InfraCatch - reflex game

Autor: Moroianu Horia-Valentin
Grupă: 334CA

Introducere

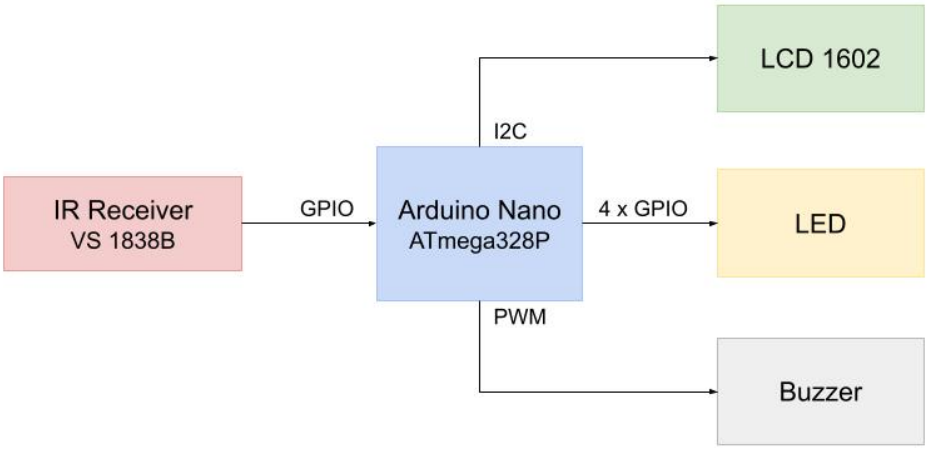
InfraCatch este un joc interactiv de testare a reflexelor. Patru LED-uri se aprind aleatoriu, unul câte unul, iar jucătorul trebuie să apese butonul corespunzător de pe o telecomandă cu infraroșu pentru a stinge LED-ul activ. Dacă reacția este greșită sau prea lentă, jucătorul pierde o viață. Feedback-ul este oferit în timp real printr-un buzzer și un LCD care arată progresul făcut. Jocul se termină atunci când jucătorul rămâne fără vieți.

Scopul jocului este de a obține un scor cât mai mare prin testarea reflexelor într-un mod distractiv și ușor de înțeles. Ideea a pornit din dorința de a crea un joc fizic, controlat de la distanță și de a învăța mai multe despre tehnologia de comunicație cu infraroșu și timer-ele interne ale unui microprocesor.

Consider că acest proiect este util pentru mine deoarece îmi oferă ocazia de a construi un sistem complet, de la început până la sfârșit, concentrat asupra interacțiunii cu utilizatorul, și cred că poate atrage atenția și altor persoane care caută un mod de relaxare, diferit de jocurile digitale clasice.

Descriere generală

Diagramă bloc:



Proiectul este construit în jurul microcontroller-ului Arduino Nano, care coordonează următoarele module:

- Receptor IR VS1838B:
  - Primește semnalele de la telecomandă și transmite codurile corespunzătoare către Arduino.
  - Verifică dacă butonul apăsat corespunde LED-ului aprins.
- LED-uri (x4):
  - Conectate la pini digitali GPIO ai Arduino-ului.
  - Se aprind aleatoriu și trebuie stinse de utilizator prin apăsarea butonului corect de pe telecomandă.
- Display LCD 1602 cu modul I2C:
  - Afișează informații despre joc, cum ar fi scorul, numărul de vieți rămase și eventuale mesaje.
  - Modulul I2C ușurează semnificativ cablajul, folosind doar două fire (SDA și SCL).
- Buzzer (cu PWM):
  - Oferă feedback sonor.
  - Emite un sunet pozitiv pentru acțiuni corecte și unul negativ pentru greșeli sau întârzieri.

