs Six 5th Battery Box

Um suporte de bateria é um ou mais compartimentos ou câmaras para armazenar uma bateria.

16. 1 pcs Two 5th Battery Box

Caixa para colocar as pilhas.

17. 1 pcs 40 pin Dupont Line

Pode ser dividido para fazer diferentes números de fios, consoante as necessidades. Estes fios de jumper podem ser usados com projetos eletrônicos, em uma prancha sem solda e direto para cabeçalhos de placa de circuito.

3. Funcionalidades

3.1 - Evitação de obstáculos: O robô é capaz de detectar e evitar obstáculos a uma distância específica usando um sensor ultrassônico.

3.2 - Controle remoto: O robô é capaz de ser controlado remotamente através de um controlador de infravermelhos, com comandos para mover-se para frente, para trás, para a esquerda e para a direita.

3.3 - Modos de operação: O robô teria diferentes modos de operação, como seguimento de linha, evitação de obstáculos e modo de movimento normal, que pode ser alternado através do controlador de infravermelhos.

3.4 - Servo motor: O robô possui um servo motor que é usado para girar o sensor ultrassônico em diferentes direções, para medir a distância de obstáculos em diferentes áreas.

3.5 - Controlo de velocidade: O robô teria a capacidade de ajustar a velocidade dos motores para evitar colisões frequentes.

4. Desenvolvimento

O desenvolvimento de um projeto de carro Arduino requer conhecimento de programação, eletrônica e robótica. O código que aqui apresentado/utilizado usa a linguagem de programação do Arduino (C + +), que é semelhante ao C + +, mas com algumas modificações. O código usa várias bibliotecas, como IRremote e Servo, que fornecem funções para controlar o movimento e o comportamento do carro. O código também usa várias constantes para definir os números dos pinos para os diferentes sensores e atuadores, bem como constantes para os diferentes comandos de movimento do carro.

O código tem várias funções que controlam o movimento e o comportamento do carro. A função "loop ()" chama essas funções continuamente num ciclo, permitindo que o carro responda a diferentes entradas de um controle remoto IR, além de evitar obstáculos e rastrear uma linha.

