BOBBY

-Autonomous robotic vehicle with avoiding obstacle system using ultrasounds-

Badau Elena Nicoleta
Chirila Radu-Ilie
Duma Dragos

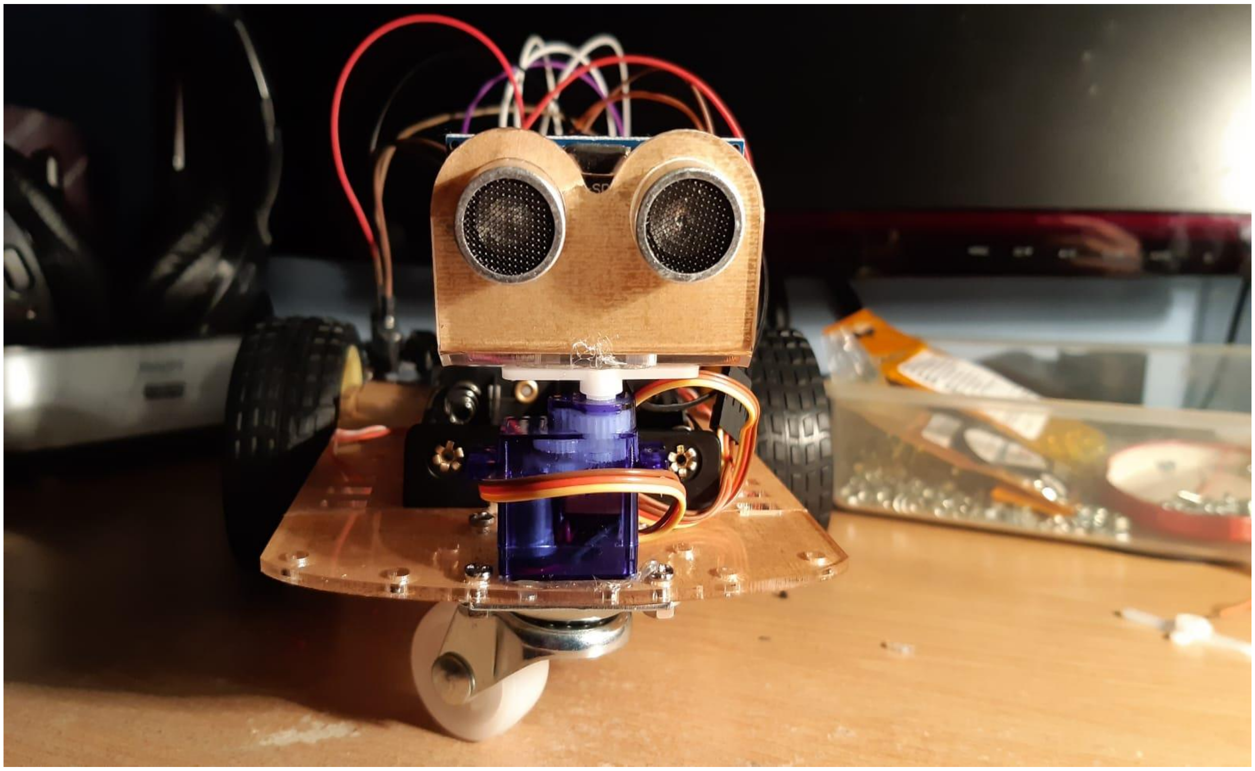


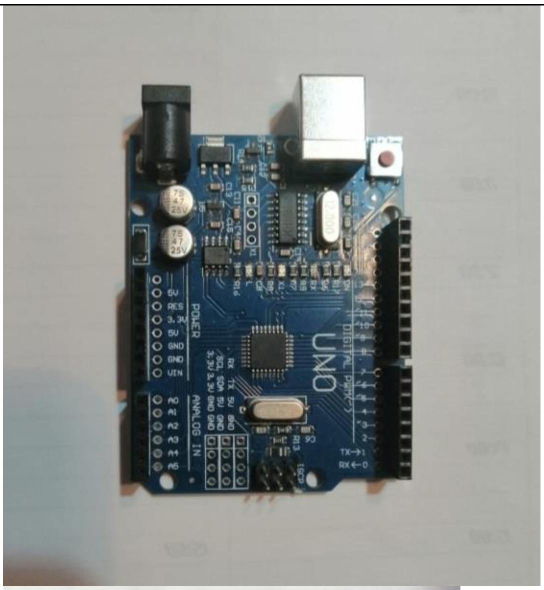
Table of Contents

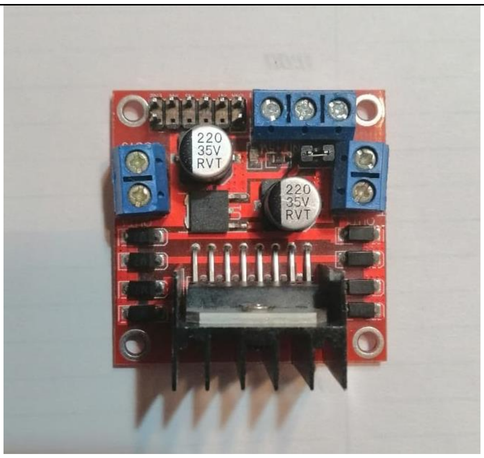
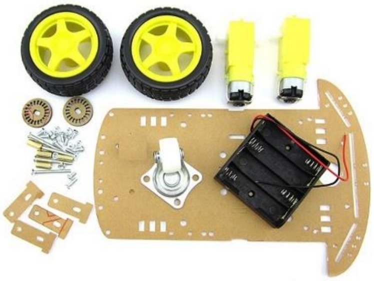
THEME PRESENTATION

