Snake

Autor

Nistor Andreea Iuliana 333CB

Introducere

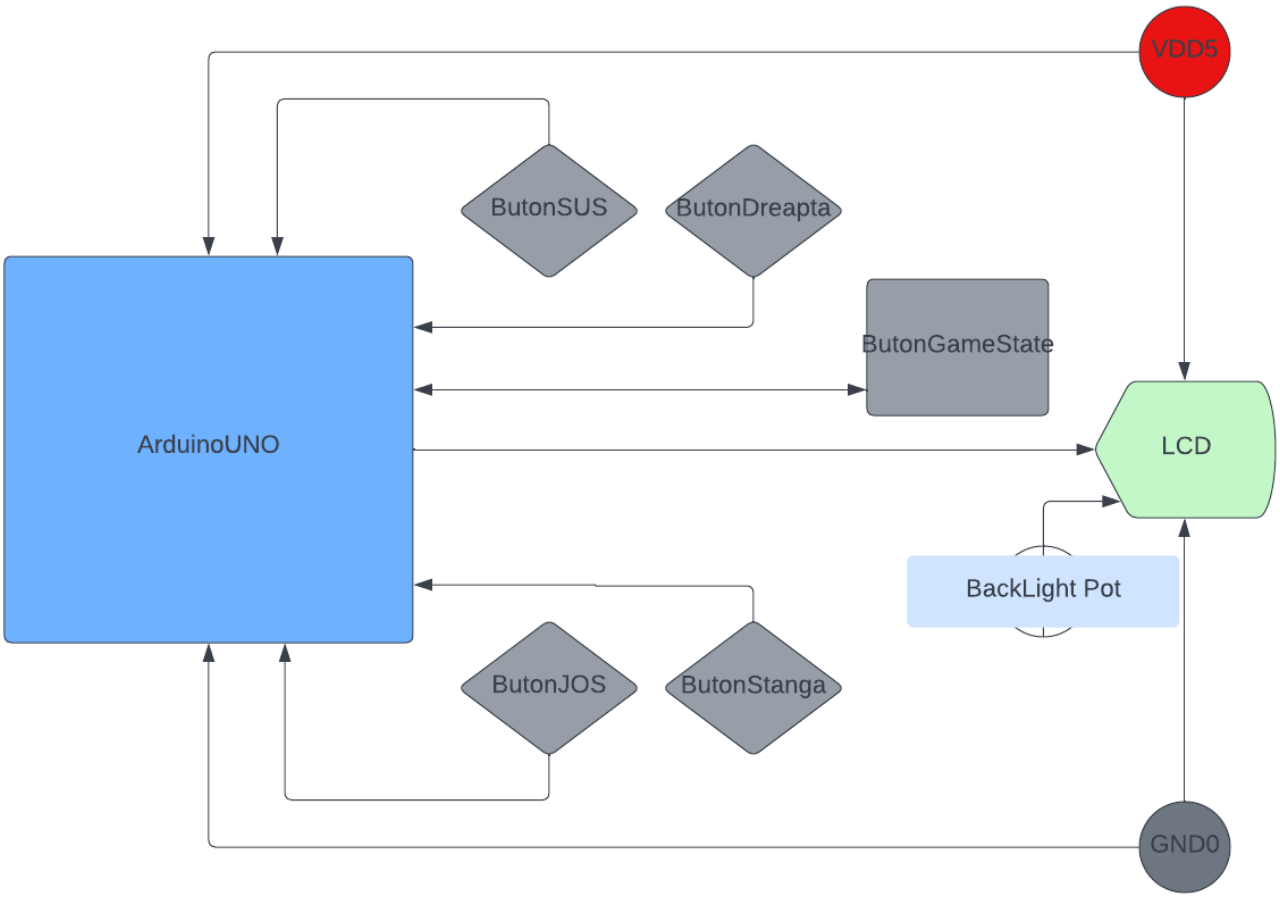
Proiectul constă în implementarea jocului Snake pe un LCD. Am vrut sa fie cât mai aproape de jocul original, astfel că am ales să păstrez controlul șarpelui din 4 butoane pentru stânga, dreapta, sus si jos. Scopul jocului este ca șarpele să crească cât mai mare fără să atingă marginile sau pe el, consumând hrana care apare random pe ecran.