5. Código utilizado no Arduino originalmente:

```
#include <IRremote.h>
/* https://github.com/z3t0/Arduino-IRremote */
#include <Servo.h>
/* Define the pin */
#define Echo A0                  //ECHO pin of obstacle avoidance module
#define Trig A1                  //Trig pin of obstacle avoidance module
#define ENB 5                    //The pin of motor drive
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 10
#define ENA 6
#define RECV_PIN 13              //Infrared receiver pins
#define LineTeacking_Pin_Right  2        //The pins of the trace module
#define LineTeacking_Pin_Middle 1
#define LineTeacking_Pin_Left   4
#define SERVO 12                 //Servo motor pins

#define FORWARD 2
#define BACK 8
#define LEFT 4
#define RIGHT 6
#define STOP 5
#define MOVE_MODE 10
#define OBSTACLES_MODE 11
#define LINETRACKING_MODE 12
/* Infrared remote control coding  ,Each remote control is different, you need to test the code first */
#define KEY_2 16718055
#define KEY_4 16716015
```

```
#define KEY_6 16734885
#define KEY_8 16730805
#define KEY_5 16726215
#define KEY_CH1 16753245
#define KEY_CH2 16736925
#define KEY_CH3 16769565
/* Read the pins of the trace module */
#define LineTeacking_Read_Right   digitalRead(LineTeacking_Pin_Right)
#define LineTeacking_Read_Middle  digitalRead(LineTeacking_Pin_Middle)
#define LineTeacking_Read_Left    digitalRead(LineTeacking_Pin_Left)
int carSpeed = 250;
IRrecv irrecv(RECV_PIN);
decode_results results;
Servo myservo;
int rightDistance = 0, leftDistance = 0, middleDistance = 0;
int Mode = MOVE_MODE;
int Direction = 5;
unsigned long recvTime;
unsigned long lineTime;
void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn();
  pinMode(Echo, INPUT);
  pinMode(Trig, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  digitalWrite(ENA, HIGH);
```

```
  digitalWrite(ENB, HIGH);
  pinMode(LineTeacking_Pin_Right, INPUT);
  pinMode(LineTeacking_Pin_Middle, INPUT);
  pinMode(LineTeacking_Pin_Left, INPUT);
  myservo.attach(SERVO);
  myservo.write(90);
}

void loop() {
  recvData();
  moveMode();
  obstaclesMode();
  linetrackingMode();
}
void moveMode() {
  if (Mode == MOVE_MODE) {
    carSpeed = 250;
    switch (Direction) {
      case FORWARD: forward(); break;
      case BACK: back(); break;
      case LEFT: left(); break;
      case RIGHT: right(); break;
      case STOP: stop(); break;
      default: break;
    }
    if (millis() - recvTime >= 500) {
      Direction = STOP;
      recvTime = millis();
    }
  }
}
```

```
/* Adjust "carSpeed" if it hits the wall frequently  */
void obstaclesMode() {
  if (Mode == OBSTACLES_MODE) {

    carSpeed = 200;
    myservo.write(90);
    delay(500);
    middleDistance = getDistance();
    if (middleDistance <= 40) {
      stop();
      delay(500);
      myservo.write(10);
      delay(1000);
      rightDistance = getDistance();

      delay(500);
      myservo.write(90);
      delay(1000);
      myservo.write(180);
      delay(1000);
      leftDistance = getDistance();

      delay(500);
      myservo.write(90);
      delay(1000);
      if (rightDistance > leftDistance) {
        right();
        delay(360);
      }
      else if (rightDistance < leftDistance) {
        left();
```

```
      delay(360);
    }
    else if ((rightDistance <= 40) || (leftDistance <= 40)) {
      back();
      delay(180);
    }
    else {
      forward();
    }
   }
   else {
    forward();
   }
  }
}
int getDistance() {
  digitalWrite(Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);
  return (int)pulseIn(Echo, HIGH) / 58;
}
/*The trace module needs to test whether the low level trigger or the high level trigger.
  It can be adjusted according to the actual situation*/
void linetrackingMode() {
  if (Mode == LINETRACKING_MODE) {
    carSpeed = 150;
    if (LineTeacking_Pin_Middle) {
      forward();
      //Serial.println("forward");
```

```
    while (LineTeacking_Pin_Middle);
   }
   if (LineTeacking_Pin_Left) {
    right();
    while (LineTeacking_Pin_Left);
   }
   else if (LineTeacking_Pin_Right) {
    left();
    while (LineTeacking_Pin_Right);
   }
   else if (LineTeacking_Pin_Left && LineTeacking_Pin_Middle) {
    right();
    while (LineTeacking_Pin_Left);
   }
   else if (LineTeacking_Pin_Right && LineTeacking_Pin_Middle) {
    left();
    while (LineTeacking_Pin_Left);
   }
   else {
    forward();
   }
  }
}
void recvData() {
  if (irrecv.decode(&results)) {
   recvTime = millis();
   switch (results.value) {
    case KEY_2:   Direction = FORWARD;  break;
    case KEY_4:   Direction = LEFT;     break;
    case KEY_6:   Direction = RIGHT;    break;
    case KEY_8:   Direction = BACK;    break;
```

```
    case KEY_5:   Direction = STOP;     break;
    case KEY_CH1:  Mode = MOVE_MODE; stop(); Serial.println("MOVE_MODE");
      delay(1000); break;
    case KEY_CH2:  Mode = OBSTACLES_MODE; stop(); Serial.println("OBSTACLES_MODE");
      delay(1000); break;
    case KEY_CH3:  Mode = LINETRACKING_MODE; stop();
Serial.println("LINETRACKING_MODE");
      delay(1000); break;
    default: break;
   }
   irrecv.resume();
  }
}
/*  The best thing to do is to input the HIGH and LOW of each pin once,
   look at the direction of each wheel of execution, and then define this function  */
void forward() {

  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("Forward");  //send message to serial monitor
}

void back() {



  digitalWrite(IN1, LOW);    //set IN1 hight level
  digitalWrite(IN2, HIGH);     //set IN2 low level
  digitalWrite(IN3, HIGH);     //set IN3 low level
  digitalWrite(IN4, LOW);    //set IN4 hight level
```

