Mașinuță teleghidată

**Profesor coordonator** **Studenți**

Trăistaru Mihai Almășan Victor Petru

Belbe Ioan Cristian

Dulău Marius Cristian

**Ianuarie 2018**

**Cuprins**

1. Enunțul problemei..........................................................................................................2
2. Soluția problemei...........................................................................................................3
   1. Codul pentru placa Arduino Mega 2560.........................................................4
   2. Codul pentru placa Arduino Uno..................................................................11
3. Poze proiect..................................................................................................................13
4. Posibilități de dezvoltare ulterioară..............................................................................17
5. Referințe.......................................................................................................................18
6. **Enunțul problemei**

Să se proiecteze un robot teleghidat folosind interfețele de comunicare oferite de plăcile Arduino. Robotul trebuie să se poată deplasa și să mai aibă încă cel puțin o funcție folosind unul sau mai mulți senzori, funcție stabilită la începutul proiectului.

1. **Soluția problemei**

Proiectul constă din construirea unei mașinuțe care are 2 moduri de funcționare:

* deplasarea în funcție de semnalul de la o telecomandă
* deplasare autonomă

Pentru ambele funcții mașinuța este capabilă să semnalizeze anumite acțiuni (stânga, dreapta, mersul cu spatele, frână).

Pentru a rezolva problema a fost nevoie de următoarele componente:

* 1 placă Arduino Mega 2560
* 1 placă Arduino Uno
* cablu USB tip B pentru programare plăcilor
* 2 breadbord-uri (unul de 400 de puncte și unul de 100 de puncte)
* 1 celulă fotosensibilă
* 4 led-uri albe
* 2 led-uri galbene
* 2 led-uri roșii
* 2 led-uri RGB
* 3 rezistențe (două de 220Ω și una de 10 kΩ)
* fire mamă-tată
* fire jumper
* 1 motor servo
* 1 senzor ultrasonic HC-SR04
* 1 senzor IR
* 1 telecomandă pentru sezorul IR
* 1 baterie de 3V pentru telecomanda IR
* 1 punte H model L298N
* 4 motoare DC
* 4 roți
* 4 baterii AA
* 1 suport pentru baterii AA
* 1 baterie de 9V
* 1 suport pentru bateria de 9V
* șasiu (2 bucați de plexiglas)
* 1 șurubelniță stea
* 6 stâlpuri de cupru pentru suprepunerea bucăților de plexiglas
* 13 șuruburi (unul de dimensiune diferită față de celelalte)
* 2 piulițe în formă de haxagon de diferite dimesiuni

Codul a fost scris în C++ folosind Microsoft Visual Studio 2015 având instalat driver-ul ViMicro (Visual Micro). S-a creat câte un program pentru fiecare placă. Placa Arduino Mega 2560 are codul pentru deplasare, iar placa Arduino Uno codul pentru semnalizare. Cele 2 plăci comunică între ele prin protocolul I2C.

* 1. **Codul pentru placa Arduino Mega 2560**

Fișierul “Masinuta.ino” conține librăriile pentru mașinuță, telecomanda IR și motorul servo. Sunt declarate variabile pentru optiunile alese de utilizator, citirea valorii luminii, valoarea pentru ledurile albe, poziția motorului servo, pinul celulei fotosensibile. Funcția de setup conține setarea pentru modulull IR, motorul servo și viteza de transmisie al interfeței Serial. Interfața Serial a fost folosită pentru depanare.

#include "Car.h"

#include <IRremote.h>

#include <Servo.h>

Car \*myCar = new Car(7, 6, 5, 2, 4, 3);

int RECV\_PIN = 11;

IRrecv irrecv(RECV\_PIN);

decode\_results results;

int photocell = 0;

int photocellReading = 0;

int value = 0;

bool wantToRead = true;

int option = 0;

Servo myServo;

int pos = 0;

// the setup function runs once when you press reset or power the board

void setup() {

irrecv.enableIRIn();

Serial.begin(9600);

myServo.attach(9);

}

S-a creat o clasă Car pentru funțiile necesare deplasării mașinuței și transmiterii de date către placa Arduino Uno.

#pragma once

#include <Arduino.h> //includem Arduino.h pentru a avea acces la instructiunile specifice Arduino

#include <Wire.h> //comunicare prin I2C

class Car

{

private:

int carSpeed;

int directionFlag;

int directionMove;

int lightValue;

int goLeft;

int goRight;

Toate variabilele de instanță și metoda de setare a motoarelor au fost declarate private pentru a nu fi accesibile în fișierul “Masinuta.ino”. În constructorul clasei Car se initializează variabilele de instanță, se setează pinii plăcii și se pornește protocolul de comunicație I2C.

