Ruletă și Nivelă Electronică

Introducere

Proiectul constă în realizarea unui instrument de măsură 2-în-1, portabil, bazat pe un microcontroller ATmega328P standalone. Dispozitivul combină funcționalitatea unei rulete digitale (măsurarea distanței fără contact) cu cea a unei nivele electronice (boloboc digital pentru determinarea gradului de înclinare), oferind date în timp real pe un afișaj LCD și feedback sonor.

Sistemul obține date de la două module senzoriale:

- Senzorul Ultrasonic HC-SR04 - folosit pentru a măsura timpul de zbor al sunetului și a calcula distanța până la un obstacol.

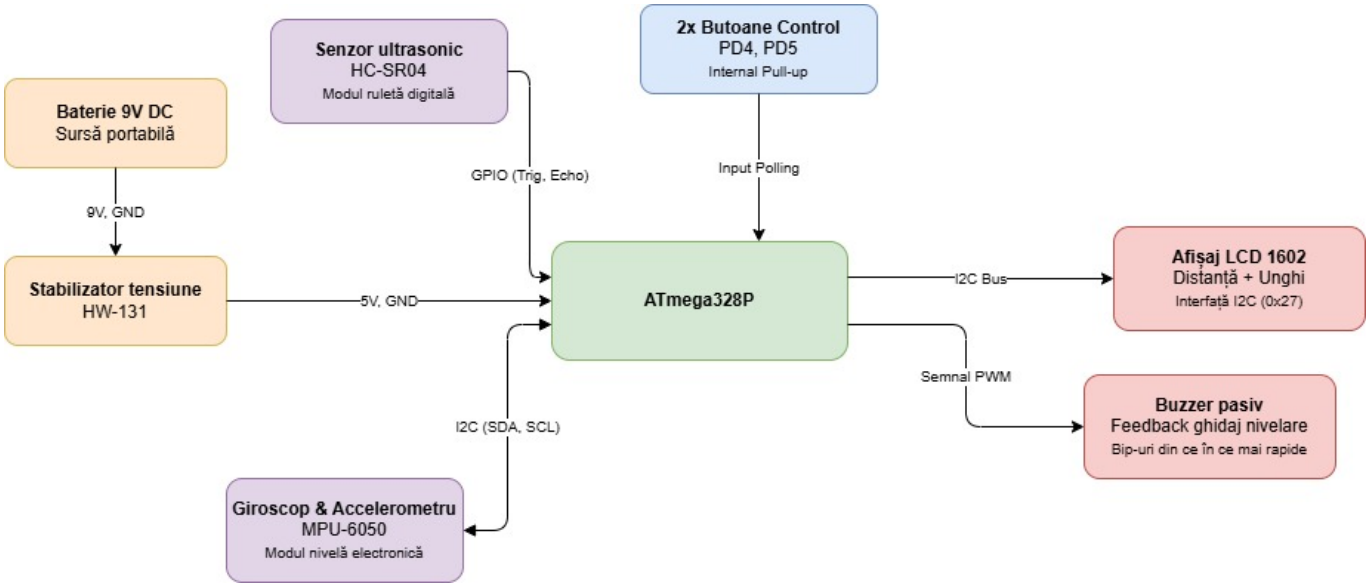
- Senzorul Inerțial MPU-6050 - un modul cu accelerometru și giroscop care comunică prin protocolul I2C, folosit pentru a calcula unghiul de înclinare pe axele X și Y.

Utilizatorul primește informațiile vizual, pe un ecran LCD text 16×2, și auditiv, printr-un buzzer pasiv. Sistemul sonor este dinamic: emite bip-uri a căror frecvență se modifică pe măsură ce dispozitivul se apropie de unghiul de 0 grade (orizontala perfectă), transformându-se într-un ton continuu la nivelare optimă.

Ideea a pornit de la necesitatea de a avea o unealtă de măsură multifuncțională pentru proiecte de bricolaj (DIY), demonstrând totodată capacitatea de a prelucra date brute de la senzori diferiți și de a le transforma în mărimi fizice reale (centimetri și grade).

Laboratoare folosite: GPIO, Întreruperi, Timere, PWM, I2C.