De asemenea, atunci când player-ul pierde, i se da posibilitatea să își scrie numele pentru a-l trece intr-o listă de highscores. Un al cincelea buton este folosit pentru a schimba state-urile curente ale jocului (main page → game play → game over → check highscore & set name → highscores → main page).

Scopul proiectului este implementarea unui joc clasic lansat în 1976 pentru însușirea cunoștințelor dobândite la cursul de Proiectarea cu Microprocesoare.

Descriere generală

Modul de funcționare este simplu: jucătorul va controla din cele 4 butoane și va putea urmări pe LCD mișcările șarpelui și apariția random a hranei.

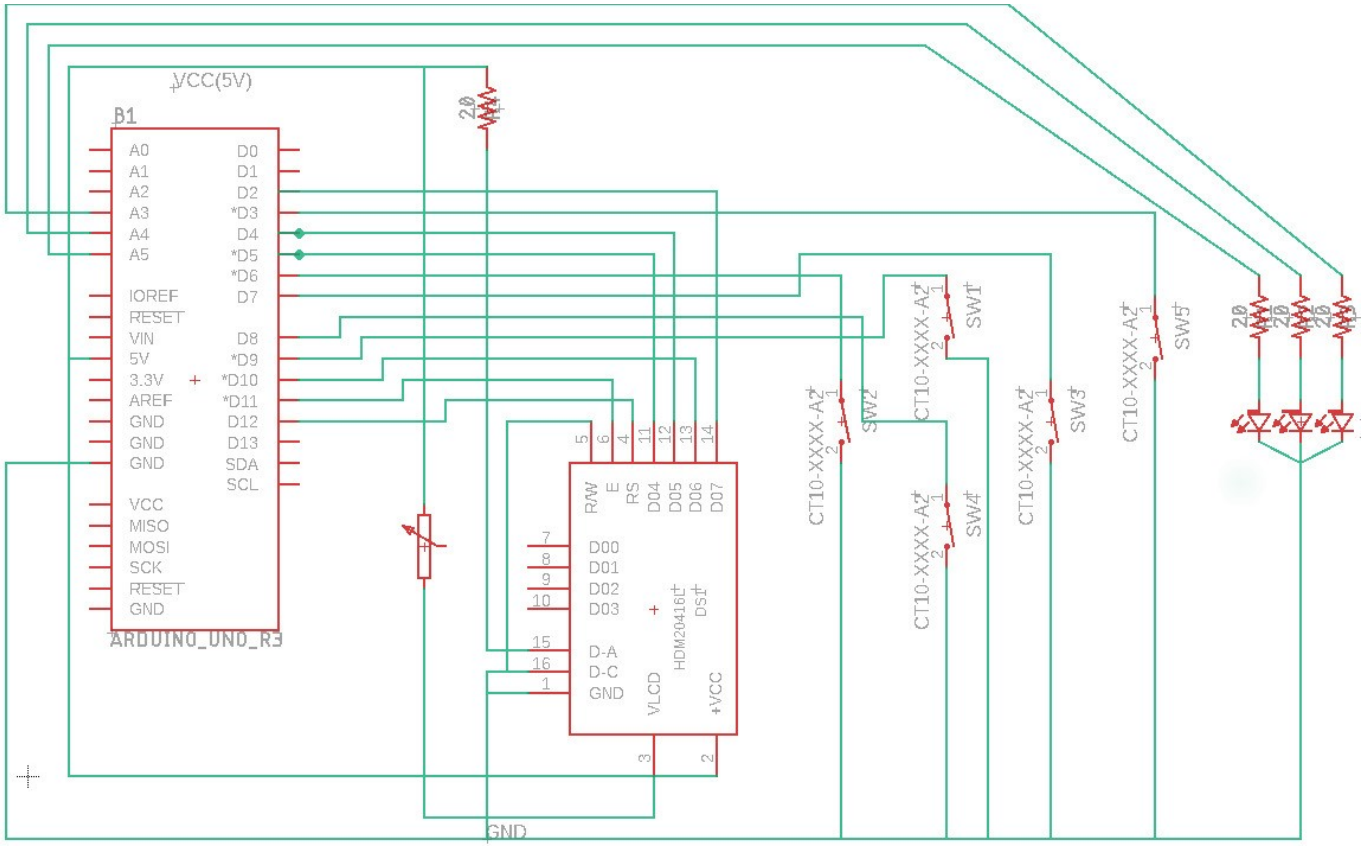


Hardware Design

Listă de piese:

- Arduino Uno
- breadboard
- LCD 16×2
- 4 butoane
- potențiometru
- led RGB

Schema electrică



Software Design

Mediul de dezvoltare folosit este Arduino IDE, schema bloc a fost realizata in Lucidchart, iar schema electrica in EAGLE. De asemenea am folosit librăriile LiquidCrystal.h pentru LCD si EEPROM.h pentru memoria EEPROM.

