SmartBumper

Introducere

Ce face proiectul: Proiectul reprezinta un sistem automatizat capabil sa masoare viteza unui vehicul in miscare folosind doi senzori cu infrarosu (IR) si sa actioneze un “bumper” (limitator de viteza/bariera fizica) controlat de un servomotor. Daca vehiculul depaseste limita de viteza stabilita (30 km/h), bumperul se ridica, iar un semnal sonor de avertizare este emis. Viteza si starea limitei sunt afisate in timp real pe un ecran LCD.

Care este scopul lui: Scopul principal este sporirea sigurantei active in trafic, in special in zonele cu limitari stricte de viteza (zone rezidentiale, scoli, treceri de pietoni), prin implementarea unui sistem fizic de descurajare a vitezei excesive.

Ideea de la care am pornit: Limitatoarele de viteza clasice (bumperele din asfalt) afecteaza toate vehiculele, inclusiv pe cele care circula regulamentar, cauzand uzura inutila a suspensiilor si disconfort. Ideea a fost crearea unui sistem “inteligent” de aerodinamica sau limitare fizica, care sa actioneze **doar** impotriva soferilor care incalca regulile, lasand calea libera pentru cei care respecta viteza.

De ce este util: Pentru societate, acest concept ar putea eficientiza fluidizarea traficului si reduce poluarea fonica/uzura auto in zonele rezidentiale.

Descriere generala

Proiectul este structurat intr-o unitate centrala de procesare (microcontrollerul ATmega328P) care interactioneaza cu doua categorii de module: de intrare (Senzori IR) si de iesire (Servomotor, Buzzer, LCD).

Interactiunea modulelor (Logica de functionare):

- **Modulul de Input (Detectie):** Cei doi senzori IR sunt amplasati la o distanta fizica fixa unul de celalalt. Trecerea vehiculului prin dreptul primului senzor genereaza un semnal care declanseaza o Intrerupere Externa (External Interrupt) pe microcontroller, determinandu-l sa porneasca imediat un Timer hardware. Trecerea pe la al doilea senzor declanseaza o a doua intrerupere, care opreste Timerul.
- **Modulul de Procesare (Calcul):** Microcontrollerul citeste valoarea Timerului si calculeaza timpul scurs cu precizie. Folosind distanta fixa si timpul masurat intre cele doua intreruperi, calculeaza viteza vehiculului.
- **Modulul de Output (Reactie):**
  - Viteza este trimisa pe magistrala I2C catre **ecranul LCD**.
  - Daca viteza calculata > 30 km/h, microcontrollerul comanda **Servomotorul** si **Buzzer-ul** prin intermediul pinilor GPIO (manipuland direct registrii de port). Buzzer-ul emite o alarma sonora

timp de 3 secunde, iar servomotorul este actionat pentru a ridica limitatorul de viteza.
- Daca viteza este sub limita, servomotorul ramane in repaus (0 grade).

Schema bloc:

Hardware Design

Lista de piese si Rolul lor in proiect:

- **Placa de dezvoltare compatibila Arduino Uno:** Functioneaza ca placa de baza pentru microcontrollerul **ATmega328P**, care reprezinta “creierul” proiectului. Acesta proceseaza intreruperile, calculeaza timpul si viteza, si comanda perifericele.
- **2 x Modul Senzor Infrarosu (IR):** Au rolul de a detecta momentul exact in care vehiculul trece prin doua puncte de referinta. Ofera un semnal digital (LOW/HIGH) atunci cand raza IR este intrerupta.
- **1 x Servomotor SG90:** Actioneaza ca mecanism fizic pentru ridicarea si coborarea bumperului (limitatorului de viteza).
- **1 x Ecran LCD 16×2 cu modul I2C:** Interfata cu utilizatorul, folosita pentru a afisa in timp real viteza calculata a vehiculului si mesajul de stare (“LIMITA DEPASITA”).
- **1 x Buzzer:** Unitate de avertizare sonora care se activeaza daca se depaseste pragul de viteza stabilit.

