UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Snake

Proiectare cu microprocesoare

Autor: Cotei Ruxandra-Maria
Grupa: 30238
Profesor îndrumător: Itu Răzvan

FACULTATEA DE AUTOMATICA
SI CALCULATOARE

13 Ianuarie 2025

Cuprins

1 Problem Statement

1.1 Enunțul Proiectului

Scopul acestui proiect este de a crea un joc simplu de tip *Snake*, în care utilizatorul controlează mișcarea șarpelui folosind patru butoane fizice conectate la un microcontroller *Arduino*. Jocul este rulat într-un mediu grafic creat în *Processing*, iar mișcarea șarpelui va fi controlată prin semnalele transmise de la Arduino. Problema principală este integrarea corectă a unui sistem hardware (butoane conectate la Arduino) cu un sistem software (*Processing* pentru desenarea și gestionarea jocului) [2], iar utilizatorul trebuie să poată controla mișcarea șarpelui fără întârzieri semnificative și fără erori datorate bouncing-ului butoanelor.

1.2 Definiția Problemei

Jocul *Snake* presupune că jucătorul controlează un șarpe pe un ecran. Șarpele se mișcă într-un spațiu 2D în patru direcții, iar obiectivul este ca acesta să consume mâncare pentru a crește în lungime. Fiecare apăsare a unui buton fizic va schimba direcția de mișcare a șarpelui, iar controlul jocului va fi realizat printr-o interfață hardware (butoane) și software (Processing).

Dificultăți întâmpinate Dificultățile întâmpinate includ:

- **Debouncing-ul butoanelor**: Butoanele fizice pot genera multiple semnale la o singură apăsare din cauza efectului de ”bouncing”. Acesta trebuie gestionat pentru a evita comenzi multiple.
- **Sincronizarea între hardware și software**: Arduino trebuie să trimită informațiile despre starea butoanelor către Processing într-un mod continuu, iar Processing trebuie să actualizeze poziția șarpelui pe ecran pe baza comenzilor primite.
- **Performanța**: Este necesar ca întregul sistem să răspundă rapid și să mențină jocul fluid, fără întârzieri sau întreruperi.
- **Desenarea anumitor forme**: Este necesar să se deseneze corect formele pentru șarpe, mâncare și alte obiecte pe ecran, având grijă la actualizarea continuă a pozițiilor și la evitarea suprapunerilor între forme.

2 Soluția pentru Jocul Snake

Jocul Snake este implementat în Processing, iar direcțiile de mișcare ale șarpelui sunt controlate de un Arduino printr-o conexiune serială. Arduino trimite semnale seriale către Processing, iar în funcție de aceste semnale, jocul actualizează poziția șarpelui pe o grilă, verifică coliziunile și controlează scorul.

2.1 Configurarea hărții și a șarpelui

În primul rând, este definită o rețea de dimensiune 30x25 care reprezintă câmpul de joc. Fiecare celulă poate fi fie liberă, fie o zonă de perete. Se creează o funcție care stabilește harta cu perete pe margini [1].

```
int cols = 30;
int rows = 25;
int[][] grid = new int[cols][rows];

void createMap() {
  for (int i = 0; i < cols; i++) {
    grid[i][0] = 1;
    grid[i][rows - 1] = 1;
  }
  for (int i = 0; i < rows; i++) {
    grid[0][i] = 1;
    grid[cols - 1][i] = 1;
  }
}
```

Această funcție creează peretele pe margini, reprezentând coliziuni.

2.2 Crearea și mișcarea șarpelui

Șarpele este reprezentat de o listă de puncte (`PVector`). Mișcarea șarpelui se face pe baza unui vector de direcție, care este actualizat la fiecare cadru.

```
void moveSnake() {
  PVector newHead = snake.get(0).copy().add(direction);
  snake.add(0, newHead);  // Add new head

  if (newHead.equals(apple)) {
    score += 10;
    spawnApple();
  } else {
    snake.remove(snake.size() - 1);  // Remove tail if no apple eaten
  }
}
```

Funcția adaugă un nou cap la șarpe în direcția curentă și verifică dacă capul a atins un măr. Dacă da, mărul este reinițializat, iar scorul crește.

2.3 Verificarea coliziunilor

Se verifică dacă capul șarpelui se ciocnește de perete sau de corpul său propriu.

```
void checkCollisions() {
  PVector head = snake.get(0);

  // Wall collision
  if (grid[(int)head.x][(int)head.y] == 1 || head.x < 0 || head.y < 0
   ↪  || head.x >= cols || head.y >= rows) {
    gameOver = true;
  }

  // Self-collision
  for (int i = 1; i < snake.size(); i++) {
    if (head.equals(snake.get(i))) {
      gameOver = true;
    }
  }
}
```

Această funcție verifică dacă capul șarpelui este în afacerea limitelor hărții sau dacă se ciocnește de orice parte a corpului său.

