Facilitarea manevrelor de parcare a unui automobil: măsurarea distanței rămase până la un obstacol folosind un microcontroler și senzori de distanță

Sebö Diana Loredana

Stan Mirun Ameteo

Coordonator: as. dr. ing. Sergiu Nimară

Enunț:

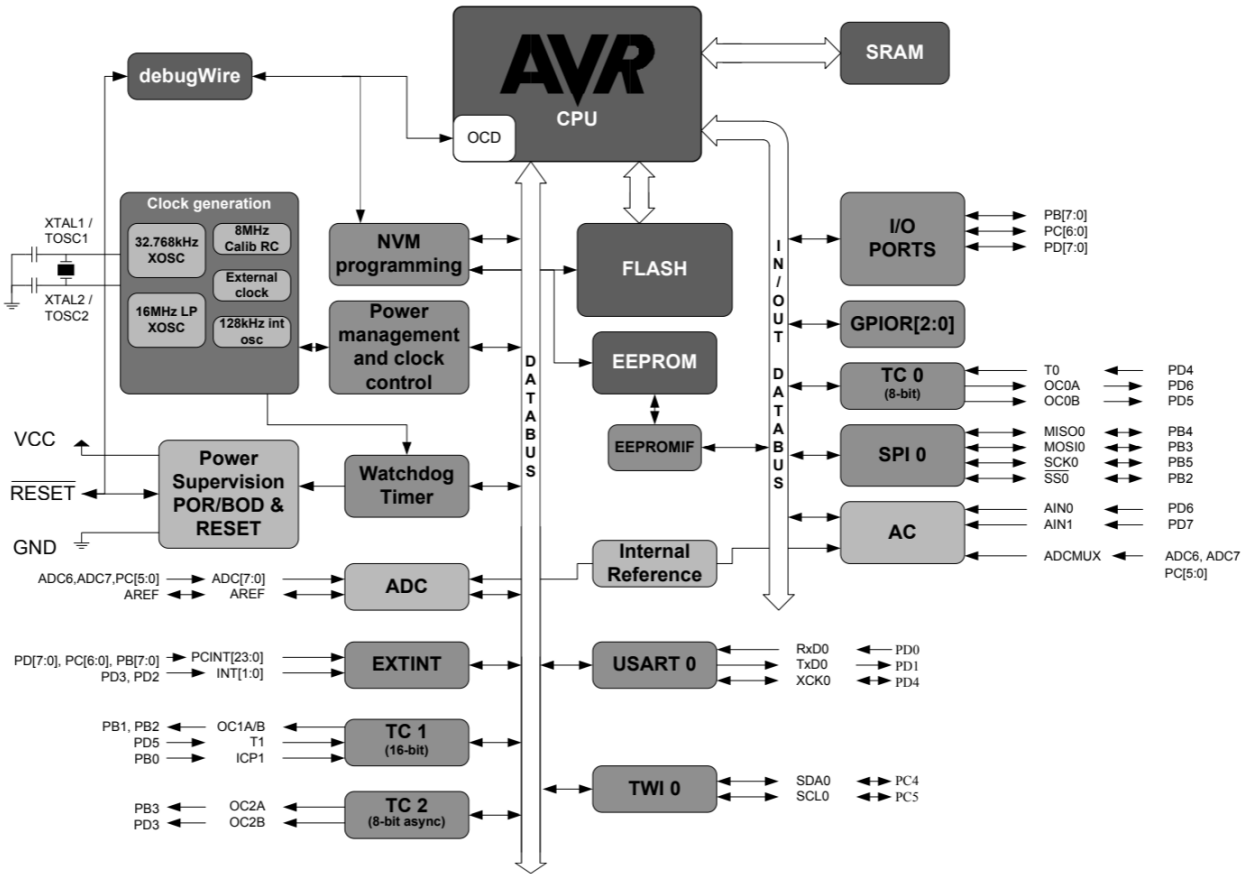
- Senzorii de distanţă se pot lega la o interfaţă serială, precum CAN, I2C sau SPI, la alegerea proiectantului;
- Se va măsura o distanţă cuprinsă între 10 şi 70 – 100 cm;
- Măsurarea distanţei va fi însoţită de o alarmă sonoră: frecvenţa sunetului va creşte pe masură ce distanţa până la obstacol scade;
- Valoarea distanţei măsurate va fi afişată, la alegerea studentului, pe afişaje cu 7 segmente sau matrice de LED-uri sau afişaj LCD;
- Se vor utiliza cel puțin 2 senzori de distanță, iar aplicația va avea un prototip practic.

