# 2048 Deluxe

Contact: Login to see contact info.

## Introducere

Proiectul constă în implementarea bine-cunoscutului joc **2048** pe un ecran monocrom **Nokia 5110** 84×48 pixeli, adăugând diverse elemente de *quality of life* precum:

- lumini RGB care reacționează la input-ul jucătorului
- sunete pentru fiecare acțiune a jucătorului:
  - selectarea opțiunilor din meniu
  - mutarea elementelor jocului
  - scurtă melodie joc pierdut/câștigat

Scopul acestui proiect este de a pune în practică toate noțiunile prezentate până acum în cadrul laboratoarelor de PM.

## Descriere generală

La pornirea jocului se va alege între a începe un joc nou și între a vedea un top cu cele mai mari scoruri de până acum. Când jucătorul alege să pornească un joc nou, pe **ecran** va fi încărcată scena jocului în care va putea să își vadă scorul curent și va putea să miște elementele stânga/dreapta sau sus/jos folosind un **modul joystick PS2**. La sfârșitul jocului, *în cazul în care a câștigat*, jucătorul va putea să își salveze scorul dacă acesta se încadrează în top-ul vechi.

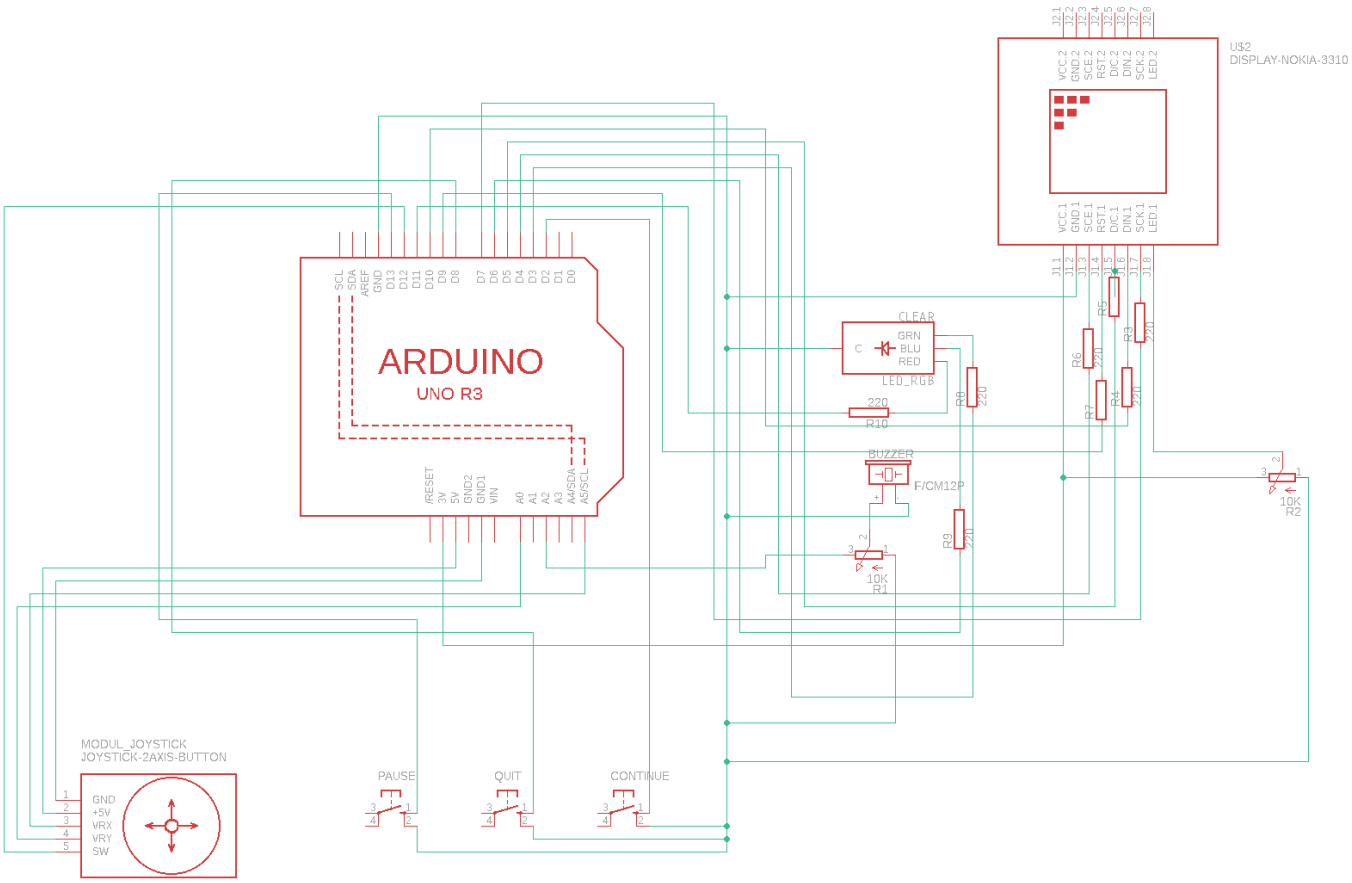
Utilizatorul va avea câteva **butoane tactile** dedicate pentru **pause**, **quit**, **continue**. Jocul conține, de asemenea, un **led RGB** decorativ care luminează în culori diferite în funcție de acțiunile utilizatorului, precum și un **buzzer** care să redea sunetele. Utilizatorul va putea seta luminozitatea ecranului, precum și volumul buzzerului folosind **potențiometre**.

