Snake Game

Autor: Grosu Gheorghe

Grupa: 334CD

Introducere

Joc de tip Snake interactiv și captivant. Jocul este afișat pe un ecran LCD și oferă mai multe niveluri și viteze, crescând astfel complexitatea și atractivitatea jocului. Jucătorul își controlează șarpele cu ajutorul unui joystick. Scopul jocului este de a obține cel mai mare scor posibil fără ca șarpele să se ciocnească de el însuși sau de obstacolele de pe ecran.

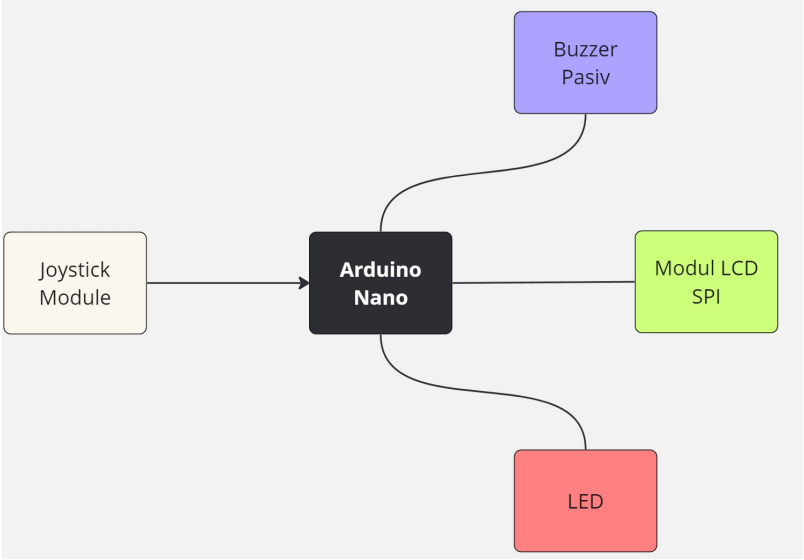
Scorul poate fi vizualizat fie în timpul jocului, fie la final, oferind astfel jucătorului un feedback asupra performanței sale. Proiectul include și un buzzer care emite diferite sunete în timpul jocului.

Am fost un mare fan al acestui joc pe telefoanele vechi (Nokia :)) si mi s-a parut interesant cum as putea implementez eu acest lucru pe un microcontroler.

Utilitatea proiectului vine in primul rand din faptul ca ma va ajuta sa inteleg tot procesul de planificare, arhitectura si legare a partii de software cu cea de hardware.

Descriere generală

Proiectul este un joc implementat pe un dispozitiv care folosește un display pentru afișarea jocului, un buzzer pentru redarea sunetelor și un joystick pentru controlul personajului. Scopul jocului este să eviți coliziunile și să obții cât mai multe puncte. Atunci când se produce o coliziune, scorul este afișat. Jucătorul poate ajusta nivelul de dificultate, inițial setat pe modul “Easy”(nivelul 1), folosind un buton(posibil tot cel din joystick). La apăsarea butonului, viteza de deplasare a personajului crește, harta se schimbă și culoarea LED-ului se modifică pentru a indica nivelul de dificultate actual sau pentru diferite efecte din timpul jocului.



O schemă bloc cu toate modulele proiectului vostru, atât software cât şi hardware însoţită de o descriere a acestora precum şi a modului în care interacţionează.

