GameBoy

Introducere

- Ce face: Proiectul nostru este un mini GameBoy bazat pe Arduino Nano, care poate rula jocuri clasice retro precum Tetris, Space Invaders și Super Mario. Acesta include un display, butoane pentru controlul direcției și a acțiunii, și un buzzer pentru sunet.

- Scop: Recreaza experiența de joc a unui console portabil retro folosind hardware simplu și accesibil. Acesta este un proiect educativ care îmbină învățarea conceptelor de electronică și programare cu un rezultat fun și interactiv.

- Inspiratie: Ideea a pornit de la dorința de a construi un mini-console de jocuri portabil fiind pasionat de jocuri retro si modificand console, astfel ca am decis ca ar fi interesat o consola mica custom.

- Utilitate: Util pentru alții deoarece poate încuraja învățarea interactivă a electronicii și programării, iar pentru mine este o oportunitate de a aplica concepte teoretice într-un proiect practic.

Descriere generală

Microcontroler - Arduino Nano: Arduino Nano va acționa ca unitatea principală de control a întregului sistem. Acesta va gestiona inputurile de la butoane, va rula jocul și va controla afișajul OLED/TFT, iar prin intermediul unui buzzer va emite sunete.

Display OLED/TFT (SPI/I2C): Display-ul va fi folosit pentru a reda grafica jocurilor. Acesta va comunica cu Arduino prin SPI sau I2C, în funcție de tipul de display ales.

Butonul D-pad (4 direcții) și 2 butoane A/B: Butoanele vor fi folosite pentru a controla mișcarea (sus, jos, stânga, dreapta) și acțiunile în joc (ex: sări, trage).

Buzzer: Buzzer-ul va emite sunete pentru diverse evenimente din joc (ex: când se câștigă un nivel sau când apare o coliziune).

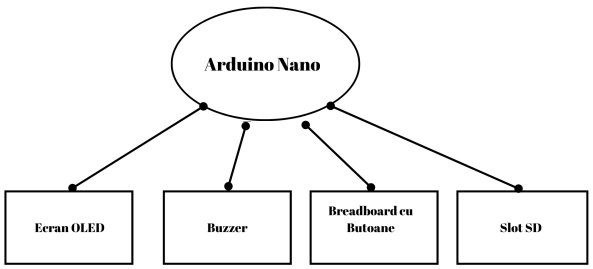
Card SD: Intr-o varianta de optimizare a fost atasat si un card SD in vederea incadrarii tuturor jocurilor in acelasi “cod”, fara reusita insa.

Codul jocurilor (Breakout, Space Invaders, Super Mario): Codul jocurilor va fi stocat în memoria flash a Arduino-ului și va interacționa cu hardware-ul pentru a genera jocuri interactive. Nivelele și grafica vor fi stocate în PROGMEM pentru a economisi memoria RAM.

Utilizatorul va putea selecta folosind butonul A jocul pe care il doresc si apoi va rula ca variantele originale.

Hardware Design

Componentă	Cant.	Link aprovizionare / Datasheet
Arduino Uno R3	1	Arduino Uno
Display OLED 128x64	1	OLED SSD1306
Modul cititor card SD	1	Modul SD Card
Card microSD (min. 2GB)	1	Orice card microSD FAT32
Fire jumper M-M	10+	Set jumper
Breadboard (optional)	1	Breadboard



1. Descrierea hardware-ului

Arduino UNO Microcontroller: ATmega328P

RAM: 2KB, Flash: 32KB

Tensiune operare: 5V

Interfețe folosite:

I2C: pentru display

SPI: pentru modulul SD card

Display OLED 128×64 (I2C) Model: SSD1306

Rezoluție: 128×64 pixeli

Interfață: I2C

Adresă implicită: 0x3C

Pini folosiți:

SDA → A4

SCL → A5

Modul SD Card (SPI) Interfață: SPI

Suportă FAT16/FAT32

Pini folosiți:

CS (Chip Select) → D10

MOSI → D11

MISO → D12

SCK → D13

2. Distribuția pinilor Arduino

3. Justificare

 1. OLED Display 128×64 (SSD1306) Interfață: I2C (Inter-Integrated Circuit)

Pini folosiți:

SDA → A4 (Serial Data)

SCL → A5 (Serial Clock)

De ce pe acești pini?

Pe Arduino Uno, interfața I2C este hardware-mapată exclusiv pe pinii A4 (SDA) și A5 (SCL). Nu pot fi schimbați fără librării speciale.