Variabile si utilizare

Definire pini și macro-uri:

- const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 10, d7 = 2 → pentru LCD LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
- #define redPin A3, greenPin A4, bluePin A5 → pini pentru Led-ul RGB
- #define BUTTON_GameState 3, BUTTON_UP 9, BUTTON_DOWN 8, BUTTON_RIGHT 7, BUTTON_LEFT 6 → pini pentru butoane
- #define GRAPHIC_WIDTH 16, GRAPHIC_HEIGHT 4, DEBOUNCE_DURATION 25 →variable pentru dimensiunea interfeței jocului și a duratei de debounce

Variabile globale:

- enum DisplayItem {GRAPHIC_ITEM_NONE, GRAPHIC_ITEM_A, GRAPHIC_ITEM_B, GRAPHIC_ITEM_NUM};
- enum {GAME_MENU, GAME_PLAY, GAME_LOSE, GAME_WIN, GAME_SCORES,GAME_NAMESCORE}

gameState;
- enum {SNAKE_LEFT,SNAKE_UP,SNAKE_RIGHT,SNAKE_DOWN} snakeDirection; → definire stări joc/snake și setare grafică a interfeței / RAM-ului jocului

- byte block[3] = {B01110, B11111, B01110,} → corp snake
- byte apple[3] = {B00100, B01110, B00100,} → corp apple

- volatile bool Interrupt → verifică dacă a avut loc o întrerupere
- bool pressed → verifică dacă un buton a fost apăsat
- const int max_dimension = GRAPHIC_HEIGHT*GRAPHIC_WIDTH → dimensiunea maximă posibilă
- size_t snakeLength = 0 → inițializare variabilă care o să salveze dimensiunea snake-ului

- int a[4] → folosit pentru a salva highscore-urile in EEPROM
- hue → folosită pentru setarea culorii led-ului RGB

- String Name1thPlace,Name2thPlace,Name3thPlace,Name4thPlace → salvăm în ordine numele care au obținut highscore
- const char chars[] = {'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'} → salvăm literele pentru setarea numelui în caz de highscore
- char setName[3] → o variabilă temporară în care salvam valorile anterioare ale numelor

- struct Pos {uint8_t x=0, y=0;};
- struct Pos applePos;
- struct Pos snakePosHistory[max_dimension] → poziții initiale snake/apple

- unsigned long lastGameUpdateTick = 0 → salvează progresiv timpul ultimului frame al jocului
- unsigned long gameUpdateInterval = 850 → rata de refesh a jocului
- uint8_t graphicRam[GRAPHIC_WIDTH*2/8][GRAPHIC_HEIGHT] → definire dimensiune RAM

Funcții

- void BtnInterrupt() → funcția de întrerupere a butonului BUTTON_GameState
- bool debounce_activate_edge(unsigned long* debounceStart), void debounce_deactivate(unsigned long* debounceStart) → funcții care stabilesc și verifică dacă valoarea logică a unui buton este cea corespunzatoare (atât după apăsarea lui, cât și dacă apar efecte mecanice/zgomot).

- void graphic_flush(), void graphic_generate_characters(), void graphic_clear(), void graphic_add_item(uint8_t x, uint8_t y, enum DisplayItem item) → funcții care generează RAM-ul jocului și prin care jocul, chiar dacă e pe un lcd 16×2, este jucat pe 4 linii. De asemenea, sunt definite și generate modelele snake-ului si ale marului

- void setLedColorHSV(int h, double s, double v)
- void setLedColor(int redValue, int greenValue, int blueValue) → funcțile date la laborator pentru a seta culoarea rgb-ului mai usor.
- void game_new_apple_pos() → generează o poziție random pentru măr. De asemenea, se asigură că un măr nu se generează pe poziția snake-ului
- void game_calculate_logic() → calculează mișcarea și direcția snake-ului, verifică dacă s-a produs o coloziune (perete/ self) și mărește rata de refresh a jocului dacă snake-ul a mâncat un măr (pana la un maxim al scorului de 20)

