

Sudoku 4x4

Student: Bogdan-Andrei Buga

Grupa: 332CB

E-mail: bogdandrei04@gmail.com

Introducere

M-am gandit sa implementez un joc clasic de Sudoku, dar limitat la o tabla de 4x4 casute, pentru a putea implementa mai usor mai multe nivele cu grad de dificultate variabil.

Descriere generala

Jucatorul are, in fata sa, 7 butoane disponibile:

- 4 butoane de **navigare** printre casutele tablei de joc (stanga, jos, sus, dreapta);
- 1 buton de **toggle** prin care schimba valoarea casutei pe care se afla cursorul;
- 1 buton de **clear**, prin care sterge o valoare scrisa de-a lungul jocului;
- 1 buton de **check**, prin care verifica daca tabla noastra este completata corect.

Gameplay

Jucatorul va primi o tabla 4x4 completata cu numere de la 1 la 4 si cu spatii libere. Numarul de spatii libere variaza in functie de dificultatea aleasa de jucator la inceputul jocului, dupa ce se termina intro-ul acestuia. Pentru a termina cu succes nivelul dat, jucatorul trebuie sa completeze spatiile cu numere de la 1 la 4 astfel incat:

- A)** fiecare numar sa apara o singura data pe linia sa;
- B)** fiecare numar sa apara o singura data pe coloana sa;
- C)** fiecare numar sa apara o singura data in cadranul sau.

Pe ecranul de joc se vor afisa atat tabla de joc, cat si pozitia cursorului mutabil pe tabla curenta de joc, precum si caracterul din dreptul acestei pozitii.

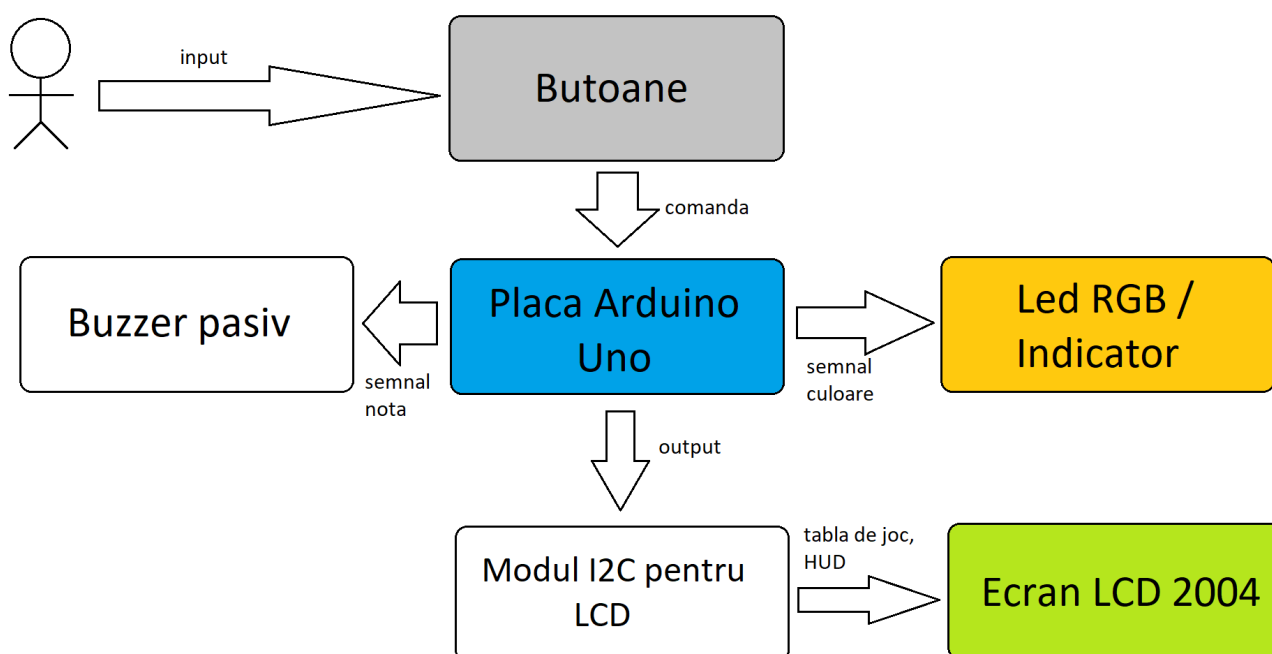
Jucatorul poate sa schimbe valoarea unui spatiu prin apasarea unui buton desemnat pentru a cicla printre numerele de la 1 la 4 sau sa stearga valoarea scrisa daca acesta considera ca nu este corecta. El nu poate schimba un numar deja existent pe tabla, fiind atentionat corespunzator intr-o asemenea situatie.