Placa de dezvoltare

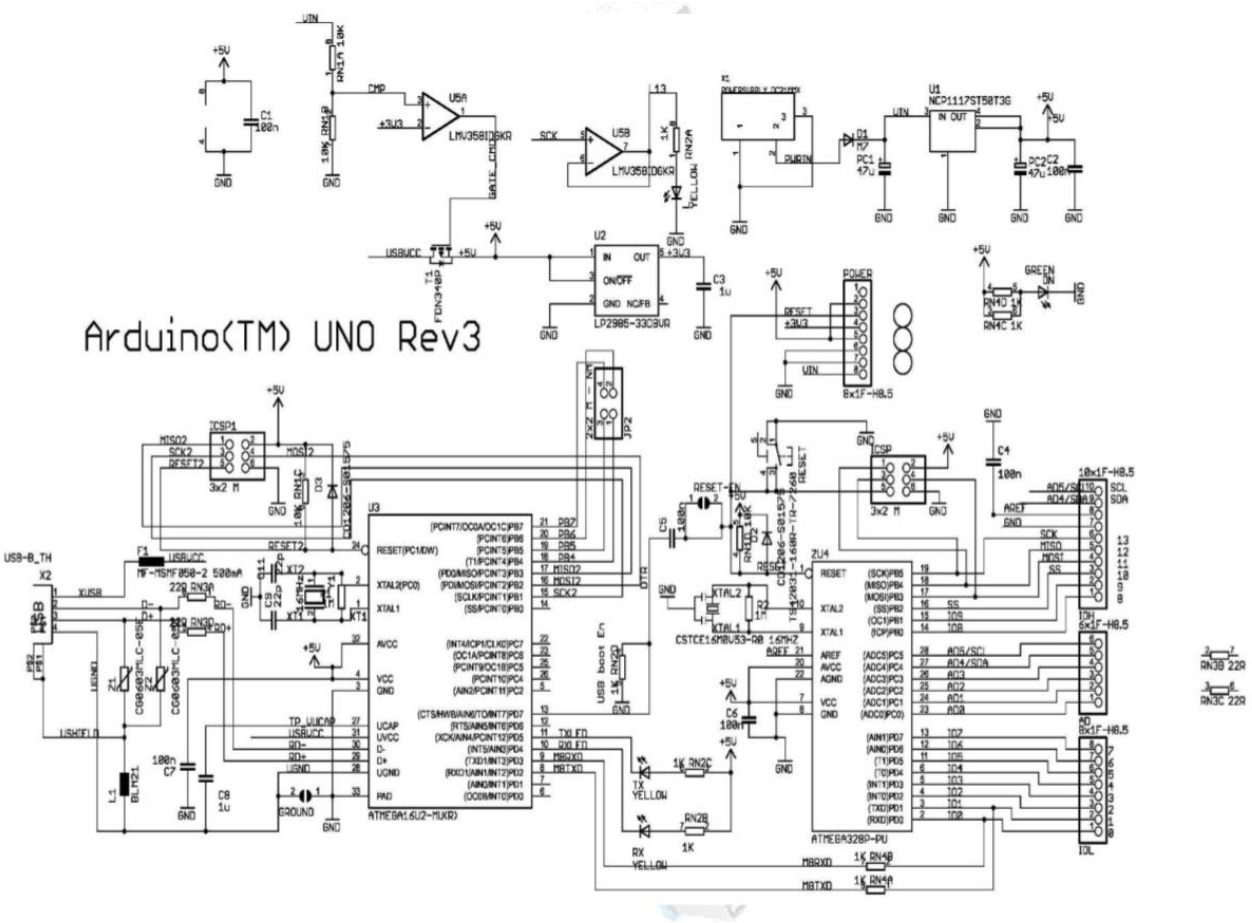
Se folosește o placă compatibilă cu Arduino Uno din revizia a 3-a. Această placă folosește microcontroller-ul ATMEGA328P-PU, pe 8 biti, care are următoarele caracteristici:

- 32 de registre cu scop general, conectate direct la ALU, permițând ca 2 registre independente să fie accesate într-o singură instrucțiune, executată într-un ciclu de tact;
- 32 kB de memorie flash programabilă, cu capacități de scriere în timpul citirii;
- 1 kB de memorie EEPROM;
- 2 kB de memorie SRAM,
- 23 de pini I/O;
- 3 numărătoare;
- Întreruperi externe și interne;
- O interfață serială programabilă USART etc.