- void game_calculate_display() → schimbă Game State-ul curent al jocului dacă s-a intâmplat un eveniment (de exemplu: jucatorul a pierdut sau a câștigat)
- void Game_INIT() → starea inițială a jocului (cea inițializată în setup)
- void Game_LOSE() → starea atunci când jucătorul pierde
- void Game_MAIN() → starea în care este prezentat ecranul de început
- void Game_WIN() → starea atunci când jucătorul câștigă
- void Game_PLAY() → stare care generează snake-ul și mărul la început
- void Game_CHECKHIGHSCORES() → verifică dacă jucătorul curent a obținut un scor mai mare decât cei salvați în memoria EEPROM
- void Game_HIGHSCORES() → afișează cei mai buni 4 jucatori cu cele mai mari score-uri și schimbă în memorie top-ul în cazul în care alt jucător a obținut un highscore mai bun
- void Test_SnakeLength() → funcție care pune un 0 în fața scorului în caz ca acesta este mai mic decat 9 (pur estetic)
- void rgb_color() → setează culoarea Led-ului RGB in funcție de cât de mare e scor-ul (verde → albastru → turcoaz → roz → portocaliu)
- void writeStringToEEPROM(int addrOffset, const String &strToWrite) → permite scrierea în memoriea EEPROM
- String readStringFromEEPROM(int addrOffset) → permite citirea din memoria EEPROM

setup()

```
void setup(){
  pinMode(redPin,OUTPUT);
  pinMode(greenPin,OUTPUT);
  pinMode(bluePin,OUTPUT);
  pinMode(BUTTON_GameState, INPUT_PULLUP);
  pinMode(BUTTON_UP, INPUT_PULLUP);
  pinMode(BUTTON_RIGHT, INPUT_PULLUP);
  pinMode(BUTTON_LEFT, INPUT_PULLUP);
  pinMode(BUTTON_DOWN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(BUTTON_GameState), BtnInterrupt,
FALLING);
  graphic_generate_characters();
  Game_MAIN();
  gameState = GAME_MENU;
}
```

Definesc:

- pinii pentru Led-ul RGB ca OUTPUT (low-impedance state) astfel permințând să treacă curent prin ei.
- pinii pentru butoane ca INPUT_PULLUP folosiți să seteze o valoare fixă pentru un pin, în cazul nostru, dacă butonul este apăsat, acesta o să fie activ pe 0.
- o întrerupe pentru butonul BUTTON_GameState aflat pe pinul 3. Atunci când este apasat, generează în avans caracterele jocului și se apelează funcția pentru interfața inițiala a jocului.

loop()

```
if(Interrupt){ // verifică dacă a apărut o întrerupere
       delay(5*DEBOUNCE_DURATION); // un delay cu rol de debounce
       switch(gameState){...} // schimbă starea curentă a jocului prematur
```