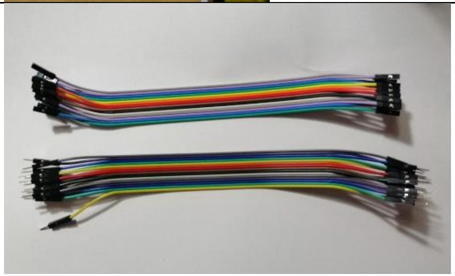
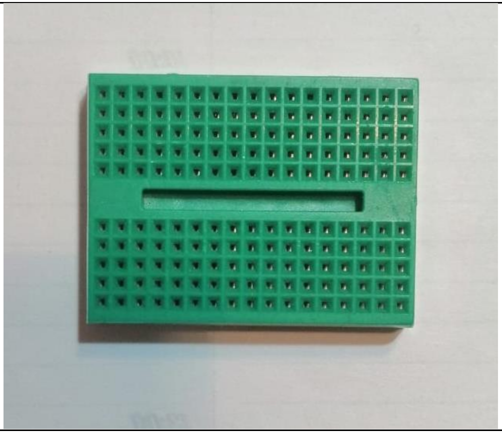
Project description

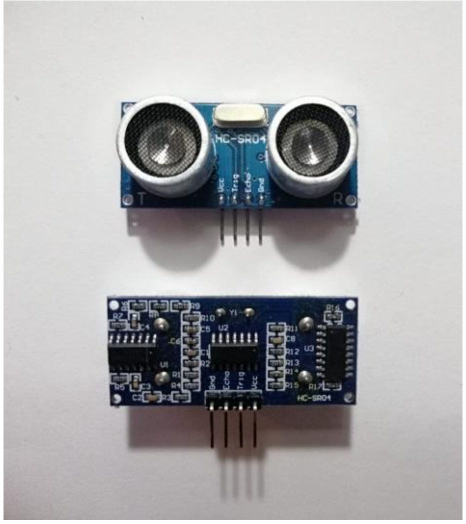
This project’s purpose is to create an object avoiding robot, with the help of Arduino technology. This action is made possible with a proximity sensor with ultrasounds, being capable to detect obstacles and at what distance they are from the robot. After that, the robot continues it’s movement in another direction.

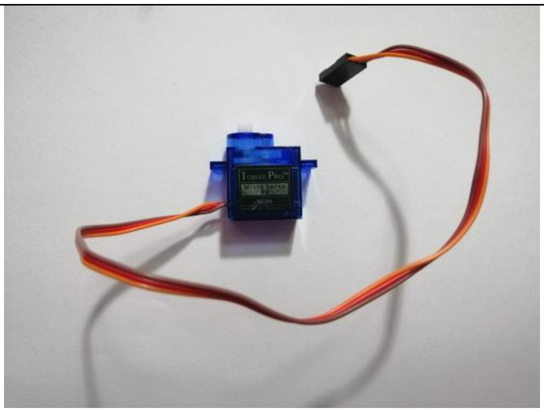
HARDWARE RESOURCES

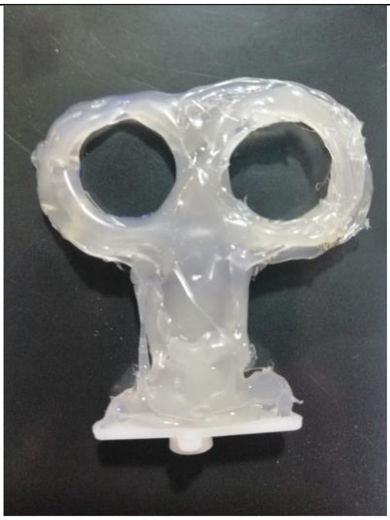
Component Name	Component Picture	Characteristics
Arduino UNO development board	 	It is a small platform built around a signal processor and is capable of retrieving data from the environment through a series of sensors and performing actions on the environment through lights, motors, actuators, and other types of mechanical devices. The processor is capable of running code written in a programming language that is very similar to C ++. Technical details: • Working voltage: 5V • Input voltage: 7-12V • Input voltage (limit): 6-20V • Digital pins: 14 (6 PWM output) • Analog pins: 6 • I / O pin current: 40 mA • 3.3V current: 50 mA • Dimensions: 69mm x 52mm x 13mm

Driver module L298N Dual Arduino compatible H-bridge		This module allows the control and speed and direction of two DC motors. The H-bridge L298N can be used with motors with a voltage between 5 and 35V DC. There is also an integrated 5V regulator, so if the supply voltage is up to 12V, there is no need for separate supply of the logic part. Technical details: • Motor voltage 5V - 35V • 5V logic circuit voltage • Motor current 2A (MAX) • Logic current 36mA • Dimensions: 43 x 43 x 27 mm.
2WD V1 chassis kit		Contains: • Support plate • 2x continuous motor current • 2x rubber wheel • Plastic wheel • Engine mounts • Screws

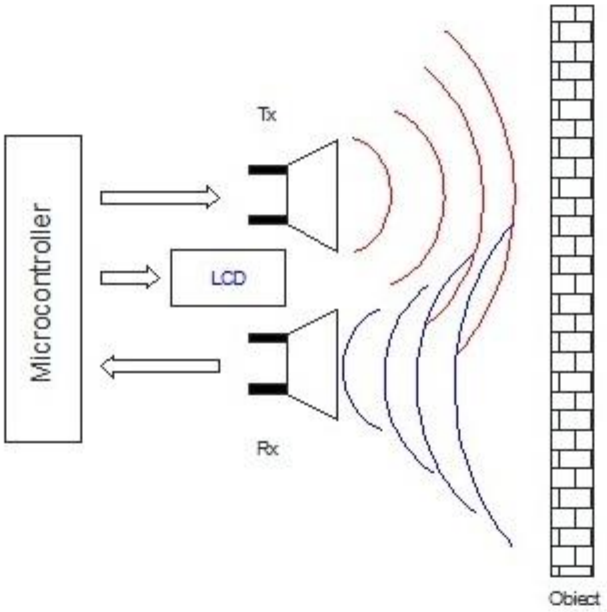
Engines MG-6-120		6V rated gear motor and Gear Ratio 1: 120. Technical details • Double shaft gear motor • Rated voltage: 6 V DC • Rated current: <300 mA • Rotation speed: 100 / min • Torque: 1 kgf.cm
Connection wires		• Length: 22 cm • 3 types: • Mother – Mother • Mother – Father • Father - Father
Breadboard mini		It is a building base for mechatronic and electronic design. Because it is not necessary to bond the wires, it is reusable.

