# Mașina de urmărire

## Introducere

Proiectul constă într-un robot autonom care urmărește persoane identificate prin intermediul unei aplicații OpenBot, care transmite date de la telefonul amplasat pe robot. Sistemul folosește tehnologii avansate pentru a detecta și urmări persoane, oferind asistență în diverse situații.

Scopul principal al proiectului este de a oferi un mijloc inovativ de a urmări și asista persoane în mișcare, fără a necesita control uman direct. Acest lucru este util pentru a reduce nevoia de personal suplimentar în anumite situații și pentru a oferi suport celor care au nevoie de ghidare sau asistență personalizată.

Ideea a pornit de la necesitatea de a găsi soluții automate pentru problemele cu care se confruntă persoanele cu mobilitate redusă sau cei care au nevoie de ghidare în spații mari. Cu un astfel de robot autonom, utilizatorii pot primi asistență personalizată, fără a depinde de personal uman în mod constant.

Prezentarea pe scurt a proiectului vostru:

- ce face
- care este scopul lui
- care a fost ideea de la care aţi pornit
- de ce credeţi că este util pentru alţii şi pentru voi

## Descriere generală

O schemă bloc cu toate modulele proiectului vostru, atât software cât şi hardware însoţită de o descriere a acestora precum şi a modului în care interacţionează.

Exemplu de schemă bloc: http://www.robs-projects.com/mp3proj/newplayer.html

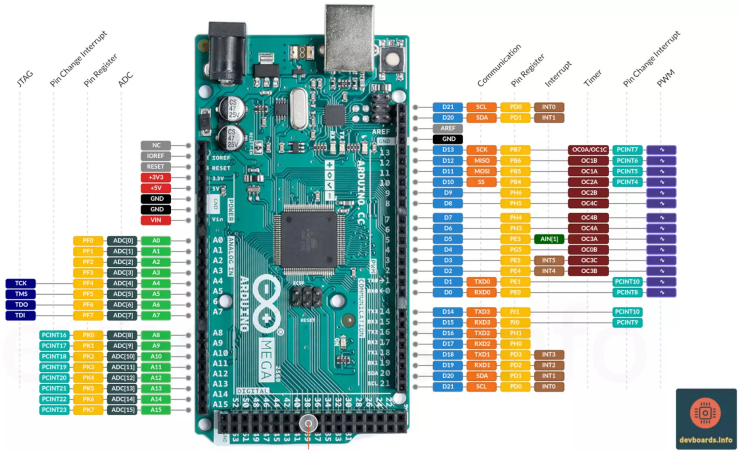
## Hardware Design

Componente:

- Arduino Mega2560 R3
- Modul driver de comanda motor cu L298N OKY3195
- 2x Modul de masurare viteza compatibil Arduino OKY3276-1
- Placa de stocare Micro SD TF Card reader Shield
- Display LCD 16×2 afisaj galben, cu modul I2C inclus
- Sasiul, ce contine 2 placi mari pentru masina, 4 motoreductoare cu ax dublu, 4 roti, 4 discuri pentru senzorii de viteza, suruburile si distantierele necesare, carcasa pentru 4 baterii
- Breadboard

⇨ ***Evidențierea componentelor si explicația folosirii pinilor acestora:***

1. Arduino Mega2560 R3:



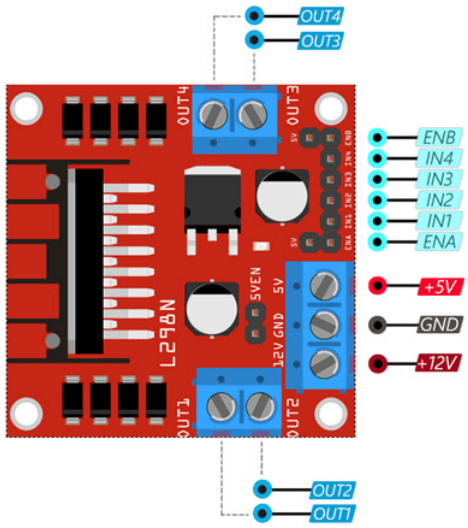

- D2, D3, D4, D5, D6, D7 → pini PWM utilizați pentru inputul driver-ului de motor, pentru a controla roțile mașinii;

