

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:

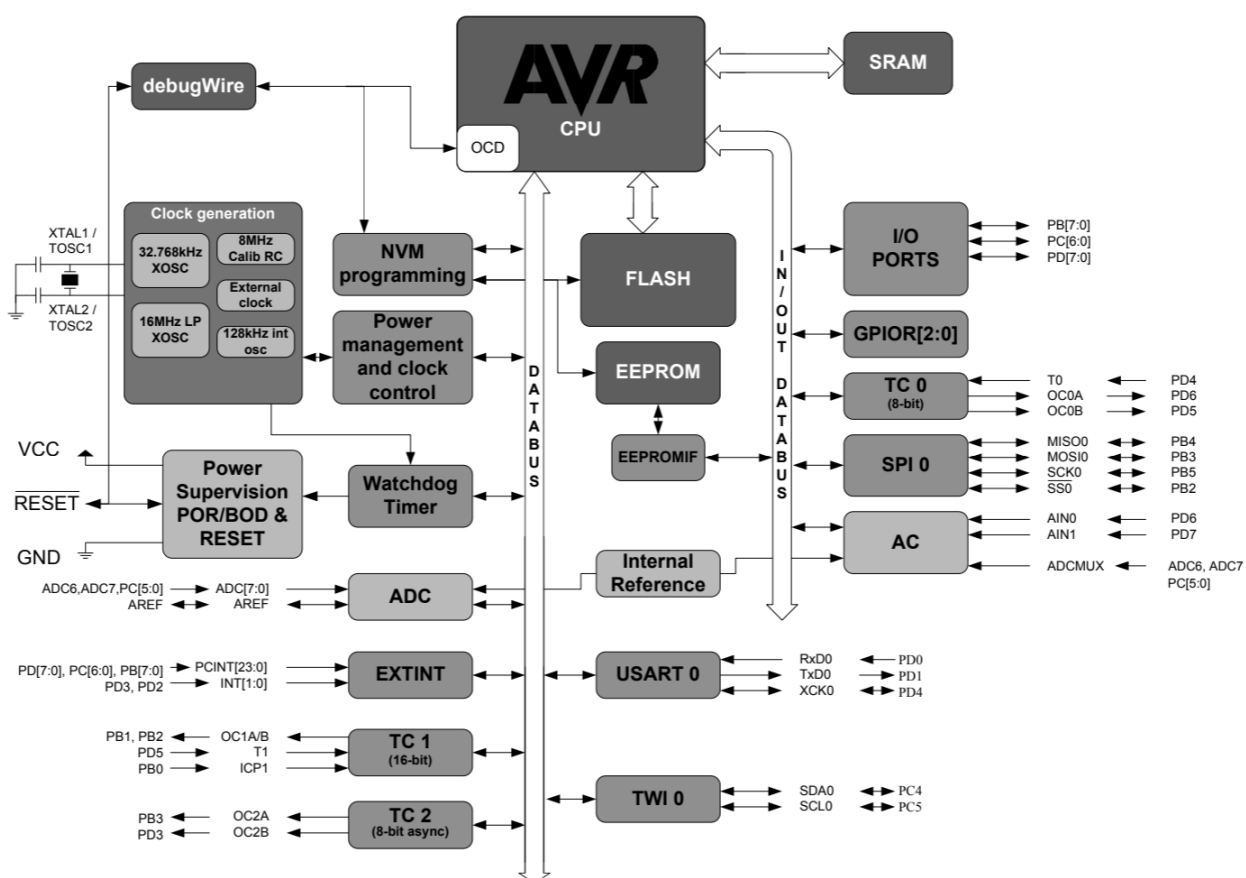
- Senzorii de distanță se pot lega la o interfață serială, precum CAN, I2C sau SPI, la alegerea proiectantului;
- Se va măsura o distanță cuprinsă între 10 și 70 – 100 cm;
- Măsurarea distanței va fi însoțită de o alarmă sonoră: frecvența sunetului va crește pe măsură ce distanța până la obstacol scade;
- Valoarea distanței măsurate va fi afișată, la alegerea studentului, pe afișaje cu 7 segmente sau matrice de LED-uri sau afișaj LCD;
- Se vor utiliza cel puțin 2 senzori de distanță, iar aplicația va avea un prototip practic.