Verificarea jocului

Jucatorul poate, de asemenea, sa verifice corectitudinea tablei curente. Daca exista vreo greseala in asezarea curenta a numerelor, jucatorul va pierde una din cele 3 vietii cu care a inceput jocul actual. Daca jucatorul isi pierde toate vietile, jocul se incheie intr-un esec.

In cazul in care nu exista greseli, jucatorului i se afiseaza pe ecran, langa tabla de joc, si cate spatii mai are de completat. Atunci cand acesta a completat toate spatiile corect si isi verifica rezultatele, jocul se termina cu un succes.

La terminarea jocului, jucatorul va avea parte de o melodie si de un mesaj corespunzator starii finale a jocului curent (succes sau esec).



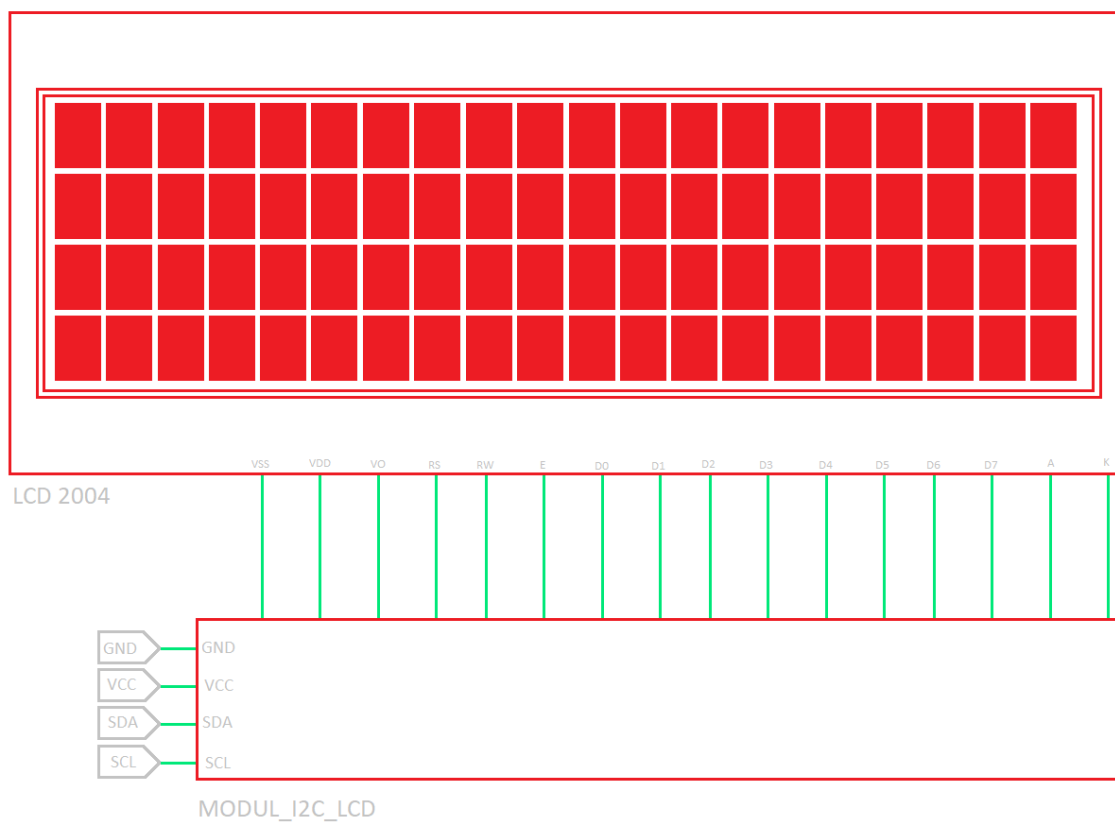
Hardware Design

Componente necesare

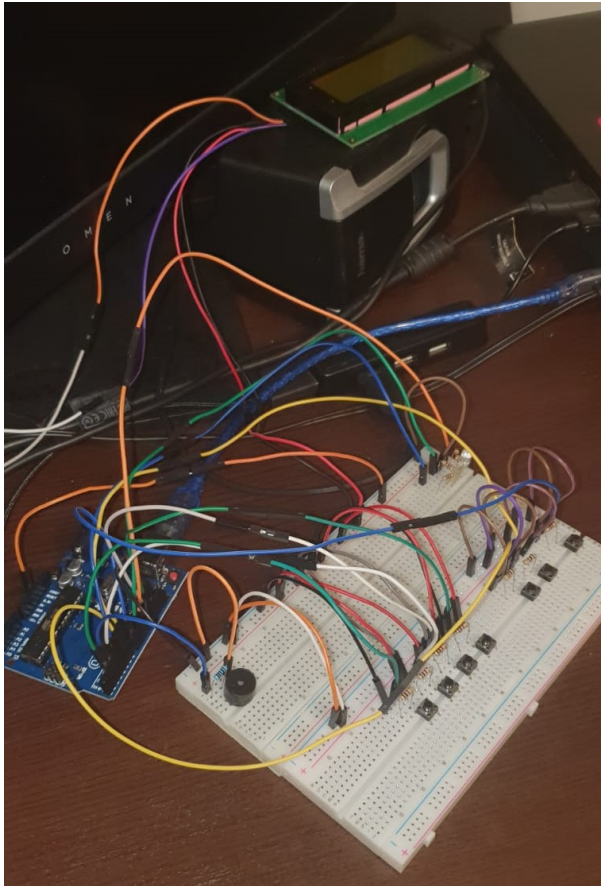
- 1 placa de dezvoltare Arduino Uno
- 1 cablu USB A-B

- 7 butoane
- 1 ecran LCD 2004
- 1 adaptor I2C pentru LCD
- 1 buzzer pasiv
- 1 LED RGB
- 10 rezistori de 1k ohmi
- 1 rezistor de 100 ohmi

Schema electrica



Layout fizic



Butoanele au fost asezate in urmatoarea ordine, de la stanga la dreapta: LEFT, DOWN, UP, RIGHT, CELAR, TOGGLE si CHECK.

Cea mai dificila parte a fost aranjarea tuturor pinilor necesari astfel incat PWM-ul din pinii pentru indicatorul LED RGB sa nu tulbure semnalul PWM-Tone trimis catre buzzer.

Software Design

Medii de dezvoltare folosite

- **Arduino 1.8.13**, pentru scrierea si testarea programului
- **Notepad++**, pentru comentarii si aranajarea codului

Variabile importante folosite

- `checkDifficulty`
 - un bool care este "true" daca s-a trecut de introducerea si nu s-a ales niciun grad de dificultate;
 - acesta devine "false" pe tot restul jocului odata cu alegerea dificultatii jocului.
- variabile de memorare a pozitiei cursorului: X si Y;

- prompt
 - un char[20] folosit pentru afisarea mesajelor de HUD pe ecranul LCD.

Funcțiile utilizate și scopul acestora

Funcții de ieșire

Aceste funcții sunt folosite pentru a transmite date către ecranul LCD, buzzer și led-ul RGB.