## Schema bloc:

# Hardware Design

| Denumire piesă | Preț piesă | Furnizor | Cantitate comandată |
|---|---|---|---|
| Arduino Uno R3 | 25 lei | cleste.ro | 1 |
| Modul joysticks PS2 | 8 lei | cleste.ro | 1 |
| Ecran Nokia 5110 85×48 | 19 lei | cleste.ro | 1 |
| Buton Tactil 6x6x5mm | 1 leu | cleste.ro | 5 |
| LED de 5mm diverse culori | 0.3 lei | cleste.ro | 50 |
| LED RGB 5mm 4 pini catod comun | 2 lei | cleste.ro | 5 |
| Breadboard 830 puncte | 15 lei | cleste.ro | 1 |
| Fire Dupont tata-tata | 4 lei | cleste.ro | 30 |
| Fire Dupont mama-tata | 4 lei | cleste.ro | 30 |
| Potentiometru liniar 10K | 3 lei | ardushop.ro | 2 |
| Buzzer pasiv | 3 lei | ardushop.ro | 5 |
| Rezistor 220R | 0.4 lei | ardushop.ro | 20 |
| Rezistor 1.2K | 0.4 lei | ardushop.ro | 20 |
| Cablu USB A-B | 4 lei | ardushop.ro | 1 |