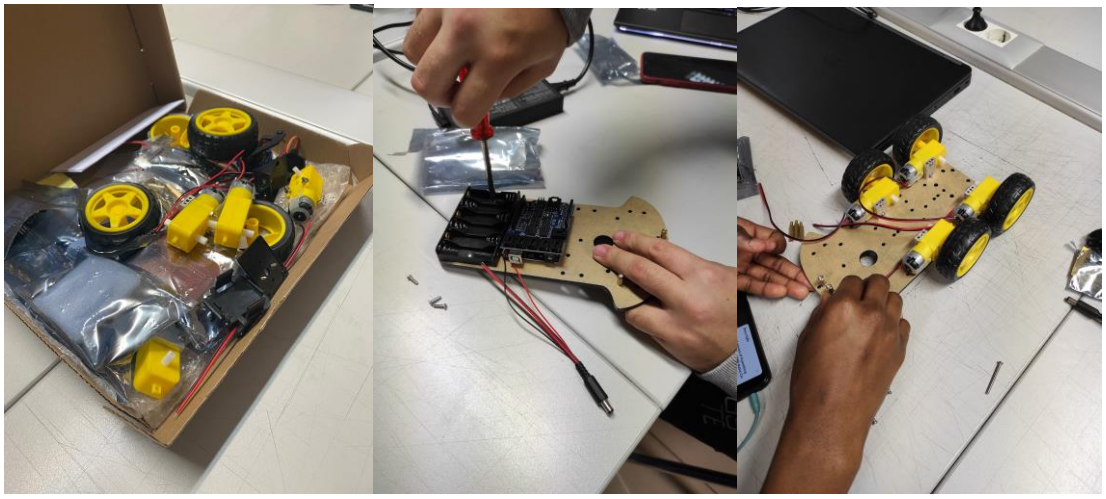
```
  Serial.println("Back");
}

void stop() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
  //Serial.println("stop");
}

void right() {

  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("right");
}

void left() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);

  Serial.println("left");
}
```

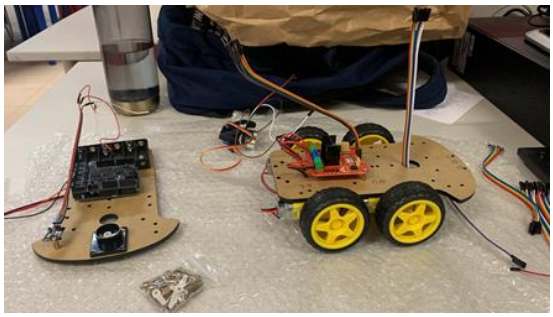
6. Conclusão

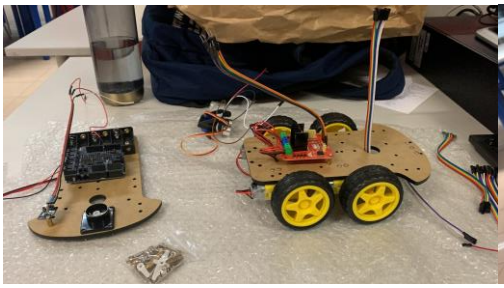
Em síntese, construir um projeto de carro Arduino pode ser uma tarefa desafiadora, exigindo conhecimentos de programação, eletrônica e robótica. No entanto, com as habilidades certas, foi uma experiência muito gratificante e divertida. Portanto, é importante rever cuidadosamente o código e garantir que todas as funções necessárias estejam definidas antes de tentar carregá-lo numa placa Arduino. Além disso, depurar e testar o código foi uma parte importante do processo de desenvolvimento, bem como ajustar os parâmetros como a velocidade do carro, as leituras do sensor e a posição do servo para ajustar o movimento e o comportamento do carro. No geral, construir um carro Arduino pode ser um projeto desafiador, mas agradável, permitindo sejam apreendidas novas habilidades e que se crie um carro totalmente autônomo.

