Memory game

Introducere

Proiectul constă într-un joculeț de memorie care funcționează în felul următor: îți alegi modul de joc (easy/hard) prin apăsarea butonului corespunzător, iar o dată ce apeși pe unul din butoane, se va afișa o secvență de lumini, fiecare led având o culoare și un sunet diferit. Jucătorul va trebui să reproducă secvența apăsând pe butoanele corespunzătoare acestora, iar dacă nimerește trece la nivelul următor.

Ideea de la care am pornit este aceea de a crea un joc, ceva care să fie și distractiv, dar care este și util, având ca scop antrenarea memoriei.

Este un mod de a combina utilul cu plăcutul, trezind în jucător și un mic spirit de competitivitate, pentru a reuși să scoată un scor cât mai bun.

Descriere generală

Flow proiect:

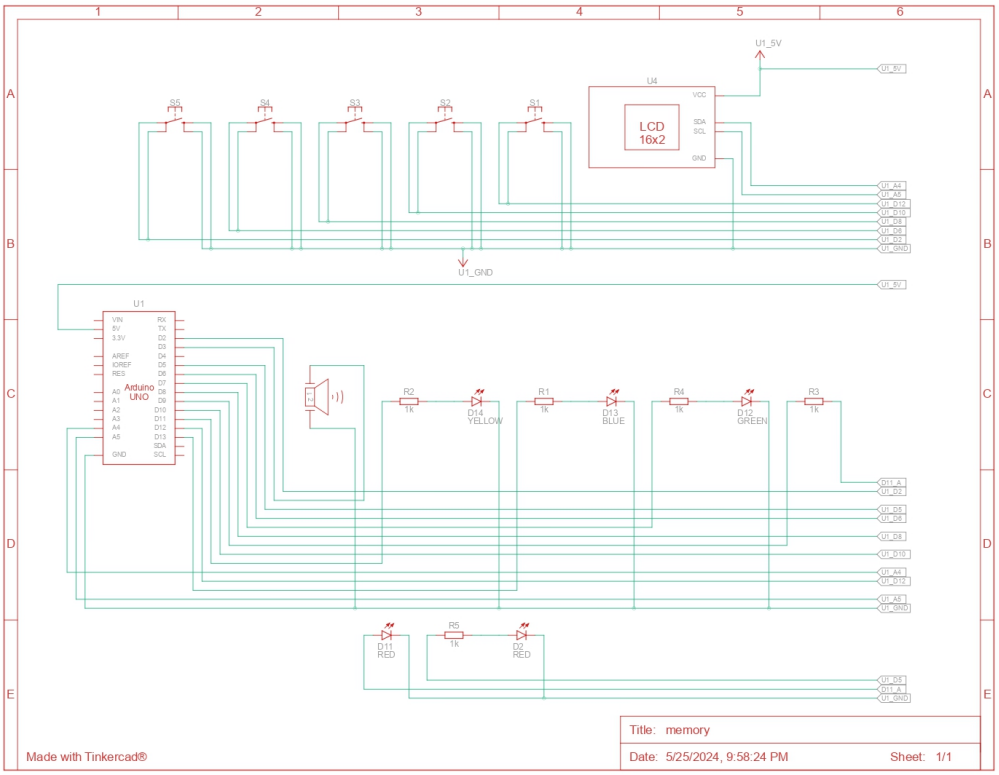
1. Jucătorul va alege modul de joc apăsând butonul de schimbare a modului de joc. Implicit, modul de joc va fi easy.
2. Jucătorul apasă pe unul din cele 4 butoane corespondente luminilor pentru a începe jocul.
3. Pentru fiecare nivel, se vor lumina un anumit număr de LED-uri și fiecare va scoate un sunet diferit prin buzzer. O dată cu avansarea în nivele va crește și numărul de LED-uri luminate.
4. Jucătorul va trebui să reproducă secvența apăsând pe butoanele corespondente LED-urilor. Scorul actual va fi afișat pe ecranul LCD, iar dacă jucătorul greșește va pierde jocul, afișându-se un mesaj.
5. După ce a pierdut, poate încerca din nou.
6. Pe modul hard al jocului, odată cu trecerea timpului și mărirea scorului, intensitatea luminii LED-urilor va scădea și viteza cu care LED-urile vor fi afișate va crește.