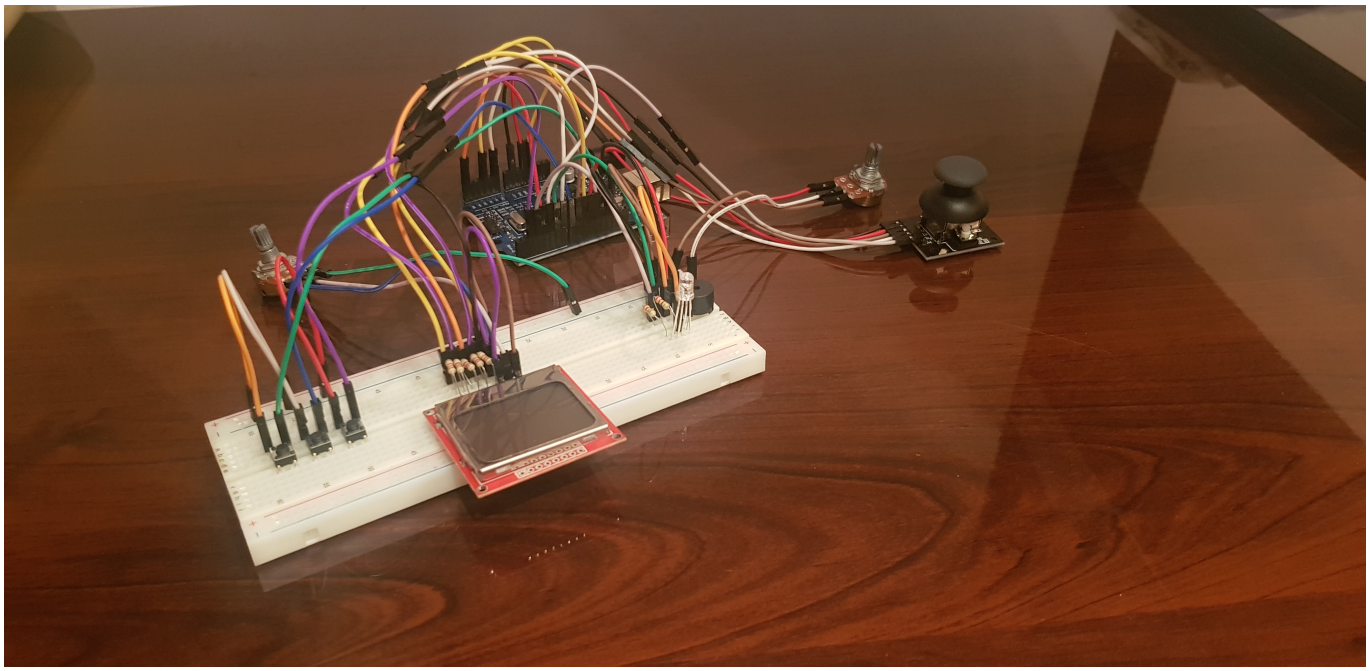
# Schema Electrică



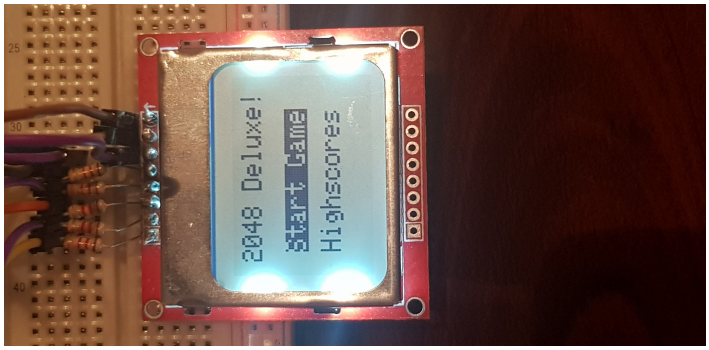
# Hardware

Mai jos se găsesc poze cu montajul hardware realizat și cu câteva imagini din joc:

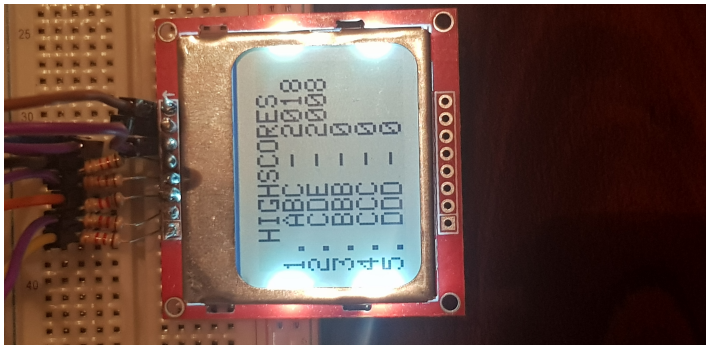
- **Montaj**



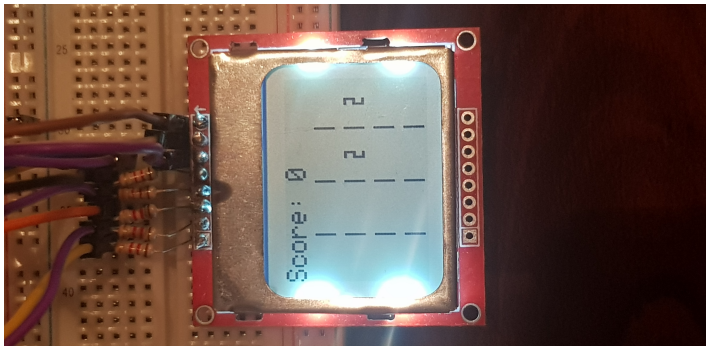
- **Meniu principal**



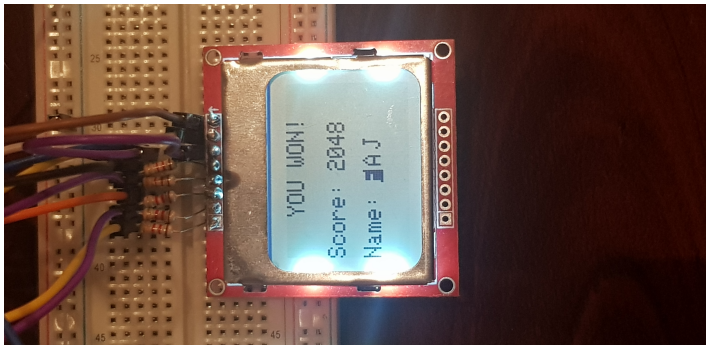

- **Highscores**




- **Start Game**




- **Game Won**




# Software Design

Mediul de dezvoltare folosit pe parcursul implementării a fost Arduino IDE .

Biblioteci folosite în implementarea proiectului:

- EEPROM.h
  - folosită pentru a stoca datele persistente (highscores) în memoria EEPROM a plăcuței Arduino
- biblioteci specifice ecranului LCD Nokia 5110
  - SPI.h
  - Adafruit_GFX.h
  - Adafruit_PCD8544.h
  - Font4x7Fixed.h
    - folosită pentru a schimba fontul caracterelor reprezentate pe LCD

## Detalii despre implementare

### Funcții și variabile pe scurt

**Variabile globale**

1. Variabila cea mai importantă
   - **g_display**
     - obiectul LCD Nokia 5110 folosit pe tot parcursul programului
2. Logica jocului
   - **g_state**, **g_option**
     - starea curentă în care se află jocul, respectiv opțiunea selectată din cadrul meniului principal
   - **g_pause**, **g_record**
     - indică dacă jocul se află în pauză sau nu, respectiv dacă jucătorul se află în interfața de înregistrare a scorului
3. Variabile ce reprezintă jocul propriu-zis
   - **g_board**, **g_max_piece**, **g_score**
     - reprezintă tabla de joc specifică 2048 (un grid 4×4); indică cea mai mare valoare dintre toate piesele de pe tabla de joc, respectiv scorul curent al jucătorului
   - **g_empty_tiles**, **g_esize**
     - vector ce conține informații despre spațiile libere/ocupate din tabla de joc; numărul de spații libere din tabla de joc
4. Variabilă auxiliară
   - **g_win_melody**
     - stochează notele muzicale prin care se va itera atunci când jucătorul câștigă jocul
5. Variabile orientate hardware
   - **g_adc_val**
     - valoarea citită de la ADC
   - **g_last_debounce**
     - folosită în procedeul de *debouncing* pentru butoanele *pause*, *quit* și *resume*

**Funcții folosite**

Funcții specifice Arduino

- **ARDUINO_MyRead**
  - întoarce valoarea obținută în urma conversiei ADC de la pinul specificat ca parametru
- **ARDUINO_ChangePin**
  - schimbă pinul de la care se primește input-ul analogic
- **ARDUINO_WriteScore**
  - scrie în memoria EEPROM a plăcuței Arduino noul highscore al jucătorului

Funcții specifice display-ului

- **DISPLAY_PrintXYZ**

  - realizează afișarea mesajelor/frame-urilor specifice numelui funcției; unele funcții sunt mai complexe și presupun executarea mai multor acțiuni decât simpla afișare de text; detalierea acestora va fi făcută mai jos
- **DISPLAY_ResetGame**
  - realizează resetarea întregului joc și reafișarea frame-ului de la începutul jocului
- **DISPLAY_GameLogic**
  - se apelează în loop și apelează diverse funcții în funcție de starea curentă a jocului
- **DISPLAY_MoveUp/Down/Left/Right**
  - mută piesele în direcția specifică funcției (+ merge numere cu aceeași valoare) și generează o piesă nouă pe tabla de joc
- **DISPLAY_GeneratePiece**
  - generează o piesă nouă (2 sau 4) într-o poziție liberă de pe tabla de joc
- **DISPLAY_CheckIfLost/Won**
  - verifică dacă jocul s-a terminat (cu victoria sau înfrângerea jucătorului)
- **ISR(TIMER1_COMPA_vect)**
  - întrerupere după timer1 la valoare din OCR1A; la fiecare 3 secunde se scad 10 puncte din scor