1. Schema bloc a microcontrollerului ATMEGA328P-PU



2. Schema bloc a plăcii de dezvoltare Arduino UNO Rev3

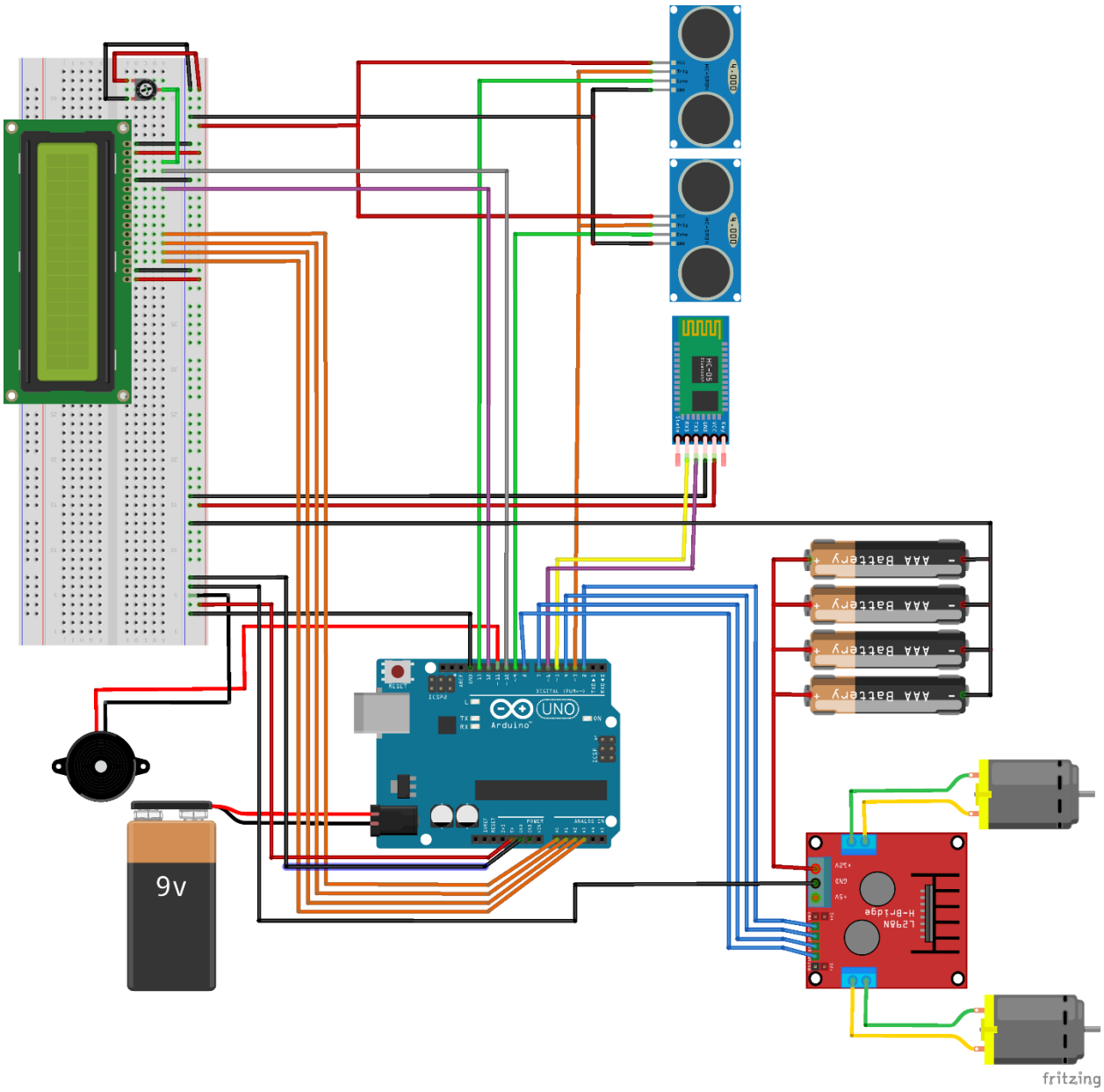


Fiecare dintre cei 14 pini digitali pot fi folosiți fie ca intrare, fie ca ieșire prin functiile *pinMode*, *digitalWrite* și *digitalRead*. Pinii operează la tensiunea de 5 volți și suportă un curent de maxim 40 de miliamperi. Unii pini oferă functionalități suplimentare:

- Serial: 0 (RX) și 1 (TX). Se folosesc pentru a primii (RX) și transmite (TX) date TTL în mod serial.
- Întreruperi externe: 2 și 3.
- PWM: 3, 5, 6, 9, 10 și 11. Permit ieșire PWM pe 8 biți prin funcția *analogWrite*.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Permit comunicare SPI prin biblioteca SPI.
- LED: 13.

Cei 6 pini analogici au o rezoluție de 10 biți (1024 de valori). Măsoara de la 0 la 5 sau chiar mai mulți volți. Maximul se poate modifica prin pinul AREF și funcția *analogReference*.

Module folosite



Interfața automobilului este o aplicație Android. Comunicarea cu placuța Arduino se face prin bluetooth. Se folosește un **modul bluetooth HC-05** și facilitatea plăcuței Arduino de a comunica serial cu modulul bluetooth.

Se folosesc doi **senzori de distanță HC-SR04** pentru a citii distanța pana la obstacole atât în față cât și în spate.

Pentru afișaj se folosește un **display LCD 1602A**.

Cele două motoare sunt controlate de către un **driver L298N**.

Senzor ultrasonic HC-SR04

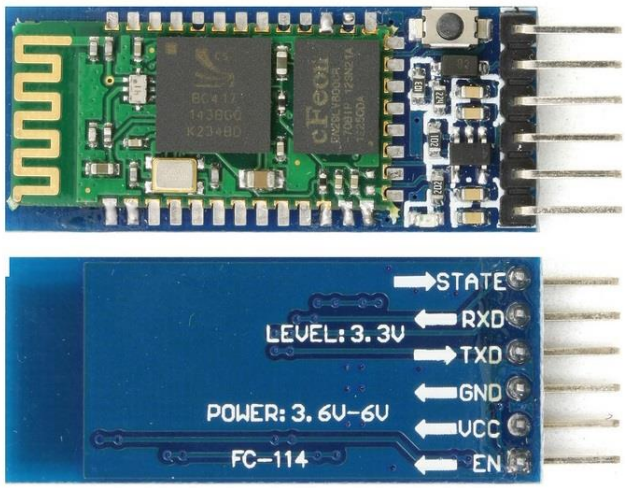
Senzorul măsoară distanțe între 2 și 400 cm, cu o precizie de 3 mm, având următorul mod de funcționare: La recepționarea unui semnal HIGH timp de 10 us pe pinul Trig, modulul transmite 8 semnale cu frecvența de 40 kHz și așteaptă întoarcerea lor. Dacă recepționează întoarcerea semnalelor, semnalul Echo este setat pe HIGH pentru durata de timp dintre transmiterea și recepționarea semnalelor ultrasonice.

Raspunsul senzorului pe pinul Echo este transformat în distanță astfel: Se citește durata de timp cât valoarea pinului Echo a fost HIGH și se înmulțește acestă valoare cu 2 *viteza sunetului* (340 m/s) / 2. Documentația sugerează înmulțirea duratei de timp în nanosecunde cu 58 pentru a obține un rezultat în centrimetri.

Pini:

- VCC: +5V
- Trig: Trigger (INPUT)
- Echo: Echo (OUTPUT)
- GND: Ground

Modul Bluetooth HC-05

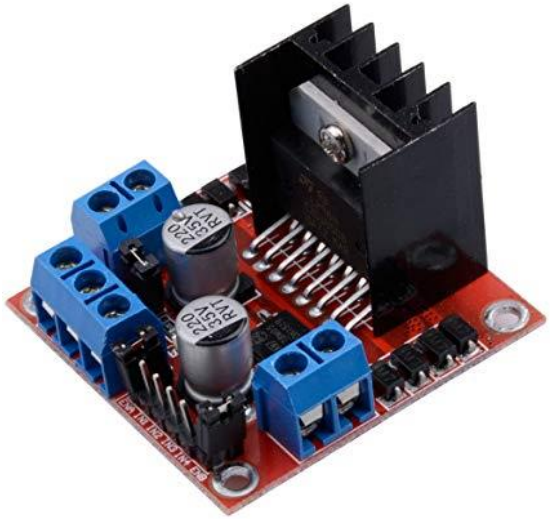


Modulul bluetooth este folosit pentru a recepționa comenzi. Comunicarea dintre acest modul si plăcuța Arduino se face prin conectarea pinilor RXD si TXD la 2 pini digitali și este facilitată de către biblioteca SoftwareSerial. Aceasta ne permite setarea a vitezei de comunicare și citirea următorului caracter.