Hardware Design

Schemă electrică: infracatch-schematic.pdf

Conectivitate:

Pin ATmega328P	Pin Arduino Nano	Componentă asociată	Funcție
GND	GND	Toate modulele	Alimentare
VCC	5V	Toate modulele	Alimentare
PB0	D8	LED0	GPIO control
PB1	D9	LED1	GPIO control
PB2	D10	LED2	GPIO control

PB3	D11	LED3	GPIO control
PC4	A4	LCD-1602	I2C - SDA
PC5	A5	LCD-1602	I2C - SCL
PD2	D2	Receptor IR	Data/INT0
PD6	D6	Buzzer	Timer0 PWM Control

Listă de componente:

Nume componentă	Link achiziție	Cantitate	Preț unitar (lei)
Arduino Nano (ATmega328p)	Link	1	24.99
Receptor Infraroșu	Link	1	8.99
Telecomandă Infraroșu	Link	1	3.6
LCD 1602	Link	1	16.99
Adaptor I2C pentru LCD 1602	Link	1	5.37
Buzzer	Link	1	1.4
LED	Link	4	0.39
Rezistor 0.5W 220Ω	Link	4	0.1
Breadboard 830 Puncte	Link	1	9.98
Breadboard 400 Puncte	Link	1	4.56
Fire tată-tată	Link	1	7.99
Fire mamă-tată	Link	1	4.45
Cablu USB-B Mini	Link	1	4.37
Preț total:			**93.18**

Software Design

Mediu de dezvoltare: *Visual Studio Code + PlatformIO*

Biblioteci externe:

- Arduino pentru compatibilitatea cu celelalte biblioteci folosite
- IRRemote pentru decodificarea semnalelor primite de la senzorul IR
- LiquidCrystal_I2C pentru afișarea stării jocului pe ecran

Implementare:

Codul este împărțit în 5 fișiere ce vor fi prezentate mai jos, ținând cont de funcționalitățile moduleleor. Deși acestea nu conțin particularități ale limbajului C++, am ales sa folosesc acest limbaj pentru compatibilitatea cu bibliotecile externe.

main.cpp / main.h: conține implementarea principală a jocului.

Programul este structurat sub forma unui atutomat cu 3 stări:

- START: Inițializează jocul și așteaptă comanda START. La primirea ei, resetează scorul și viețile, pornește timerul, setează seed-ul random și trece la GAME_PLAYING.
- GAME_PLAYING: Gestionează jocul activ — aprinde LED-uri, așteaptă răspuns IR și actualizează scorul sau viețile. Dacă timpul expiră sau răspunsul e greșit, scade o viață. Dacă viețile ajung la 0

sau se apasă START, trece la GAME_OVER.
- GAME_OVER: Oprește LED-urile, afișează scorul final și redă un sunet în funcție de performanță. Așteaptă comanda START pentru a reveni la GAME_START.

Variabile globale folosite:

- `game_state` – stare actuală a jocului
- `valid_ir` – flag pentru semnal IR valid
- `led_timeout` – flag dacă timpul LED-ului a expirat
- `score, high_score, lives` – folosite pentru a urmari progresul jucatorului

timers.cpp / timers.h: realizează controlul duratei de aprindere a LED-urilor, generarea de sunete prin buzzer și extragerea unui seed pentru randomizare.

Timer1 - Folosit pentru a controla cât timp rămâne aprins un LED.

- `initLedTimer()` – setează Timer1 în mod CTC, cu prescaler 1024 și OCR corespunzător MAX_LED_DELAY (2s).
- `startLedTimer()` – pornește cronometrarea si resetează TCNT1.
- `stopLedTimer()` – oprește timerul.
- `decreaseLedDelay()` – reduce treptat durata LED-ului (până la MIN_LED_DELAY), pentru a crește dificultatea jocului.
- ISR `TIMER1_COMPA_vect` – semnalează timeout-ul LED-ului în main.cpp (led_timeout = true).

Timer0 - Folosit pentru a genera tonuri audio la frecvențe date.