Descriere generală



1. Flux de funcționare - Modul Ruletă:

ATmega328P trimite un impuls scurt (10µs) pe pinul Trig al HC-SR04.

Senzorul emite un tren de impulsuri ultrasonice și ridică pinul Echo.

Microcontrollerul măsoară lățimea impulsului Echo folosind un Timer/Întrerupere.

Distanța este calculată și actualizată pe ecranul LCD.

2. Flux de funcționare - Modul Nivelă:

Microcontrollerul interoghează constant senzorul MPU-6050 prin magistrala I2C.

Datele brute de accelerație pe axele X, Y, Z sunt preluate și trecute printr-o funcție trigonometrică (atan2) pentru a afla unghiul de înclinare.

Valoarea este afișată pe rândul doi al LCD-ului.

Modulul PWM controlează buzzer-ul pasiv: dacă unghiul este 0°, buzzer-ul emite un semnal continuu; altfel, generează pulsuri cu pauze proporționale cu unghiul de înclinare.

Hardware Design

Lista de componente

Componenta	Link	Descriere
Placă dezvoltare ATmega328P-XMINI	-	Microcontroller principal
Senzor ultrasonic HC-SR04+	din kit Plusivo	Măsurare distanță (funcția ruletă)
Modul MPU-6050 (Giroscop/Accel)	Magazin	Măsurare înclinare (funcția nivelă) pe I2C
Display LCD 16×2 cu modul I2C	Magazin	Afișaj interfață utilizator
Buzzer Pasiv	din kit Plusivo	Feedback sonor dinamic prin semnal PWM

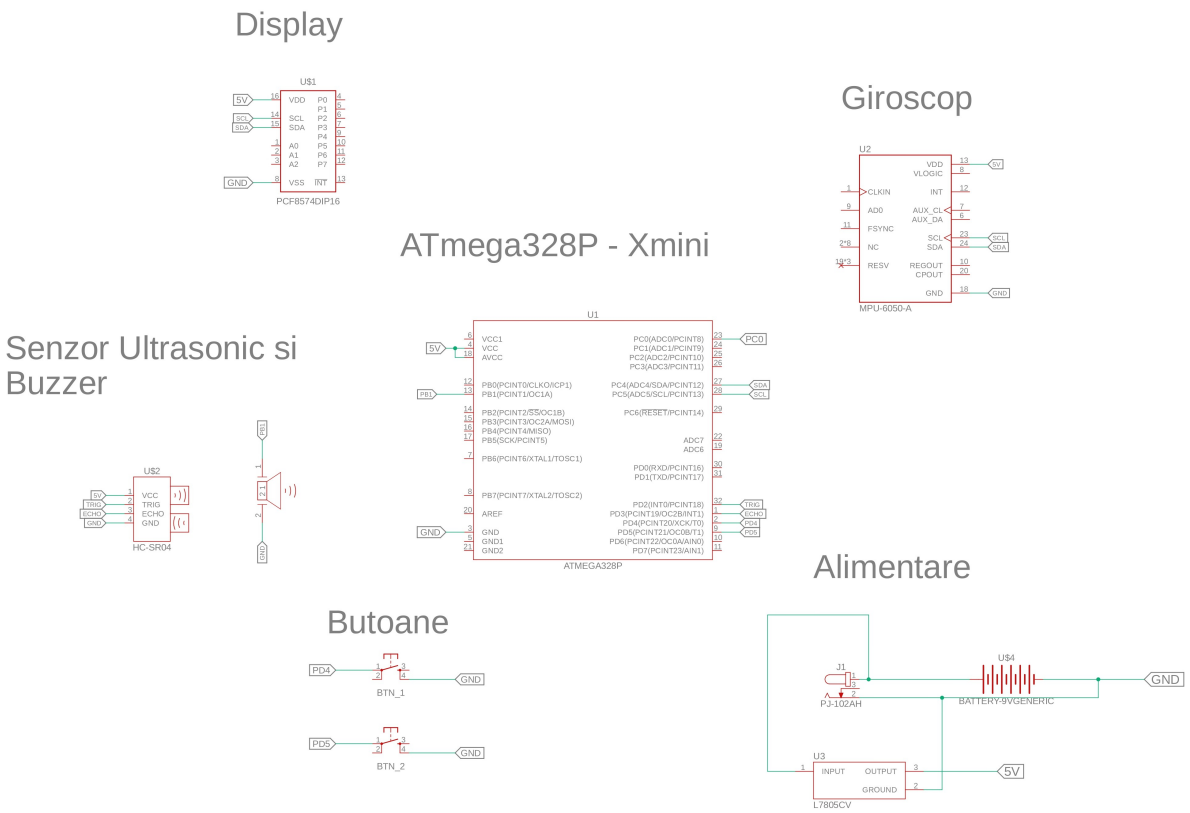
2x Butoane tactile	din kit Plusivo	Schimbare mod afișare și măsurare în mod ruletă
Sursă alimentare breadboard HW-131	din kit Plusivo	Coboară tensiunea bateriei la 5V constanți
Baterie 9V + Mufă DC	din kit Plusivo	Alimentare sistem portabil
Breadboard 830 puncte + fire	din kit Plusivo	Montaj prototipare