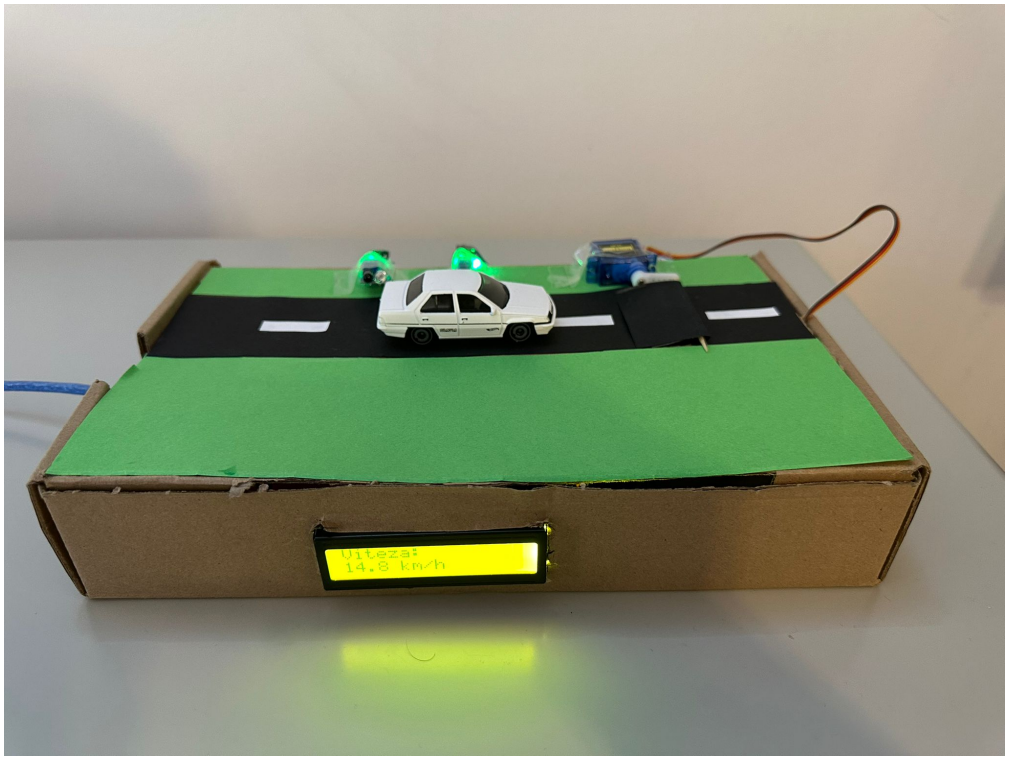
Conectarea componentelor si justificarea pinilor folositi:

- **Senzorul IR 1 (Start):** Conectat la pinul Arduino **D2**, care corespunde pinului fizic **PD2** al ATmega328P.
  - *Justificare:* Pinul PD2 este pinul hardware dedicat pentru Intreruperea Externa 0 (**INT0**). Acest lucru permite microcontrollerului sa reactioneze instantaneu (asincron) la trecerea masinii, fara a folosi polling in bucla principala.
- **Senzorul IR 2 (Stop):** Conectat la pinul Arduino **D3**, care corespunde pinului fizic **PD3** al ATmega328P.
  - *Justificare:* Pinul PD3 este pinul dedicat pentru Intreruperea Externa 1 (**INT1**), pentru a inregistra exact momentul in care masina a parcurs distanta si a opri Timerul intern.
- **Ecranul LCD I2C:** Conectat la pinii Arduino **A4** (SDA) si **A5** (SCL), corespunzatori pinilor fizici **PC4** si **PC5** ai ATmega328P.
  - *Justificare:* Acesti doi pini sunt pinii hardware dedicati pentru magistrala TWI (Two-Wire Interface / I2C) din arhitectura microprocesorului, permitand o comunicare directa si eficienta prin manipularea registrilor TWBR, TWCR, TWSR, TWDR.
- **Servomotorul SG90:** Conectat la pinul Arduino **D9**, care corespunde pinului **PB1**.
  - *Justificare:* Folosit ca pin GPIO de iesire. Miscarea servomotorului va fi generata exclusiv din software (manipuland starea pinului si timpii).
- **Buzzer-ul:** Conectat la pinul Arduino **D12**, care corespunde pinului **PB4**.
  - *Justificare:* Utilizat ca pin GPIO general. Setarea bitului corespunzator in registrul PORTB declanseaza instantaneu alarma sonora.