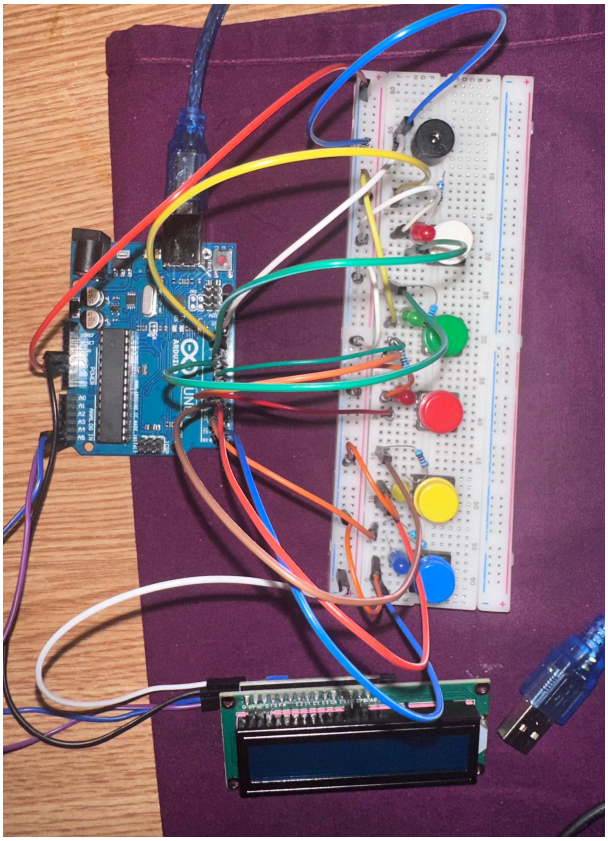
Hardware Design

Listă piese:

- Arduino UNO R3
- Breadboard
- LCD display I2C
- Fire tată-mamă, tată-tată
- 5 LED-uri (4 pentru joc, 1 pentru modul de joc)
- 5 rezistențe
- 5 butoane
- Buzzer

M-am jucat puțin și cu simulările din Tinkercad înainte de a pune totul pe plăcuță.





Am modificat puțin schema electrică deoarece am descoperit că este bine să folosești pinii 2 și 3 pentru întreruperi, pe care i-am folosit pentru butonul de schimbare a modului de joc.

Software Design

- Mediul de dezvoltare: **Arduino IDE 2.3.2**
- Librării folosite: **Wire.h** și **LiquidCrystal_I2C.h** pentru comunicarea I2C cu LCD display-ul.

Descrierea implementării

- ***setup() și smallButtonISR()***: aici inițializez toate LED-urile, butoanele, buzzer-ul, mesajul inițial de pe display, starea inițială a jocului și adaug o întrerupere pentru butonul de selectare a modului de joc (hard/normal). Aceasta modifică variabila volatilă hardModeChanged, folosind debouncing pentru că observasem că nu se selecta modul bine fără.

```
void smallButtonISR() {
  static unsigned long lastDebounceTime = 0;
  unsigned long debounceDelay = 100;

  if (millis() - lastDebounceTime > debounceDelay) {
    hardModeChanged = true;
    lastDebounceTime = millis();
  }
```

```
}

void setup() {
  pinMode(blueLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(modeLED, OUTPUT);

  pinMode(blueButton, INPUT_PULLUP);
  pinMode(yellowButton, INPUT_PULLUP);
  pinMode(redButton, INPUT_PULLUP);
  pinMode(greenButton, INPUT_PULLUP);
  pinMode(smallButton, INPUT_PULLUP);

  pinMode(buzzerPin, OUTPUT);
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Memory Game!");
  lcd.setCursor(0, 1);
  lcd.print("Highest Score ");
  lcd.print(highestScore);

  randomSeed(analogRead(0));
  gameState = WAIT_FOR_START;
  attachInterrupt(digitalPinToInterrupt(smallButton), smallButtonISR,
FALLING);
}
```

In loop, avem mai multe stări ale jocului:

- WAIT_FOR_START
- GENERATE_SEQUENCE
- DISPLAY_SEQUENCE
- HARD_DISPLAY_SEQUENCE
- WAIT_FOR_BUTTON_PRESS
- GAME_OVER
- LEVEL_UP