1. playIntro()
 - Trimite note muzicale către buzzer pentru a cânta melodia de introducere aleasă ("We Are Number One" de Stefan Karl Stefansson).
2. playVictoryJingle()
 - Trimite note muzicale către buzzer pentru a cânta o melodie compusă manual care felicită jucătorul pentru victoria obținută.
 - Transmite culoarea verde către indicator.
3. playGameOverJingle()
 - Trimite note muzicale către buzzer pentru a cânta o melodie compusă manual care îi spune, subtil, jucătorului că și-a pierdut toate vietile și nu își mai poate continua jocul.
 - Transmite culoarea roșie către indicator.
4. beep(int note)
 - Trimite către buzzer o notă muzicală dată ca parametru pentru o durată scurtă de timp.
5. printGameScreen()
 - Afisează, pe ecranul LCD, toate detaliile jocului curent: tabla de joc, poziția cursorului, caracterul din dreptul acestuia și vietile rămase.
6. printWarning()
 - Trimite către buzzer o notă muzicală specifică unei mutări ilegale.
 - Jucătorul este avertizat pe ecranul LCD că nu poate modifica poziția curentă, care nu era spațiu pe tabla originală.
 - Transmite culoarea mov / magenta către indicator.
7. printGoodCheck(char no_spaces)
 - Trimite către buzzer o notă muzicală specifică unui joc bine jucat până la momentul verificării inclusiv.
 - Scrie pe ecranul LCD atât un mesaj încurajator, cât și numărul de spații rămase (stocat în parametrul "no_spaces").
 - Transmite indicatorului o culoare ce indică progresul jocului: albastru, dacă sunt completate mai puțin de jumătate din spații, sau turcoaz, în caz contrar.
8. printLevelComplete()
 - Transmite către buzzer melodia de victorie definită în *playVictoryJingle()*.
 - Termină jocul cu mesaje adecvate scrise pe ecranul LCD.
 - Transmite culoarea verde către indicator.
9. printBadCheck(char lives)
 - Trimite către buzzer o notă muzicală specifică unei greseli.
 - Jucătorul va fi instiintat pe ecranul LCD câte vieti mai are (număr stocat în parametrul "lives").
 - Transmite indicatorului o culoare roșie de intensitate invers proporțională cu numărul de vieti rămase.

Funcții de gameplay

Aceste funcții implementează partea de back-end a jocului, ocupându-se de verificările necesare pentru ca jocul să se desfășoare cum trebuie.

1. `checkBoardElement(char board[16], char original[16], char y, char x)`
 - verifică dacă elementul tablei "board" de pe poziția dată (Linia "y", Coloana "x") este unic pe linia sa, unic pe coloana sa și unic pe cadranul său (cadran = bloc 2×2 rezultat prin împărțirea tablei de joc în 4 pătrate egale), numai în cazul în care poziția curentă reprezintă un spațiu pe tabla inițială (tabla "original").
 - Returnează "true" dacă verificarea menționată este validă sau "false" în caz contrar.
2. `checkBoard(char board[16], char original[16])`
 - Verifică așezarea corectă a tuturor numerelor pe tabla curentă ("board") prin apelurile funcției `checkBoardElement(board, original, i, j)`, unde *i* și *j* iau valori în intervalul [0,4).
 - Returnează "false" dacă s-a găsit cel puțin un element numeric așezat incorect pe tabla curentă sau "true" în caz contrar.
3. `setup()`
 - Inițializează ecranul LCD, caracterele speciale, pinii de pe placa Arduino și porneste introducerea jocului.
 - La începutul fiecărui joc, jucătorul va avea 3 vieți.
4. `loop()`
 - Pentru început, trebuie selectată dificultatea dorită a jocului. În funcție de alegerea jucătorului, se va genera o tabla de joc aleator dintr-o bază de date de câte 35 de table de joc pentru fiecare dificultate și, astfel, jocul va începe.
 - Pentru fiecare buton apăsător, se vor lua următoarele decizii:
 - **LEFT** : deplasează cursorul la stânga pe tabla de joc;
 - **DOWN** : deplasează cursorul în jos pe tabla de joc;
 - **UP** : deplasează cursorul în sus pe tabla de joc;
 - **RIGHT** : deplasează cursorul la dreapta pe tabla de joc;
 - **CLEAR** : șterge valoarea numerică scrisă în dreptul unui spațiu de pe tabla originală;
 - **TOGGLE** : scrie o valoare numerică în dreptul unui spațiu sau modifică această valoare dacă a fost deja scrisă;
 - **CHECK** : verifică corectitudinea jocului actual; dacă există minim o greșeală, jucătorul va pierde o viață, iar dacă își pierde toate cele 3 vieți, nu va mai putea continua jocul curent; dacă nu există nicio greșeală, i se arată câte spații libere mai are de completat sau, dacă toate spațiile au fost completate corect, înseamnă că a terminat runda actuală și jocul se încheie cu un succes al jucătorului.
 - Dacă jocul nu s-a terminat, se va afișa tabla curentă de joc, împreună cu celelalte elemente de HUD (poziția cursorului, caracterul curent, viețile rămase).