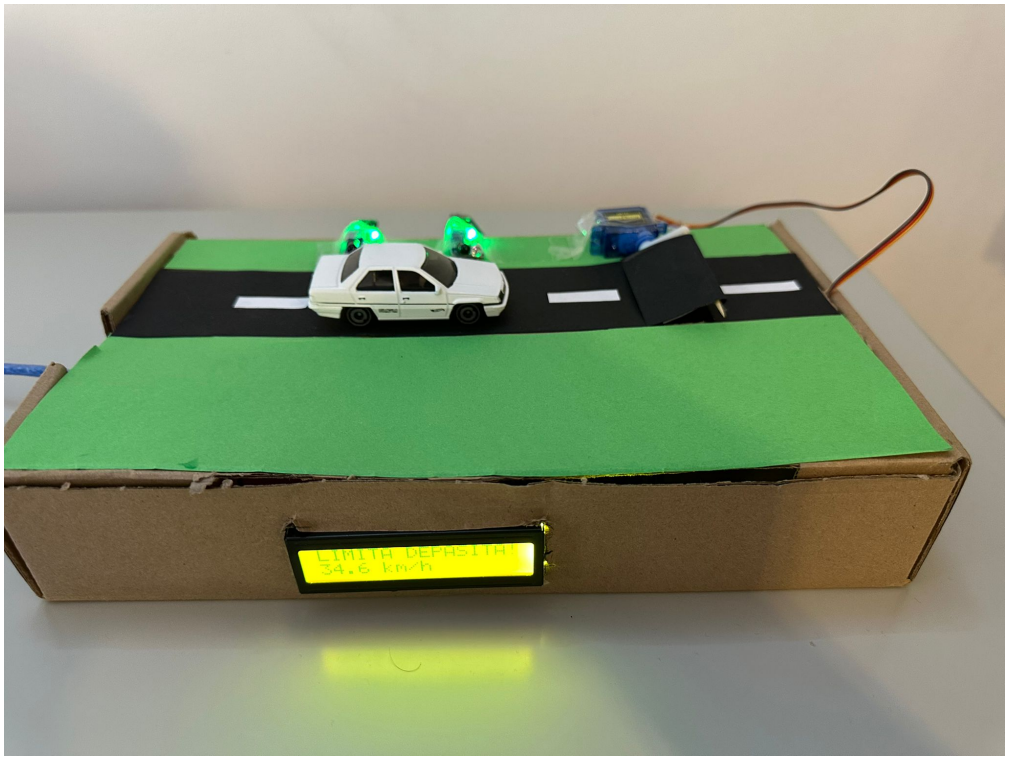
Schema electrica:

Schema ilustreaza conexiunile logice si de alimentare prezentate in sectiunea anterioara. Microcontrollerul este alimentat prin regulatorul placii, iar de pe liniile de 5V/GND sunt alimentate toate perifericele externe.

Proiectul in functiune:



Bumper coborat. Vehiculul circula cu viteza legala (< = 30 km/h)



Bumper ridicat. Vehiculul circula cu viteza ilegala (> 30 km/h)

Software Design

Mediul de dezvoltare si biblioteci folosite: Pentru dezvoltarea firmware-ului am utilizat platforma **PlatformIO** integrata in mediul de dezvoltare VS Code, folosind toolchain-ul `avr-gcc`.

Alegerea bibliotecilor: Proiectul foloseste exclusiv bibliotecile standard C pentru AVR (`<avr/io.h>`, `<avr/interrupt.h>`, `<util/delay.h>`, `<stdio.h>`). Am evitat in mod intentionat bibliotecile third-party (precum `LiquidCrystal_I2C.h` sau `Servo.h`) pentru a construi drivere proprii si a implementa comunicarea cu perifericele la nivel de registru.