//motor A

int enA;

int in1;

int in2;

//motor B

int enB;

int in3;

int in4;

//ultrasonic

int trigPin;

int echoPin;

long duration;

int leftDist;

int rightDist;

//servo

bool rotateServo = false;

void engineSetup(int motor1Value, int motor2Value, int motor3Value, int motor4Value);

public:

Car(int enableA, int input1, int input2, int enableB, int input3, int input4);

void forward();

void backward();

void left();

void right();

void brake();

void stop();

void forwardSpeed();

void backwardSpeed();

void translateIR(unsigned long value);

void setLightValue(int value);

void sendMessage();

bool needToRotateServo();

void setRotateServo(bool val);

int ultrasonicDistance();

void setDistances(int letf, int right);

void driveAutonomous();

};

În funcția loop() din fișierul .ino se citește valoarea luminii din încăpere, transmiterea acestei valorii pentru ledurile albe către obiectul de tip Car și citirea opțiunii utilizatorului pentru modul de funcționare a mașinii. Se poate alege între deplasarea comandată de la telecomandă sau deplasare autonomă. În cazul în care se alege deplasarea comandată de la telecomandă funcția loop permite achiziția de date de la telecomandă și transmiterea acestora la obiectul myCar, unde aceste date comandă deplasarea autovehiculului. Dacă se alege deplasarea autonomă se primesc date de la obiect pentru acționarea motorului servo și transmiterea distanței atât din stânga cât și din dreapta înapoi la acesta. De asemenea, la finalulu funcției loop() se transmit date că placa Arduino Uno, indiferent de opțiunea aleasă.

// the loop function runs over and over again until power down or reset

void loop() {

photocellReading = analogRead(photocell);

if (photocellReading > 255) value = 0;

else value = 255 - photocellReading;

if (irrecv.decode(&results))

{

Serial.println(results.value);

//0xFF22DD = codul pentru butonul 0

if (results.value == 0xFF22DD)

{

wantToRead = true;//PREV

option = 0;

myCar->stop();//oprim masina

irrecv.resume();

Serial.println("PREV");

delay(1000);

}

if (wantToRead)

{

switch (results.value)

{

case 0xFF30CF: option = 1; //butonul 1

break;

case 0xFF18E7: option = 2; //butonul 2

break;

}

wantToRead = !wantToRead;//nu vrem sa citim alta optiune

irrecv.resume();

}

else

{

switch (option)

{

case 0: wantToRead = true;

break;

case 1: myCar->translateIR(results.value);

irrecv.resume();

break;

case 2: myCar->driveAutonomous();

if (myCar->needToRotateServo())

{

for (pos = 90; pos <= 180; pos++)

{

myServo.write(pos);

delay(10);

}

int left = myCar->ultrasonicDistance();

delay(1);

for (pos = 180; pos >= 0; pos--)

{

myServo.write(pos);

delay(10);

}

În cazul primei opțiuni de funcționare (deplasare comandată prin intermediul telecomenzii) se preia valoarea transmisă prin infraroșu și se trimite la obiectul mașină și se prelucrează datele pentru deplasare (metoda translateIR(results.value)).

void Car::translateIR(unsigned long value)

{

switch (value)

{

case 0xFF18E7: directionMove = 1;

goLeft = goRight = 0;//nu trebuie sa semnalizam stanga/dreapta

forwardSpeed(); //viteza pentru mersul cu fata

forward(); // mergem in fata

break;

case 0xFF4AB5: directionMove = 2; //semnalizare pentru mersul cu spatele

goLeft = goRight = 0;//nu trebuie sa semnalizam stanga/dreapta

backwardSpeed(); //viteza pentru mersul cu spatele

backward(); // mergem cu spatele

break;

case 0xFF10EF: goRight = 0; // nu mergem in dreapta

goLeft++;

if (goLeft == 1) directionMove = 3; //semnalizam ca mergem in stanga

else if (goLeft == 2)

{

left(); // mergem in stanga

goLeft = 0;//resetare

}

break;

int right = myCar->ultrasonicDistance();

delay(1);

myCar->setRotateServo(false);

myCar->setDistances(left, right);

for (pos = 0; pos <= 90; pos++)

{

myServo.write(pos);

delay(10);

} //rotire motor servo inapoi la pozitia initiala

}

break;

}

}

}

myCar->setLightValue(value);

myCar->sendMessage();

Serial.println(option);

