Micro Invaders

Introducere

Prezentarea pe scurt a proiectului:

- Proiectul constă într-un joc interactiv inspirat de „Chicken Invaders”, cu afișaj pe un ecran LCD, ce poate fi controlat cu ajutorul unui joystick și al unui buton.
- Scopul proiectului este de a simula comportamentul unei console de jocuri de tip handheld. Acest concept presupune integrarea tuturor componentelor hardware (microcontrolerul, ecranul, joystick-ul și butonul) într-un format compact și ergonomic.
- Ideea proiectului a pornit de la consolele pe care le-am folosit de-a lungul anilor, cum ar fi Nintendo DS sau Steam Deck, și de la un joc clasic al copilăriei mele: “Chicken Invaders”. Practic, mi-am propus să construiesc propria versiune a unei astfel de console, realizată de la zero și la o scară mult mai mică.
- Proiectul este, de asemenea, un exercițiu bun de optimizare și adaptare a codului unui joc video pentru un microcontroller, care are mai multe limitări (memorie redusă, clock speed scăzut, etc.).

Descriere generală

Jocul este implementat pe microcontroller-ul ATmega328P și include următoarele funcționalități:

- Afișaj video: Comunicarea cu ecranul LCD se realizează prin protocolul SPI.
- Controlul mișcării: Coordonatele navei sunt controlate prin joystick. Tensiunile potențiometrelor aferente axelor X și Y sunt convertite în valori numerice prin modulul ADC integrat pe microcontroller.
- Acțiune (“shoot”): Butonul este conectat la un pin digital (GPIO). Pentru a filtra zgomotul mecanic (contact bounce) fără a folosi funcții blocante precum delay(), se utilizează un algoritm de debouncing bazat pe Timere.

Hardware Design

Bill of Materials

Nr. crt.	Nume componentă	Specificații	Cantitate
1	Arduino Nano	Placă de dezvoltare bazată pe microcontroller-ul ATmega328P	1
2	Modul ecran LCD	Modul LCD de 1.44'' cu SPI și Controller ST7735 (128×128 px)	1
3	Joystick analogic	Modul joystick cu 2 potențiometre (axa X, axa Y)	1
4	Buton (Push button)	Microîntrerupător tactil	1
5	Breadboard	Placă de test 830 de puncte	1
6	Fire de conexiune	Fire cu izolație PVC (22AWG)	1 set
7	Cablu USB	Pentru alimentare și programare	1

Descriere componente

- Placa Arduino Nano: Aceasta conține microcontroller-ul ATmega328P care gestionează logica jocului, afișajul pe display și citește inputurile date de joystick și butoane.
- Ecran LCD: Pe acest ecran va fi afișat jocul “Micro Invaders” prin protocolul SPI.
- Joystick: Acesta conține 2 potențiometre care determină poziția navei pe ecran prin maparea valorilor sale la coordonatele X și Y.
- Push button: Folosit pentru a trage cu proiectile către inamici.