Elementul de noutate: Spre deosebire de limitatoarele de viteza clasice care afecteaza fluiditatea traficului indiferent de comportamentul soferului, noutatea software consta in algoritmul de decizie in timp real. Microcontrollerul calculeaza viteza instantaneu si activeaza bariera fizica **exclusiv** pentru vehiculele care incalca regulamentul, transformand o solutie de trafic pasiva intr-una activa si inteligenta.

Laboratoare utilizate ca referinta:

- **GPIO:** A fost utilizat pentru actionarea Buzzer-ului (pin PB4) si a Servomotorului (pin PB1). Prin manipularea directa a bitilor din registrii `PORTB` si `DDRB`, am implementat controlul perifericelor. Semnalul de control pentru servomotor a fost generat exclusiv software prin pini GPIO (mentinand pinul HIGH o anumita durata in microsecunde, urmata de starea LOW), fara a bloca resursele hardware.
- **Intreruperi Externe (INT0 si INT1):** Senzorii IR au fost configurati sa declanseze o intrerupere pe front descrescator (modificand registrii `EICRA` si `EIMSK`). Citirea senzorilor prin intreruperi hardware elimina latenta metodei de “polling”, asigurand o inregistrare exacta (la nivel de microsecunda) a momentului trecerii masinii.
- **Timere:** Am configurat `Timer1` (16-biti) in modul CTC (Clear Timer on Compare Match) cu un prescaler de 64, generand o intrerupere fix la fiecare 1 milisecunda (`OCR1A = 249`). Aceasta funcionalitate creeaza o variabila de sistem (`systicks`) fiabila pentru contorizarea timpului scurs, complet independenta de intarzierile din bucla principala.
- **I2C / TWI:** Folosit pentru ecranul LCD. Am construit propriul protocol scriind direct in registrii `TWBR`, `TWCR`, `TWDR`, setand frecventa magistralei la 100 kHz si generand manual conditiile de START, STOP si WRITE.

Scheletul proiectului:

Ierarhia de fisiere a proiectului in PlatformIO:

```
SmartBumper/
└── src/
    ├── main.c
    ├── i2c.h
    ├── i2c.c
    ├── lcd.h
    └── lcd.c
```

Descrierea componentelor software:

- **main.c**: Contine logica principala a aplicatiei, initializarile pentru pinii GPIO, Timere si Intreruperi Externe, precum si algoritmul de calcul al vitezei si controlul bumperului.
- **i2c.h**: Defineste prototipurile functiilor necesare pentru configurarea si utilizarea magistralei hardware I2C/TWI a microcontrollerului.
- **i2c.c**: Implementeaza functiile de initializare, start, stop si scriere de octeti pentru protocolul I2C prin manipularea directa a registrilor hardware (`TWBR`, `TWCR`, `TWDR`).
- **lcd.h**: Defineste adresa I2C a ecranului si prototipurile functiilor de control (afisare caractere, stergere ecran, setare pozitie cursor).
- **lcd.c**: Implementeaza driverul ecranului LCD 16×2, traducand si transmitand comenzile si datele de afisat sub forma de pachete compatibile cu controlerul ecranului.

In fundal, Timer1 numara milisecundele. Trecerea vehiculului declanseaza `INT0`, care salveaza timpul initial (`t_start = systicks`). Trecerea prin cel de-al doilea senzor declanseaza `INT1`, salvand `t_stop` si ridicand un flag (`measure_complete = 1`). Bucla infinita `while(1)` proceseaza flag-ul, calculeaza viteza (V = d/t), o afiseaza pe LCD si comanda perifericele de avertizare (buzzer si servo) in functie de pragul setat de 30 km/h.