2.4 Citirea semnalului serial

Pentru a controla direcția șarpelui, se folosește un Arduino care trimite semnale seriale. Semnalele pot fi ”UP”, ”DOWN”, ”LEFT” sau ”RIGHT”, iar în funcție de acest semnal, direcția șarpelui este schimbată.

```
void readSerialInput() {
  while (myPort.available() > 0) {
    String input = myPort.readStringUntil('\n');
    if (input != null) {
      input = input.trim();
      if (input.equals("UP") && direction.y == 0) {
        direction.set(0, -1);
      } else if (input.equals("DOWN") && direction.y == 0) {
        direction.set(0, 1);
      } else if (input.equals("LEFT") && direction.x == 0) {
        direction.set(-1, 0);
      } else if (input.equals("RIGHT") && direction.x == 0) {
        direction.set(1, 0);
      }
    }
  }
}
```

Această funcție citește datele trimise de Arduino și schimbă direcția în consecință.

2.5 Restabilirea jocului

După un final de joc, există un ecran care permite utilizatorului să apese un buton pentru a restarta jocul. Funcția de restart șterge pozițiile anterioare și reinițializează toate valorile de început ale jocului.

```
void restartGame() {
  snake.clear();
  initializeSnake();
  spawnApple();
  direction.set(1, 0);
  score = 0;
  gameOver = false;
  loop();
}
```

Această funcție resetează jocul la starea inițială.

2.6 Arduino pentru controlul butoanelor

În partea de hardware, butoanele sunt conectate la Arduino și sunt folosite pentru a trimite semnale către Processing pentru a schimba direcția șarpelui.

```
boolean debounce(int pin, int index) {
  int reading = digitalRead(pin);

  if (reading != lastButtonState[index]) {
    lastDebounceTime[index] = millis();
  }

  if ((millis() - lastDebounceTime[index]) > debounceDelay) {
    if (reading != buttonState[index]) {
      buttonState[index] = reading;
      if (reading == LOW) {
        return true;
      }
    }
  }

  lastButtonState[index] = reading;
  return false;
}
```

Această funcție implementează un debounce pentru butoane, prevenind multiple citiri eronate atunci când butonul este apăsat.

```
if (debounce(UP_BUTTON, 0)) {
    Serial.println("UP");
  }
  if (debounce(DOWN_BUTTON, 1)) {
    Serial.println("DOWN");
  }
```

```
7     if (debounce(LEFT_BUTTON, 2)) {
8       Serial.println("LEFT");
9     }
10    if (debounce(RIGHT_BUTTON, 3)) {
11      Serial.println("RIGHT");
12    }
```

Aceasta funcție citește semnalele de la butoane și le trimite pe serial la Processing.

3 Circuitul electronic

Circuitul folosește patru butoane conectate la pinii digitali ai plăcii Arduino, fiecare asociat unei direcții: sus, jos, stânga și dreapta. Rezistențele de pull-down sunt folosite pentru a asigura un semnal clar la apăsarea butoanelor, prevenind citiri false. Conexiunile includ:

- Pinul `2` conectat la butonul pentru direcția ”sus”.
- Pinul `3` conectat la butonul pentru direcția ”jos”.
- Pinul `4` conectat la butonul pentru direcția ”stânga”.
- Pinul `5` conectat la butonul pentru direcția ”dreapta”.
- Fiecare buton are o rezistență pentru a preveni fluctuațiile de semnal.

Diagrama schematică a circuitului este prezentată mai jos:

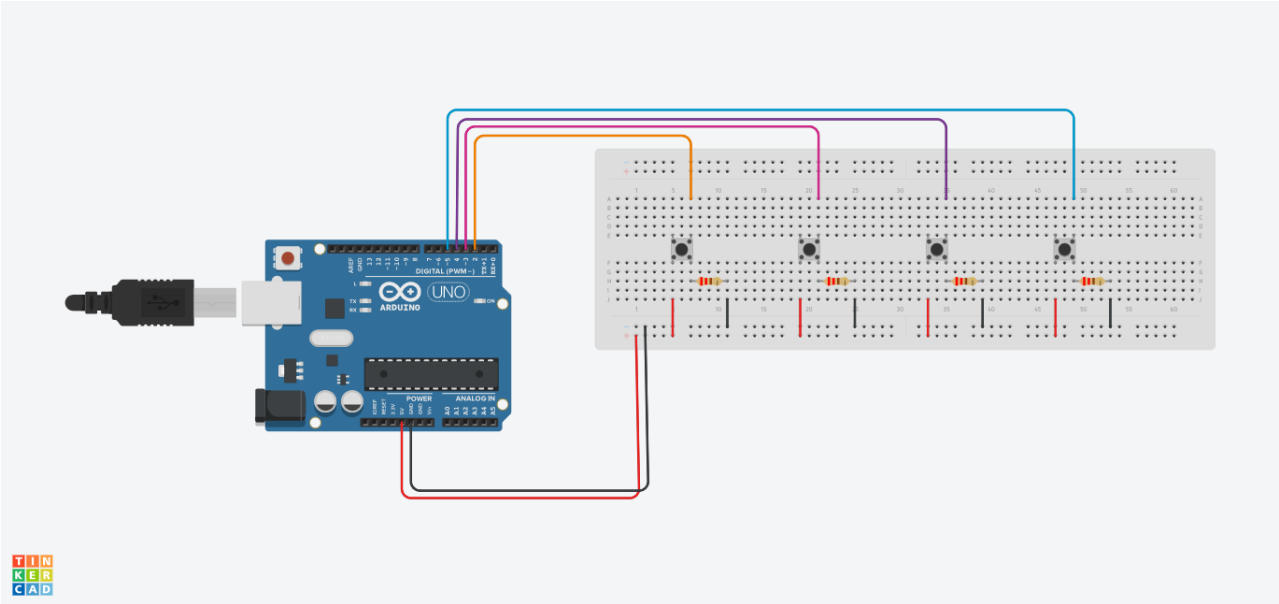

Figura 1: Schema circuitului pentru conexiunea butoanelor la Arduino.

4 Imaginea proiectului

Imaginea de mai jos ilustrează ansamblul proiectului final, inclusiv conexiunile hardware și modul în care butoanele sunt utilizate pentru a controla jocul. Acest setup este folosit pentru a integra componentele software și hardware într-un mod funcțional.

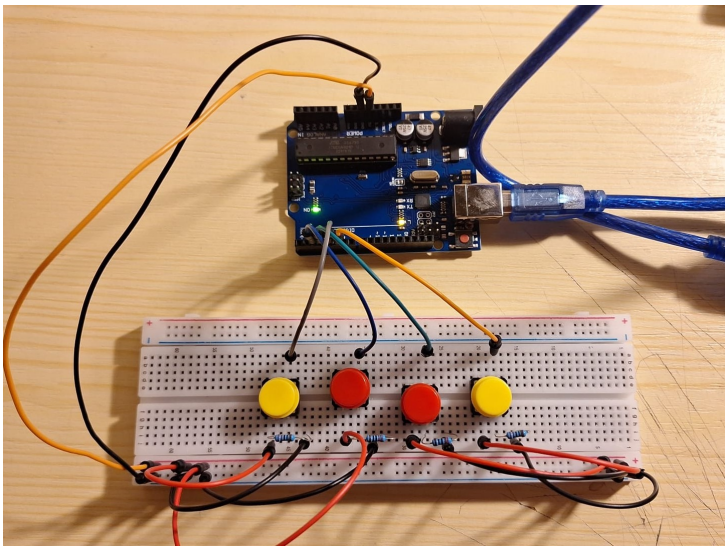

Figura 2: Schema circuitului pentru conexiunea butoanelor la Arduino.

În cadrul acestui proiect, interfața grafică a jocului ”Snake” este realizată folosind Processing. Pe ecranul de joc, se afișează o grilă care reprezintă câmpul de joc, iar șarpele este desenat ca o secvență de pătrate, cu capul șarpelui având o culoare distinctă. Mâncarea apare aleator pe grilă și, atunci când șarpele o consumă, acesta crește în lungime. Pe parcursul jocului, coliziunile cu pereții sau cu corpul șarpelui sunt verificate, iar în cazul unui eșec, jocul se oprește, iar utilizatorului i se oferă opțiunea de a începe o nouă rundă.

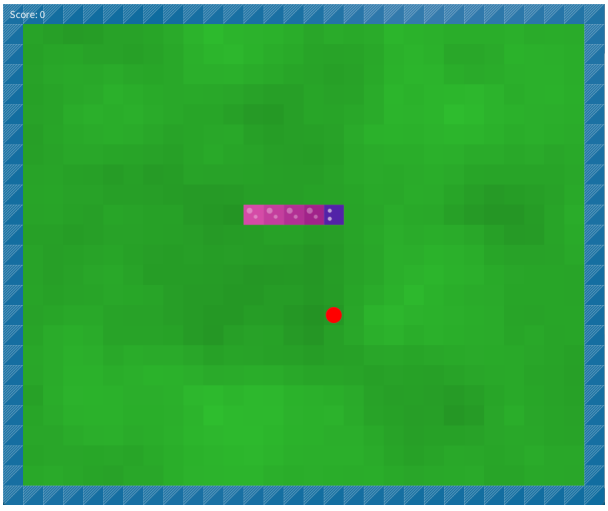

Figura 3: Schema circuitului pentru conexiunea butoanelor la Arduino.

Referințe

[1] Ben Fry și Casey Reas, *Processing: A Programming Handbook for Visual Designers and Artists*, MIT Press, 2007.

[2] Paul Horowitz și Winfield Hill, *The Art of Electronics*, 3rd, Cambridge University Press, 2015.