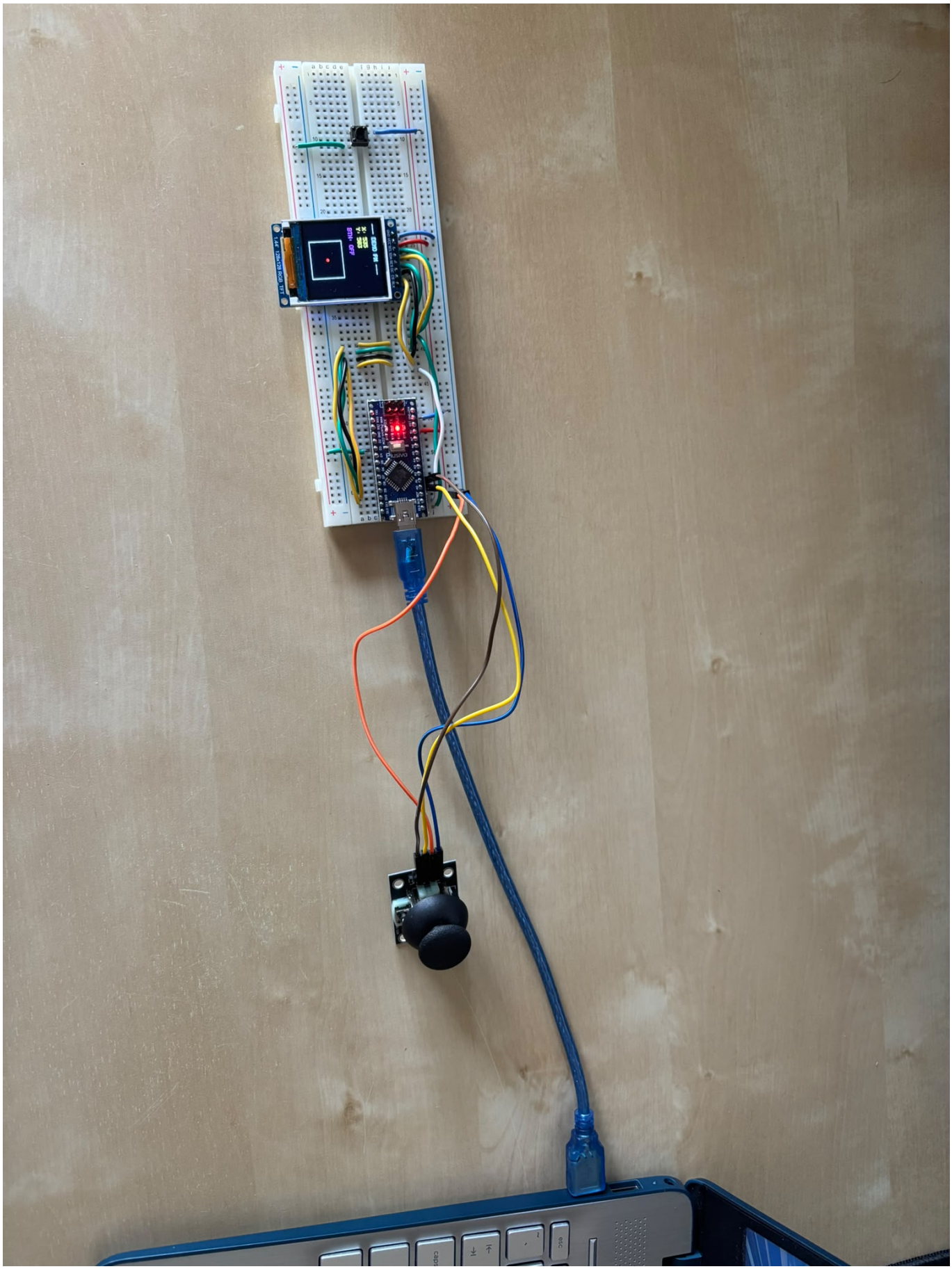
Pinout

Componentă	Pin Componentă	Pin Arduino Nano	Port / Registru ATmega328P	Descriere
Display LCD	VCC	5V	—	Alimentare logică ecran.
	GND	GND	—	Masă circuit.
	SCL	Pin 13	PORTB (PB5)	**Hardware SPI Clock**: Sincronizează transferul rapid de date.
	SDA	Pin 11	PORTB (PB3)	**Hardware SPI MOSI**: Linie master-to-slave pentru trimis pixeli.
	RES	Pin 9	PORTB (PB1)	Reset hardware la inițializarea ecranului.
	DC	Pin 8	PORTB (PB0)	**Data/Command**: Schimbă starea între comenzi (0) și pixeli (1).
	CS	Pin 10	PORTB (PB2)	**Chip Select**: Activează/dezactivează magistrala SPI pentru ecran.
	BLK	3.3V	—	Alimentare LED Backlight (lumină fundal).
Joystick	5V	5V	—	Alimentare potențiometre interne.
	GND	GND	—	Referință masă (0V).
	VRx	Pin A0	ADC0 (Canal 0)	Convertor Analog-Digital: Citește poziția X (valori 0-1023).

	VRy	Pin A1	ADC1 (Canal 1)	Convertor Analog-Digital: Citește poziția Y (valori 0-1023).
Push Button	Pin 1	Pin 3	PORTD (PD3)	Citire rapidă stare buton. Are Pull-up intern activat.
	Pin 2	GND	—	Închide circuitul la masă când butonul este apăsat.

Schematic

Demo



Link video: https://youtu.be/86-g_5LCc6M?si=7hPv4MY68I2BOmY2

Software Design

Mediu de dezvoltare

Aplicația a fost dezvoltată pe VS Code cu extensia PlatformIO.