- `initBuzz()` – configurează Timer0 în mod CTC pentru ieșire pe pinul PD6 (OC0A).
- `buzz(freq, duration)` – redă un sunet la frecvența și durata cerută. Folosește `timer_freq_prescale()` pentru a calcula automat prescalerul și OCR-ul necesar si controlează durata folosind întreruperea `TIMER0_COMPA_vect`. (*Cod inspirat din acest articol.*)

Random seed: `extractTimers()` – combină valorile din TCNT0, TCNT1 și TCNT2 într-un uint32_t pentru a genera un seed aleator folosit la alegerea LED-urilor.

lcd.cpp / lcd.h: gestionează afișajul LCD al jocului (mesaje de început, de final, scor, și vieți).

- vectorul `heart` reprezintă un caracter personalizat (o inimă stilizată), folosită pentru a reprezenta viețile jucătorului.
- `initLCD()` — inițializează LCD-ul și configurează simbolul inimă pentru afișare.
- `displayStart()` — afișează instrucțiunile inițiale pentru a porni jocul și controlul LED-urilor.
- `displayScore(lives, score)` — arată numărul de vieți rămase și scorul curent pe ecran.
- `displayGameOver(score, high_score)` — prezintă scorul final și high score-ul la încheierea jocului.

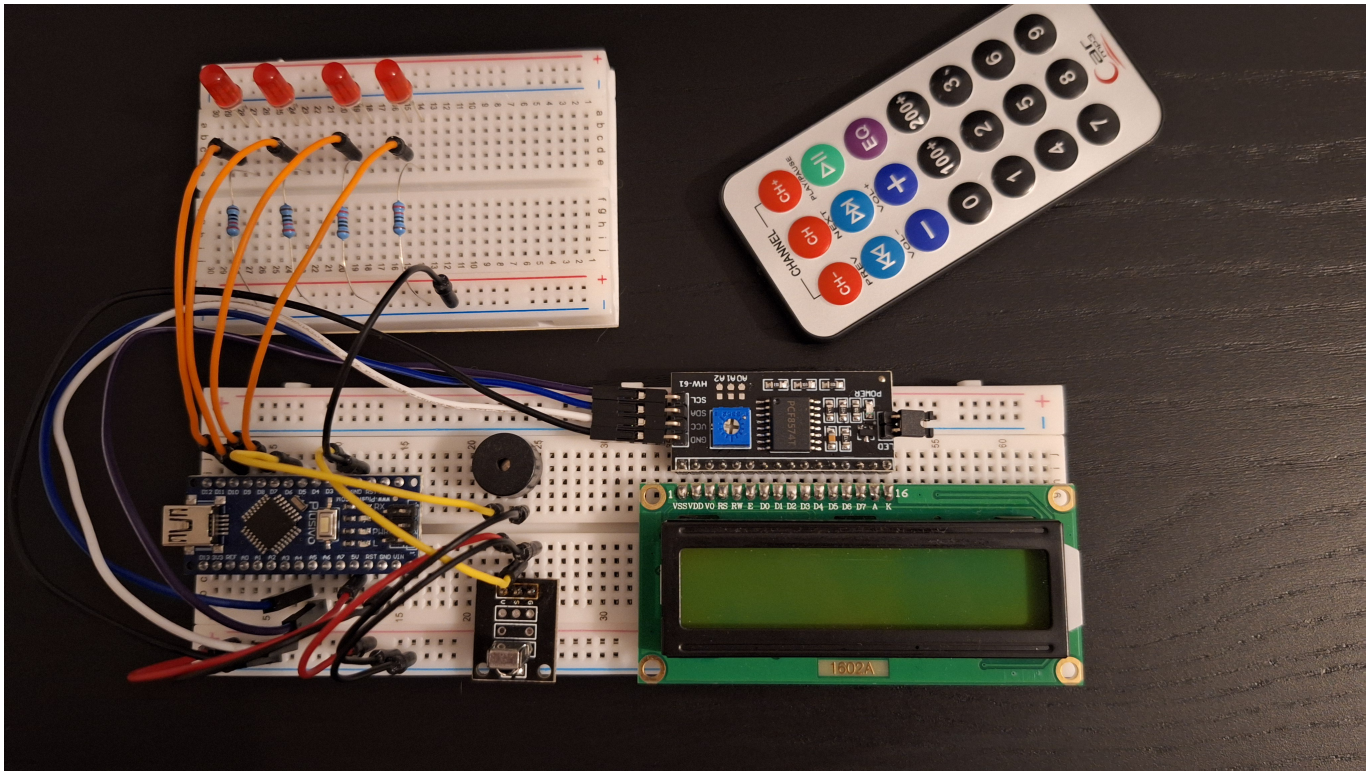
sounds.cpp / sounds.h: se ocupă de redarea melodiilor și sunetelor pentru stările jocului.

