Document de specificații proiectare

# 1. Introducere

Proiectul "Sistem inteligent pentru recunoașterea semnelor de circulație utilizând machine learning" își propune dezvoltarea unei soluții hardware-software capabile să identifice și să clasifice semnele de circulație în timp real. Aceasta este o componentă esențială în sistemele avansate de asistență a șoferului (ADAS) și în vehiculele autonome.

## 1.1 Scopul documentului

Scopul prezentului document este de a descrie specificațiile tehnice, arhitecturale și funcționale ale proiectului. Se urmărește asigurarea unei baze solide pentru dezvoltarea, testarea și evaluarea sistemului, luând în considerare cerințele de performanță, securitate și scalabilitate ale aplicației embedded ML.

# 2. Prezentare generală și abordări de proiectare

## 2.1 Prezentare generală

Sistemul se bazează pe o platformă hardware Arduino Mega integrată cu senzori și module de captură, precum și un model de machine learning dezvoltat cu Create ML pentru clasificarea semnelor rutiere. Modelul antrenat este apoi transferat pe un microcontroller sau un dispozitiv edge (ex: Raspberry Pi) pentru inferență locală.

## 2.2 Presupuneri / Constrângeri / Riscuri

- Presupuneri: Sistemul funcționează cu imagini clare captate de camere. Setul de date este etichetat corect. Utilizatorii folosesc sistemul în condiții normale de trafic.

- Constrângeri: Limitările hardware ale plăcii Arduino, memoria redusă, latențe posibile în procesarea imaginii.

- Riscuri: Clasificări eronate din cauza condițiilor de iluminare, confuzii între semne similare, probleme de alimentare sau conectivitate.

# 3. Considerații de proiectare

## 3.1 Obiective și linii directoare

Obiectivele proiectului includ:  
- Identificarea precisă a semnelor de circulație folosind rețele neuronale convoluționale.  
- Funcționarea în timp real pe un sistem embedded.  
- Cost redus și scalabilitate.  
- Ușurință în integrarea cu sisteme de navigație sau alte subsisteme ale vehiculului.

## 3.2 Metode de dezvoltare

Metoda aleasă este una iterativă, cu pași de prototipare și testare frecventă. Se folosesc:   
- Arduino pentru controlul hardware.  
- Create ML pentru antrenarea modelelor.  
- Xcode pentru testare și validare.  
- Teste de performanță în condiții diverse (zi, noapte, vreme nefavorabilă).

## 3.3 Strategii de arhitectură

Sistemul este compus dintr-un modul hardware (Arduino Mega + senzori + cameră), și un model ML antrenat pe macOS și exportat pentru rulare pe un dispozitiv de procesare locală (ex: Raspberry Pi). Comunicarea între componente se face prin semnale electrice, iar modelul ML clasifică imaginile captate de cameră.

# 4. Arhitectura Sistemului și Proiectarea Arhitecturii

## 4.1 Vedere logică

Fluxul general este:  
1. Captarea imaginii cu semne de circulație.  
2. Preprocesarea imaginii (dacă este cazul).  
3. Clasificarea cu modelul ML.  
4. Returnarea rezultatului și afișarea sa sau transmiterea către un sistem de alertare.

## 4.2 Arhitectură hardware

Componenta hardware include:  
- Arduino Mega pentru controlul senzorilor și motoarelor.  
- Cameră video pentru captarea imaginilor.  
- Senzori suplimentari (ultrasonic, IR) pentru detectarea obstacolelor.  
- Modul de afișare (LCD sau LED-uri pentru output vizual).

## 4.3 Arhitectură software

Software-ul este împărțit în două:  
- Codul C++ pentru Arduino (control hardware).  
- Modelul ML antrenat cu Create ML pe macOS, exportat pe platforma de testare (macOS/RPi).

# 5. Proiectarea sistemului

## 5.1 Proiectarea bazei de date

Fiind un sistem embedded, nu se utilizează o bază de date relațională complexă. Datele relevante (precum clasificările și numărul de detecții) pot fi salvate temporar în memoria dispozitivului sau transmise prin serial/USB către un sistem superior pentru analiză.

## 5.2 Conversii de date

Imaginile captate de cameră pot fi convertite din format RGB în grayscale pentru optimizarea inferenței. De asemenea, datele brute pot fi normalizate și redimensionate conform cerințelor modelului ML.

## 5.3 Interfețe utilizator

Sistemul poate include:  
- Afișaj LCD pentru vizualizarea semnului detectat.  
- LED-uri pentru semnalizare vizuală (culoare roșie pentru semn STOP, galben pentru ATENȚIE, etc).  
- Semnale sonore prin buzzer în cazuri critice (STOP, interzis, etc).

## 5.4 Proiectarea interfețelor cu utilizatorul

Interacțiunea este minimă și intuitivă. Utilizatorul poate porni/opri sistemul, iar acesta afișează automat semnele detectate. Se pot implementa butoane pentru testare sau resetare, precum și un mod de calibrare (ex. setare sensibilitate cameră).