Librării şi surse 3rd-party

Pentru controlul ecranului TFT și randarea primitivelor grafice, au fost utilizate următoarele librării externe:

- **Adafruit_GFX Library**: Oferă funcțiile de bază pentru desenarea figurilor geometrice (linii, dreptunghiuri).
- **Adafruit_ST7735 Library**: Asigură comunicarea SPI dintre ATmega328P și modulul LCD ST7735.

Structuri de date

- **struct Enemy**: Grupează variabilele de poziție curentă (x, y), coordonatele anterioare utilizate pentru ștergerea grafică (oldX, oldY), poziția de referință pentru mișcarea sinusoidală (baseY) și un indicator boolean de stare (alive). Inamicii sunt stocați într-un vector de astfel de structuri.
- **struct Spark**: Reprezintă proiectilul lansat de inamici. Stochează variabilele de poziție (curentă și anterioară), viteza individuală de cădere pe axa Y (speed) și starea curentă a acestuia (active) pentru a permite reutilizarea structurii.
- **Variabilele specifice jucător**: Spre deosebire de inamici, starea jucătorului este gestionată prin variabile globale individuale, deoarece acesta reprezintă o entitate unică pe ecran și nu necesită gruparea într-un vector. Acestea reține poziția curentă, coordonatele anterioare pentru ștergerea grafică, scorul acumulat și starea jocului (“isGameOver”).
- **Matrice binare stocate în memoria Flash (PROGMEM)**: Pentru designul grafic al navelor și al inamicilor s-au utilizat vectori constanți de tipul “unsigned char”. Această abordare reține bitmap-urile direct în memoria Flash, urmând a fi citite de către metoda de redare pe ecran “drawBitmap()”.

Algoritmi implementaţi

- **Algoritmul de detecţie a coliziunilor pe bază de distanţă absolută (AABB simplificat)**: Pentru verificarea coliziunilor (Glonţ-Inamic, Inamic-Jucător, Scânteie-Jucător), s-a implementat un algoritm bazat pe determinarea distanţei absolute pe axele X şi Y. Acesta verifică suprapunerea casetelor (boxes) de încadrare ale obiectelor, fiind o variantă optimizată matematic ce evită

operaţiile costisitoare hardware (precum ridicarea la pătrat şi radicalul) specifice calculului distanţei Euclidiene.

- **Algoritmul de debouncing**: Implementat în rutina de întrerupere (ISR) a Timer-ului 2. Algoritmul monitorizează starea brută a pinului digital legat la buton, validează schimbarea de stare doar dacă aceasta rămâne stabilă timp de 20ms şi elimină zgomotul provocat de contactul mecanic.
- **Algoritmul de generare a mişcării sinusoidale**: Pentru a obţine traiectoria curbată şi fluidă a inamicilor pe axa Y, s-a implementat o ecuaţie de oscilaţie armonică bazată pe funcţia trigonometrică “sin()”, unde poziţia finală y este calculată dinamic în funcţie de poziţia curentă x:

$$y = baseY + \sin(x \times 0.15) \times 8$$

Funcții implementate

Logica jocului, controlul perifericelor și randarea grafică sunt gestionate prin intermediul următoarelor funcții:

- **setup()**: Configurarea inițială a sistemului. Inițializează regiștrii de direcție ai porturilor (DDRB, DDRD) stabilind pinii ca ieșire/intrare, configurează multiplexorul ADC și registrul de control, setează Timer 2 în modul CTC pentru generarea întreruperilor la 1ms și pornește ecranul TFT, afișând ecranul inițial.
- **loop()**: Nucleul asincron al programului. Rulează pe baza cadrelor de timp (gameTicks), executând logica jocului o dată la ~30ms pentru a asigura un framerate stabil. Gestionează citirea potențiometrelor joystick-ului și apelează funcțiile de actualizare (updateEnemies(), updatePlayerBullet(), updateSparks()). De asemenea, loop() conține și cod specific jucătorului, precum definirea limitelor ecranului, desenarea navei, afișarea scorului și verificarea dacă jucătorul a tras spre inamici.
- **ISR(TIMER2_COMPA_vect)**: Funcția de întrerupere a Timer-ului 2. Incrementează ceasul intern al jocului și rulează algoritmul de debouncing hardware pentru citirea stabilă a butonului de tras.
- **readADC(uint8_t channel)**: Schimbă canalul multiplexorului ADC, pornește conversia hardware, așteaptă finalizarea acesteia și returnează valoarea brută (0-1023) a joystick-ului.
- **spawnSpark()**: Caută un slot liber în vectorul de proiectile pentru a activa și lansa o „scânteie” cu viteză aleatorie direct de sub inamicul care trage.
- **respawnEnemies()**: Activează toți cei 8 inamici și îi așază ordonat pe două rânduri, la coordonatele de pornire, pentru a genera un nou val de joc.
- **updateEnemies()**: Actualizează poziția pe ecran a inamicilor. Verifică dacă grupul a atins marginile laterale pentru a inversa direcția de mers și calculează șansa aleatorie (~2%) ca un inamic să genereze un proiectil la oricare cadru. Pentru efectul sinusoidal de mișcare al flotei, se păstrează valoarea Y de bază (baseY), care este utilizată ulterior în ecuația de actualizare a coordonatei Y (flock[i].y). De asemenea, funcția gestionează și logica de coliziune dintre jucător și inamici.
- **updatePlayerBullet() / updateSparks()**: Funcțiile responsabile de mișcarea proiectilelor. Mută pozițiile pe ecran (în sus pentru jucător, în jos pentru inamici), aplică algoritmul de ștergere a

pixelilor vechi, redesenează obiectele și verifică continuu coliziunile cu țintele prin calcularea distanțelor absolute (AABB).

- **drawShip() / drawEnemy()**: Funcții dedicate desenării acestor obiecte pe ecran. Pentru a obține un aspect detaliat, designul de tip pixel-art al navei și al inamicilor (microcontrolere, de unde și numele jocului, „Micro Invaders”) a fost creat manual prin definirea unor matrice de tip bitmap. Pe lângă aceste bitmap-uri, au fost adăugate detalii grafice suplimentare folosind funcția “fillRect()”. Pentru ștergerea sprite-urilor de pe pozițiile vechi, se redesenează suprafața locală utilizând culoarea fundalului (ST77XX_BLACK).
- **showGameOver()**: Oprește jocul, colorează ecranul în roșu, afișează scorul final și blochează execuția într-o buclă de așteptare până la o nouă apăsare a butonului.
- **resetGame()**: Readuce toate variabilele globale la starea inițială, curăță ecranul, resetează vectorii de proiectile și apelează funcția de respawn pentru a genera un val nou de inamici.

Rezultate Obţinute

Rezultatul final este o mini-consolă handheld construită pe un breadboard, cu un design compact. Ecranul, butonul și placa Arduino Nano sunt perfect integrate, singurele elemente care “ies în afară” fiind cablul USB și joystick-ul. Pe această platformă rulează un joc arcade, interactiv și suficient de provocator.

Link video: https://youtu.be/wmc5tNVgK2Y?si=Ei97VLyqL2z5He_Q

Concluzii

Implementarea hardware m-a provocat să mă gândesc mult mai bine la organizarea componentelor pe placă. Cu această ocazie, am învățat să folosesc letconul, fiind necesar să lipesc un header de pini pentru ecranul LCD. De asemenea, am tăiat și am dezizolat firele la dimensiuni cât mai mici pentru a obține o lizibilitate mai mare a circuitului, lucru care a făcut consola mult mai ușor de utilizat.

Partea software mi-a plăcut în mod deosebit deoarece am văzut ce înseamnă să programezi un joc de la zero pe un microcontroler și pe un ecran mic, de 128×128 pixeli. Librăriile utilizate au fost de mare ajutor pentru definirea și redarea elementelor grafice, oferind multe funcționalități de bază. Pe parcursul proiectului s-a resimțit destul de tare memoria limitată a microcontrolerului ATmega328P, fiind necesar sa gestionez resursele cu atenție. Codul final ocupă ~47% din memoria FLASH și ~17% din memoria RAM.

În final, consider că am reușit să obțin un proiect complet, care mi-a dezvoltat atât abilitățile practice, cât și cele de programare.

Download

Link Github: https://github.com/AlexStoianACS/Micro-Invaders

Export to PDF