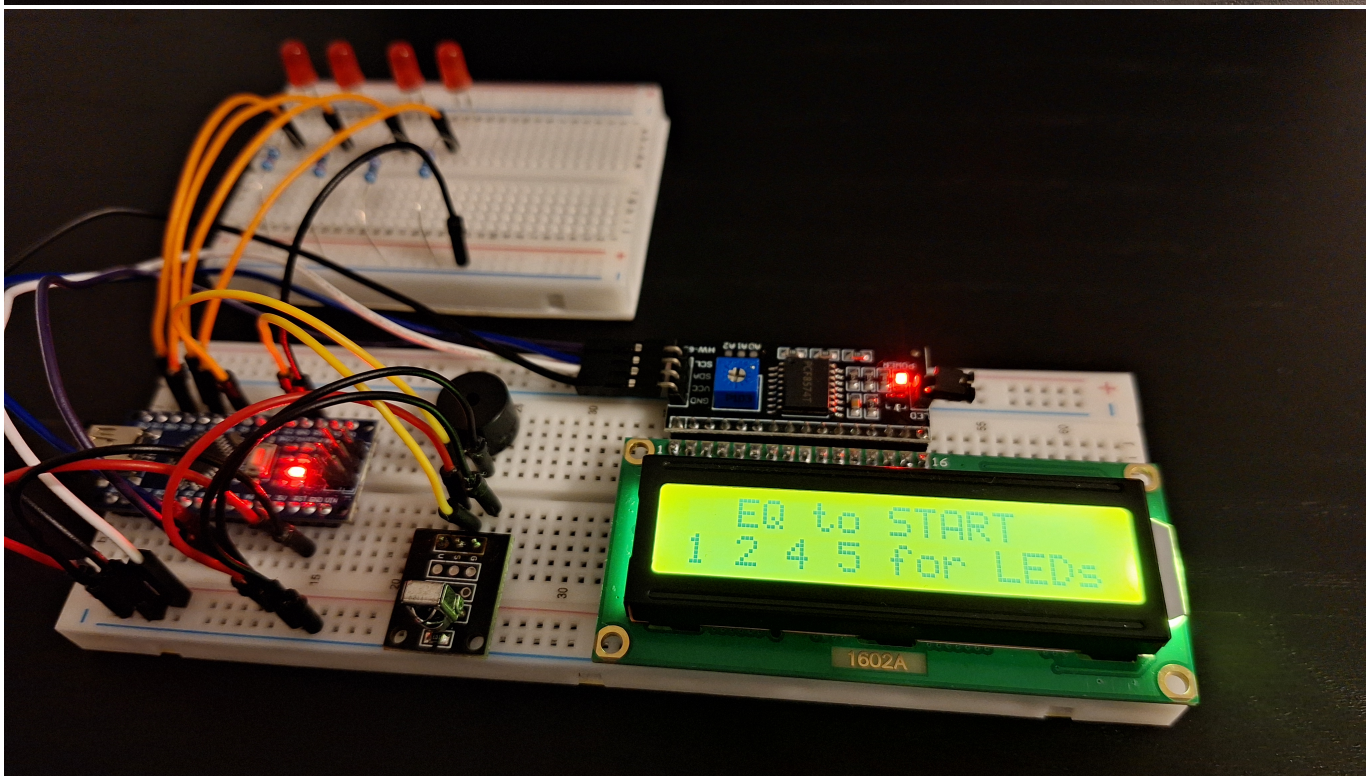
- `playStart()` — melodie scurtă de început pentru startul jocului.
- `playWin()` — melodie de victorie la obținerea unui high score.
- `playFail()` — melodie de eșec/skip.

