Radar cu 2 senzori ultrasonici, alerta si afisaj

Introducere

Punctul de plecare a fost problema vitezi excesive de pe drumurile publice din Romania, una dintre principalele cauze ale accidentelor grave de circulatie.

Acest proiect a fost conceput pentru a masura viteza medie a masinilor pe un anumit segment de drum si declanseaza o alarma sonora daca viteza legala este depasita. Sistemul activeaza primul senzor ultrasonic si porneste un timer in momentul in care un vehicul este detectat. Cand vehiculul ajunge la cel de-al doilea senzor, ce este amplasat la o distanta stiuta, timer-ul se opreste si se calculeaza viteza medie pe baza timpului si a distantei dintre senzori. Aceasta este afisata pe un ecran LCD prin intermediul modulului I2C. Daca viteza masurata depaseste o limita prestabilita (configurata manual), un buzzer emite un semnal de alarma, sunetul creste si descreste pe parcursul a cateva secunde.

Proiectul este util deoarece poate fi amplasat pe orice segment de drum pentru colectarea datelor reale despre viteza si a semnala contraventiile din trafic. Acesta incurajeaza soferii sa respecte viteza legala prin feedback-ul audio imediat.

Descriere generală

Sistemul este compus din urmatoarele module hardware si software:

Module Hardware

- **ATmega328P Xplained Mini** - placa de microcontroler principala ce coordoneaza toata logica, masoara timpul si comunicatia
- **Senzorul ultrasonic 1** - amplasat la inceputul segmentului de drum masurat, detecteaza cand un vehicul intra in zona si porneste timer-ul prin intrerupere externa
- **Senzorul ultrasonic 2** - amplasat la finalul segmentului masurat, detecteaza cand vehiculul iese si opreste timer-ul prin intrerupere externa
- **Buzzer pasiv** - emite un semnal de alarma generat prin PWM cand viteza legala este depasita, intensitatea sunetului creste si scade pe parcursul a cateva secunde
- **LCD 16×2 cu modulul I2C** - afiseaza viteza calculata si statusul sistemului in timp real, conectata prin interfata TWI.
- **Breadboard + fire jumper** - folosite pentru fiecare conexiune intre componente

Module Software

- **Handler intreruperi** - configureaza intreruperile externe pe pinii Echo ai ambilor senzori. Cand senzorul 1 semnaleaza, timer-ul incepe masurarea. Cand senzorul 2 semnaleaza, timer-ul se opreste si se foloseste timpul scurs.
- **Calcul viteza** - calculeaza viteza medie ca: v = distanta/timp, unde distanta este separarea fizica intre cei 2 senzori (este constanta)
- **Control buzzer PWM** - daca viteza depaseste pragul prestabilit, generam un semnal PWM pe pinul buzzer-ului. Duty cycle-ul este modulat in timp pentru a crea un efect de alarma care creste si apoi scade.
- **Driver LCD I2C** - foloseste perifericul TWI al ATmega328P pentru a trimite datele de viteza catre modulul I2C conectat la LCD, formatate ca “Viteza: XX km/h”.

Hardware Design

Lista componente

Componenta	Model / Specificatii	Cantitate
Placa microcontroller	ATmega328P Xplained Mini	1
Senzor ultrasonic de distanta	HC-SR04	2
Afisaj LCD	LCD 16×2	1
Modul I2C pentru LCD	CEY4005	1
Buzzer pasiv	Compatibil 3.3V-5V	1
Breadboard	830 puncte	1
Fire jumper	Male-to-male, culori diverse	~20
Fire jumper mama-tata	Female-to-male, culori diverse	4
Cablu USB	Micro-USB	1
Sursa de alimentare	5V prin USB sau externa	1

Componente folosite

Componenta	Rol in proiect
ATmega328P Xplained Mini	Microcontroller principal - proceseaza semnalele de la senzori, calculeaza viteza si controleaza afisajul si buzzer-ul
HC-SR04 #1 (Senzor 1)	Senzor ultrasonic - detecteaza trecerea unui obiect prin primul punct de masurare
HC-SR04 #2 (Senzor 2)	Senzor ultrasonic - detecteaza trecerea aceluiasi obiect prin al doilea punct de masurare
LCD 16×2 cu modul I2C (LCD_I2C)	Afiseaza mesajele “Speed Radar” si viteza calculata in km/h
Buzzer pasiv	Semnalizare sonora la detectarea unui obiect
Breadboard + fire jumper	Interconectarea componentelor