Exemplu de schemă bloc: http://www.robs-projects.com/mp3proj/newplayer.html

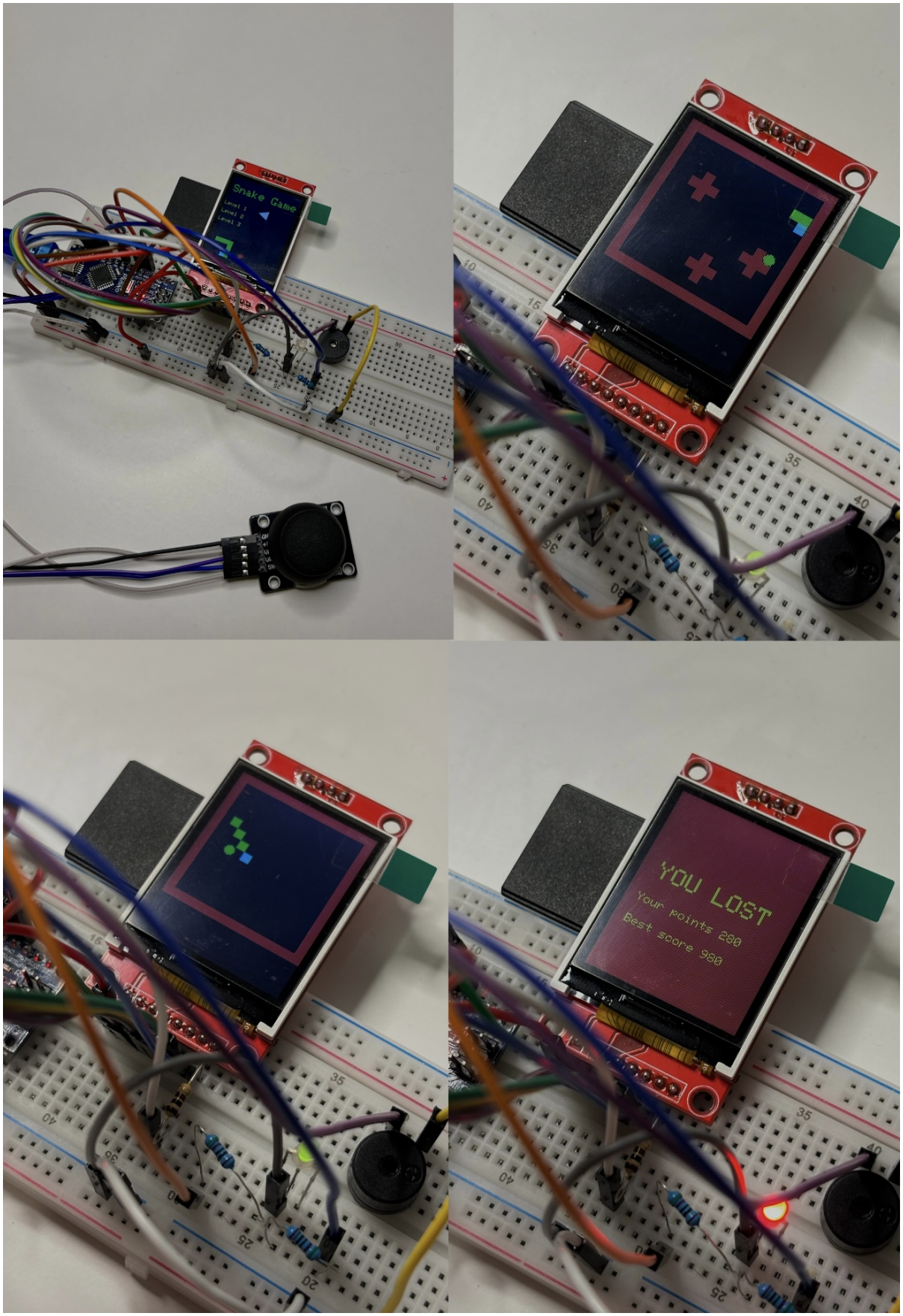
Hardware Design

Lista de piese:

- Placă de Dezvoltare Compatibilă cu Arduino Nano (ATmega328p)
- Modul LCD SPI de 1.8'' (128×160)
- Modul Joystick Biaxial
- Breadboard HQ
- Fire (Mama-Tata, Tata-Tata)
- Buzzer Pasiv
- LED
- Rezistente

Schema hardware

Schema electrica



Software Design

Descrierea codului aplicației (firmware)

1. Mediu de dezvoltare: - **Platformă de dezvoltare**: Arduino IDE

```
  Arduino IDE este utilizat pentru scrierea, compilarea și încărcarea
firmware-ului pe un microcontroller Arduino.
```

2. Librării şi surse 3rd-party:

• **TFT Library**: `Adafruit_ST7735`

- Această librărie este folosită pentru a controla ecranul TFT.

• **SPI Library**: `SPI`

- SPI (Serial Peripheral Interface) este utilizat pentru comunicația cu ecranul TFT.

• **EEPROM Library**: `EEPROM`

- Această librărie permite stocarea persistentă a datelor, cum ar fi scorurile maxime.

• **Tone Library**: `tone()`

- Functia `tone()` este utilizată pentru a genera sunete pe pinul difuzorului.

3. Algoritmi şi structuri implementate: - **Structuri de date**:

`Joystick` Structura `Joystick` se ocupă de citirea pozițiilor analogice ale joystick-ului și de calcularea direcțiilor următoare pentru deplasarea șarpelui.

`GameState`

Structura `GameState` gestionează stările jocului și timpul de cadru pentru fiecare stare, facilitând tranzițiile între stări și verificarea momentului potrivit pentru actualizarea jocului.