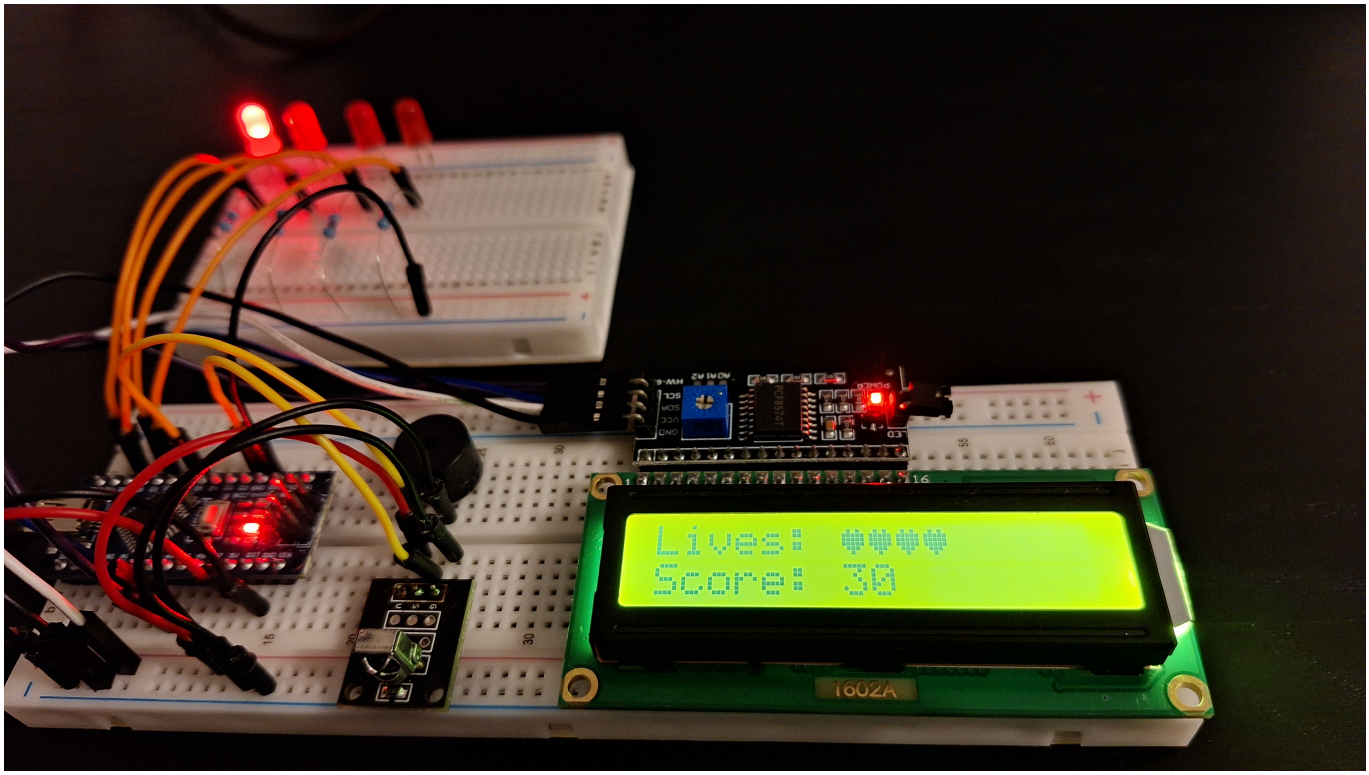
random.cpp / random.h: modul de generare a numerelor pseudo-aleatoare, optimizat pentru microcontrolere.