Ultrasonic proximity sensor HC-SR04		Pins: • VCC: + 5VDC • Trig: Trigger (OUTPUT) • Echo: Echo (INPUT) • GND: GND Operating mode: • The transmitter emits a high frequency sound signal, it hits the object and is reflected to the receiver; Technical details: • Measuring angle: 30 • Effective angle: 15 (2 * 15 = 30) • Electrical intensity: 15mA • Dimensions: 45mm x 20mm x 15mm • Range: 2-400 cm

Servomotor SG90		This servomotor is specially designed for low power applications. It can rotate about 180 degrees (90 degrees in both directions). Pins: 1) Ground 2) + V, Power 3) Signal Technical details: • Power supply voltage: 4.8V; • Low power consumption; • Operating speed: 0.12 s / 60o @ 4.8 V; • Torque in lock at 4.8V: 1.8 kgf * cm; • PWM frequency: 50Hz; • Operating temperature: -30 ° C - + 60 ° C. • Dimensions: 21.5 x 11.8 x 22.7 mm

9V Battery		In order to carry out this project, two 9 volt batteries were needed, one connected to the Arduino development board and the other to the L298N module;
Ultrasonic sensor support for servomotor		The support is made manually with the help of a soldering gun. This is a negative mold of the ultrasound sensor, to which the servomotor will be attached.

IMPLEMENTATION OF THE APPLICATION



FUNCTIONING PRINCIPLE

Figure 1

The movement of the robot is conditioned by the signal transmitted / received by the ultrasound sensor (figure 1).

It detects objects within its range by emitting sound signals, and measuring the time required to return them. Because the signal speed is constant (this is the speed of sound), the distance between the robot / sensor and the object is very easy to determine (figure 2).

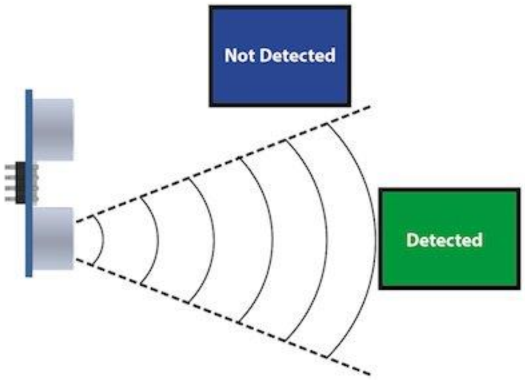


Figure 2

The transmitter and the receiver are disposed at a distance of 1 centimeter from each other, each having an effective angle of 15 degrees. Because of this angle, some objects may not be detected because they are not within its range (Figure 3).

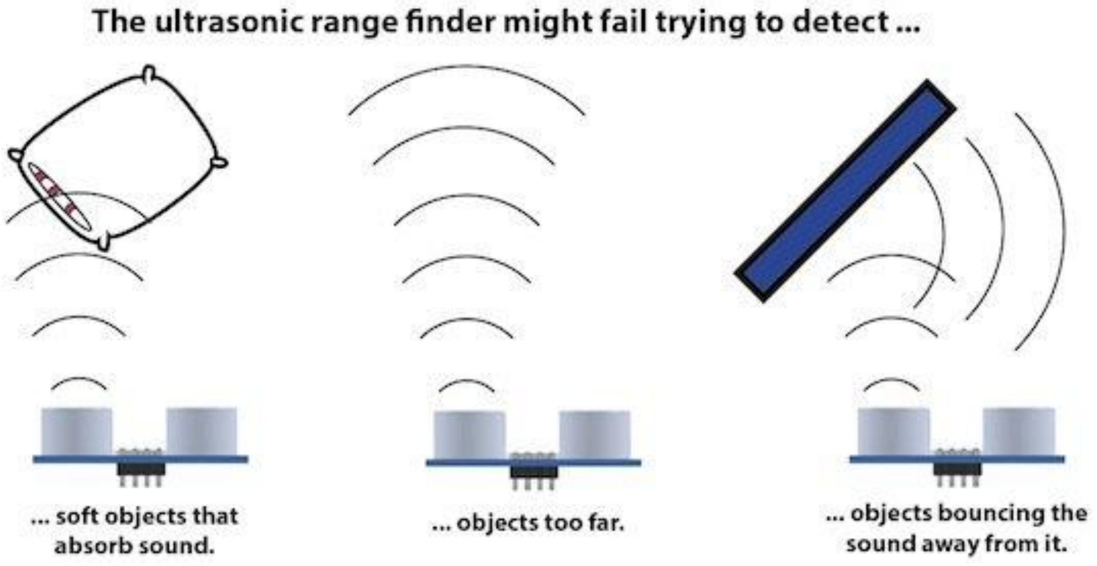

The ultrasonic range finder might fail trying to detect ...
... soft objects that absorb sound.
... objects too far.
... objects bouncing the sound away from it.

Figure 3

The accuracy of the sensor or even its operation may be limited by several factors. The material and consistency of the detected object are very important. When colliding sound waves with blunt objects, a higher percentage of them are reflected back to the receiver, the position of the object being easy to determine. On the other hand, soft materials absorb much of the sound waves, and the position of the body becomes difficult to decipher.