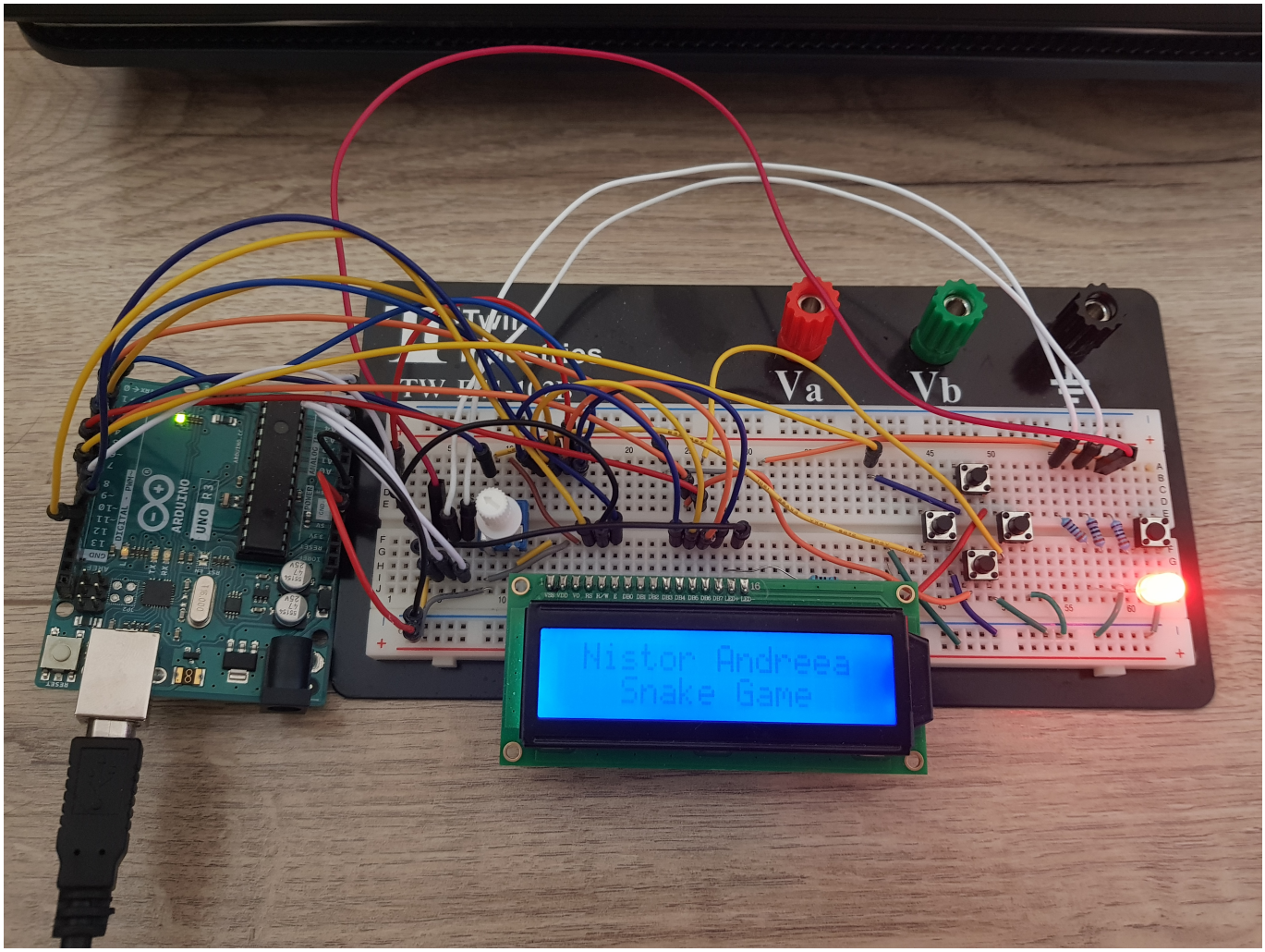
```
la apăsarea butonului GameState
 Interrupt = false;} // resteaza valoarea setată in întrerupere până la
următoarea întrerupere
```