Pini:

- VCC: +5V
- GND: Ground
- TXD: transmiterea serială (OUTPUT)
- RXD: recepționare serială (INPUT)

Driver L298N



Driverul de motoare L298N permite controlul două motoare DC printr-o interfață cu 14 pini. Controlul individual al motoarelor este dat de următorul tabel de adevăr:

IN1/IN3	IN2/IN4	Efect
LOW	LOW	Motorul este oprit
LOW	HIGH	Motorul merge înainte
HIGH	LOW	Motorul merge înapoi
HIGH	HIGH	Invalid

Driverul oferă și posibilitatea de modulare a turației pentru fiecare motor. Această funcție nu este folosită, motoarele funcționând mereu la turație maximă.

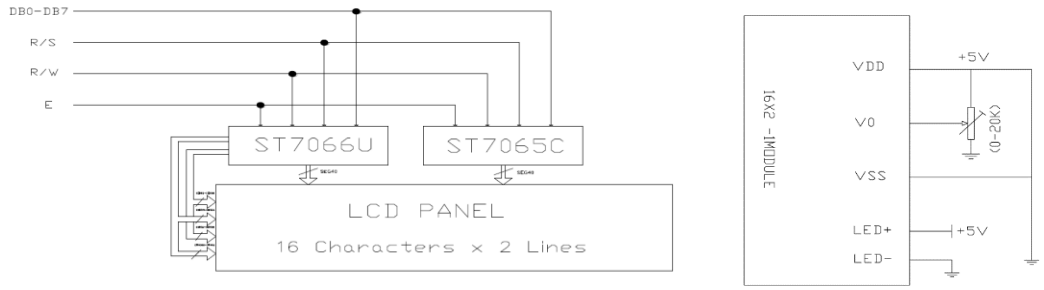
Pini:

- VCC (12v)
- VCC (5v)
- GND
- IN1 – control direcție pentru motorul A
- IN2 – control direcție pentru motorul A
- IN3 – control direcție pentru motorul B
- IN4 – control direcție pentru motorul B
- ENA – control turație pentru motorul A
- ENB – control turație pentru motorul B

Display LCD 1602A

Se folosește un display LCD 1602A, cu 2 linii a câte 16 caractere. Acest display este compatibil cu driverul Hitachi HD44780 folosit de către Arduino, lucru care face posibilă utilizarea bibliotecii LiquidCrystal pusă la dispoziție de către Arduino.

Implementarea se va folosi de modul de lucru cu 4 biți de date. De asemenea pinii E (Enable) și RS (Register Select) vor fi conectați direct la pinii digitali de date ai placuței Arduino. Întrucât nu se vor citii informații de la ecran, pinul RW (read/write) nu va fi folosit.



Coumunicarea ecranul prin intermediul bibliotecii LiquidCrystal se face foarte ușor. Biblioteca permite poziționarea pe rând și pe coloană și scrierea unui caracter sau a unui șir de caractere.

Pini:

- LED+ și LED- reprezintă alimetarea backlight
- VSS, VDD și V0 alimentează panoul LCD și permit ajustarea contrastului folosind un potențiometru
- E este intrarea de activare
- D0-7 sunt intrări de date
- RS permite selectarea modului de interpretare a informațiilor primite pe intrările de date (ca și instrucțiune sau ca și date)
- RW permite selectarea operatiei (citirea sau scriere unei valori)