Placa de dezvoltare

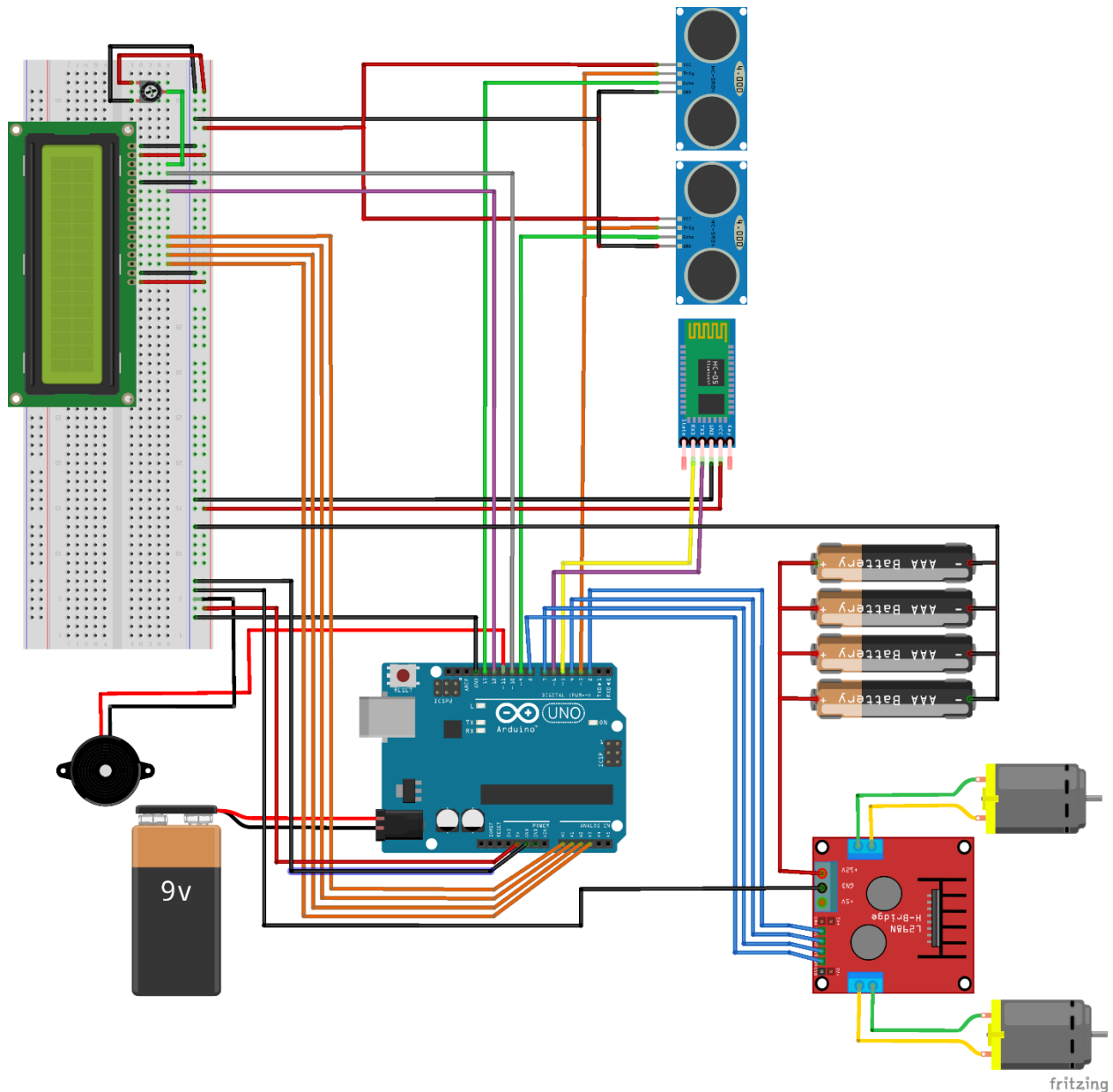
Se folosește o placă compatibilă cu Arduino Uno din revizia a 3-a. Această placă folosește microcontroller-ul ATMEGA328P-PU, pe 8 biti, care are următoarele caracteristici:

- 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



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.

Cei doi **senzori de distanță HC-SR04** permit citirea distanței măsurate prin transmiterea unui semnal digital de o anumită durată. Senzorul transmite plăcuței Arduino durata de timp între trimiterea semnalului și recepționarea lui la întoarcere. Se transformă durata de timp în distanță folosind soluția implementată de biblioteca HCSR04.

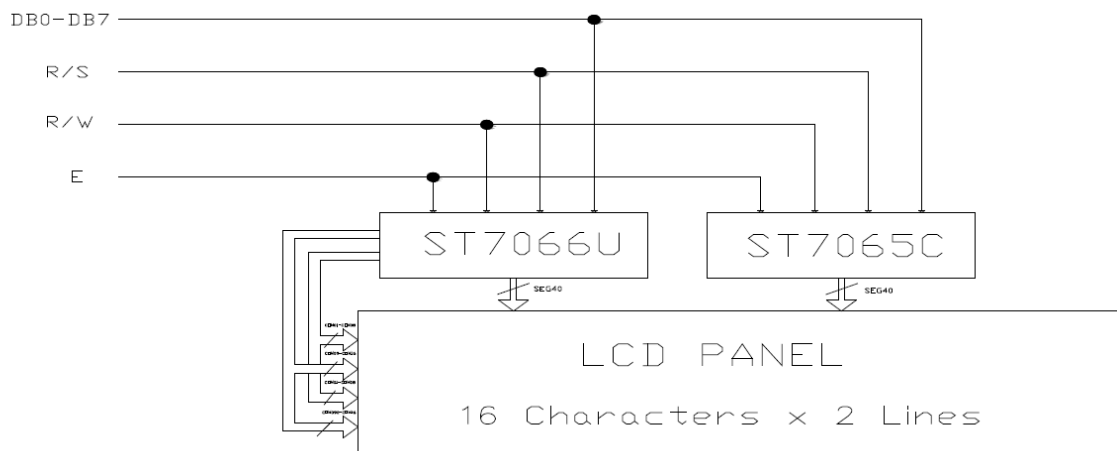
Pentru afișaj se va folosește un **display LCD 1602A**.

Cele două motoare sunt controlate de către un **driver L298N**. Modulul oferă facilități de modulare a semnalului transmis către motoare, dar acestea nu se folosesc.

Sistemul de afișaj

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 plăcuței Arduino.



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;

//instantiarea senzorilor folosind biblioteca HCSR04
HCSR04 sensors (PIN_SENSOR_TRIG, new int[2] { PIN_SENSOR_FRONT, PIN_SENSOR_BACK }, 2);
//instantiarea modului Bluetooth folosind biblioteca SoftwareSerial
SoftwareSerial Bluetooth (PIN_BLUETOOTH_RECEIVE, PIN_BLUETOOTH_TRANSMIT);
//instantiarea modului 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 modului 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