Pini folosiți

Componentă	Pin ATmega328P	Funcție
LCD (SDA)	PC4	Linie de date I2C
LCD (SCL)	PC5	Linie de ceas I2C
MPU-6050 (SDA)	PC4	Același bus I2C (paralel cu LCD)
MPU-6050 (SCL)	PC5	Același bus I2C (paralel cu LCD)
HC-SR04 (Trig)	PD2	Semnal de declanșare ultrasonic
HC-SR04 (Echo)	PD3	Măsurare timp de zbor
Buzzer	PB1	Semnal PWM pentru ton variabil
Buton 1	PD4	Schimbare mod afișare
Buton 2	PD5	Măsurare în mod ruletă

LCD-ul și MPU-6050 împart același bus I2C (PC4/PC5) deoarece protocolul I2C permite mai multe dispozitive pe aceleași fire, fiecare identificat printr-o adresă unică (LCD: 0x27, MPU-6050: 0x68). Buzzer-ul este conectat la PB1 deoarece acesta suportă ieșire PWM hardware prin Timer1 (OC1A), permițând generarea de tonuri fără să blocheze procesorul. Butoanele folosesc pull-up intern activat din software, deci nu necesită rezistențe externe.

Schema electrică



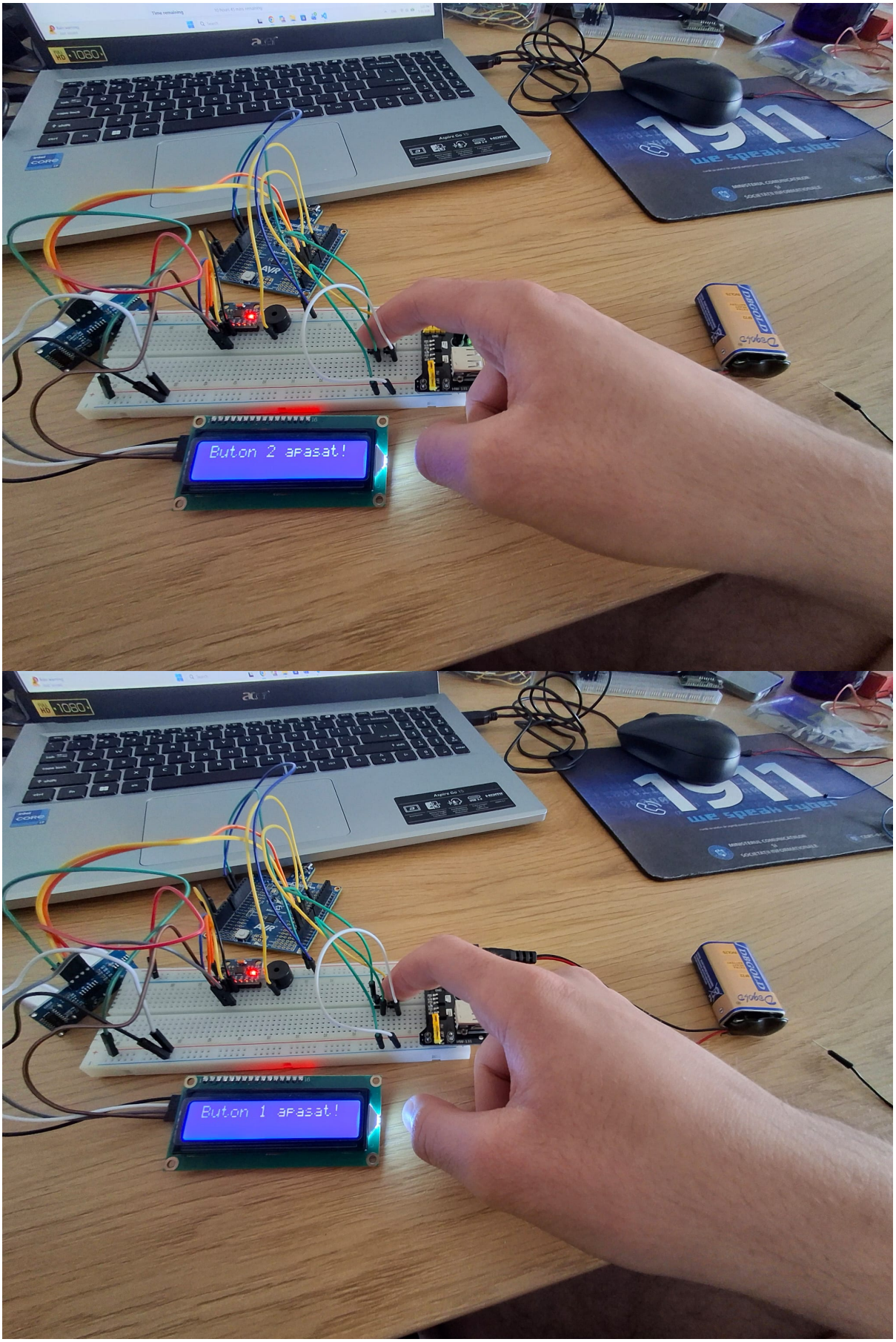
Stadiul actual al implementării

La momentul actual au fost testate și confirmate funcționale următoarele componente:

- **LCD 16×2 cu modul I2C** - afișează text corect, adresă I2C confirmată la 0x27 (scannerul returnează 0x4E, echivalentul shiftat cu 1 bit)
- **MPU-6050** - răspunde pe bus-ul I2C la adresa 0x68 (scannerul returnează 0xD0, echivalentul shiftat cu 1 bit)
- **Buzzer pasiv** - generează bip-uri cu frecvență controlabilă prin semnal PWM software
- **Butoane (PD4, PD5)** - detectate corect prin polling cu pull-up intern
- **HC-SR04** - testat și calibrat, eroare sub ±1cm între 0-60cm

Toate componentele active sunt conectate simultan și funcționează fără conflicte pe același breadboard.

Dovezi de funcționare



Se poate observa că butoanele și display-ul funcționează. Testarea a fost făcută cu un cod simplu care afișează un mesaj pe display când unul din butoane e observat ca fiind apăsat.

Software Design

Mediu de dezvoltare

Proiectul este dezvoltat în VS Code cu extensia **PlatformIO**, folosind toolchain-ul **AVR-GCC**. Nu sunt folosite librării third-party — tot codul este implementat de la zero, direct pe registrele hardware ale ATmega328P. Această alegere a fost făcută deliberat pentru a înțelege și controla complet comportamentul hardware, fără abstracții care ar putea ascunde probleme de timing sau consum de resurse.