Pinii folositi

Pin ATmega328P	Eticheta	Componenta	Rol
PB2	PB2	Buzzer	Semnal digital pentru activarea buzzer-ului
PD2	ECHO1	HC-SR04 #1	Receptie semnal ECHO de la senzorul 1 (input)
PD3	ECHO2	HC-SR04 #2	Receptie semnal ECHO de la senzorul 2 (input)
PD4	TRIG1	HC-SR04 #1	Trimitere puls TRIG catre senzorul 1 (output)
PD5	TRIG2	HC-SR04 #2	Trimitere puls TRIG catre senzorul 2 (output)
PC4	SDA	LCD I2C	Linie de date I2C pentru LCD
PC5	SCL	LCD I2C	Linie de ceas I2C pentru LCD
VCC	+5V	Toate componentele	Alimentare 5V
GND	GND	Toate componentele	Masa comuna

Am ales pinii ECHO pe PD2/PD3 deoarece corespund intreruperilor externe INT0 si INT1 ale ATmega328P Xplained Mini, permitand capturarea precisa a timpului semnalelor ultrasunetelor prin intreruperi hardware, fara polling.

Comunicatia I2C (SDA/SCL pe PC4/PC5) pentru a putea transmite semnalele.

Schema electrica

Nu am gasit in KiCad placa ATmega328P Xplained Mini si am folosit echivalentul ATmega328P-P

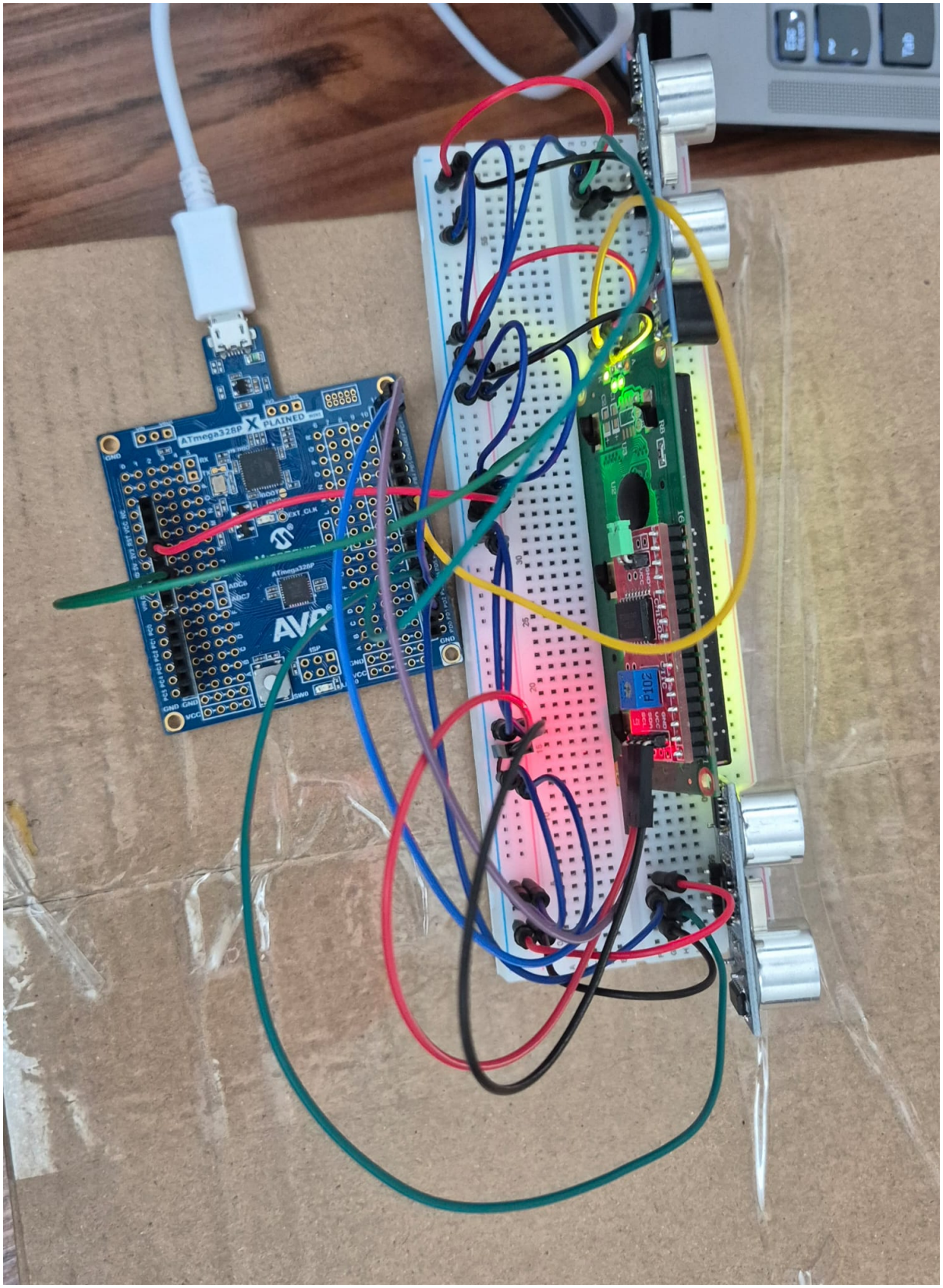
Schema prezinta:

- **ATmega328P Xplained Mini** in centru, ca unitate de control
- **HC-SR04 #1** conectat pe pinii TRIG1 (PD4) si ECHO1 (PD2) - reprezinta primul senzor de viteza
- **HC-SR04 #2** conectat pe pinii TRIG2 (PD5) si ECHO2 (PD3) - reprezinta al doilea senzor de viteza, distanta este plasata la distanta pe care o cunoastem
- **LCD I2C** conectat pe magistrala I2C (SDA=PC4, SCL=PC5) cu alimentare de 5V
- **Buzzer** connectat pe PB2 pentru activare si alimentarea este de 5V + ground

Principiu de functionare

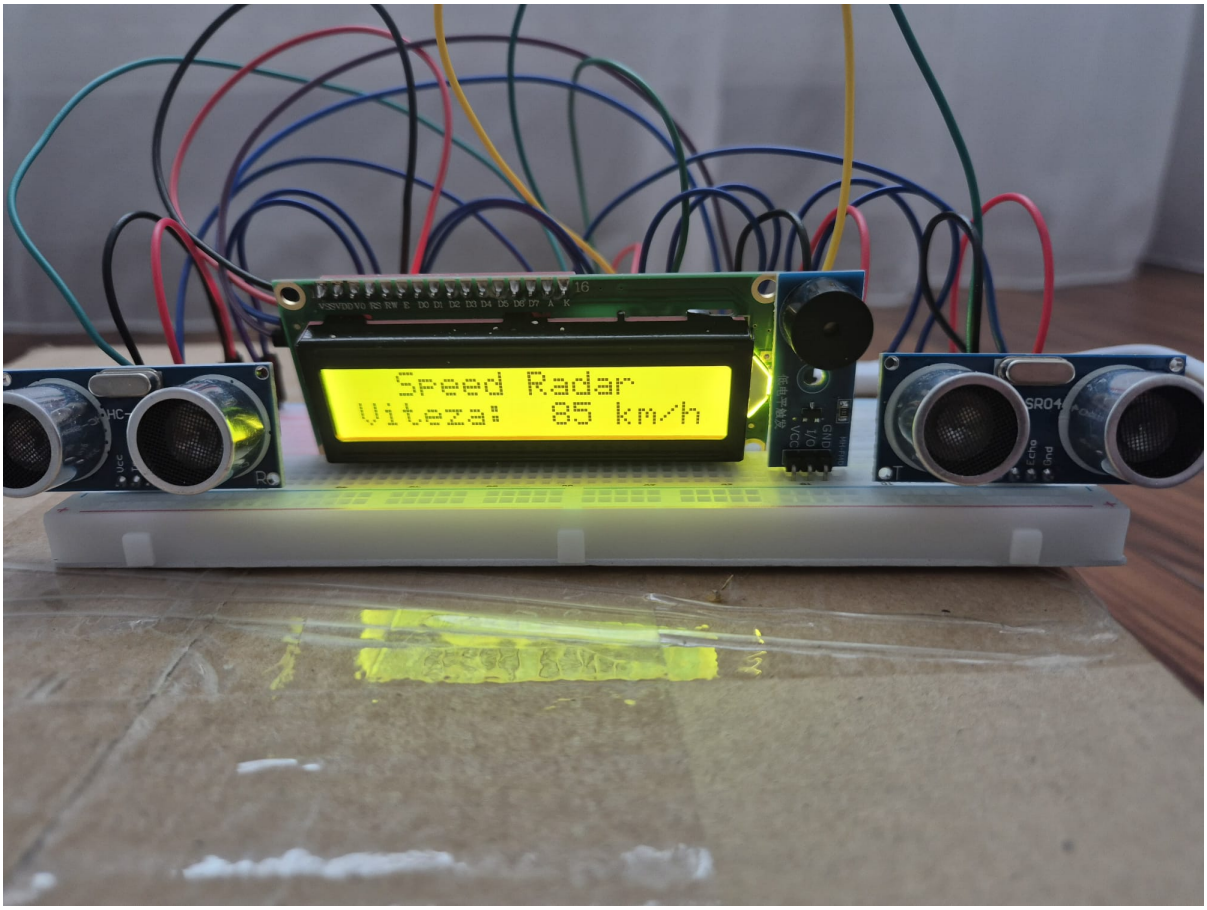
Cei 2 senzori HC-SR04 sunt plasati la o distanta fixa si cunoscuta **d** (in cm) pe acelasi ax. Cand un obiect trece prin deptul primului senzor, microcontroller-ul porneste un cronometru. Cand acelasi obiect trece prin dreptul celui de-al doilea senzor, cronometrul se opreste si calculeaza viteza.

Imagini Hardware



Imaginea de mai sus prezinta cum am legat cablurile. Cei 2 senzori plasati la capetele unui suport rigid, LCD-ul cu modulul I2C in centru si buzzer-ul atasat lateral. LCD-ul afiseaza “Speed Radar / Viteza

XX km/h”.



In imaginea de mai sus se observa clar cum LCD-ul prin I2C reuseste sa afiseze viteza corespunzatoare calculata.