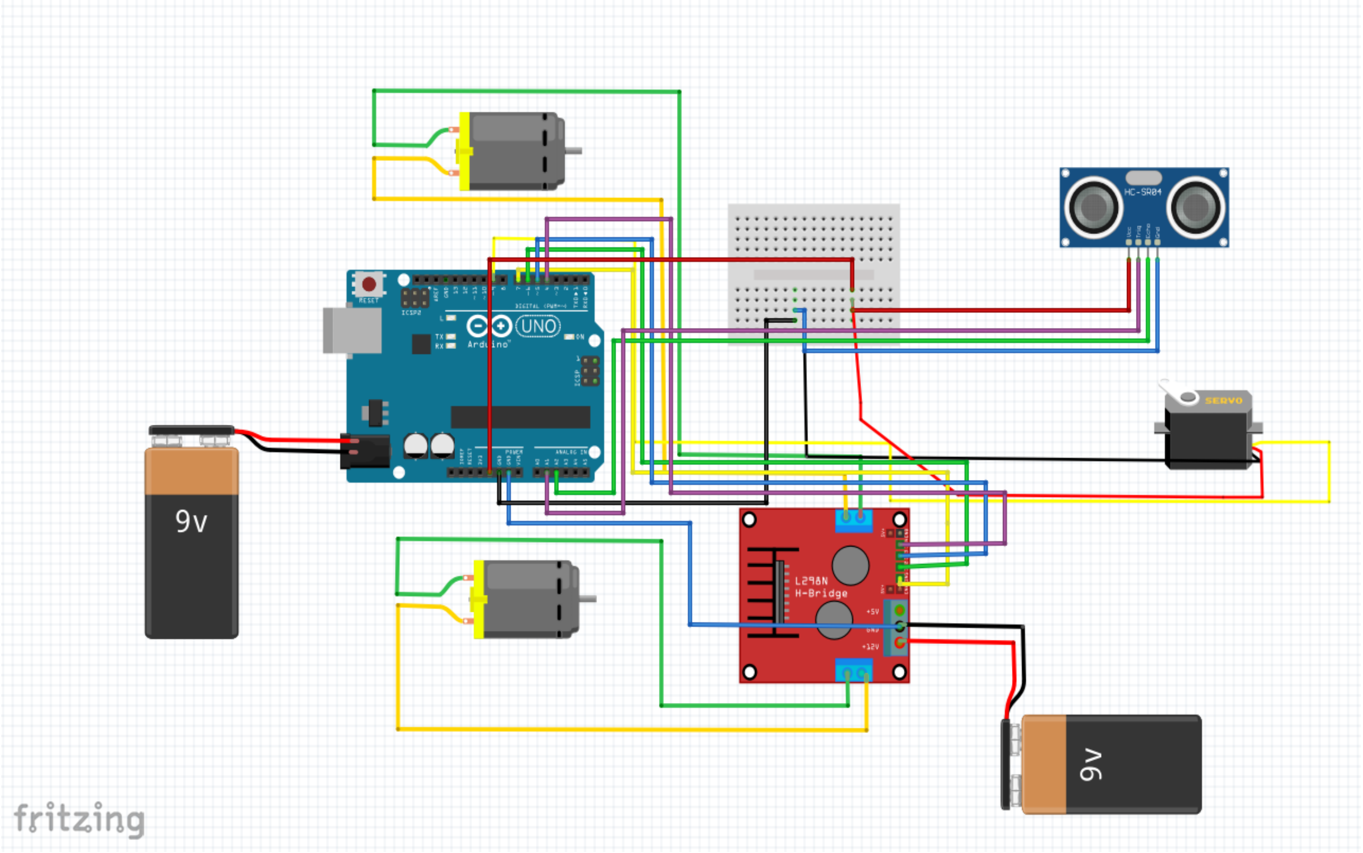
Another factor to consider is the distance from the body. The HC-SR04 ultrasonic proximity sensor has a range of 2 to 400 centimeters. Any object at a distance or smaller becomes very difficult / impossible to position.

The orientation of the object is important, because the angle of reflection depends on the angle at which the object is in front of the sensor.

SOFTWARE AND ASSEMBLY

Printre primii pași in constructia unui robot sau a unui șistem cu ajutorul tehnologiei Arduino este crearea diagramei cu toate componentele electronice din cadrul acestuia (controllere, placi de dezvoltare, fire, motoare, senzori).