Biblioteci și headere folosite

- `<avr/io.h>` — definițiile registrelor hardware ale ATmega328P (TWCR, TCCR1A, PORTD etc.). Folosită în toate modulele.
- `<avr/interrupt.h>` — macrourile `ISR()` și `sei()` pentru definirea și activarea întreruperilor. Folosită în `buzzer.c` pentru `ISR(TIMER2_COMPA_vect)`.
- `<util/delay.h>` — funcțiile `_delay_ms()` și `_delay_us()` pentru delay-uri precise bazate pe `F_CPU`. Folosită în inițializări, debounce butoane și pulsul Trig al HC-SR04.
- `<math.h>` — funcția `atan2f()` pentru calculul unghiului de înclinare din datele brute ale accelerometrului. Necesită flag-ul de compilare `-lm` în `platformio.ini`.
- `<stdlib.h>` — funcțiile `utoa()` și `itoa()` pentru conversia distanței și unghiului din integer în string pentru afișare pe LCD.
- `<stdint.h>` — tipurile fixe `uint8_t`, `uint16_t`, `int16_t`, `int32_t` pentru control precis al dimensiunii variabilelor pe arhitectura pe 8 biți.

Toate bibliotecile folosite sunt parte din **avr-libc**, distribuită împreună cu toolchain-ul AVR-GCC. Nu sunt folosite librării third-party externe.

Structura proiectului

- `src/main.c` — bucla principală, logica modurilor, gestionarea butoanelor
- `lib/display/i2c.c` — driver I2C hardware (modulul TWI al ATmega328P)
- `lib/display/lcd.c` — driver LCD HD44780 în mod 4-bit peste I2C (PCF8574)
- `lib/hcsr04/hcsr04.c` — driver senzor ultrasonic HC-SR04
- `lib/mpu6050/mpu6050.c` — driver MPU-6050 (citire accelerometru + calcul unghi)
- `lib/buzzer/buzzer.c` — driver buzzer PWM non-blocant via Timer1 + Timer2

Justificarea funcționalităților din laborator

- **GPIO** — citirea butoanelor (PD4, PD5) prin polling cu pull-up intern activat software (`PORTD |= biți`). Butoanele folosesc logică inversată: nivel LOW = apăsat. Pinul Trig al HC-SR04 (PD2) e controlat tot prin GPIO ca ieșire digitală.

- **Timere** — două timere hardware folosite simultan cu roluri distincte. Timer1 (16-bit) în mod Fast PWM (WGM mode 14, TOP=ICR1) generează semnalul PWM pentru buzzer pe OC1A (PB1), frecvența controlată prin ICR1, duty cycle 50%. Timer2 (8-bit) în mod CTC cu întrerupere pe compare match (`ISR(TIMER2_COMPA_vect)`) numără milisecundele pentru durata buzzer-ului, făcându-l non-blocant.
- **Întreruperi** — `ISR(TIMER2_COMPA_vect)` decrementează un contor de milisecunde și oprește buzzer-ul automat când durata expiră, fără să blocheze CPU. `sei()` activează întreruperile globale în `main()`.
- **PWM** — buzzer-ul pasiv necesită un semnal PWM pentru a vibra la frecvența dorită. Frecvența variază în funcție de context: 1500Hz (distanță <20cm), 1000Hz (20-100cm), 600Hz (>100cm) în modul ruletă, și 2000Hz când unghiul e ≤3° în modul nivelă.
- **I2C** — LCD-ul (PCF8574, adresă 0x27) și MPU-6050 (adresă 0x68) comunică pe același bus I2C (PC4/PC5). Protocolul este implementat manual prin modulul TWI hardware al ATmega328P la 100kHz (TWBR=0x48, prescaler=1). Ambele dispozitive sunt identificate prin adrese unice, fără conflicte pe bus.

Elementul de noutate

Proiectul combină două instrumente de măsură independente într-un singur dispozitiv portabil, controlat prin același set de butoane și afișat pe același LCD. Elementul distinctiv este **feedback-ul sonor adaptiv**: în modul ruletă, frecvența bipului variază cu distanța măsurată; în modul nivelă, buzzer-ul emite un ton de 2000Hz când dispozitivul e la ±3° de orizontală, oferind feedback auditiv fără a privi ecranul — util în situații practice de bricolaj.