}

Valorile 0xFF18E7, 0xFF4AB5, 0xFF10EF, 0xFF5AA5, 0xFF38C7 corespund butoanelor 2, 8, 4, 6, respectiv 5 de pe telecomandă.

case 0xFF5AA5: goLeft = 0; // nu mergem in stanga

goRight++;

if (goRight == 1) directionMove = 4;//semnalizam ca mergem in dreapta

else if (goRight == 2)

{

right();// mergem in stanga

goRight = 0;//resetare

}

break;

case 0xFF38C7: directionMove = 5; //semnalizare pentru frana

brake(); //frana

if (carSpeed == 0) engineSetup(LOW, LOW, LOW, LOW); // brake

break;

}

delay(10);

}

Metoda engineSetup are 4 parametrii și setează modul de funcționare a motoarelor DC, adică direcția de rotație a roților, cât și setarea vitezei de deplasare.

void Car::engineSetup(int motor1Value, int motor2Value, int motor3Value, int motor4Value)

{

digitalWrite(in1, motor1Value);//motor dreapta

digitalWrite(in2, motor2Value);//motor dreapta

digitalWrite(in3, motor3Value);//motor stanga

digitalWrite(in4, motor4Value);//motor stanga

analogWrite(enA, carSpeed);

analogWrite(enB, carSpeed);

}

Pentru motoarele de pe o parte (stânga sau dreapta) se trimit 2 valori care indică sensul de rotație. Dacă se asignează perechea (HIGH, LOW) atunci motoarele se învârt în față. Perechea (LOW, HIGH) va învârti motoarele în sens opus. Perechile (LOW, LOW) sau (HIGH, HIGH) va opri motoarele de pe partea în care sunt aplicate. Pentru deplasarea spre dreapta roțile din stânga se învârt în față, iar cele din dreapta se învârt în spate. Pentru deplasarea spre stânga roțile din dreapta se învârt în față, iar cele din stânga se învârt în spate.

Metoda brake() scade viteza mașinii. Metoda stop() aplică metoda brake() până când viteza mașinii este egală cu 0. Apoi oprește motoarele folosind metoda engineSetu folosind apelul engineSetup(LOW, LOW, LOW, LOW).

În cadrul clasei Car avem 2 metode pentru ajustarea vitezei cand mergem în față sau în spate.

void Car::forwardSpeed()

{

if (directionFlag == 0)

{

if (carSpeed <= 200) carSpeed += 50;

else carSpeed = 255;

//daca mergem in fata crestem viteza

}

else

{

//mergem cu spatele si trebuie sa scadem viteza pentru a ajunge la 0 si apoi sa mergem in primul if

brake();

if (carSpeed == 0) directionFlag = 0; //flag pentru mersul cu fata

}

}

void Car::backwardSpeed()

{

if (directionFlag == 0)

{

//daca mergem in fata si vrem sa mergem cu spatele trebuie scadem viteza

if (carSpeed > 50) carSpeed -= 50;

else

{

carSpeed = 0;

directionFlag = 1;//flag pentru a merge cu spatele

}

}

else

{

//vrem sa mergem cu spatele si trebuie sa crestem viteza ca sa mergem cat mai repede

if (carSpeed < 255) carSpeed += 25;

else carSpeed = 255;

}

}

Când mașina este pe modul de deplasare autonom ne folosim de senzorul ultrasonic pentru a determina distanța pânâ la obstacol. Acest senzor emite o unda care când lovește un obstacol se reflectă. Senzorul măsoară timpul de la emitere până la receptare. Viteza undei este de 340 m/s = 0.034 cm/µs (viteza sunetului). Cunoscând timpul și viteza undei putem determina distanța până la obstacol. Aceasta distanță se împarte la 2 pentru că unda a parcurs de 2 ori distanța respectivă: o dată de la emitere până la întâlnirea cu obiectul și a doua oară când s-a reflectat.

int Car::ultrasonicDistance()

{

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

Principiul de funcționare în modul autonom este următorul: robotul merge în față. Când întâlnește un obiect în cale la o distanță mai mică de 15 de centimetri se oprește. Motorul servo se rotește în stânga și în dreapta pentru ca senzorul ultrasonic să determine distanțele în ambele părți. Robotul va da cu spatele, apoi va merge în direcția în care direcția la un alt obstacol este mai mare.

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

// The speed of sound is 340 m/s or 29 microseconds per centimeter.