- D20, D21 → pini de comunicare I2C SDA, respectiv SCL cu display-ul LCD 16×2;

- D45, D47, D49, D51 → pini pentru comunicarea cu modulul de card SD (MISO, MOSI, SCK, CS);

- D52, D53 → pini digitali, utilizați pentru inputul senzorilor de viteza;

2. Driver L298N:




- IN1 & IN2 - Pini de intrare a motoarelor I și II. Utilizare: Controlarea direcției de rotație a celor două motoare;

- IN3 & IN4 - Pini de intrare a motorului III și IV. Utilizare: Controlarea direcției de rotație a celor două motoare;

- ENA - Activează semnalul PWM pentru motoarele I și II;

- ENB - Activează semnalul PWM pentru motoarele III și IV;

- OUT1 & OUT2 - Pinii de ieșire ale motoarelor I și II;

- OUT3 & OUT4 - Pinii de ieșire ale motoarelor III și IV;

# Software Design

- **Mediu de dezvoltare**

Mediul de dezvoltare utilizat este Arduino IDE, o opțiune excelentă pentru începători datorită setup-ului său simplu și intuitiv.

- **Stadiul Actual al Implementării**

Proiectul a suferit multe probleme pe parcurs, iar din cauza gestionării proaste a timpului, scopul acestui proiect nu poate fi dus pana la capat. Însă, am încercat să folosesc componentele, pentru a face ceva mai simplu. Proiectul implementează tot ce mi-am propus, având următoarele funcționalități software:

- detectează și citește semnalele de la senzorii de viteză. În funcție de semnalele recepționate, calculează turațiile pe minut (RPM) ale motoarelor.

- utilizează un LCD I2C pentru a afișa RPM-ul motoarelor în timp real.

- un modul SD card este folosit pentru a salva într-un fișier text valorile RPM alături de timestamp-uri.

- un sistem de mișcare aleatorie pentru vehicul este controlat prin pini digitali și PWM pentru motoare.

- **Motivația Alegerii Bibliotecilor**

1. **LiquidCrystal_I2C.h**: Această bibliotecă este folosită pentru a controla afișajul LCD cu interfață I2C. Metodele de print, clear și setCursor sunt folosite intensiv pentru a afișa informații pe LCD.
2. **SD.h**: Biblioteca SD este folosită pentru a interacționa cu cardurile SD. Permite inițializarea cardului, deschiderea, citirea și scrierea fișierelor pe cardul SD. Este crucială pentru salvarea datelor de RPM.

- **Elementul de Noutate al Proiectului**

Având în vedere minusurile scopului principal al proiectului, unicitatea proiectului constă acum în integrarea unui sistem de monitorizare a turațiilor motoarelor în timp real, cu afișare pe LCD și salvare pe cardul SD, în timp ce vehiculul se mișcă aleatoriu. Aceasta adaugă un strat de complexitate și utilitate, permițând utilizatorilor să monitorizeze performanța motoarelor și să salveze datele pentru analize ulterioare.

- **Utilizarea Funcționalităților din Laborator în cadrul Proiectului**

1. **GPIO**: Cunoștințele generale din acest laborator au oferit baza pentru folosirea componentelor simple, precum pinii digitali pentru controlul motoarelor.
2. **PWM**: Cunoștințele din acest laborator au fost esențiale pentru înțelegerea controlului motoarelor folosind PWM, chiar și atunci când s-au folosit biblioteci ajutătoare.
3. **SPI**: Conectarea mai multor dispozitive folosind SPI a fost de ajutor în implementarea interfeței pentru SD card.
4. **I2C**: Modul în care dispozitivele master și slave comunică folosind doar două linii (SDA și SCL) a fost crucial pentru înțelegerea și implementarea controlului LCD-ului I2C.