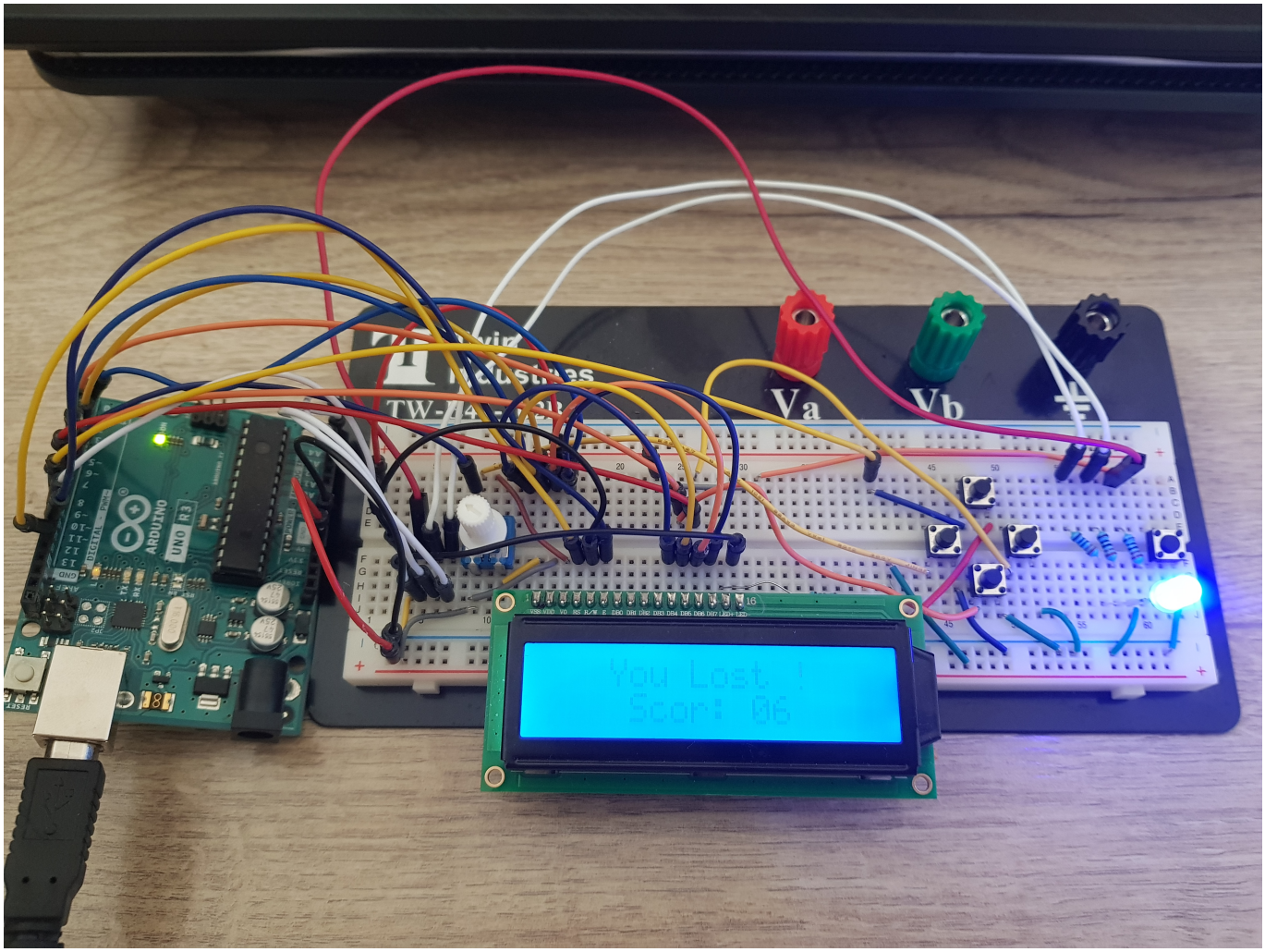
```
if(digitalRead(BUTTON_UP) == pressed){ // verifică dacă butonul respectiv a
fost apelat
    if(debounce_activate_edge(&debCountBUTTON_UP)){ -// verifică debounce-ul
      snakeDirection=SNAKE_UP; // schimbă direcția snake-ului
    }
}else{
    debounce_deactivate(&debCountBUTTON_UP); // în cazul în care butonul nu
a fost apăsat, resetează timer-ul de debounce
}
```

Același tipar de verificare este apelat pentru restul butoanelor care definesc mișcarea snake-ului.

```
if(millis()-lastGameUpdateTick > gameUpdateInterval){
  game_calculate_logic();
  game_calculate_display();
  graphic_clear();
  rgb_color();
  lastGameUpdateTick = millis();
} // verifica constant dacă jocul trebuie să-ți dea refresh să recalculeze
starea jocului, culoarea led-ului, dimensiunea, pozitia snake-ului și a
mărului
```

Rezultate Obţinute


Va
Vb
Nistor Andreea
Snake Game



Demo youtube: https://www.youtube.com/watch?v=DwJfF1_lE9Y

Concluzii

Simpla idee că aș putea realiza un joc singură m-a motivat să duc până la capăt acest proiect. Consider că a fost un mod captivant prin care am aplicat cunoștințe dobândite în cadrul cursului de PM.

Download

proiect_pm.zip

Jurnal

- 13.04.2022 : Alegere proiect

- 20.04.2022 : Realizare pagină
- 04.05.2022 : Implementare proiect
- 18.05.2022 : Adăugare funcționalitate nouă
- 20.05.2022 : Realizare schemă electrică EAGLE
- 23.05.2022 : Finalizare pagină wiki

Bibliografie/Resurse

https://ocw.cs.pub.ro/courses/pm/lab/lab3-2022

https://www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce

https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt

https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

Export to PDF

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

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2022/imacovei/andreea.nistor2208

Last update: **2022/05/25 19:18**