// The ping travels out and back, so to find the distance of the

// object we take half of the distance travelled.

return duration / 29 / 2;

}

void Car::driveAutonomous()

{

int distance = ultrasonicDistance();//preluam distanta pana la obstacol

//Serial.println(distance);

if (distance >= 15)

{

directionMove = 1;//mergem in fata

sendMessage();

//forwardSpeed();

forward();

}

else

{

stop();//am dat de un obstacol si trebuie sa ne oprim

setRotateServo(true);//dam control in loop pentru a roti motorul servo

if (leftDist != 0 && rightDist != 0)

{

carSpeed = 255;

backward();

directionMove = 2;//dam cu spatele

sendMessage();//trimitem datele

delay(500);

if (rightDist > leftDist)

{

right();

directionMove = 4;//daca distanta la un obstacol in partea dreapta e mai mare ca in partea stanga mergem in dreapta

}

else

{

left();

directionMove = 3;//daca distanta la un obstacol in partea stanga e mai mare ca in partea dreapta mergem in stanga

}

Metoda sendMessage() trimite date către placa Arduino Uno. Se trimit date valoarea ledurilor albe(faruri), valoarea pentru ledurile roșii(stopuri) și direcția în care se indreaptă robotul.

sendMessage();//trimitem datele

delay(1250);

leftDist = rightDist = 0;//setam valorile pe 0 pentru a nu intra in acest if

setRotateServo(false);

}

}

}

Pentru ca placa Mega să comunice cu placa Uno, pinul 20 de la Mega se conectează la pinul A4 de la Uno, iar pinul 21 de la Mega se conectează la pinul A5 de la Uno.

void Car::sendMessage()

{

//Serial.println(value);

//Serial.println(direction);

int stopuri = lightValue >= 10 ? 10 : 0;

Wire.beginTransmission(8);

Wire.write(lightValue);//faruri

Wire.write(stopuri);//stopuri

Wire.write(directionMove);//directie

Wire.endTransmission();

}

* 1. **Codul pentru placa Arduino Uno**

Fișierul “Sistem\_lumini.ino” conține librăria pentru comunicarea prin I2C cu placa Arduino Mega, variabile pentru direcție, faruri, stopuri, timp și pini. În funcția setup() se definește tipul fiecărui pin și se pornește protocul de comunicare și interfața serială. Interfața Serial a fost utilizată pentru depanare.

#include <Wire.h>

int valueFaruri = 0;

int valueStopuri = 0;

int directie = 1;

int blink = 0;

int nr = 255;

int time;

int faruri = 10;

int stopuri = 6;

int stangaFata = 5;//semnalizare stanga

int dreaptaFata = 11;//semnalizare dreapta

Avem definită funcția de primire a mesajului de la placa Mega.

int bStangaSpate = A2;

int rgStangaSpate = A3;

int bDreaptaSpate = A0;

int rgDreaptaSpate = A1;

// the setup function runs once when you press reset or power the board

void setup() {

pinMode(stangaFata, INPUT\_PULLUP);

pinMode(dreaptaFata, INPUT\_PULLUP);

pinMode(bStangaSpate, INPUT\_PULLUP);

pinMode(rgStangaSpate, INPUT\_PULLUP);

pinMode(bDreaptaSpate, INPUT\_PULLUP);

pinMode(rgDreaptaSpate, INPUT\_PULLUP);

pinMode(faruri, OUTPUT);

pinMode(stopuri, OUTPUT);

Wire.begin(8);

Wire.onReceive(receiveEvent);

Serial.begin(9600);

}

În funcția loop() se setează valorile pentru pinii farurilor și a stopurilor folosind analogWrite. De asemenea, în funcție de variabile directie aprinde/stinge ledurile de semnalizare stânga, dreapta și în spate.

void receiveEvent(int howMany)

{

valueFaruri = Wire.read();

valueStopuri = Wire.read();

directie = Wire.read();