Funcții pentru led/buzzer/butoane/joystick

- **LED_SetColorRGB**
  - setează culoarea led-ului RGB folosind reprezentarea RGB
- **LED_SetColorHSV**
  - setează culoarea led-ului RGB folosind reprezentarea HSV
- **BUZZER_PlaySound**
  - redă un sunet/o melodie în funcție de parametru
- **ISR(PCINT0_vect)**
  - întreruperi externe pentru butoanele *pause* și *quit*
- **ISR(PCINT2_vect)**
  - întrerupere externă pentru butonul *continue*
- **ISR(ADC_vect)**
  - întrerupere pentru conversia ADC; folosită pentru a citi valorile specifice axelor Ox sau Oy ale joystick-ului

**Analiza programului**

În funcția de **setup** se pregătește configurația inițială a jocului:

- se pornesc *ADC* și *întreruperea de ADC*
- se setează întreruperea după *timer1* la valoarea din *OCR1A*, adică la fiecare 3 secunde
- se creează două piese de **1024** pe tabla de joc pentru a putea analiza rapid toate funcțiile apelate în caz de victorie. Când jocul este resetat folosind funcția **DISPLAY_ResetGame** aceste două piese amplasate special nu se vor mai afla în tabla de joc.
- se realizează inițializarea led-ului, a buzzerului, a celor 3 butoane, a joystick-ului și a display-ului
- se activează întreruperile externe pentru *PCINT*-uri
- se setează led-ul RGB pe alb și display-ul printează meniul principal apelând **DISPLAY_PrintMenu**

În funcția de **loop** se apelează doar **DISPLAY_GameLogic** care se va ocupa de toată logica jocului. În funcția de *GameLogic* se apelează diverse alte funcții după valoarea variabilei *g_state*. La începutul programului variabila are valoarea *quit_state* care indică faptul că fie ne aflăm la începutul rulării, fie că s-a apăsat butonul *quit*, iar jocul a fost resetat. În această stare se citește cu **ARDUINO_MyRead** valoarea de pe axa Oy a joystick-ului. Dacă scade/crește față de pragul de repaus, atunci se schimbă valoarea variabilei *g_option* fie în *scores* (joystick-ul a fost mutat în jos și a fost selectat *Highscores*), fie în *start* (joystick-ul a fost mutat în sus și a fost selectat *Start Game*). În această stare se printează meniul principal cu selecția conform *g_option*, se redă un sunet specific operației de *select* și se schimbă culoarea led-ului.

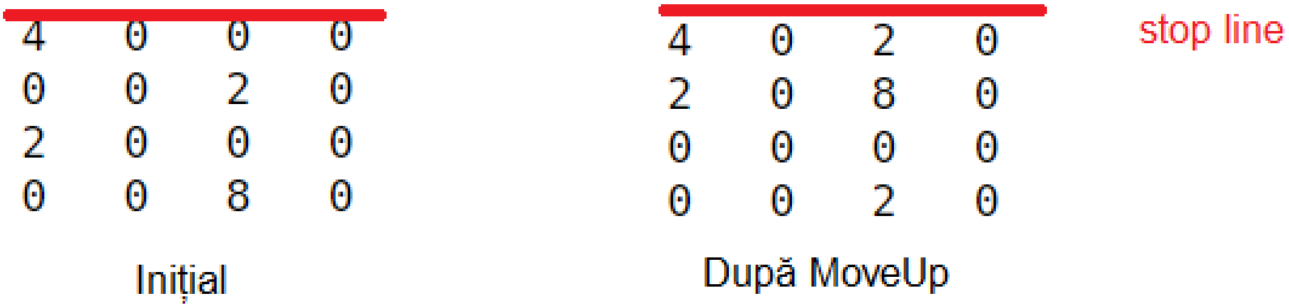
Cât timp jocul se află în starea de *quit_state* singurele butoane care pot schimba starea sunt **quit** (care duce la resetarea jocului și, implicit, la întoarcerea în *quit_state* cu *g_option* setat pe *start*) sau **continue** care va schimba starea fie în *score_state*, fie în *game_state*.