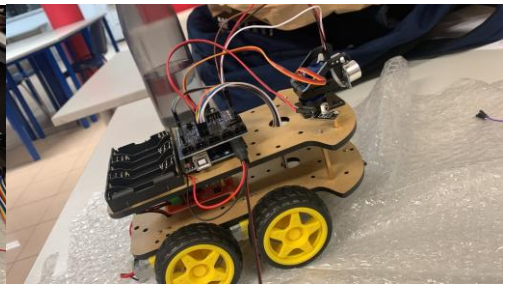
7. Montagem do carro

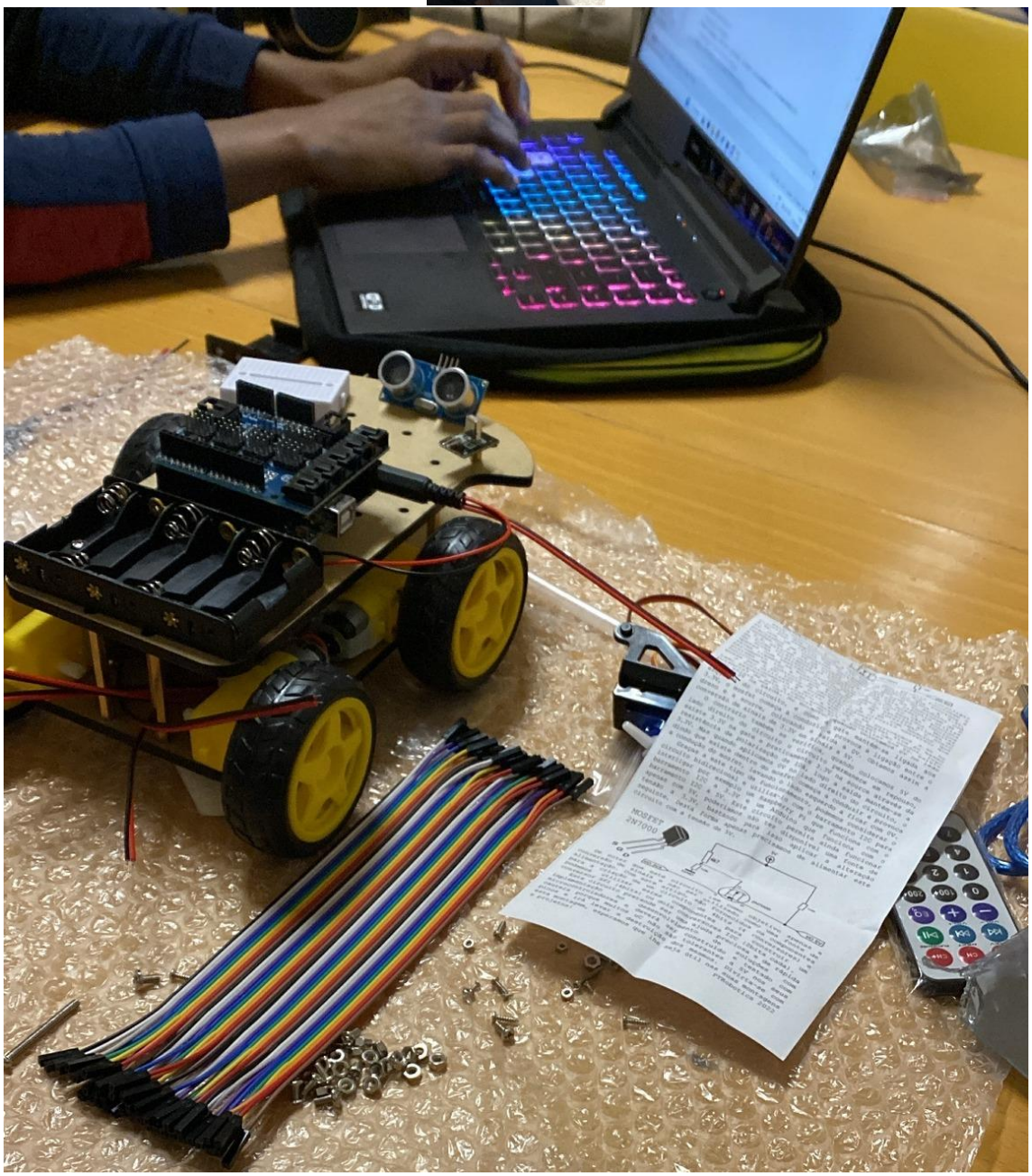
Seguidamente encontram-se algumas fotos da montagem do carro com Arduino.











8. Bibliografia

- Apontamentos das aulas de Arquiteturas Avançadas de Computadores;
- Vídeos sobre montagem de arduinos, consultas online;
-