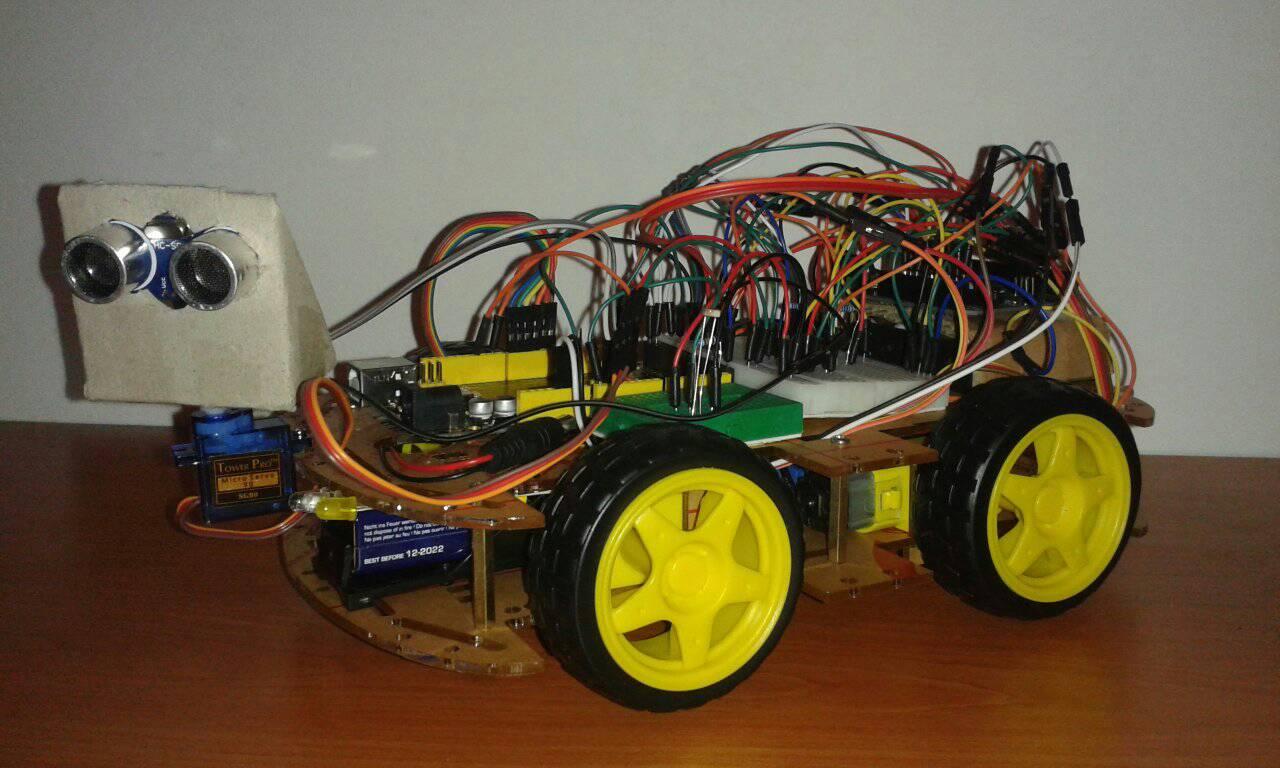
//Serial.println(value);

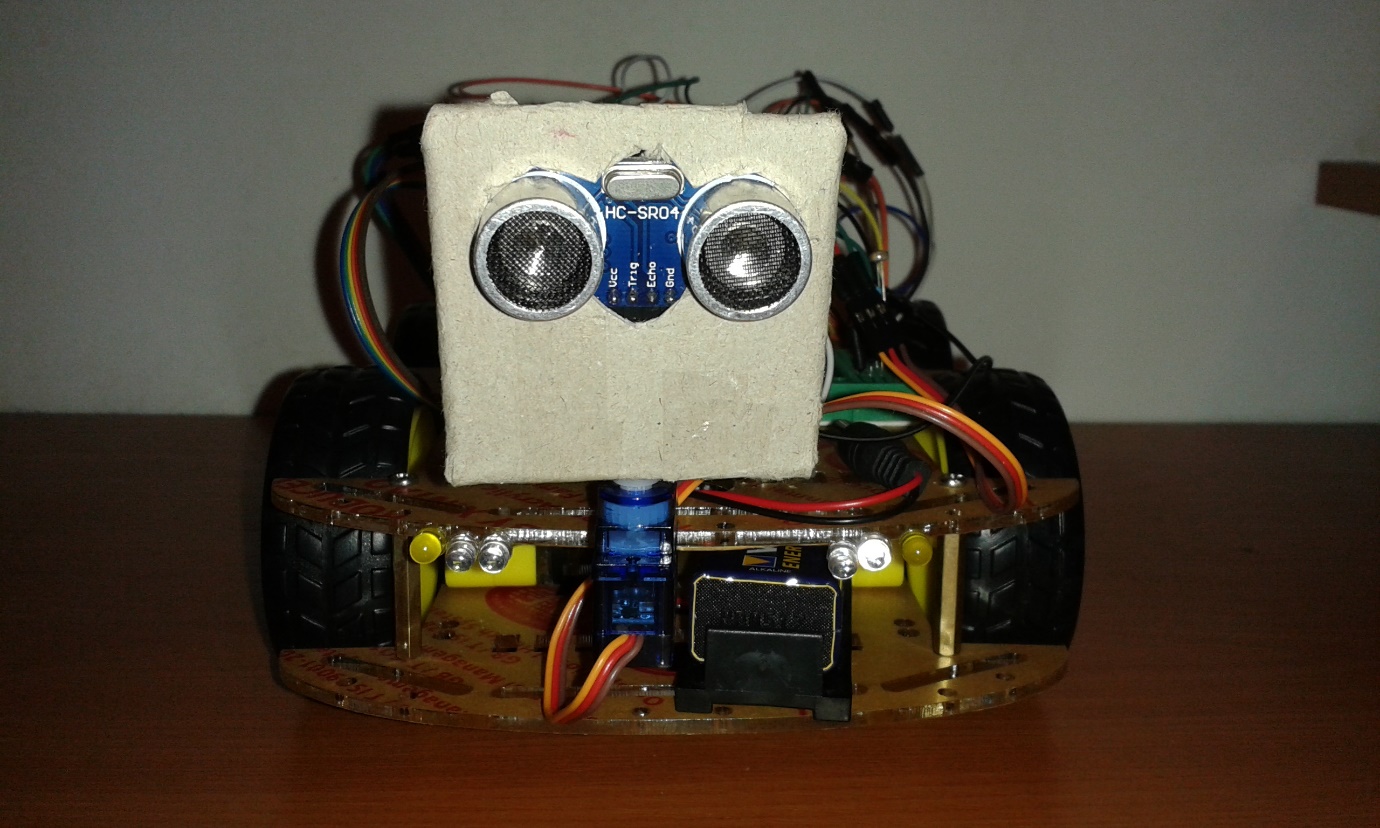
//Serial.println(directie);