`Snake`

Structura `Snake` gestionează pozițiile șarpelui, lungimea sa și funcțiile pentru resetare, mișcare și creștere. De asemenea, verifică dacă șarpele a colizionat cu el însuși.

Șarpele este reprezentat prin structura `Snake`, care conține: * **`length`**: lungimea șarpelui. * **`positions`**: array de tip `byte` ce stochează pozițiile fiecărui segment al șarpelui.

Motive pentru această reprezentare:

1. **Eficiență Memorie**: Folosirea `byte` ocupă doar 1 octet, economisind memorie RAM limitată pe Arduino.

2. **Simplitate și Performanță**: Acces rapid și simplu la pozițiile șarpelui pentru operațiuni de mișcare și creștere.

3. **Gestionare Ușoară a Lungimii**: Lungimea și pozițiile sunt ușor de manipulat pentru actualizări rapide în timpul jocului.

- **Algoritmi**:

1. Controlul direcției șarpelui pe baza intrărilor de la joystick.
2. Verificarea coliziunilor cu pereții, obstacolele și corpul șarpelui.
3. Gestionarea creșterii șarpelui și actualizarea scorului.
4. Generarea aleatorie a pozițiilor merelor.
5. Stocarea și citirea scorurilor maxime din EEPROM.

Funcțiile implementate

Funcții principale:

- **setup()**

1. Inițializează comunicarea serială.
2. Configurează pinii pentru joystick și LED-urile RGB.
3. Inițializează ecranul TFT.
4. Afișează ecranul de selecție a nivelului.
5. Inițializează starea și timpul de cadru pentru joc.

* **loop()**

1. Verifică dacă trebuie să actualizeze starea jocului pe baza timpului de cadru.
2. Citește direcțiile de la joystick.
3. Execută funcțiile corespunzătoare fiecărei stări (selecție nivel, inițializare nivel, rulare nivel, game over).

Funcții de stare:

* **state_level_select()**

1. Permite utilizatorului să aleagă nivelul dorit folosind joystick-ul.
2. Afișează săgeata de selecție pe ecran.
3. Trecerea la starea de inițializare a nivelului atunci când utilizatorul apasă butonul de selecție.

* **state_level_init()**

1. Inițializează nivelul selectat, resetând șarpele și plasând obstacole și mere pe hartă.
2. Trecerea la starea de rulare a nivelului.

* **state_level_running()**

1. Actualizează poziția șarpelui pe baza direcțiilor de la joystick.
2. Verifică coliziunile și gestionează creșterea șarpelui.
3. Actualizează scorul și generează sunete atunci când șarpele mănâncă un măr.
4. Trecerea la starea de game over în caz de coliziune.

* **state_game_over()**

1. Afișează scorul final și scorul maxim pe ecran.
2. Salvează scorul maxim în EEPROM dacă este cazul.
3. Afișează un mesaj de finalizare a jocului.

Funcții utilitare:

1. **level_select_init()**

- Afișează ecranul de selecție a nivelului.

2. **renderIfDifferent(int pos, int tile)**

- Redă graficul corespunzător pentru un tile dacă este diferit de cel precedent.

3. **render_grass(int pos_x, int pos_y)**

- Redă grafica pentru iarbă.

4. **render_snake_head(int pos_x, int pos_y)**

- Redă grafica pentru capul șarpelui.

5. **render_snake_body(int pos_x, int pos_y)**

- Redă grafica pentru corpul șarpelui.

6. **render_apple(int pos_x, int pos_y)**

- Redă grafica pentru un măr.

7. **render_red_apple(int pos_x, int pos_y)**

- Redă grafica pentru un măr roșu.

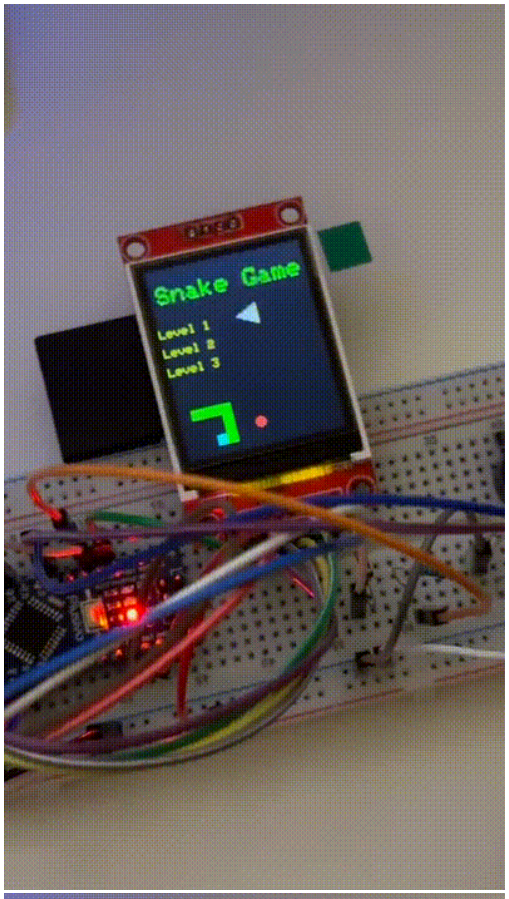
8. **render_obstacle(int pos_x, int pos_y)**