Figure 7- Circuit Diagram



COMPONENT LINKS AND CONNECTIONS

- HC-SR04 ultrasound proximity sensor

• Gnd (through breadboard) - Gnd development board

• Echo - pin A2

• Trig - pin A1

• Vcc (through breadboard) - 5V development board

-Servomotor SG90

• Gnd (through breadboard) - Gnd development board

• Vcc (through breadboard) - 5V development board

• Signal - pin 11

- Driver module L298N double deck H

• In1 - pin 7

• In2 - pin 6

• In3 - pin 5

• In4 - pin 4

• 12V - terminal + battery1 9V

• Gnd - terminal - battery1 9V

• Gnd - Gnd development board

• Out1 - Terminal 1 Motor1 MG-6-120

• Out2 - Terminal 2 Motor1 MG-6-120

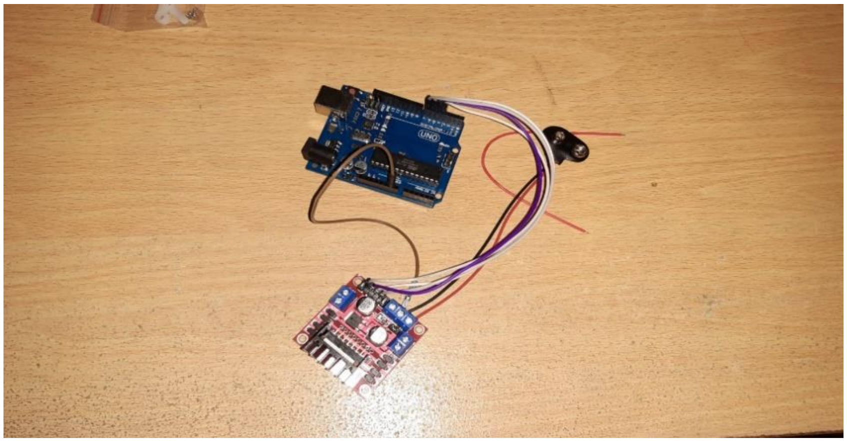
• Out3 - Terminal 1 Motor2 MG-6-120

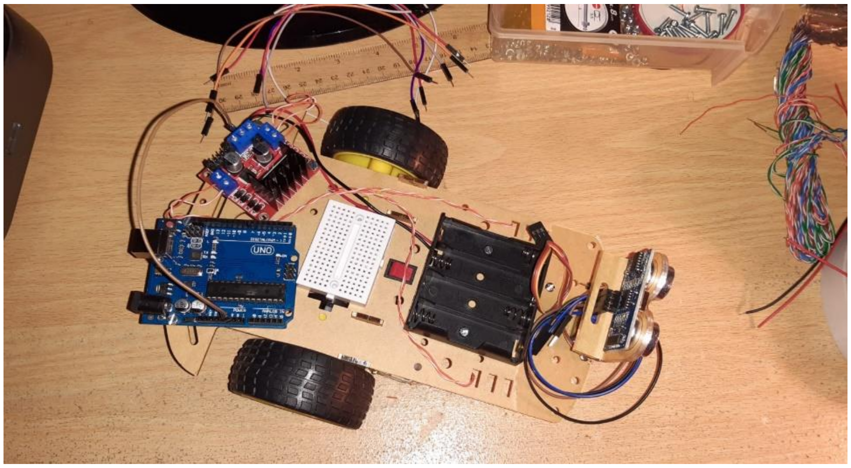
• Out4 - Terminal 2 Motor2 MG-6-120

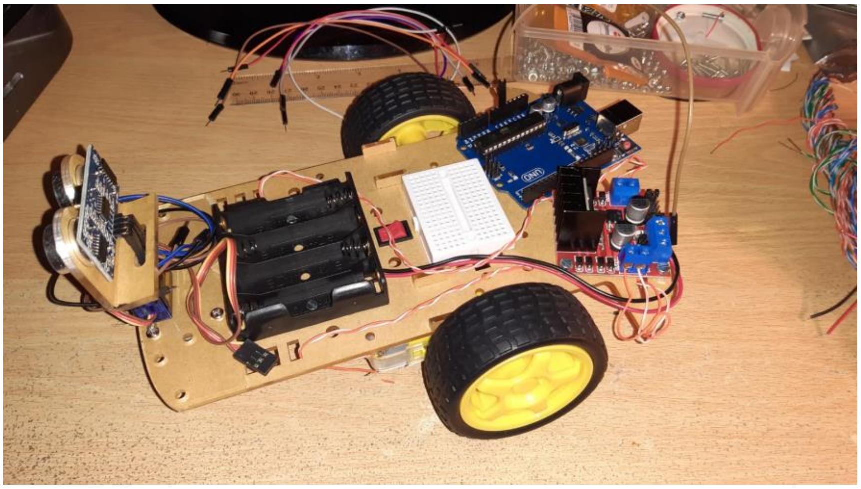
-9V battery2

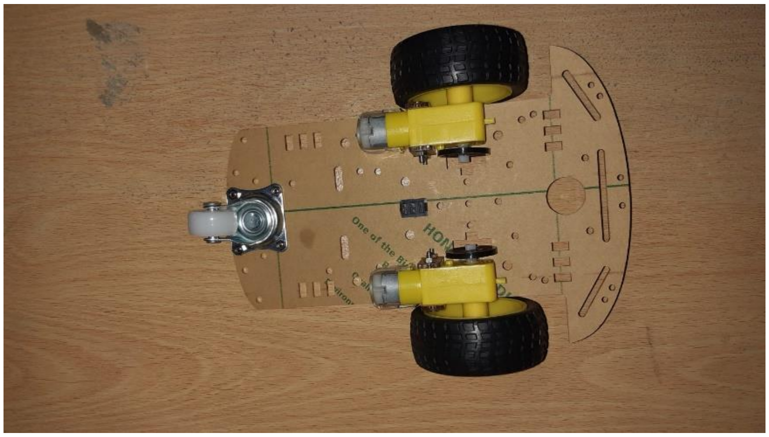
• Terminal + - power supply jack jack plate development

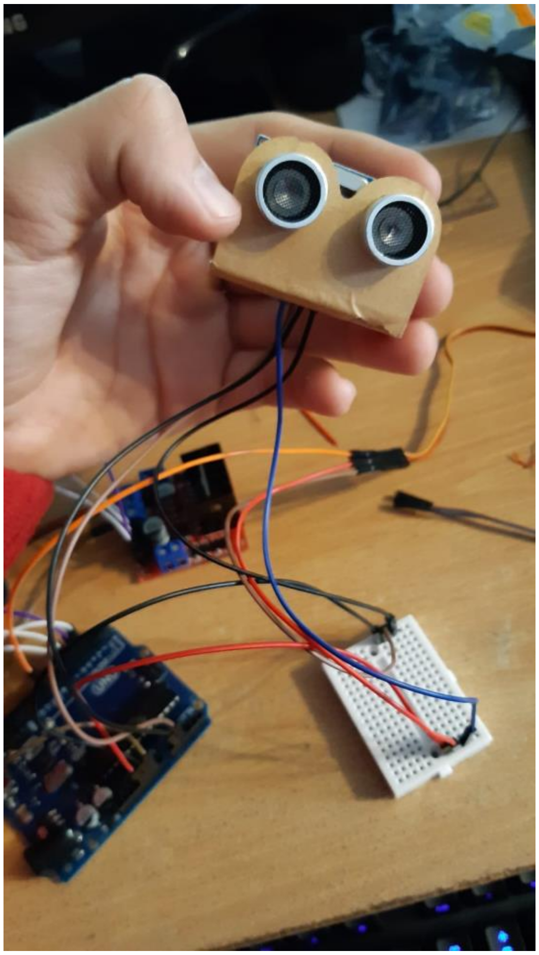
• Terminal - - food jack jack plate development