- **Explicații Detaliate ale Funcțiilor**

1. **setup()**: se inițializează mai întâi comunicarea serială la o viteză de 9600 baud pentru debugging. Apoi, se configurează pinii pentru motoare ca ieșiri utilizând pinMode() și se setează pinii senzorilor de viteză 1 și 2 ca intrări cu pinMode(SPEED_SENSOR_1, INPUT) și pinMode(SPEED_SENSOR_2, INPUT). Funcțiile de întrerupere sunt atașate la senzorii de viteză folosind attachInterrupt() pentru a contoriza impulsurile detectate. Afișajul LCD este inițializat și iluminarea de fundal este activată cu lcd.init() și lcd.backlight(). Viteza inițială a motoarelor este setată prin intermediul funcției analogWrite(), iar generatorul de mișcări aleatoare este inițializat prin randomSeed(). În final, cardul SD este inițializat cu SD.begin(SD_CS_PIN).
2. **loop()**: se verifică dacă a trecut o secundă folosind millis() - lastMoveTime > 1000, pentru a putea da altă comandă de mișcare mașinii. Se calculează apoi turațiile pe minut (RPM) pentru fiecare motor, bazate pe impulsurile contorizate, stocate în variabilele rpm1 și rpm2. Contorii de impulsuri pulseCount1 și pulseCount2 sunt resetați la zero pentru următoarea măsurătoare. Valorile RPM sunt afișate pe LCD cu ajutorul funcției lcd.print(). Funcția writeRPMToSD(rpm1, rpm2) este apelată pentru a salva valorile RPM pe cardul SD. O acțiune aleatorie pentru mișcarea vehiculului este aleasă folosind random(). În funcție de acțiunea aleasă, switch(action) execută mișcarea corespunzătoare: înainte, înapoi, stânga sau dreapta.
3. **countPulse1() și countPulse2()**: incrementează contorii de impulsuri la fiecare impuls detectat de senzorii de viteză.
4. **writeRPMToSD()**: se deschide fișierul “rpm_data.txt”, iar valorile RPM sunt scrise în acest fișier text cu ajutorul funcțiilor dataFile.print() și dataFile.println(). La final, fișierul este închis pentru a finaliza operația de scriere.
5. **moveForward(), moveBackward(), turnLeft() și turnRight()**: stările pinilor sunt setate

utilizând digitalWrite(). Aceste funcții permit controlul direcției motoarelor, facilitând mișcarea vehiculului înainte, înapoi, virarea la stânga sau la dreapta.

## Rezultate Obţinute

Dovada funcționării componentelor:

https://youtube.com/shorts/A4ZfONLVZZQ?feature=shared

Mișcarea mașinii:

https://youtube.com/shorts/dyA15-3WLrQ?feature=shared

Afișarea RPM-ului primit de la senzori pe ecranul LCD

https://youtu.be/UrJFegWzkhE?feature=shared

## Concluzii

- Ideea proiectului a fost una interesantă, dar complexă, dar totuși, prin intermediul laboratoarelor, am reușit până la urmă să duc la bun sfârșit ceva funcțional, chiar dacă scopul proiectului nu a fost atins. Fiind prima experiență, mă declar mulțumit pentru cât am încercat.
- Am avut multe probleme cu anumite cabluri de la motoare care nu reușeau să stea în driverul de motor, am pierdut mult timp cu această problemă, deoarece de fiecare dată mi întâmpla să nu îmi meargă toate motoarele. Soluția optimă era să mai cumpăr un driver L298n și o carcasă de 8 baterii AA, doar că târziu am realizat și timpul nu a fost de partea mea.

## Download

Codul sursă:

**george_vlad.simion_332cb_pm_proiect.zip**

## Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

# Bibliografie/Resurse

### Resurse Hardware:

- https://www.electroniclinic.com/lm393-speed-sensor-with-arduino-using-l9110-motor-driver-circuit-and-code-explained/
- https://projecthub.arduino.cc/arduino_uno_guy/i2c-liquid-crystal-displays-5eb615
- https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/
- https://www.instructables.com/Micro-SD-Card-Tutorial/

### Resurse Software:

- https://www.arduino.cc/reference/en/libraries/liquidcrystal-i2c/
- https://www.arduino.cc/reference/en/libraries/sd/

### Laboratoare:

- https://ocw.cs.pub.ro/courses/pm/lab/lab0-2023
- https://ocw.cs.pub.ro/courses/pm/lab/lab3-2023-2024
- https://ocw.cs.pub.ro/courses/pm/lab/lab5-2023-2024
- https://ocw.cs.pub.ro/courses/pm/lab/lab6-2023-2024

Export to PDF