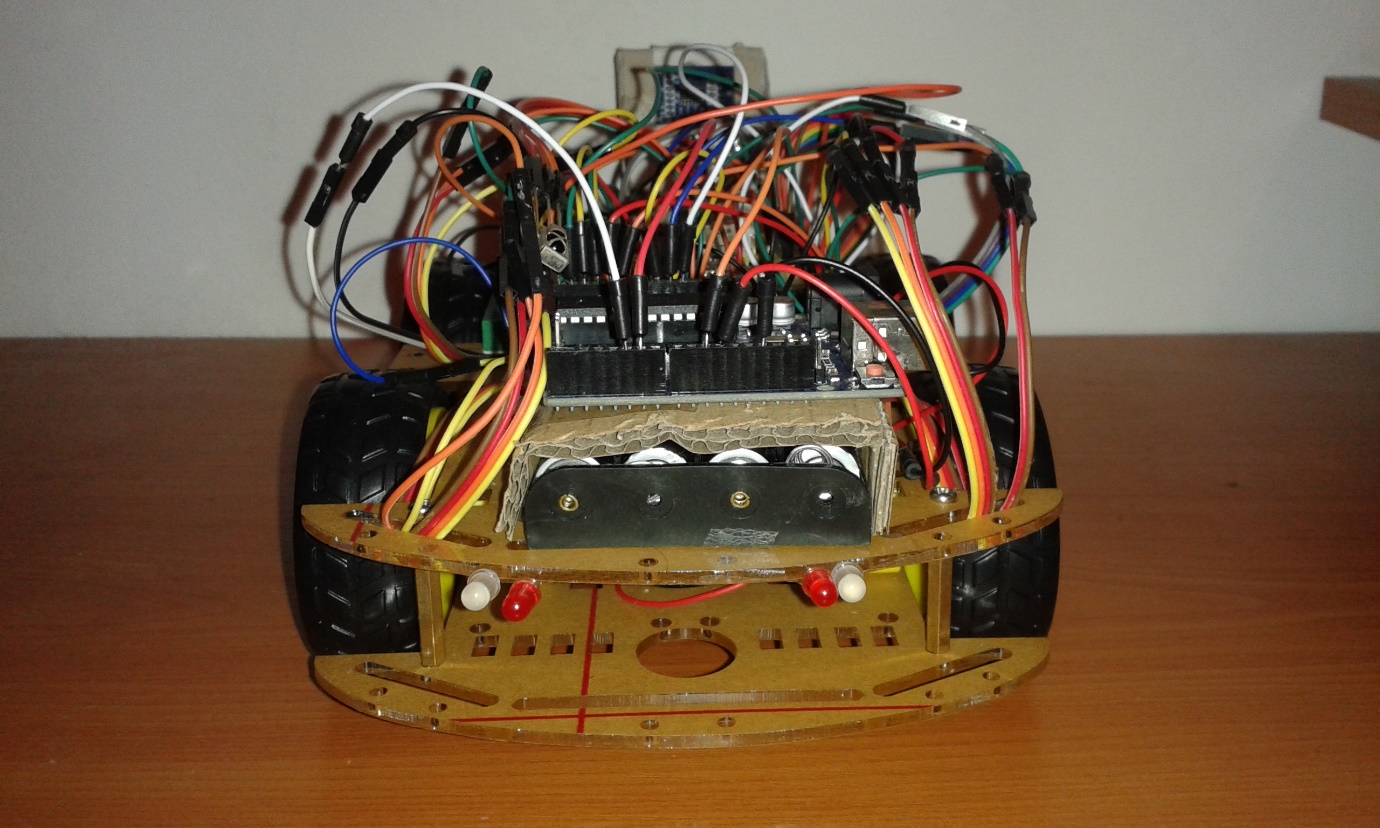
//Serial.println();