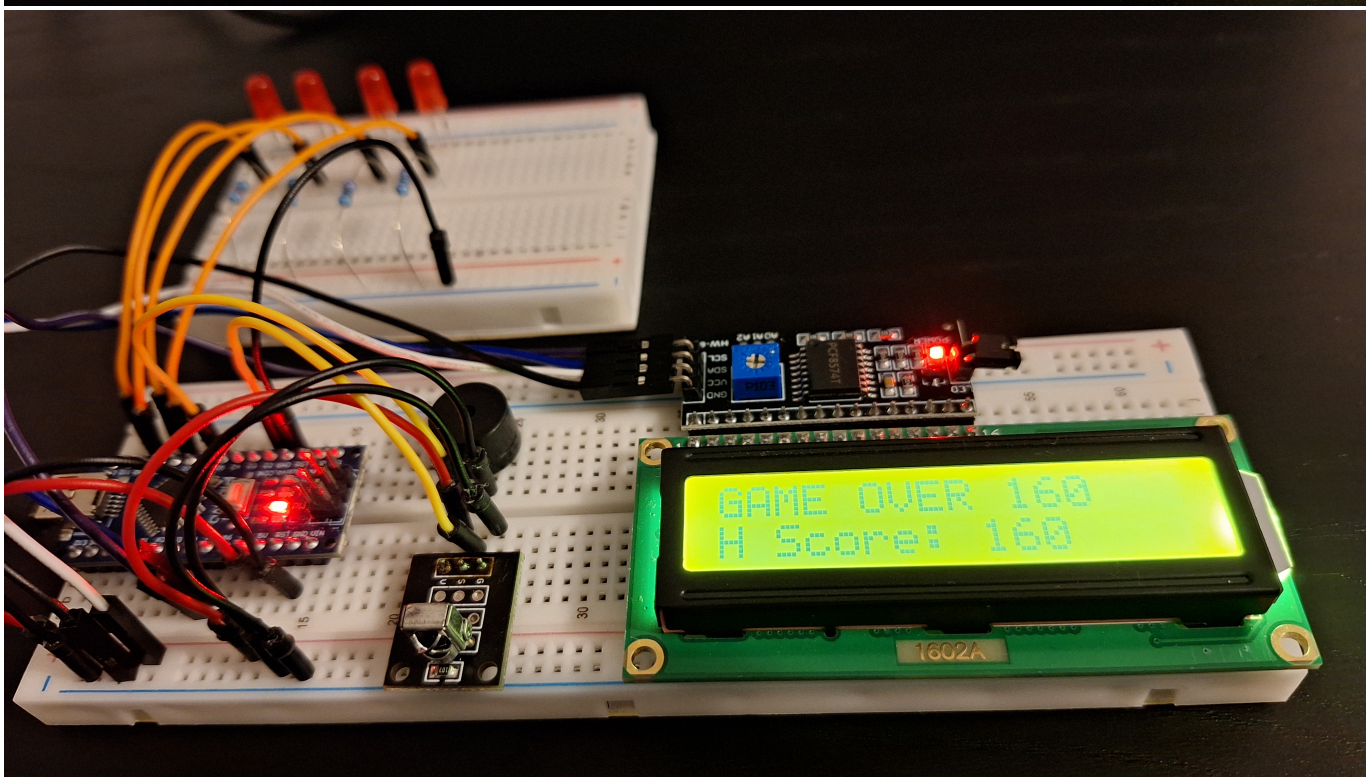
- `setSeed(seed)` — inițializează generatorul cu o valoare externă pentru diversificarea secvenței de numere aleatoare.
- `nextRand()` — returnează un număr pseudo-aleator pe 8 biți folosind algoritmul `xorshift32` usor modificat. Acesta lucreaza doar cu operații bitwise și produce rapid valori pe 8 biți, folosite ulterior pentru selectarea celor 4 LED-uri.

Rezultate Obţinute









Demo

Un scurt demo al proiectului poate fi găsit aici.

Download

Toate fișierele acestui proiect pot fi găsite aici.

Bibliografie/Resurse

Resurse Hardware:
ATmega328P Datasheet
Arduino Nano Pinout
Autodesk Fusion

Resurse Software:
PlatformIO
IRremote
LiquidCrystal I2C
Xorshift32 PRNG
Generating Tones with Timers

Export Page to PDF