# 6. Scenarii de utilizare

Scenariu 1: Vehiculul se apropie de o intersecție cu semnul STOP. Sistemul detectează semnul și emite un semnal sonor + vizual.  
Scenariu 2: Sistemul detectează semnul 'Cedează Trecerea' în condiții de lumină slabă. Clasificarea este reușită și apare mesajul corespunzător.  
Scenariu 3: În condiții meteo nefavorabile (ceață), sistemul întâmpină dificultăți dar reușește totuși identificarea cu o acuratețe scăzută.

# 7. Proiectare de detaliu

## 7.1 Proiectare hardware de detaliu

Hardware-ul include:  
- Arduino Mega 2560: microcontrolerul principal, cu suficiente I/O pentru senzori și module.  
- Cameră video: conectată la un Raspberry Pi (sau alt edge device) pentru procesarea imaginii.  
- Senzor ultrasonic HC-SR04: pentru detecția obstacolelor.  
- Servomotor MG90S: pentru simularea unui sistem de reacție (ex. braț care semnalizează).  
- LED-uri și buzzer pentru feedback vizual și auditiv.  
- Alimentare: sursă de 5V stabilizată pentru Arduino și accesorii.

## 7.2 Proiectare software de detaliu

Software-ul este format din două componente:  
- Cod Arduino (C++): gestionează senzori, primește comenzi de la modulul ML și generează acțiuni (aprindere LED, semnal acustic).  
- Model ML (Create ML + Swift): antrenat pe macOS folosind seturi de date etichetate cu semne rutiere. Modelul este exportat pe un dispozitiv edge (ex. Raspberry Pi cu CoreML sau tflite) pentru inferență.

## 7.3 Proiectare detaliată de securitate

Sistemul fiind local, nu necesită măsuri avansate de securitate de rețea. Totuși, sunt luate în considerare:  
- Protejarea accesului la fișierul modelului ML.  
- Izolarea sursei de alimentare pentru a evita interferențe sau defecțiuni hardware.  
- Filtrarea datelor de intrare (imaginile) pentru a evita inputuri corupte sau anormale.

## 7.4 Proiectare de detaliu pentru performanța sistemului

Sistemul este testat pentru:  
- Acuratețe: modelul ML trebuie să atingă minim 90% pe setul de test.  
- Timp de inferență: maxim 1 secundă de la captarea imaginii până la reacție.  
- Stabilitate: funcționare continuă minim 1 oră fără erori.  
- Robusteză: sistemul trebuie să funcționeze în medii cu lumină variabilă.

## 7.5 Proiectare detaliată a comunicațiilor interne (între componente)

Comunicarea se face astfel:  
- Camera -> Modul ML: captură imagine + procesare pe edge (ex: Raspberry Pi).  
- Modul ML -> Arduino: prin USB/serial, trimite codul semnului recunoscut.  
- Arduino -> LED/Buzzer/Servo: reacție la semnul detectat.  
- Toate componentele sunt sincronizate cu timpi bine definiți pentru latență minimă.

# 8. Diagrama de arhitectură a sistemului

Diagrama arhitecturală poate fi descrisă textual astfel:  
  
1. Captură imagine:  
- Cameră video capturează o imagine a mediului rutier.  
  
2. Procesare imagine:  
- Imaginea este preluată de un dispozitiv edge (ex. Raspberry Pi), unde este procesată cu ajutorul modelului ML exportat din Create ML.  
  
3. Clasificare semn:  
- Modelul ML clasifică imaginea și returnează eticheta semnului detectat (ex. STOP, Cedează, etc).  
  
4. Transmitere rezultat:  
- Rezultatul este trimis prin USB/serial către placa Arduino Mega.  
  
5. Reacție sistem:  
- Arduino aprinde LED-urile, emite semnale sonore sau controlează servo în funcție de semnul detectat.

# 9. Concluzii și perspective de dezvoltare

Sistemul dezvoltat demonstrează aplicabilitatea tehnologiilor de machine learning în medii embedded, cu scopul creșterii siguranței rutiere. Prin utilizarea platformei Arduino și a antrenării modelului pe Create ML, proiectul reușește să ofere o soluție scalabilă, accesibilă și eficientă.  
  
Rezultatele obținute arată o rată bună de recunoaștere în scenarii reale, iar timpii de reacție sunt adecvați pentru utilizare în vehicule sau sisteme de testare. În plus, costul redus și flexibilitatea permit integrarea în diverse aplicații de tip ADAS sau vehicule autonome educaționale.  
  
Perspectivele de dezvoltare includ:  
- Extinderea setului de semne recunoscute.  
- Îmbunătățirea modelului cu date augmentate și condiții diverse.  
- Migrarea modelului ML pe hardware dedicat AI (ex. Coral TPU).  
- Integrarea cu GPS și sisteme de hartă pentru context geografic.  
- Implementarea unei aplicații mobile pentru monitorizarea în timp real a rezultatelor.