UNO

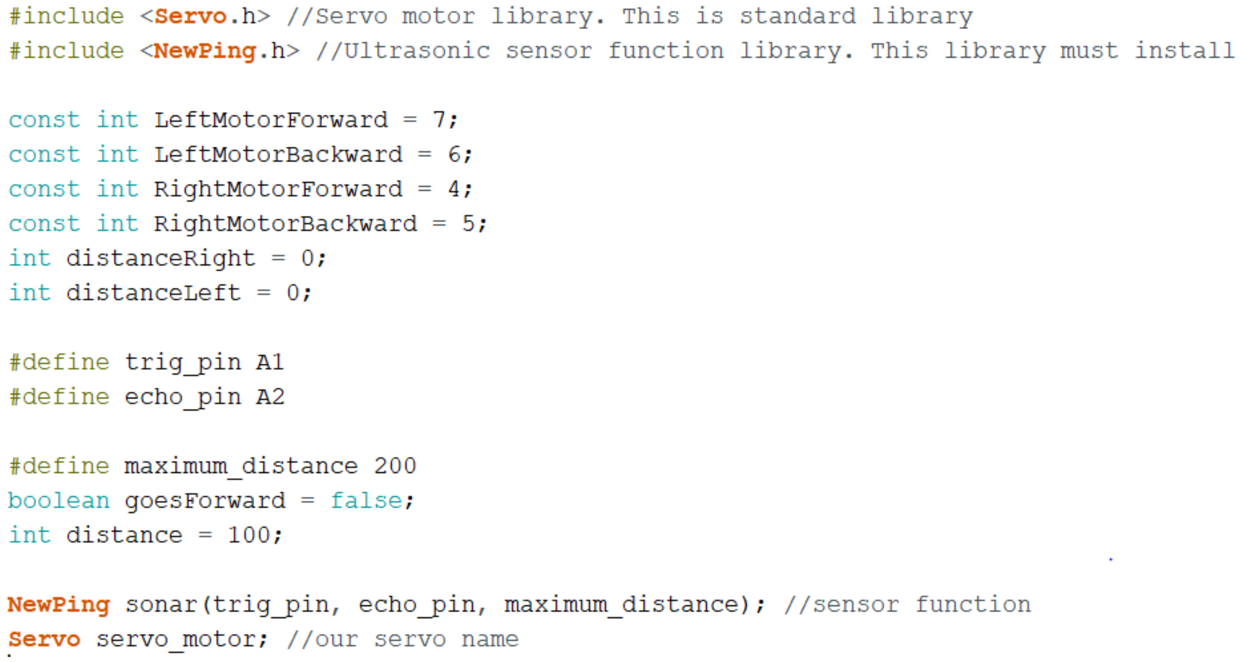

UNO


UNO



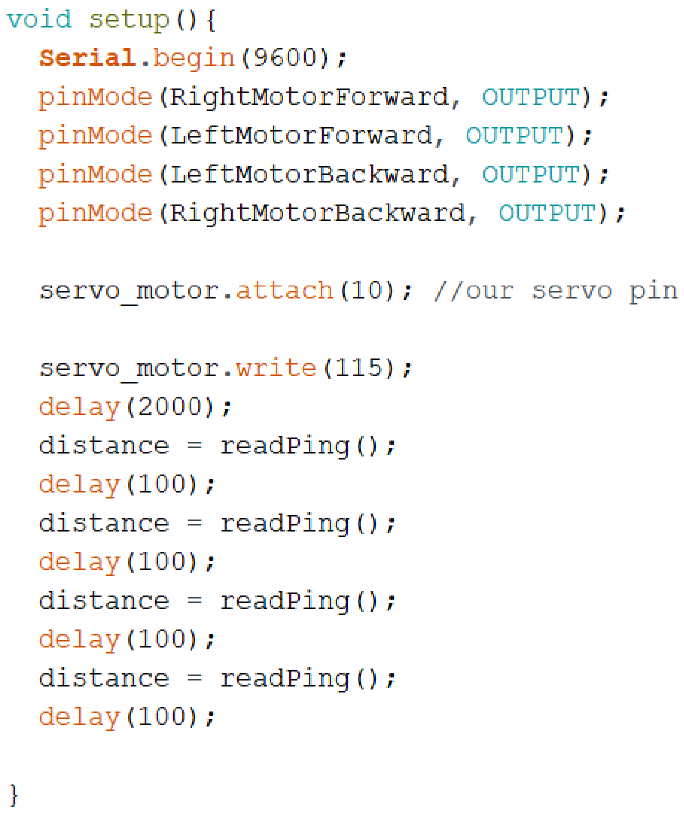


SOFTWARE



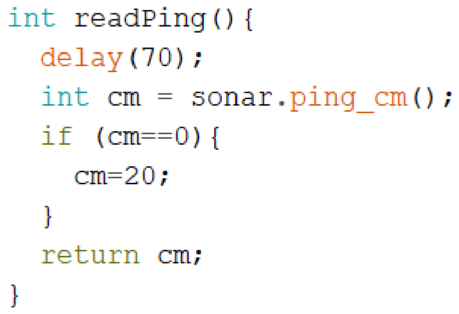
```
#include <Servo.h> //Servo motor library. This is standard library
#include <NewPing.h> //Ultrasonic sensor function library. This library must install

const int LeftMotorForward = 7;
const int LeftMotorBackward = 6;
const int RightMotorForward = 4;
const int RightMotorBackward = 5;
int distanceRight = 0;
int distanceLeft = 0;

#define trig_pin A1
#define echo_pin A2

#define maximum_distance 200
boolean goesForward = false;
int distance = 100;

NewPing sonar(trig_pin, echo_pin, maximum_distance); //sensor function
Servo servo_motor; //our servo name
```

The first block of code consists of libraries with the declaration of engines and distances for our sensor.