I2C permite conectarea mai multor dispozitive pe aceeași magistrală (display, senzori etc.) – eficient în proiecte cu pini limitați.

 Modul Card SD (cu interfață SPI) Interfață: SPI (Serial Peripheral Interface)

Pini folosiți (standard SPI pe Arduino Uno):

MOSI → D11 (Master Out Slave In – date trimise de Arduino)

MISO → D12 (Master In Slave Out – date primite de la SD)

SCK → D13 (Serial Clock – semnal de ceas)

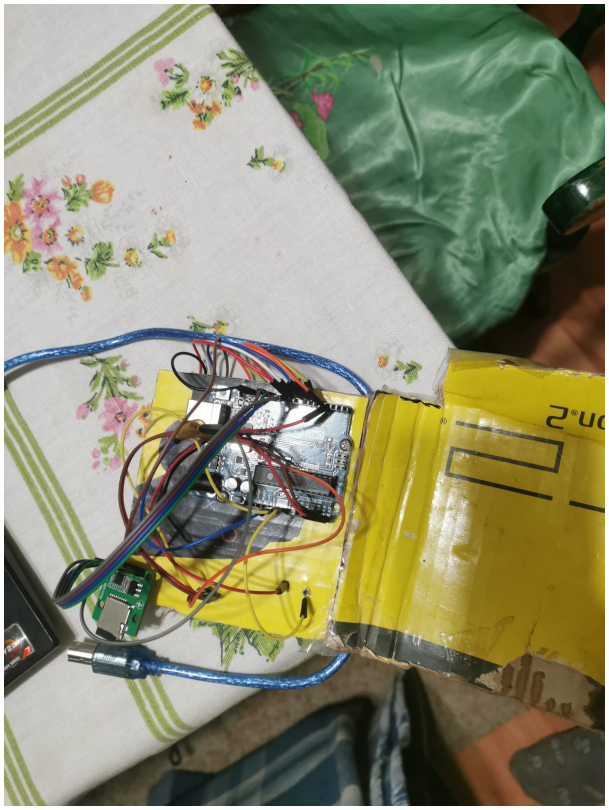
CS → D10 (Chip Select – activare SD)

De ce pe acești pini?

Pe Arduino Uno, interfața SPI este hardware-mapată pe pinii D11, D12, D13.

CS (Chip Select) poate fi orice pin digital liber, dar convențional este D10 pentru compatibilitate cu librăriile.

4. Galerie Proiect



Software Design

Descrierea codului aplicaţiei (firmware):

- mediu de dezvoltare (Arduino)

Meniu Principal (`menu/`): - Implementat folosind Arduino și un display OLED SSD1306 - Oferă o interfață grafică pentru selecția jocurilor - Include efecte sonore pentru feedback - Gestionează tranzițiile între diferite stări (boot, meniu, jocuri)

Cele Trei Jocuri:

1. **Joc1 (Super Mario):**

1. Implementare 2D a unui joc de platformă
2. Controlul personajului prin D-pad
3. Sistem de coliziuni și fizică simplă
4. Terenul este generat procedural pe măsură ce jucătorul avansează. Acest lucru este evident din mai multe componente cheie:

Terenul este stocat într-un buffer circular (vector) numit terrainHeights cu o dimensiune de TERRAIN_SIZE:

```
      Pe măsură ce jucătorul se mișcă înainte, noi blocuri de teren sunt
generate în timp real prin funcția advanceTerrainWindowForward():
      Generarea terenului este gestionată de generateNextBlockHeight() care
creează modele variate de teren, inclusiv:
      Găuri (spații goale)
      Platforme
      Trepte (sus și jos)
      Secțiuni plate
      Sistemul folosește o abordare de tip fereastră unde:
      Doar o porțiune din teren este păstrată în memorie
      Terenul vechi este eliminat pe măsură ce se generează teren nou
      Terenul se înfășoară folosind aritmetica modulo pentru a menține
buffer-ul circular
      Generarea terenului este randomizată dar urmează anumite modele pentru
a asigura jucabilitatea:
      Menține o diferență maximă de înălțime
      Creează găuri de diferite dimensiuni
      Generează platforme la anumite înălțimi
      Creează modele de trepte care urcă și coboară
```

1. Coliziunile se bazează pe verificarea poziției pixelilor: Dacă jucătorul atinge un bloc (1), mișcarea este oprită. Coliziunile sunt implementate pentru sus/jos/stânga/dreapta și pentru detecția solului. Sistemul previne „căderea prin sol” și permite sărituri doar dacă jucătorul se află pe o platformă.
2. Mișcare: Jucătorul are viteză verticală și orizontală. Se aplică gravitație (un număr constant adunat la viteza verticală). Săriturile sunt limitate la o durată sau înălțime maximă.