Implementare

```
#include <SoftwareSerial.h>
#include <HCSR04.h>
#include <LiquidCrystal.h>

//declararea pinilor folosiți
int PIN_DRIVER_IN1 = 2;
int PIN_SENSOR_TRIG = 3;
int PIN_DRIVER_IN2 = 4;
int PIN_BLUETOOTH_TRANSMIT = 5;
int PIN_BLUETOOTH_RECEIVE = 6;
int PIN_DRIVER_IN3 = 7;
int PIN_DRIVER_IN4 = 8;
int PIN_SENSOR_FRONT = 9;
int PIN_LCD_RS = 10;
int PIN_BUZZER = 11;
int PIN_LCD_EN = 12;
int PIN_SENSOR_BACK = 13;
int PIN_LCD_D4 = A0;
int PIN_LCD_D5 = A1;
int PIN_LCD_D6 = A2;
int PIN_LCD_D7 = A3;


//instanțierea senzorilor folosind biblioteca HCSR04
HCSR04 sensors (PIN_SENSOR_TRIG, new int[2] { PIN_SENSOR_FRONT, PIN_SENSOR_BACK }, 2);
//instanțierea modulului Bluetooth folosind biblioteca SoftwareSerial
SoftwareSerial Bluetooth (PIN_BLUETOOTH_RECEIVE, PIN_BLUETOOTH_TRANSMIT);
//instanțierea modulului LCD folosind biblioteca LiquidCrystal
LiquidCrystal LCD (PIN_LCD_RS, PIN_LCD_EN, PIN_LCD_D4, PIN_LCD_D5, PIN_LCD_D6, PIN_LCD_D7);


char val; //= valoarea transmisă prin aplicația Android către modulul Bluetooth

void setup() {
  //inițializarea pinilor pentru driver-ul de motoare ca ieșire
  pinMode(PIN_DRIVER_IN1, OUTPUT);
  pinMode(PIN_DRIVER_IN2, OUTPUT);
  pinMode(PIN_DRIVER_IN3, OUTPUT);
  pinMode(PIN_DRIVER_IN4, OUTPUT);
  pinMode(PIN_BUZZER, OUTPUT);
  //începerea comunicării cu modulul Bluetooth, la baud rate de 9600
  Bluetooth.begin(9600);
  //inițializarea modulului LCD și stabilirea modului de utilizare ca având 2 linii a câte 16 coloane
  LCD.begin(16, 2);
  LCD.clear();
  //tipărirea pe ecran a mesajelor care indică senzorii
  LCD.print("FRONT: ");
  LCD.setCursor(0, 1);
```

```
  LCD.print(" BACK: ");
}

//transmiterea semnalelor corespunzătoare către motoare pentru mers înainte
void forward() {
  digitalWrite(PIN_DRIVER_IN1, HIGH);
  digitalWrite(PIN_DRIVER_IN2, LOW);
  digitalWrite(PIN_DRIVER_IN3, HIGH);
  digitalWrite(PIN_DRIVER_IN4, LOW);
}

//transmiterea semnalelor corespunzătoare către motoare pentru mers înapoi
void backward() {
  digitalWrite(PIN_DRIVER_IN1, LOW);
  digitalWrite(PIN_DRIVER_IN2, HIGH);
  digitalWrite(PIN_DRIVER_IN3, LOW);
  digitalWrite(PIN_DRIVER_IN4, HIGH);
}

//transmiterea semnalelor corespunzătoare către motoare pentru executarea unui viraj la dreapta
(roata din stânga merge în față iar cea din dreapta, în spate)
void right() {
  digitalWrite(PIN_DRIVER_IN1, LOW);
  digitalWrite(PIN_DRIVER_IN2, HIGH);
  digitalWrite(PIN_DRIVER_IN3, HIGH);
  digitalWrite(PIN_DRIVER_IN4, LOW);
}

//transmiterea semnalelor corespunzătoare către motoare pentru executarea unei mișcări circulare
la dreapta (roata din dreapta stă pe loc iar cea din stânga merge în față)
void circleRight() {
  digitalWrite(PIN_DRIVER_IN1, LOW);
  digitalWrite(PIN_DRIVER_IN2, LOW);
  digitalWrite(PIN_DRIVER_IN3, HIGH);
  digitalWrite(PIN_DRIVER_IN4, LOW);
}

//transmiterea semnalelor corespunzătoare către motoare pentru executarea unui viraj la stânga
(roata din dreapta merge în față iar cea din stânga, în spate)
void left() {
  digitalWrite(PIN_DRIVER_IN1, HIGH);
  digitalWrite(PIN_DRIVER_IN2, LOW);
  digitalWrite(PIN_DRIVER_IN3, LOW);
  digitalWrite(PIN_DRIVER_IN4, HIGH);
}

//transmiterea semnalelor corespunzătoare către motoare pentru executarea unei mișcări circulare
la stânga (roata din stânga stă pe loc iar cea din dreapta merge în față)
void circleLeft() {
  digitalWrite(PIN_DRIVER_IN1, HIGH);