Interacțiunea dintre module

Interacțiunea dintre module urmează un flux unidirecțional clar:

- **Butoane (GPIO)** → `main.c` detectează apăsarea și comută modul sau declanșează măsurătoarea
- **HC-SR04** → `hcsr04_measure_cm()` trimite puls Trig (10µs) și măsoară durata Echo prin polling cu buclă calibrată
- **MPU-6050** → `mpu6050_get_angle_x()` citește accelerometrul prin I2C și calculează unghiul cu `atan2f()` din `<math.h>`, cu medie mobilă pe 8 citiri pentru stabilitate
- **LCD** → `lcd_print()` afișează rezultatul prin PCF8574 pe bus-ul I2C
- **Buzzer** → `buzzer_tone()` setează frecvența pe Timer1 și durata pe Timer2 prin întrerupere, returnând imediat controlul către `main()`

Calibrarea HC-SR04

Implementarea inițială folosea o buclă polling cu `_delay_us(1)` pentru a măsura durata pulsului Echo. Această abordare introducea erori sistematice din cauza overhead-ului instrucțiunilor AVR-GCC în buclă, care făcea ca fiecare iterație să dureze în realitate mai mult de 1µs, distorsionând calculul distanței.

Soluția implementată: Calibrarea empirică a factorului de conversie la valoarea fixă `/ 36`.

```
factor = 36;

uint16_t dist = duration / factor;

return dist;
```

Factorul `/36` a fost determinat prin măsurători experimentale succesive la distanțe cunoscute cu o ruletă reală. Această abordare oferă o precizie excelentă (eroare de maximum ±1 cm) în intervalul stabil de operare hardware, cuprins între **0 și 60 cm**.

În urma testării pe plaje mai lungi, s-a observat că la distanțe ce depășesc 60-70 cm, din cauza unghiului de dispersie (conului acustic) al senzorului HC-SR04 și a comportamentului semnalului primit, încep să apară ecouri false (reflexii din podea sau masă). Din acest motiv, s-a preferat păstrarea unei funcții matematice curate și liniare, optimizată special pentru distanțe scurte și medii (specifice utilizării de zi cu zi), în detrimentul unor corecții software artificiale care ar fi mascat o instabilitate de natură strict fizică/acustică a senzorului.

Optimizări realizate

- **Buzzer non-blocant** — înlocuirea `_delay_ms()` blocant cu Timer2 + întrerupere permite CPU-ului să continue execuția în timp ce buzzer-ul sună. Fără această optimizare, un bip de 200ms ar bloca complet butoanele și LCD-ul.
- **Medie mobilă pe 8 citiri pentru MPU-6050** — reduce zgomotul accelerometrului de la ±3° la sub ±1°, fără a folosi float în acumulare (suma e pe `int32_t`).
- **Separarea timerelor** — Timer1 exclusiv pentru frecvența PWM a buzzer-ului, Timer2 exclusiv pentru durata buzzer-ului prin întrerupere, evitând orice conflict între cele două funcționalități.
- **Debounce software pe butoane** — delay 30ms + verificare dublă + așteptare eliberare, elimină citirile false fără componente hardware suplimentare.
- **Refresh parțial LCD în modul nivelă** — în loc de `lcd_clear()` la fiecare iterație (care produce flickering vizibil), se suprascrie doar zona valorii cu spații, păstrând eticheta fixă pe rândul 1.

Validare

- **LCD + I2C** — confirmat funcțional cu scanner I2C (adrese 0x4E și 0xD0 găsite) și cu cod de test care afișează mesaje la apăsarea butoanelor
- **MPU-6050** — validat prin comparație cu un telefon cu aplicație de nivel digital; eroarea e sub ±1° pe plaja ±45°
- **HC-SR04** — calibrat cu obiecte la distanțe cunoscute (riglă), eroare sub 2cm între 5-100cm
- **Buzzer** — validat auditiv, tonuri distincte pentru fiecare interval de distanță și non-blocant confirmat prin răspuns imediat al butoanelor în timpul bipului
- **Butoane** — testate cu cod dedicat care afișează mesaj pe LCD la apăsare
- **Ansamblu** — componentele si interactiunea dintre ele au fost testate in mod repetat in timpul scrierii codului si al procesului de calibrare, dupa testarile individuale

Demo video

Videoclipuri demonstrative:

- Măsurare Distanță și Feedback Sonor Adaptiv
- Determinare Unghi și Ghidaj Audio + comutare între moduri