În cazul *score_state* se apelează funcția **DISPLAY_PrintScores** care va afișa primele 5 cele mai mari scoruri obținute în joc sub forma *X. [][][] - SCORE* unde **X** reprezintă un număr de la 1 la 5, **[]** reprezintă un caracter de la A la Z iar **SCORE** reprezintă scorul jucătorului identificat de cele 3 caractere. Aceste date sunt preluate din memoria EEPROM a plăcuței Arduino, iar fiecare intrare reprezintă 5 bytes (3 pentru numele jucătorului și 2 pentru scor - *int* este reprezentat pe 2 bytes). Din această stare se poate reveni doar la starea *quit_state* apăsând **quit**.

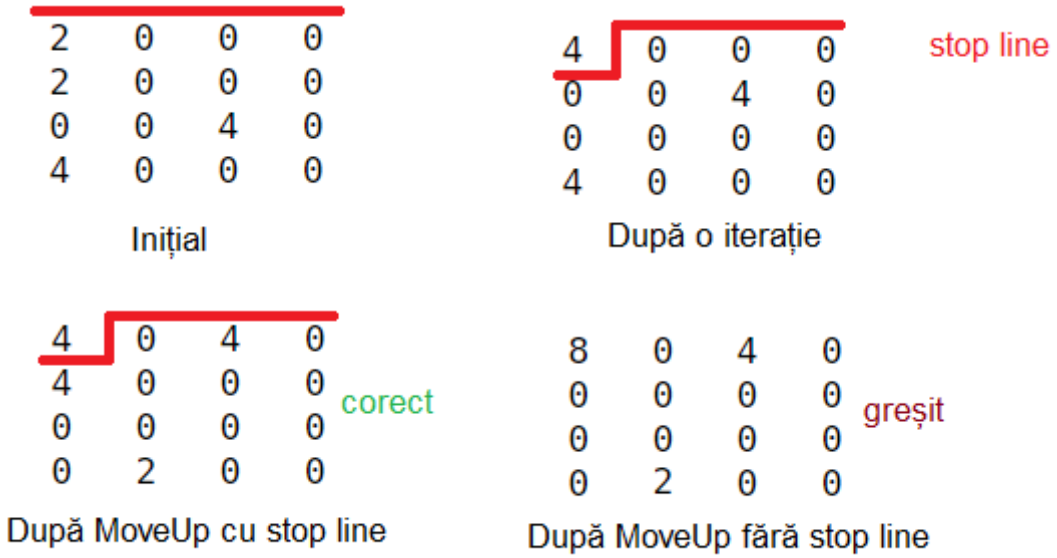
În cazul *game_state* se citesc atât valoarea pe Oy cât și valoarea pe Ox a joystick-ului pentru a ști ce funcție trebuie apelată în continuare. Este posibil să se apeleze o singură funcție la un moment dat dintre cele 4 funcții **DISPLAY_MoveUp/Down/Left/Right** în funcție de valorile citite pe x și pe y, apoi se va apela funcția **DISPLAY_PrintBoard** care va afișa scorul și piesele curente de pe tabla de joc. *PrintBoard* apelează **DISPLAY_PrintPiece** pentru a afișa centrat orice piesă indiferent de valoare folosind calcule care țin cont de valorile posibile ale pieselor și de numărul de pixeli ocupați pe axa Ox a display-ului (3 pixeli + 1 pixel spațiu).

Funcțiile *MoveXYZ* folosesc același algoritm cu modificări în funcție de direcție, pentru simplitate voi explica ce se întâmplă în cazul *MoveUp*, pentru celelalte cazuri ideea este asemănătoare. *MoveUp* iterează prin fiecare linie, coloană cu coloană căutând o piesă validă (valoarea != 0). Când întâlnește o astfel de piesă începe să o „tragă” în sus până la o linie de stop; aici există două cazuri:

1. nu există o altă piesă până la linia de stop sau, există o altă piesă, dar de valoare diferită de piesa curentă




1. există o altă piesă de aceeași valoare cu piesa curentă, se va executa combinarea pieselor și se va actualiza linia de stop




În cazul 2. se verifică dacă prin combinarea celor 2 piese s-a obținut **2048** apelând **DISPLAY_CheckIfWon**. La final după ce toate piesele au fost mutate în locul corespunzător se redă un sunet specific acțiunii de *move*, led-ul își schimbă culoarea și se generează o noua piesă apelând **DISPLAY_GeneratePiece**.