Rezultate Obținute

[Introducerea jocului](#)

[Proba 1](#)

- Joc usor (pe dificultatea EASY)
- Navigarea printre casutele tablei de joc
- Scrierea, modificarea si stergerea unei casute
- Verificari corecte
- Completarea nivelului dat cu succes

Proba 2

- Joc greu (pe dificultatea HARD)
- Incercarea de a face CLEAR sau TOGGLE pe un numar neschimbabil
- Greseli la verificare si pierderea vietilor
- Terminarea nivelului dat cu esec

Concluzii

A fost un proiect interesant, la care mi-a placut sa luzez si pe care mi-as fi dorit sa-l fac si mai amplu. Datorita limitarilor de memorie, nu am reusit sa incadrez decat 35 de configuratii initiale ale tablei de joc pentru fiecare dificultate, tinta initiala fiind de 60 de configuratii initiale per dificultate. De asemenea, mi-ar fi placut daca as fi putut folosi 2 buzzere simultan, dar placa de dezvoltare Uno nu permite decat unui singur buzzer sa functioneze la un moment dat.

Download

[Codul sursa - Sudoku 4x4 pe LCD](#)

[Documentatia proiectului \(PDF\)](#)

Jurnal

- 27/04/2021 : Alegerea temei
- 18/05/2021 : Adunarea tuturor componentelor necesare
- 27/05/2021 : Finalizarea design-ului hardware
- 31/05/2021 : Finalizarea design-ului software
- 04/06/2021 : Finalizarea documentatiei

Bibliografie / Resurse

<https://ocw.cs.pub.ro/courses/pm/prj2010/mcarjaliu/sudoku4x4> (proiectul de la care am plecat)

<https://arduinogetstarted.com/tutorials/arduino-lcd-i2c>, pentru functiile LCD folosite in codul sursa.

<https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/>, pentru transmiterea notelor si a melodiilor pe buzzer.

<https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/>, pentru alegerea pinilor specifici indicatorului LED RGB.

<https://www.youtube.com/watch?v=iEXPkv7Ijgc> (sursa de inspiratie pentru alegerea melodiei de introducere)

https://ocw.cs.pub.ro/courses/_media/pm/prj2010/mcarjaliu/60_sudokus_4x4_easy.pdf (configuratiile EASY de joc)

https://ocw.cs.pub.ro/courses/_media/pm/prj2010/mcarjaliu/60_sudokus_4x4_difficult.pdf (configuratiile HARD de joc)

From:

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

Permanent link:

http://ocw.cs.pub.ro/courses/pm/prj2021/avaduva/sudoku_4x4



Last update: **2021/06/19 11:47**