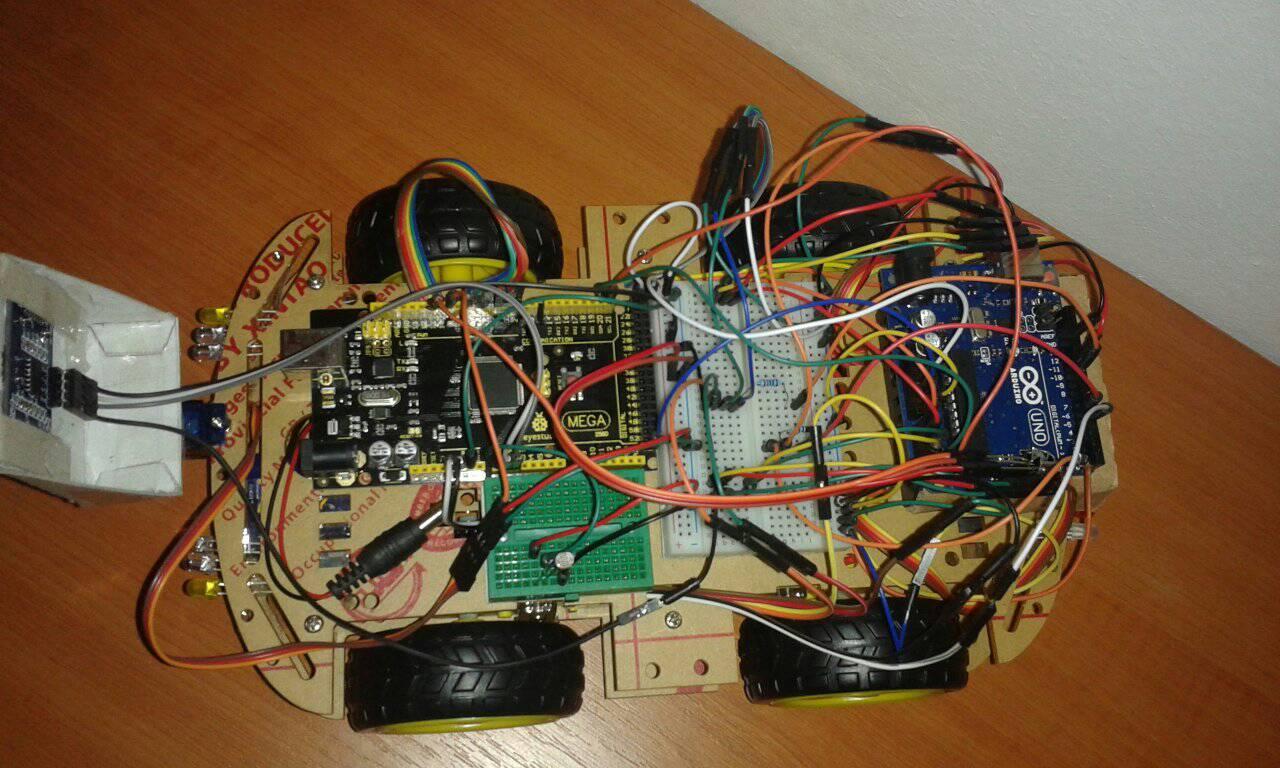
}

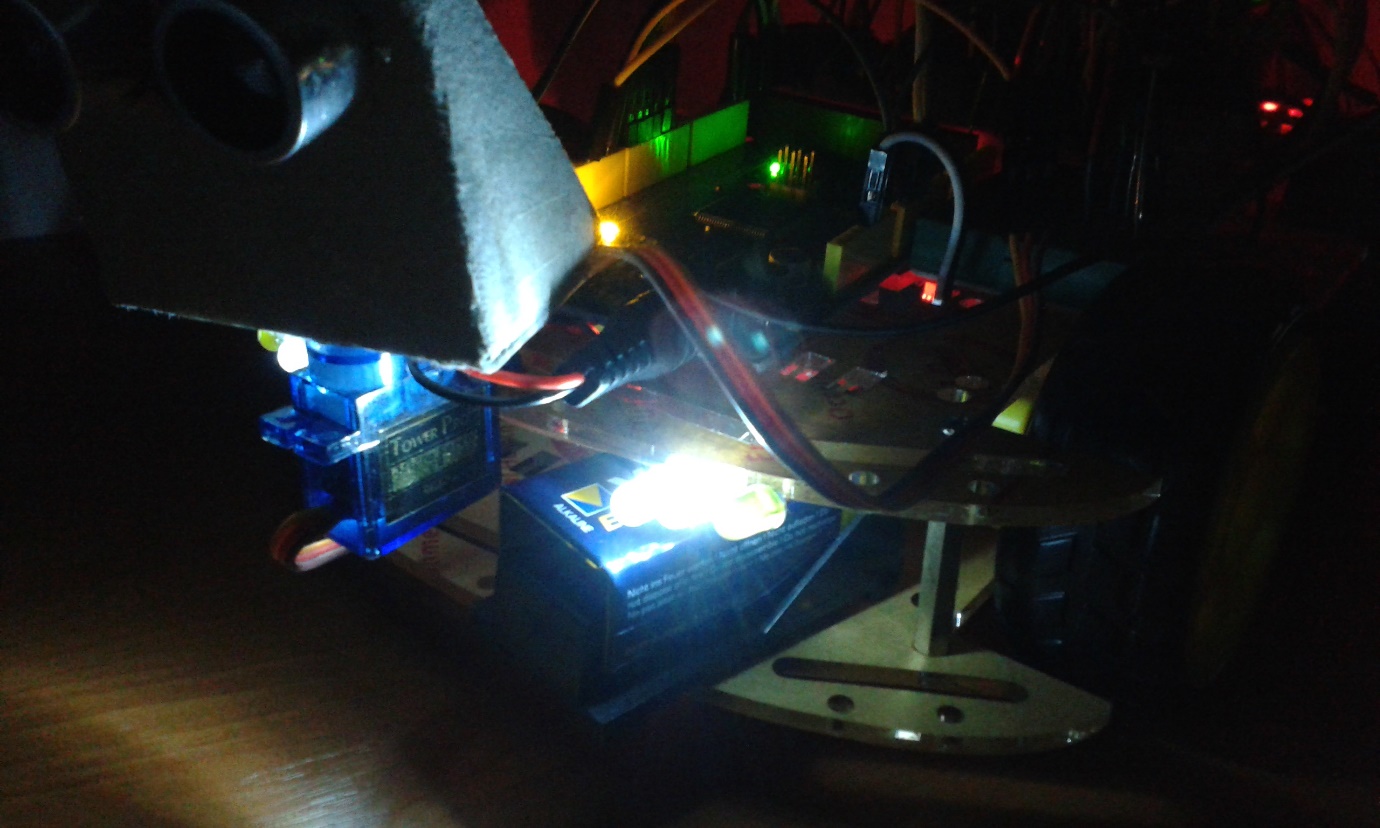
1. **Poze proiect**

****

****

****

****

****

****



1. **Posibilități de dezvoltare ulterioară**

Printre posibilitățile de dezvoltare ulterioară se numără:

* urmărirea unui traseu
* controlul mașinii de la distanță folosind un joystick/ecran touchscreen/ tastatura/ telefon mobil
* ghidare după semnele de circulație/ gesturile unui om/ voce
* crearea unei hărți a camerei în care se află mașinuța
* competiție între mașini pe un traseu (cu reguli din Formula 1)
* interfață grafică (panou de bord) pentru datelor achiziționate cât timp mașinuța este pornită
* urmărirea unui obiect de o anumită culoare
* detecția coliziunilor și semnalizarea acestora
* comunicarea prin Wi-Fi cu alte mașinuțe
* deplasare între două locații folosind GPS
* programare pe plăci Raspberry Pi/FPGA în loc de Arduino
* parcare lateral între 2 obiecte
* slalom printre obstacole

1. **Referințe**

<https://arduino-info.wikispaces.com/IR-RemoteControl>

<http://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/>