*GeneratePiece* iterează prin vectorul g_empty_tiles selectând doar spațiile goale (valoarea == *empty*) din tabla de joc și apoi selectează aleatoriu un spațiu din cele goale în care să pună un 2 sau un 4 (90% șansă pentru 2 și 10% pentru 4). Dacă în urma generării unei piese noi nu mai există niciun spațiu liber pe tabla de joc se verifică dacă jocul a fost sau nu pierdut apelând **DISPLAY_CheckIfLost**. *CheckIfLost* verifică dacă se poate realiza cel puțin o combinație mutând sus/jos/stânga/dreapta, caz în care jocul nu a fost încă pierdut, altfel jocul s-a terminat și jucătorul a pierdut.

*CheckIfWon* este o funcție simplă în care se verifică dacă jucătorul a obținut **2048**, caz în care starea curentă se schimbă în *win_state*. Alte stări posibile din *game_state* sunt *pause_state* (a fost apăsat butonul *pause*), *quit_state* (a fost apăsat butonul *quit*) sau, conform celor menționate mai sus, în stare *lose_state*.

Starea *pause_state* afișează pe ecran un mesaj corespunzător (*GAME PAUSED!*) și setează *g_pause* pe 1, oprind scăderea scorului cu 10 puncte care se întâmplă în timpul stării *game_state*. Această stare ascunde tabla de joc pentru a nu oferi jucătorului timp de gândire fără penalizarea scorului. Din *pause_state* se poate reveni în *game_state* apăsând pe butonul *continue* care va afișa tabla de joc de la momentul de timp anterior apăsării butonului de *pause*.

În starea *lose_state* se afișează un mesaj corespunzător, se redă o melodie scurtă specifică înfrângerii și se setează led-ul pe roșu. În final se resetează jocul apelând **DISPLAY_ResetGame** care zeroizează tabla de joc, resetează starea la *quit_state*, precum și celelalte variabile la valorile inițiale.

Starea *win_state* este mai specială (funcția **DISPLAY_PrintWin**). Inițial, se urmează pașii de la *lose_state* afișând un mesaj corespunzător, melodia din *g_win_melody* și schimbând culorile led-ului; după ce se termină această etapă, se realizează citirea valorilor de pe Ox și Oy ale joystick-ului pentru a itera prin 3 caractere 'AAA' care pot fi schimbate în intervalul A-Z. Când jucătorul și-a ales numele, acesta va apăsa pe butonul *continue* pentru a începe operația de înregistrare a scorului său în top. Dacă se apasă pe *quit* în loc de *continue*, atunci se va reseta jocul, iar scorul nu va fi înregistrat. Înregistrarea scorului presupune obținerea datelor din memoria EEPROM și identificarea primului scor mai mic decât cel obținut de jucător. Dacă nu există niciun scor mai mic decât cel curent, atunci nu se înregistrează, în caz contrar, scorul curent îl înlocuiește pe cel din top, iar restul scorurilor mai mici ca

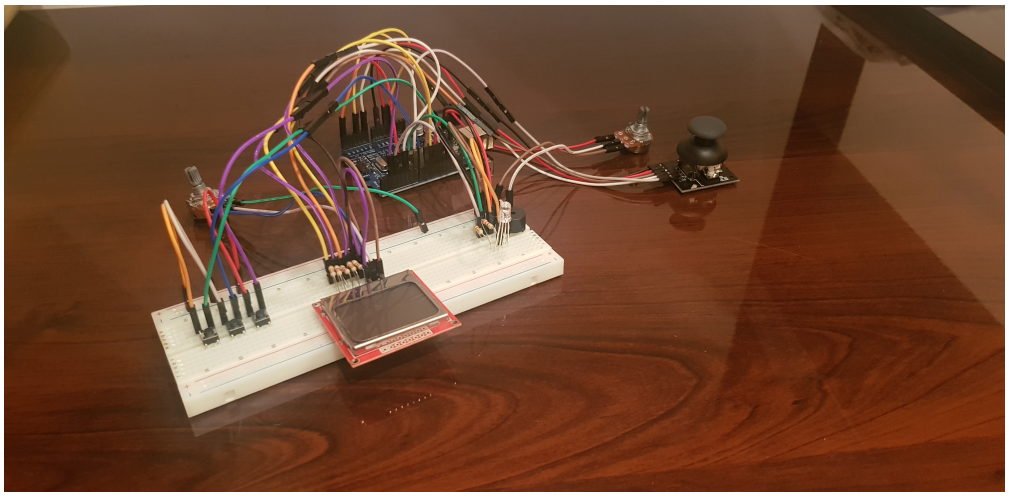
acesta sunt coborâte cu o poziție (inclusiv cel înlocuit).