In imaginea de mai sus se observa cum senzorul 1 a detectat un obiect, a trimis semnal, dureaza pana ajunge obiectul la senzorul 2. Cand obiectul a ajuns la senzorul 2 acesta il detecteaza, trimite semnal iar microcontroller-ul calculeaza viteza, trimite catre I2C sa o afiseze pe LCD. Executia se opreste pentru cateva secunde, iar apoi incepem iar detectarea pe senzorul 1 prin resetare.

Software Design

Mediu de dezvoltare

Proiectul a fost dezvoltat folosind **PlatformIO** cu framework-ul **Arduino** pe platforma **atmelavr**, pentru microcontrollerul ATmega328P Xplained Mini la 16 MHz.

Fisierul de configurare platformio.ini defineste:

- upload prin xplainedmini protocol
- monitor serial la 9600 baud pe /dev/ttyACM0
- flag de compilare -DF_CPU=16000000UL si standard -std=gnu99

Biblioteci folosite

Proiectul **nu foloseste biblioteci third-party**. Toate driverele sunt implementate de la zero direct pe registrele AVR:

Biblioteca AVR-libc	Rol in proiect
avr/io.h	Acces la registrele hardware (TCCR1A, TWCR, PORTD, etc.)
avr/interrupt.h	Definirea ISR-urilor si activarea intreruperilor globale (sei())
util/atomic.h	Citire atomica a variabilei g_uptime_ms din contextul main (ATOMIC_BLOCK)
util/delay.h	Delay-uri precise pentru pulsul Trig al HC-SR04 si initializarea LCD-ului
stdlib.h	Conversie numere la string (utoa, ultoa) pentru afisare UART si LCD

Motivatia alegerii: Am dorit sa impelementez totul de la zero deoarece este mult mai usor pentru intelegerea algoritmilor si a placutei.