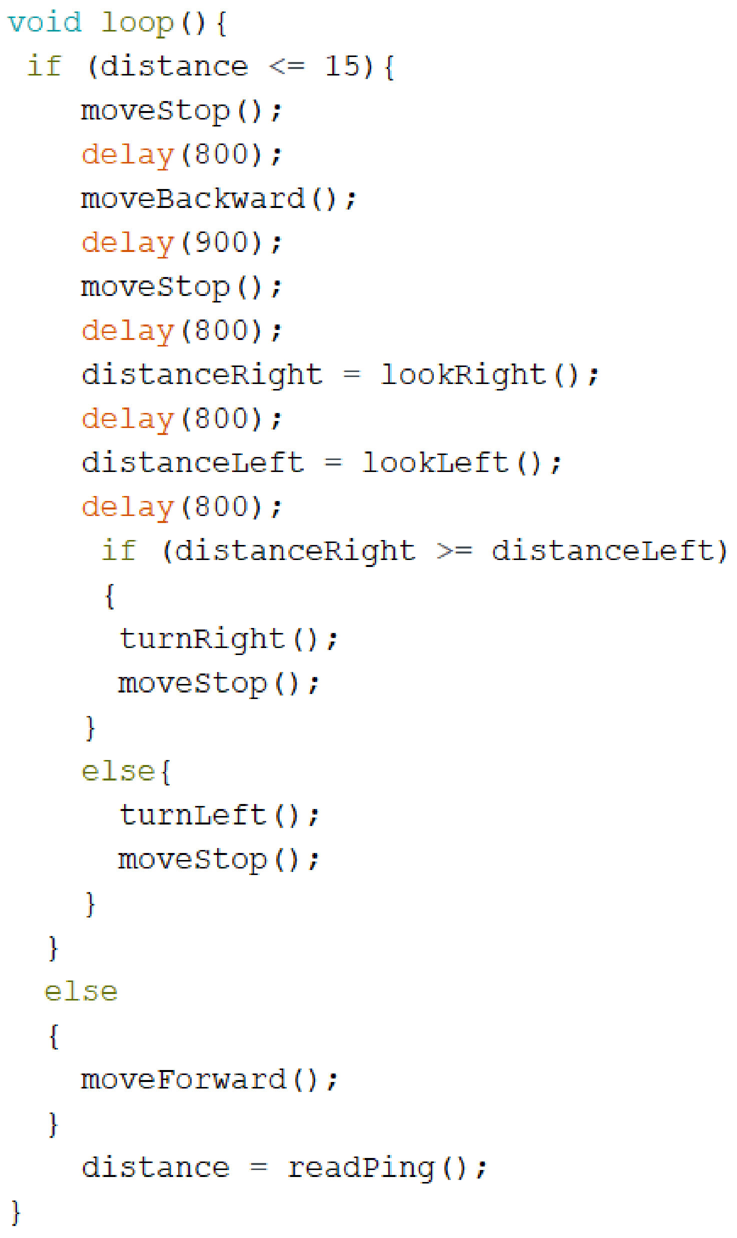
```
void setup(){
  Serial.begin(9600);
  pinMode(RightMotorForward, OUTPUT);
  pinMode(LeftMotorForward, OUTPUT);
  pinMode(LeftMotorBackward, OUTPUT);
  pinMode(RightMotorBackward, OUTPUT);

  servo_motor.attach(10); //our servo pin

  servo_motor.write(115);
  delay(2000);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);

}
```

This is our setup function following the declaration part, where the engines are initialized within the pins and the delay for the servo motor is written.



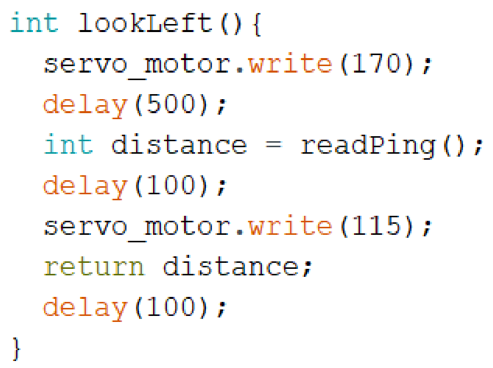
```
int readPing(){
  delay(70);
  int cm = sonar.ping_cm();
  if (cm==0){
    cm=20;
  }
  return cm;
}
```

readPing function is pretty straight forward, it only increments the distance read by the sensor.

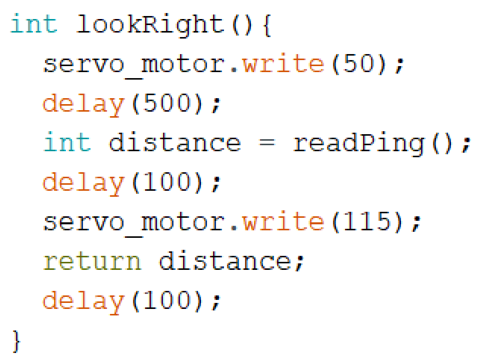


```
void loop(){
 if (distance <= 15){
    moveStop();
    delay(800);
    moveBackward();
    delay(900);
    moveStop();
    delay(800);
    distanceRight = lookRight();
    delay(800);
    distanceLeft = lookLeft();
    delay(800);
     if (distanceRight >= distanceLeft)
     {
      turnRight();
      moveStop();
    }
    else{
      turnLeft();
      moveStop();
    }
  }
  else
  {
    moveForward();
  }
    distance = readPing();
}
```

This is the main loop. The movement of the robot is based on the distance read by the sensor.