După ce jucătorul a câștigat/pierdut jocul, se realizează resetarea acestuia și întoarcerea în meniul principal. Acum acesta poate porni un joc nou sau poate vedea noul top.

## Rezultate Obţinute

Proiectul a ieșit în mare parte exact cum mi-am propus de la început. Din păcate am rămas foarte rapid fără pini liberi pe plăcuța Arduino și a trebuit să renunț la ideea originală în care foloseam mai multe led-uri și buzzere pentru *quality of life*, dar am reușit să mă încadrez în pinii puși la dispoziție de plăcuță. În afară de acest mic inconvenient am realizat exact tot ceea ce am dorit să fac pentru acest proiect. Personal, consider că rezultatul final este mult mai bun față de imaginea pe care o aveam la început .

## Link YouTube 4K



## Concluzii

Când am început lucrul efectiv pentru proiect am avut multe momente când nu știam cu siguranță dacă ceea ce mi-am propus era în totalitate posibil de realizat, majoritatea problemelor venind din partea lucrului cu LCD-ul; am trecut prin multe iterații ale mesajelor afișate și ale reprezentării pieselor pe ecran. Am căutat foarte multe biblioteci care să ofere suport pentru un font mai mic astfel încât să încapă 4 numere pe 4 cifre pe o linie. Din păcate, majoritatea bibliotecilor disponibile pentru lucrul cu Nokia 5110 nu aveau nici măcar pe aproape la fel de multe funcționalități precum biblioteca din partea ADAFRUIT. Pe această problemă am pierdut cel mai mult timp, în final alegând să rămân la ADAFRUIT și găsind după foarte multe căutări o bibliotecă de fonturi compatibile cu *Adafruit_GFX*.

Un alt **challenge**, din nefericire, a venit încă de la început când am primit piesele. Ecranul Nokia 5110 nu a venit lipit (nu mai există modelul gata lipit) și a trebuit să realizez lipirea ecranului pe coloana de

pini . Rezultatul putea fi mai bun, dar cel puțin funcționează cum trebuie .

Cu toate acestea, am reușit să duc proiectul la bun sfârșit și consider că am reușit să aprofundez mai mult și mai bine noțiunile prezentate la laborator. Proiectul a fost într-adevăr o provocare foarte bună care a venit ca un suport pentru ce ne-a fost prezentat atât la curs cât și în cadrul laboratoarelor.

De-a lungul procesului de realizare a proiectului am obținut motivația de a face și alte astfel de proiecte orientate pe partea de hardware. Concluzionând, consider că proiectul a fost un challenge bine-venit și sunt foarte satisfăcut de rezultatul final .

## Download

Arhiva cu sursele:

2048_Deluxe.zip

## Jurnal

- 23.04.2021 - Am creat pagina de wiki.
- 25.04.2021 - Am completat secțiunile necesare din wiki conform assignment-ului de pe moodle.
- 07.05.2021 - Am finalizat în mare parte jocul 2048.
- 13.05.2021 - Am terminat proiectul + adăugat toate componentele.
- 22.05.2021 - Am făcut clean up codului sursă + coding style + completat pagina de wiki.
- 23.05.2021 - Am terminat pagina de wiki.
- 26.05.2021 - Am pus poze + link youtube.

## Bibliografie/Resurse

## Biblioteci externe

- Adafruit_GFX.h
- Adafruit_PCD8544.h
- Font4x7Fixed.h

# Resurse Eagle

- adafruit.lbr
  - schema Eagle pentru Arduino Uno R3
- diy-modules.lbr
  - diverse scheme Eagle compatibile cu Arduino (cum ar fi display-ul Nokia 5110)
- SparkFun-LED.lbr
  - led RGB catod comun

# Bibliografie/alte resurse

- Laboratoarele de pe OCW
- Google
- Pinout Arduino Uno
- Datasheet ATMEGA328P
- draw.io
- Eagle
- Arduino IDE

### Pagina în format PDF

---

2048-deluxe.pdf