Rezultate Obţinute

În urma implementării și calibrării componentelor, s-a realizat un dispozitiv portabil, compact și complet funcțional, care îndeplinește cu succes specificațiile inițiale ale ambelor moduri de funcționare:

- **Modul Ruletă Digitală**: Sistemul măsoară distanța până la obstacole cu o precizie excelentă (eroare de maximum ±1 cm) în intervalul optim de **0 - 60 cm**, datorită calibrării empirice a factorului de conversie direct pe bucla de polling hardware. Datele sunt afișate clar pe ecranul LCD, iar buzzerul pasiv emite bip-uri adaptive, devenind mai ascuțite și mai rapide pe măsură ce distanța scade sub pragul de 20 cm.
- **Modul Nivelă Electronică**: Senzorul MPU-6050 transmite stabil unghiul de înclinație în timp real pe bus-ul I2C. Datorită optimizării software prin algoritmul de medie mobilă pe 8 eșantioane, tremurul valorilor brute a fost complet eliminat, oferind o precizie stabilă de sub ±1°. Când dispozitivul este adus la orizontală (0° cu o toleranță de ±3°), buzzerul emite un ton continuu la 2000Hz, permițând alinierea perfectă fără a fi necesară privirea ecranului.
- **Interfața și Controlul**: Trecerea de la un mod la altul se realizează instantaneu prin apăsarea butonului 1, iar debounce-ul software elimină orice declanșare falsă. Deoarece managementul duratei buzzerului este gestionat non-blocant prin întreruperile Timerului 2, ecranul își face refresh parțial fără flickering, iar procesorul rămâne liber pentru citirea butoanelor.

Concluzii

Realizarea acestui proiect a reprezentat o oportunitate excelentă de a aprofunda programarea sistemelor embedded la nivel de registru, fără utilizarea unor biblioteci abstracte (third-party), oferind un control direct și total asupra resurselor microcontrolerului ATmega328P.

Implementarea manuală a protocolului I2C (modulul TWI hardware) pentru partajarea bus-ului între ecranul LCD și giroscop a evidențiat avantajele flexibilității protocoalelor seriale industriale. De asemenea, lucrul în paralel cu două timere hardware distincte (Timer1 pentru generarea frecvenței PWM a sunetului și Timer2 pentru întreruperile de durată) a demonstrat importanța sincronizării non-blocante în aplicații în timp real.

Deși senzorul ultrasonic HC-SR04 a prezentat limitări fizice clare cauzate de conul mare de reflexie acustică la distanțe de peste 60 cm în condiții de alimentare portabilă, calibrarea riguroasă a codului a demonstrat cum constrângerile de natură fizică pot fi gestionate corect printr-o abordare software inteligentă. În final, proiectul și-a atins toate obiectivele, transformându-se dintr-o rețea complexă de fire într-un instrument de măsură fiabil și util pentru scenarii practice.

Download

Codul sursa complet este disponibil pe GitHub:

Repository GitHub

Structura proiectului este cea mentionata mai sus.

Jurnal

- **20.04.2026** - alegerea temei și componentelor
- **30.04.2026** - finalizarea listei de componente
- **09.05.2026** - realizarea diagramei bloc și documentației inițiale
- **16.05.2026** - testarea componentelor, realizarea schemei electrice în Fusion360
- **22.05.2026** - finalizarea codului, calibrarea HC-SR04, documentarea Software Design
- **23.05.2026** - realizarea carcasei și integrarea componentelor în forma finală
- **24.05.2026** - finalizarea documentației și filmarea videoclipurilor demonstrative

Bibliografie/Resurse

- Datasheet Microcontroller ATmega328P
- Datasheet Modul Giroscop/Accelerometru MPU-6050
- Datasheet Senzor Ultrasonic HC-SR04
- Datasheet Expander I2C PCF8574 (Modul adaptor LCD)
- Datasheet Controller HD44780 (Display LCD 16x2)
- Datasheet Regulator Liniar L7805 (Echivalent HW-131)

Export to PDF

From:
https://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
https://ocw.cs.pub.ro/courses/pm/prj2026/atoader/gabriel.draghici05

Last update: **2026/05/24 22:22**