- ***WAIT_FOR_START***: verifică 2 cazuri, dacă s-a produs o întrerupere cauzată de butonul de schimbare a jocului, sau dacă unul din celelalte butoane s-a apăsat, caz în care starea jocului devine GENERATE_SEQUENCE
- ***GENERATE_SEQUENCE***: adaugă noul LED generat random la secvența curentă.

```
void generateSequence() {
  sequence[currentLevel] = random(0, 4);
```

```
  sequenceIndex = 0;
  gameState = hardMode ? HARD_DISPLAY_SEQUENCE : DISPLAY_SEQUENCE;
}
```

- ***DISPLAY_SEQUENCE/HARD_DISPLAY_SEQUENCE***: ambele vor apela aceeași funcție, însă cu valoarea parametrului diferită. În funcție de modul de joc, luminozitatea și viteza cu care LED-urile sunt afișate se vor schimba, LED-urile luminându-se treptat cu PWM.

```
void fadeLED(int ledPin, int startBrightness, int endBrightness, int
duration) {
  int stepDelay = duration / abs(endBrightness - startBrightness);
  for (int brightness = startBrightness; brightness != endBrightness;
brightness += (startBrightness < endBrightness ? 1 : -1)) {
    analogWrite(ledPin, brightness);
    delay(stepDelay);
  }
  analogWrite(ledPin, endBrightness);
}

void displaySequence(bool hard) {
  int leds[] = {blueLED, yellowLED, redLED, greenLED};
  int brightness = hard ? max(255 - 30 * currentLevel, 0) : 255; // Adjust
brightness for hard mode
  unsigned long currentMillis = millis();

  int basePauseDuration = 600;
  int minPauseDuration = 400;
  int pauseDuration = hard ? max(basePauseDuration - 20 * currentLevel,
minPauseDuration) : basePauseDuration;

  if (displayState == 0 && currentMillis - lastMillis >= pauseDuration) {
    lastMillis = currentMillis;
    fadeLED(leds[sequence[sequenceIndex]], 0, brightness, 300);
    tone(buzzerPin, 1000 + (leds[sequence[sequenceIndex]] * 200), 300);
    displayState = 1;
  } else if (displayState == 1 && currentMillis - lastMillis >= 300) {
    lastMillis = currentMillis;
    fadeLED(leds[sequence[sequenceIndex]], brightness, 0, 300);
    sequenceIndex++;
    displayState = 0;
    if (sequenceIndex > currentLevel) {
      gameState = WAIT_FOR_BUTTON_PRESS;
      buttonPressIndex = 0;
    }
  }
}
```

- ***GAME_OVER***: toate LED-urile se aprind și se va auzi un sunet produs de buzzer, display-ul afișeaza mesajul ”Game over!” și modul de joc revine la WAIT_FOR_START.
- ***LEVEL_UP***: se mărește variabila currentLevel de fiecare dată când jucătorul ghicește ordinea corectă pentru acel nivel.

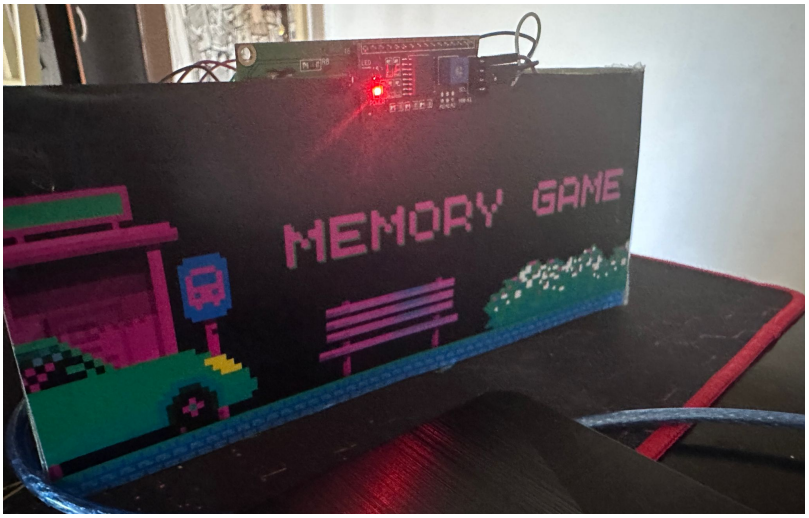
Laboratoare folosite:

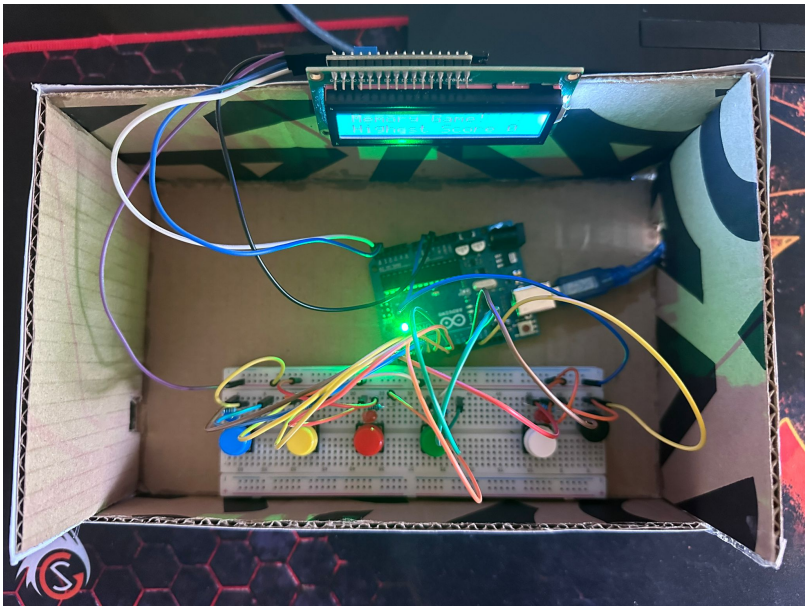
1. Întreruperi, timere: https://ocw.cs.pub.ro/courses/pm/lab/lab2-2023
2. PWM, timere: https://ocw.cs.pub.ro/courses/pm/lab/lab3-2023-2024
3. I2C: https://ocw.cs.pub.ro/courses/pm/lab/lab6-2023-2024

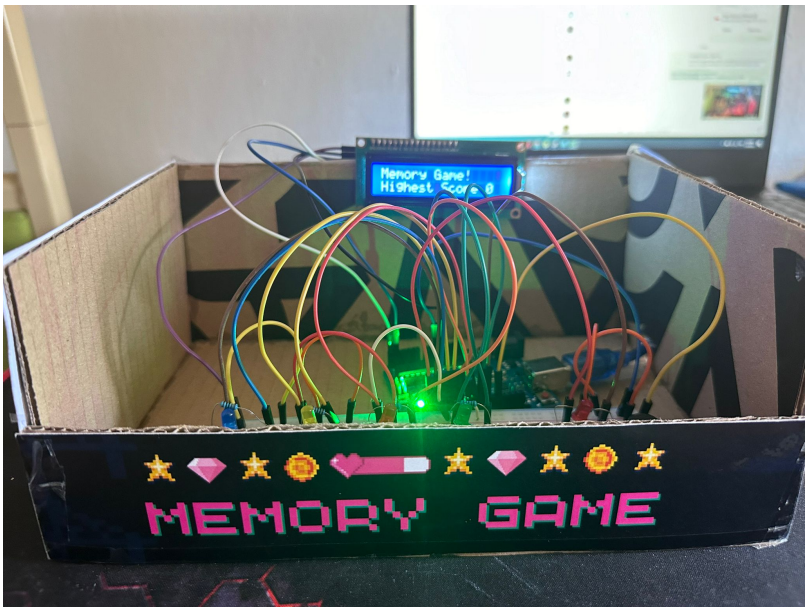
Tot codul sursă se poate găsi pe Github-ul meu la link-ul următor:
https://github.com/RazvanMihaiPopa/Memory-Game

Rezultate Obţinute

Link Demo: https://youtu.be/HsIMO3y1GlQ






Memory Game!
MEMORY GAME

Concluzii

A fost un proiect interesant, mereu am vrut să încerc să lucrez cu o plăcuță Arduino și mă bucur că am avut ocazia. Jocul chiar a ieșit drăguț după părerea mea. Unele din dificultățile pe care le-am întâmpinat au fost să îmi dau seama cum să fac butoanele să funcționeze, pentru că inițial nu se detecta nimic când se apăsau, sau cum să integrez laboratorul cu întreruperi. În final, am reușit să trec și peste ele și consider că a fost o experiență mișto.

Download

Puteți descărca această arhivă care conține cam toate cele prezentate mai sus.
memory_game_prm333cc.zip

Bibliografie/Resurse

Resurse Software:

- https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/
- https://www.arduino.cc/reference/en/libraries/liquidcrystal-i2c/
- etc.

Resurse Hardware:

- http://cleste.ro
- http://optimusdigital.ro

Export to PDF