2. **Joc2 (Space Invaders):**

1. Sistem de scor
2. Sistem de inamici: Inamicii sunt aranjați în rânduri, fiecare inamic fiind un obiect cu coordonate și stare (activ/inactiv). Se deplasează de la stânga la dreapta în sincron. Dacă un inamic ajunge la margine, toți coboară cu o linie și schimbă direcția.
3. Sistem de proiectile: Proiectilele sunt stocate într-un array de obiecte active. La apăsarea butonului A, se creează un nou glonț cu poziția curentă a jucătorului. Se verifică coliziunile între proiectile și

inamici. Dacă o coliziune este detectată, inamicul și glonțul dispar, iar scorul crește.
4. Boss: După distrugerea unui număr de inamici, apare un inamic special (“boss”). Are viață mai mare și se mișcă independent. Necesită mai multe proiectile pentru a fi distrus.

3. **Joc3 (Breakout):**

1. Scop: Jucătorul controlează o paletă care lovește o minge cu scopul de a distruge blocurile de sus.
2. Sistem de blocuri distructibile
3. Fizică pentru mișcarea mingii: Mingea are o velocitate (vx, vy) și poziție (x, y). Se deplasează la fiecare ciclu de joc, iar direcția se schimbă la coliziune: Cu margini → inversare vx, Cu tavan → inversare vy, Cu paleta → inversare vy și modificare dx în funcție de locul lovirii, Cu blocuri → inversare dy și eliminarea blocului
4. Blocuri distructibile: Reprezentate ca o matrice de 1 și 0. La coliziune, blocul devine 0 și mingea își schimbă direcția.
5. Multiplicare bile (power-up): După un scor atins sau după lovirea unui bloc special, se adaugă o a doua minge în joc. Fiecare minge este gestionată independent (poziție, coliziune, fizică). Se adaugă provocare și complexitate în gameplay.

4. **Optimizari**:

1. Debouncing software pentru butoane – elimină dublările de input
2. Reducerea consumului RAM: Folosirea tipurilor PROGMEM pentru texte constante, Eliminarea bitmapurilor mari sau nefolosite, Reducerea dimensiunii codului (Cod modular, fără duplicări), Separare clară între logică, randare și input, Ajustarea frecvenței PWM pentru sunete mai clare la buzzer

Încercările de Integrare Software a Meniului și Jocurilor

Pentru a aduce toate componentele proiectului împreună – meniul principal și cele 3 jocuri – au fost testate trei soluții majore de structurare software. Fiecare variantă a venit cu avantaje și limitări, iar mai jos sunt explicate pe rând.

1. Structură cu 4 fișiere separate: Meniu + 3 jocuri independente Descriere:

Fiecare joc era implementat într-un fișier .ino separat.

Meniul principal era, de asemenea, un sketch separat.

Se trecea manual din IDE între fișiere pentru a compila și încărca un joc anume.

Avantaje:

Codul era clar compartimentat și ușor de gestionat.

Fiecare fișier se compila fără erori, separat.

Limitări:

Nu exista o tranziție reală între meniu și jocuri — doar testare individuală.

Nu permitea rularea unui singur firmware complet, ceea ce era un dezavantaj pentru experiența utilizatorului.

Încărcarea manuală era nepractică pentru un proiect final funcțional.

2. Integrarea Meniului cu un joc + celelalte două separate Descriere:

Meniul principal era integrat într-un sketch împreună cu unul dintre jocuri.

Celelalte două jocuri rămâneau în fișiere separate.

Se testa funcționalitatea meniului și a tranziției către un joc.

Avantaje:

Permitea testarea logicii de selecție și tranziție între meniu și un joc real.

A demonstrat că este posibilă integrarea parțială într-un singur fișier.

Limitări:

Limitări de memorie au apărut deja la adăugarea unui singur joc complet.

Includerea tuturor celor 3 jocuri în același sketch devenea imposibilă pe Arduino Uno din cauza spațiului limitat de flash și RAM.

3. Integrarea completă într-un singur sketch (Eșec) Descriere:

S-a încercat integrarea meniului și a celor 3 jocuri într-un singur fișier .ino.

Codul menținea un state machine (enum sau variabilă) care comuta între stări: meniu / joc1 / joc2 / joc3.

Se folosea Adafruit_GFX + Adafruit_SSD1306 pentru randarea pe ecran.