Calibrarea elementelor de senzoristica: Pentru a asigura o functionare corecta, au fost realizate doua calibrari software:

1. **Ajustarea distantei:** Distanta dintre cei doi senzori IR de pe macheta a fost definita ca macro (`DISTANCE_M 1`) pentru a calcula corect viteza in m/s si ulterior in km/h. Valoarea de 1 metru definita in cod este o distanta teoretica, stabilita in urma testelor pentru a facilita demonstrarea functionalitatii sistemului. Deoarece pe macheta senzorii sunt montati la doar cativa centimetri distanta, simularea manuala a diferitelor praguri de viteza la scara reala ar fi fost extrem de dificil de controlat.

2. **Calibrarea servomotorului SG90:** Deoarece motoarele difera mecanic, s-au testat mai multe latimi de puls (delay in microsecunde). S-au stabilit empiric valorile `PULS_INITIAL_US = 500` pentru pozitia ascunsa a bumperului si `PULS_RIDICAT_US = 800` pentru inaltarea corespunzatoare la detectia depasirii vitezei.

Optimizari:

- **Unde:** In constructia string-ului pentru afisarea pe LCD in functia `main()`.
- **Cum/De ce:** Am evitat folosirea functiei complexe `sprintf` cu argumente de tip `float`. Includerea suportului float pentru `sprintf` ar fi adaugat un overhead masiv in memoria Flash. In schimb, am optimizat extragand partea intreaga (`vit_int = (int)speed_kmh`) si prima zecimala (`vit_dec = (int)((speed_kmh - vit_int) * 10)`) si folosind formatare simpla de intregi (`%d.%d`). Aceasta tactica a redus dramatic memoria ocupata.
- **Unde:** In rutinele de tratare a intreruperilor (ISR).
- **Cum/De ce:** Calculele matematice grele (impartiri, conversii float) au fost plasate in interiorul buclei `while(1)`, nu direct in ISR. Intreruperile realizeaza doar atribuiri simple si rapide de variabile. Optimizarea mentine timpul de executie al ISR-urilor la minimum, prevenind fenomenul de “interrupt starvation” si blocarea microprocesorului.

Rezultate Obtinute

In urma asamblarii si scrierii firmware-ului, a rezultat un sistem hardware-software perfect functional,

capabil sa reactioneze in timp real la evenimente fizice. Sistemul masoara cu succes timpul dintre cele doua puncte de referinta folosind rutine de tratare a intreruperilor (ISR) foarte rapide, evitand polling-ul. Viteza este calculata corect in bucla principala (raportata la distanta teoretica de 1 metru impusa pentru usurinta testarii manuale a senzorilor) si este afisata pe ecranul LCD folosind driverul I2C creat de la zero. In cazul depasirii pragului de 30 km/h, sistemul de actiune (servomotorul si alarma sonora) este declansat instantaneu prin manipularea directa a pinilor GPIO.

Concluzii

Proiectul “SmartBumper” reprezinta o aplicatie completa de embedded systems care integreaza multiple periferice hardware si concepte software intr-un produs coeziv, capabil sa functioneze in timp real.

Implementarea acestui sistem a contribuit semnificativ la aprofundarea programarii la nivel low-level a microcontrolerului ATmega328P. Dezvoltarea firmware-ului a consolidat intelegerea si aplicarea unor concepte critice precum: configurarea Timerelor hardware pentru masurarea precisa a timpului, gestionarea Intreruperilor Externe pentru reactii asincrone imediate, manipularea directa a registrilor GPIO pentru controlul mecanic si acustic, precum si implementarea de la zero a protocolului de comunicatie I2C.

In final, proiectul demonstreaza modul in care conceptele tehnice independente, pot fi imbinate cu succes intr-un dispozitiv interactiv si functional, pentru a rezolva o problema reala din domeniul sigurantei rutiere.

Download

Arhiva completa cu codul sursa:

smartbumper.zip

Link repository:

SmartBumper

Bibliografie/Resurse

Resurse Hardware:

- ATmega328P Datasheet (published by Microchip)
- Documentatie tehnica Controller LCD si Modul interfata I2C

Resurse Software:

- Materialele de Laborator PM pentru: GPIO, Intreruperi Externe, Timere si I2C / TWI.
- PlatformIO Documentation