Elementul de noutate

Fata de proiectele clasice de masurare a distantei cu un singur senzor, acest proiect introduce:

- **Masurarea vitezei medii** pe un segment de drum fix, folosind doi senzori independenti si un timer hardware de 16 biti, fara un singur senzor
- **Driver TWI/I2C custom** scris complet de la zero, cu functii proprii: TWI_init, TWI_start, TWI_stop, TWI_write
- **Driver LCD 16×2 prin PCF8574** implementat manual cu protocolul 4-bit si pulsarea semnalului Enable
- **Alarma PWM cu efect gradat**, duty cycle-ul buzzerului variaza continuu intre 30% si 70% din ISR-ul Timer2, creand un sunet gradat, nu un bip fix
- **Masina de stari** explicita cu 3 stari (STATE_WAIT_S1, STATE_WAIT_S2, STATE_HOLD) pentru gestionarea fluxului de detectie

Justificarea utilizarii functionalitatilor din laborator

Lab 2 - Intreruperi: Timer2 este configurat in mod overflow cu prescaler 64, generand o intrerupere la fiecare ~1ms (ISR(TIMER2_OVF_vect)). Aceasta intrerupere indeplineste doua roluri: incrementeaza cronometrul global g_uptime_ms (folosit pentru masurarea intervalului dintre senzori) si actualizeaza duty cycle-ul buzzerului cand alarma e activa. Variabila g_uptime_ms este declarata volatile si citita cu ATOMIC_BLOCK pentru a preveni race conditions.

Lab 3 - Timere si PWM: Buzzerul pasiv este conectat pe pinul PB2 (OC1B al Timer1). Timer1 este configurat in **Fast PWM mode 14** (TOP=ICR1) cu prescaler 8, generand o frecventa de ~2kHz potrivita pentru buzzer. Se foloseste **modul inverting** (COM1B1+COM1B0) deoarece buzzerul este activ-LOW - pinul e HIGH la BOTTOM si scade la match OCR1B, astfel buzzerul suna cand OCR1B < ICR1. Pornirea/oprirea alarmei se face prin conectarea/deconectarea OC1B din TCCR1A, fara a opri timer-ul.

Lab 6 — I2C (TWI): Comunicatia cu modulul LCD (PCF8574 @ adresa 0x27) este implementata

manual prin registrele TWCR, TWDR, TWSR, TWBR. Secventa respecta protocolul I2C: START → adresa slave cu bit W → date → STOP. TWBR=72 cu TWSR=0 seteaza viteza la ~100kHz (I2C standard). LCD-ul primeste comenzi si date in format 4-bit prin expandorul PCF8574, fiecare nibble fiind trimis cu pulsarea semnalului Enable.

Scheletul proiectului si interactiunea dintre functionalitati

Proiectul este organizat intr-un singur fisier main.c cu urmatoarea structura:

```
/* Initializari (in main) */
gpio_init()      -> seteaza pinii Trig (output) si Echo (input cu pull-up)
USART_init()     -> configureaza UART la 9600 baud pentru debug
TWI_init()       -> activeaza pull-up intern pe SDA/SCL, seteaza TWBR
LCD_init()       -> secventa de reset 4-bit + comenzi de configurare LCD
timer2_init()    -> Timer2 overflow ~1ms, activeaza TOIE2
buzzer_init()    -> Timer1 Fast PWM mode 14, ICR1=999 (2kHz), pin HIGH
initial
sei()            -> activeaza intreruperi globale

/* Bucla principala - masina de stari (FSM) */
STATE_WAIT_S1:
  sonar_cm(S1) -> daca d < 15cm:
    t1_ms = uptime_ms()
    trece in STATE_WAIT_S2

STATE_WAIT_S2:
  sonar_cm(S2) -> daca d < 15cm:
    delta_ms = uptime_ms() - t1_ms
    kmh = (SENSOR_DIST_CM * 36 * 10) / delta_ms
    afiseaza pe LCD si UART
    daca kmh > SPEED_LIMIT_KMH -> alarma = 1
    trece in STATE_HOLD
  daca timeout 10s -> reset in STATE_WAIT_S1

STATE_HOLD (5 secunde):
  alarm_effect_update(alarma) -> buzzer_start() sau buzzer_stop()
  dupa 5s -> reset in STATE_WAIT_S1
```