Probleme întâlnite:

La un anumit punct, ecranul OLED nu mai răspundea deloc.

După debugging, s-a constatat că librăria Adafruit_SSD1306 bloca execuția în cazul în care primea prea multe apeluri către display() într-un sketch foarte mare.

Consumul de RAM era aproape de 100% ⇒ crăpări ale programului.

S-a încercat trecerea la U8g2 (bibliotecă mai stabilă și mai versatilă), însă:

U8g2 folosește mai mult RAM și nu era sustenabil pe Arduino Uno pentru proiectul complet.

3. Încercare finală: Boot de pe SD Card (Eșec) Idee:

Deoarece memoria internă era insuficientă, s-a încercat o soluție „avansată” de a stoca fișiere .ino sau binare pe SD card și a le încărca dinamic (ca bootloader sau loader extern).

Ce s-a făcut:

Integrarea cardului SD și implementarea funcției SD.begin() și a citirii de fișiere binare.

Încercarea de a scrie un bootloader simplificat care să detecteze și să ruleze un fișier .bin sau .hex de pe card.

Rezultat:

Arduino Uno nu suportă execuția dinamică de cod din memorie externă.

Nu există suport nativ pentru „self-reprogramming” din sketch user-space.

Soluția ar necesita un bootloader scris la nivel de AVR (în afara Arduino IDE), prea complex pentru constrângerile proiectului.

Concluzie

Cea mai viabilă soluție:

Se păstrează meniu + un singur joc într-un sketch, iar celelalte se încarcă manual pentru demonstrație.

Motive:

Limitele de RAM și flash pe Arduino Uno.

Conflictele între librării (în special Adafruit vs SD vs U8g2).

Ecranul OLED devine instabil la coduri mari.

Necesitatea unei interfețe clare pentru testare și prezentare.

Biblioteci Folosite și Motivația Alegerii

Alegerea acestor biblioteci a fost dictată de compatibilitatea cu platforma Arduino Uno și dimensiunea restrânsă a memoriei flash (32 KB). Toate bibliotecile sunt bine întreținute și documentate.

Elementul de Noutate

Elementul de noutate al proiectului constă în: - Integrarea a trei jocuri clasice într-un singur sistem portabil - Implementarea unui meniu interactiv cu feedback sonor - Sistem de control unificat pentru toate jocurile - Adaptarea jocurilor pentru un display OLED de dimensiuni reduse

Utilizarea Funcționalităților din Laborator

Funcționalitățile din laborator sunt utilizate în următoarele moduri: 1. **Controlul I/O:**

1. Utilizarea pinilor digitali pentru butoane
2. Implementarea debouncing-ului pentru butoane

2. **Comunicare Serială:**

1. Debugging prin Serial Monitor
2. Comunicare I2C cu display-ul OLED

3. **Gestionarea Muzica:**

1. Controlul buzzer-ului pentru efecte sonore cu PWM folosind tune()

* 5. Scheletul Proiectului și Interacțiunea

Arhitectura Software: 1. **Sistem de Stări:**

1. STATE_BOOT: Secvența de pornire
2. STATE_MENU: Meniul principal
3. STATE_GAME1/2/3: Stările jocurilor

2. **Interacțiunea între Componente:**

1. Meniul principal gestionează selecția jocurilor
2. Fiecare joc poate reveni la meniu
3. Sistem unificat de control pentru toate jocurile

3. **Validarea Funcționalităților:**

1. Testarea butoanelor prin `test_dpad/`
2. Verificarea display-ului în secvența de boot
3. Testarea tranzițiilor între stări
4. Validarea sistemului de scor și a coliziunilor în fiecare joc

* Concluzie

Proiectul demonstrează o implementare software robustă care integrează multiple jocuri într-un sistem portabil. Arhitectura modulară permite ușoara adăugare de noi jocuri sau funcționalități. Sistemul de meniu oferă o experiență de utilizare intuitivă, iar implementarea hardware-ului este optimizată pentru performanță pe platforma Arduino.

Rezultate Obţinute

Mai multe versiuni ale aceleasi idei care se complimenteaza in complexitatea proiectului si inoperabilitatea completa a lui

Concluzii

O experienta interesanta dar nu as mai repetea-o, doar cu un proiect mai simplu

Download

Am atasat arhiva cu toate implementarile facute in stadiul lor dinaintea prezentarii

gameboy-all-versions.zip

Jurnal

Draga jurnalule, am lucrat peste 40 de ore dar nu zilnic ci cumulat

Export to PDF