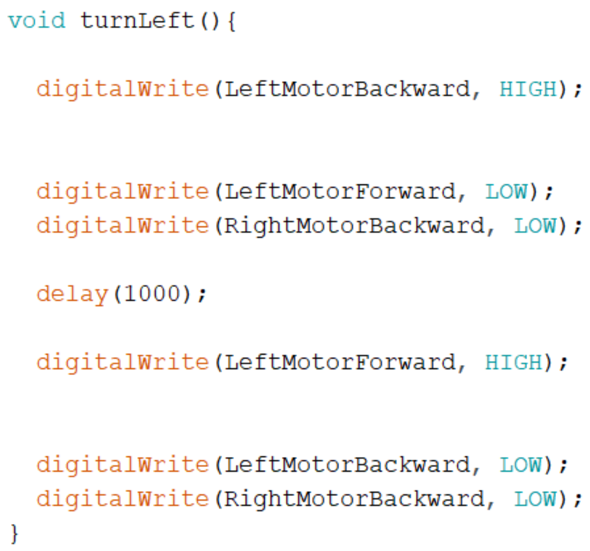


```
int lookLeft(){
  servo_motor.write(170);
  delay(500);
  int distance = readPing();
  delay(100);
  servo_motor.write(115);
  return distance;
  delay(100);
}
```

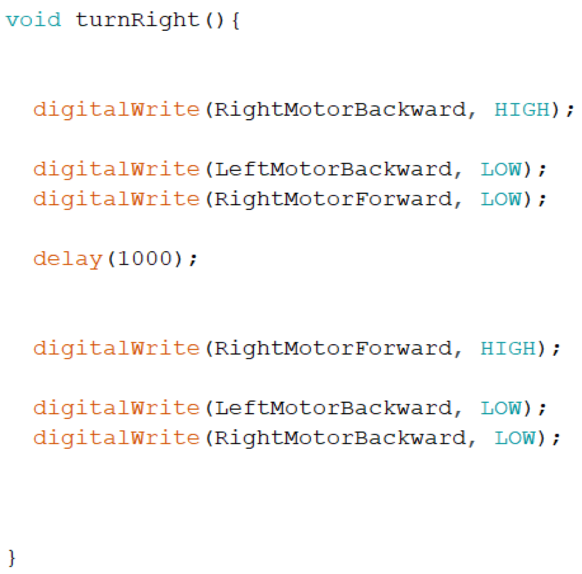


```
int lookRight(){
  servo_motor.write(50);
  delay(500);
  int distance = readPing();
  delay(100);
  servo_motor.write(115);
  return distance;
  delay(100);
}
```

lookLeft and lookRight functions are similar, only the value in the first line is different but the rotation distance is equal on both sides with respect to a middle imaginary axis.

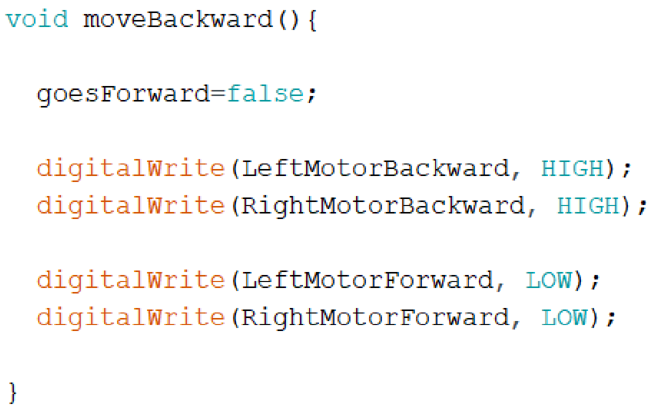


```
void turnLeft(){

  digitalWrite(LeftMotorBackward, HIGH);


  digitalWrite(LeftMotorForward, LOW);
  digitalWrite(RightMotorBackward, LOW);

  delay(1000);

  digitalWrite(LeftMotorForward, HIGH);


  digitalWrite(LeftMotorBackward, LOW);
  digitalWrite(RightMotorBackward, LOW);
}
```

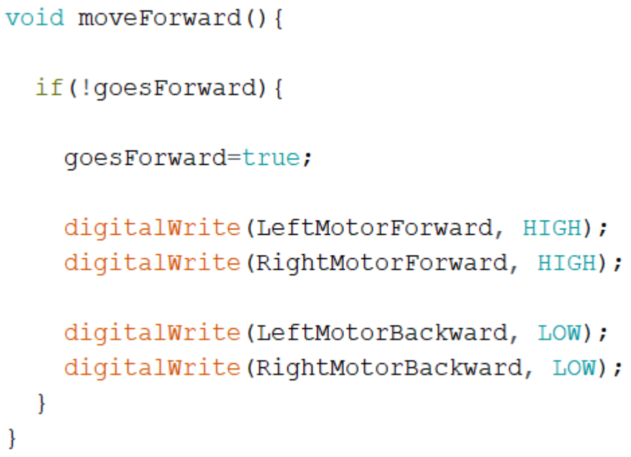


```
void turnRight(){


  digitalWrite(RightMotorBackward, HIGH);

  digitalWrite(LeftMotorBackward, LOW);
  digitalWrite(RightMotorForward, LOW);

  delay(1000);


  digitalWrite(RightMotorForward, HIGH);

  digitalWrite(LeftMotorBackward, LOW);
  digitalWrite(RightMotorBackward, LOW);



}
```

For turning the vehicle we use only one wheel at the time, while the other is at rest.



```
void moveBackward(){

  goesForward=false;

  digitalWrite(LeftMotorBackward, HIGH);
  digitalWrite(RightMotorBackward, HIGH);

  digitalWrite(LeftMotorForward, LOW);
  digitalWrite(RightMotorForward, LOW);

}
```



```
void moveForward(){

  if(!goesForward){

    goesForward=true;

    digitalWrite(LeftMotorForward, HIGH);
    digitalWrite(RightMotorForward, HIGH);

    digitalWrite(LeftMotorBackward, LOW);
    digitalWrite(RightMotorBackward, LOW);
  }
}
```

Moving forward and backward is done by activating at the same time both engines and making them spin in the direction we want the robot to move.

BIBLIOGRAPHY

https://www.youtube.com/watch?v=tXsP9STxdBc

https://www.youtube.com/watch?v=HZ1tEEOjAHE

https://www.arduino.cc/en/reference/servo

https://playground.arduino.cc/Code/NewPing/

http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

http://www.electromatic.ro/ro/senzori/item/35-senzorideproximitateultrasonici

https://www.senzori-ultrasonici.ro/principiul-ultrasonic

http://fritzing.org/home/

https://www.instructables.com/id/Obstacle-Avoiding-Robot-Arduino-1/

https://bitbucket.org/teckel12/arduino-new-ping/downloads/

https://www.arduinolibraries.info/libraries/servo

http://fritzing.org/download/