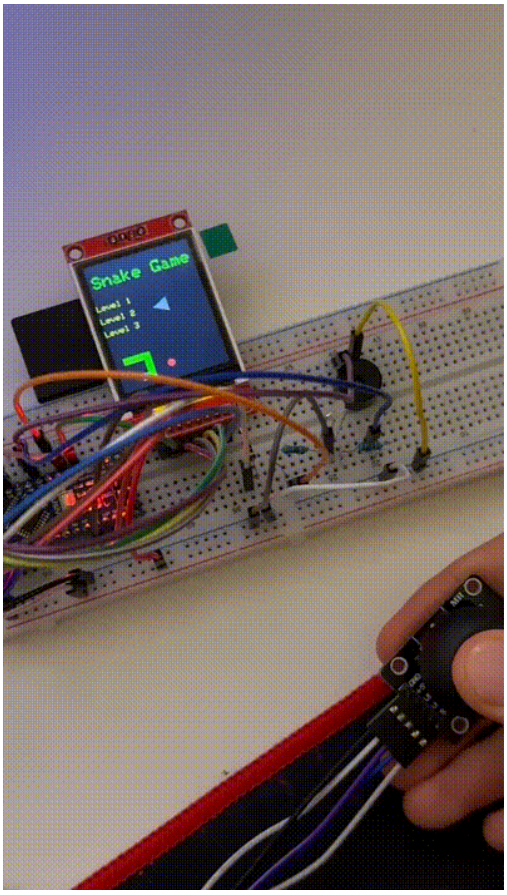
- Redă grafica pentru un obstacol.

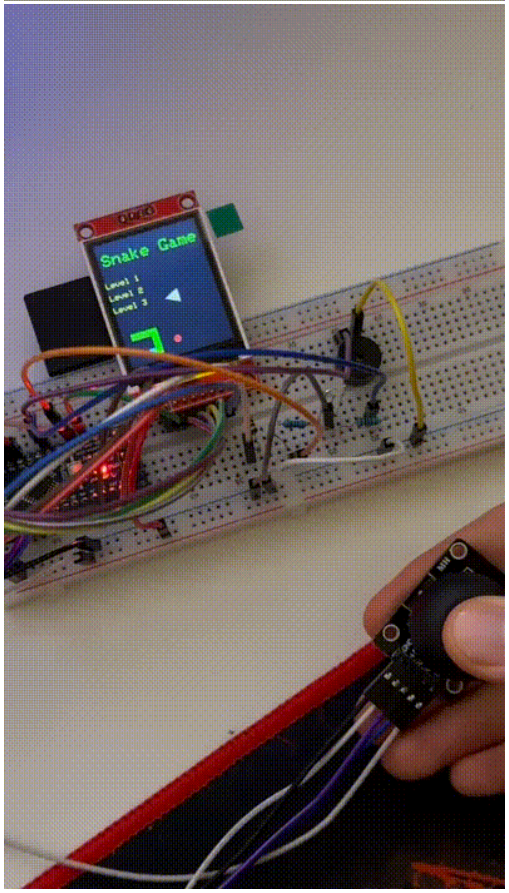
9. **get_random_pos()**

- Generează o poziție aleatorie pentru plasarea unui măr pe hartă.

Rezultate Obţinute


Snake Game
Level 1
Level 2
Level 3


Snake Game
Level 1
Level 2
Level 3


Snake Game
Level 1
Level 2
Level 3

Concluzii

Download

snake.zip

Jurnal

- 28/04/2024 - Alegerea temei proiectului si descrierea sumara
- 29/04/2024 - Comandarea componentelor hardware
- 02/05/2024 - Crearea paginii proiectului si completarea partiala a acesteia
- 12/05/2024 - Adaugarea schemei electrice si a schemei hardware
- 22/05/2024 - Adaugarea software designului
- 24/05/2024 - Finisare proiect

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

https://randomnerdtutorials.com/guide-to-1-8-tft-display-with-arduino/

Export to PDF