  digitalWrite(PIN_DRIVER_IN2, LOW);
```

```
  digitalWrite(PIN_DRIVER_IN3, LOW);
  digitalWrite(PIN_DRIVER_IN4, LOW);
}

//declanșarea unui sunet cu frecvența de 800 Hz și durata de 300 ms
void honk() {
  tone(PIN_BUZZER, 800, 300);
}

//transmiterea semnalelor corespunzătoare către motoare pentru oprirea lor
void motorStop() {
  digitalWrite(PIN_DRIVER_IN1, LOW);
  digitalWrite(PIN_DRIVER_IN2, LOW);
  digitalWrite(PIN_DRIVER_IN3, LOW);
  digitalWrite(PIN_DRIVER_IN4, LOW);
}

//se testează valorile pinilor motoarelor pentru mers înainte, respectiv înapoi
int goingForward() {
  return digitalRead(PIN_DRIVER_IN1) == HIGH
    && digitalRead(PIN_DRIVER_IN2) == LOW
    && digitalRead(PIN_DRIVER_IN3) == HIGH
    && digitalRead(PIN_DRIVER_IN4) == LOW;
}

int goingBackwards() {
  return digitalRead(PIN_DRIVER_IN1) == LOW
    && digitalRead(PIN_DRIVER_IN2) == HIGH
    && digitalRead(PIN_DRIVER_IN3) == LOW
    && digitalRead(PIN_DRIVER_IN4) == HIGH;
}

//afișarea pe modulul LCD a unor spații ce au ca scop alinierea informațiilor
void printPadding(int distance) {
  if (distance < 10)
    LCD.print("  ");
  else if (distance < 100)
    LCD.print(" ");
}

void loop() {
  //preluarea informațiilor de la senzori și afișarea lor pe LCD
  int front_dist = sensors.dist(0);
  LCD.setCursor(7, 0);
  printPadding(front_dist);
  if (front_dist == 0)
    LCD.print("INFINIT");
  else {
    LCD.print(front_dist);
    LCD.print(" CM    ");
  }
```

```
  //se face o pauză înaintea citirii următorului senzor, pentru a nu citi o valoare eronată
  delay(50);

  int back_dist = sensors.dist(1);
  LCD.setCursor(7, 1);
  printPadding(back_dist);
  if (back_dist == 0)
    LCD.print("NaN");
  else {
     LCD.print(back_dist);
     LCD.print(" CM    ");
  }

  //dacă distanța până la un obstacol este mai mică de 25 cm și se merge în direcția obstacolului, se
claxonează și se opresc motoarele
  if (front_dist != 0 && front_dist < 25 && goingForward()) {
    honk();
    motorStop();
  }
  if (back_dist != 0 && back_dist < 25 && goingBackwards()) {
    honk();
    motorStop();
  }

  //se face o pauză înaintea citirii bluetooth-ului, pentru a nu citi o valoare eronată
  delay(100);

  //preluarea valorii de la modulul Bluetooth și alegerea, în funcție de valoare, a acțiunii de executat
  Bluetooth.listen();
  val = Bluetooth.read();
  if(val != -1) {
    Bluetooth.print("Received value: ");
    Bluetooth.println(val);
    if (val == 'w' || val == '8')
      if (front_dist != 0 && front_dist > 10)
        forward();
      else
        honk();
    else if (val == 'x' || val == '2')
      if (back_dist != 0 && back_dist > 10)
        backward();
      else
        honk();
    else if (val == 'a' || val == '4')
      left();
    else if (val == 'd' || val == '6')
      right();
    else if (val == 'q')
      circleLeft();
    else if (val == 'e')
```

```
      circleRight();
    else if (val == 'B')
      honk();
    else if (val == 's')
      motorStop();
  }
}
```

Bibliografie

- Documentația plăcii de dezvoltare Arduino UNO R3
- Documentația microcontrollerului ATMega328
- Documentația senzorului de distanță HCSR04
- Documentația ecranului LCD 1602A
- Documentația driverului pentru motoare Dual L298N
- Documentația modulului Bluetooth HC-05

Proiectul se poate regăsi și la adresa: *https://github.com/dianasebo/ProiectSI*