Formula de calcul viteza:

```
v [km/h] = (distanta [cm] * 36) / timp [ms]

Derivare:
  v = d/t = (d_cm / 100) / (t_ms / 1000) [m/s]
  v_kmh = v_ms * 3.6
  => v_kmh = (d_cm * 36) / t_ms
```

In cod se foloseste *10 si impartire intreaga pentru a obtine o zecimala fara float:

```
uint32_t kmh = ((uint32_t)SENSOR_DIST_CM * 36UL * 10UL) / delta_ms;
// kmh_int  = kmh / 10   (partea intreaga)
// kmh_frac = kmh % 10   (prima zecimala)
```

Calibrarea senzorilor

Senzorii HC-SR04 au fost calibrati prin urmatoarele ajustari:

- **Pragul de detectie** DETECT_CM = 15 a fost ales experimental: la distante mai mari, senzorii detectau zgomot de fond sau obiecte din mediu; la 15cm este suficient de sensibil pentru un obiect care trece prin fata senzorului
- **Timeout sonar:** functia sonar_cm() are doua garduri: 10000us pentru asteptarea Echo HIGH si 30000us pentru durata pulsului Echo - valorile corespund la ~5m distanta maxima, eliminand blocajele la obiecte absente
- **Timeout radar:** daca S2 nu detecteaza in 10 secunde dupa S1, sistemul se reseteaza automat - evitand blocarea in STATE_WAIT_S2 cand un obiect trece doar prin fata primului senzor
- **Distanta fizica** SENSOR_DIST_CM = 15 este constanta configurabila - se ajusteaza in functie de montajul real al senzorilor

Optimizari realizate

1. **Fara floating point:** Calculul vitezei foloseste exclusiv aritmetica pe intregi (uint32_t), reducand semnificativ dimensiunea binarului si timpul de executie pe AVR
2. **Fara delay-uri blocking in ISR:** ISR-ul Timer2 este minimal (incrementare + calcul OCR1B) - nu contine delay-uri sau operatii I2C, prevenind latenta in masurarea timpului
3. **ATOMIC_BLOCK pentru variabile volatile:** g_uptime_ms este pe 32 biti; pe AVR (8-bit) citirea sa nu este atomica nativ, deci ATOMIC_BLOCK(ATOMIC_RESTORESTATE) previne citirea unui valor intermediar corupt
4. **Modul inverting PWM** pentru buzzer activ-LOW: in loc sa inversam logic in software la fiecare ciclu, configuram hardware-ul sa faca asta (COM1B0+COM1B1), reducand overhead-ul
5. **Pull-up intern pe pinii Echo:** PORTD |= (1«S1_ECHO_PIN) elimina necesitatea rezistentelor externe de pull-up

Validarea functionalitatilor

Fiecare modul a fost validat separat inainte de integrare:

- **UART debug:** toate evenimentele importante sunt loggate serial (S1=Xcm, S2! delta=Xms, Viteza: X.X km/h, Reset) - verificabile cu monitorul serial din PlatformIO
- **Sonar individual:** testat cu obiecte la distante cunoscute, verificand ca sonar_cm() returneaza valori corecte inainte de integrarea in FSM
- **LCD:** testat independent cu mesaje fixe pentru a verifica initializarea I2C si afisarea corecta pe ambele randuri
- **Buzzer PWM:** testat cu buzzer_start() apelat manual, verificand efectul de “respiratie” al sunetului
- **FSM integrata:** testata cu obiecte miscate manual prin fata senzorilor, verificand tranzitiile de

stare prin output-ul UART

Demo video

Pentru vizualizare:

https://youtu.be/z9fMCQ8Eq-I

Se poate observa cum masina trece prin fata lui si in primul run aceasta ramane acolo avand o viteza totala sub limita de 30km/h, iar in cel- de-al doilea run aceasta depasteste limita si porneste alarma

Download

github: https://github.com/NaxonRO/pm-proiect