Consola de jocuri - MINESWEEPER - Anghel Andrei - Stelian

Introducere

Ideea mea initiala a fost aceea de a implementa faimosul joc **Minesweeper** pe un ecran **LCD** de dimensiune 128×64. In timpul realizarii montajului pieselor, am realizat potentialul proiectului si am modificat anumite componente hardware pentru a obtine o **consola de jocuri portabila personalizata.**

Consola prezinta un ecran LCD, buzzer si controller cu 4 butoane pentru directii si 2 butoane de selectie.

Practic consola poate rula orice joc implementabil pe un ecran LCD de dimensiuni mici, dar in cazul proiectului meu am implementat un joc **Minesweeper de dimensiune 4×8**. Astfel, cele 4 butoane pentru directii sunt folosite pentru a deplasa cursorul pe ecran, iar butoanele de selectie au functiile REVEAL - afiseaza continutul unui camp ascuns si FLAG - pune un steag pe campul curent.

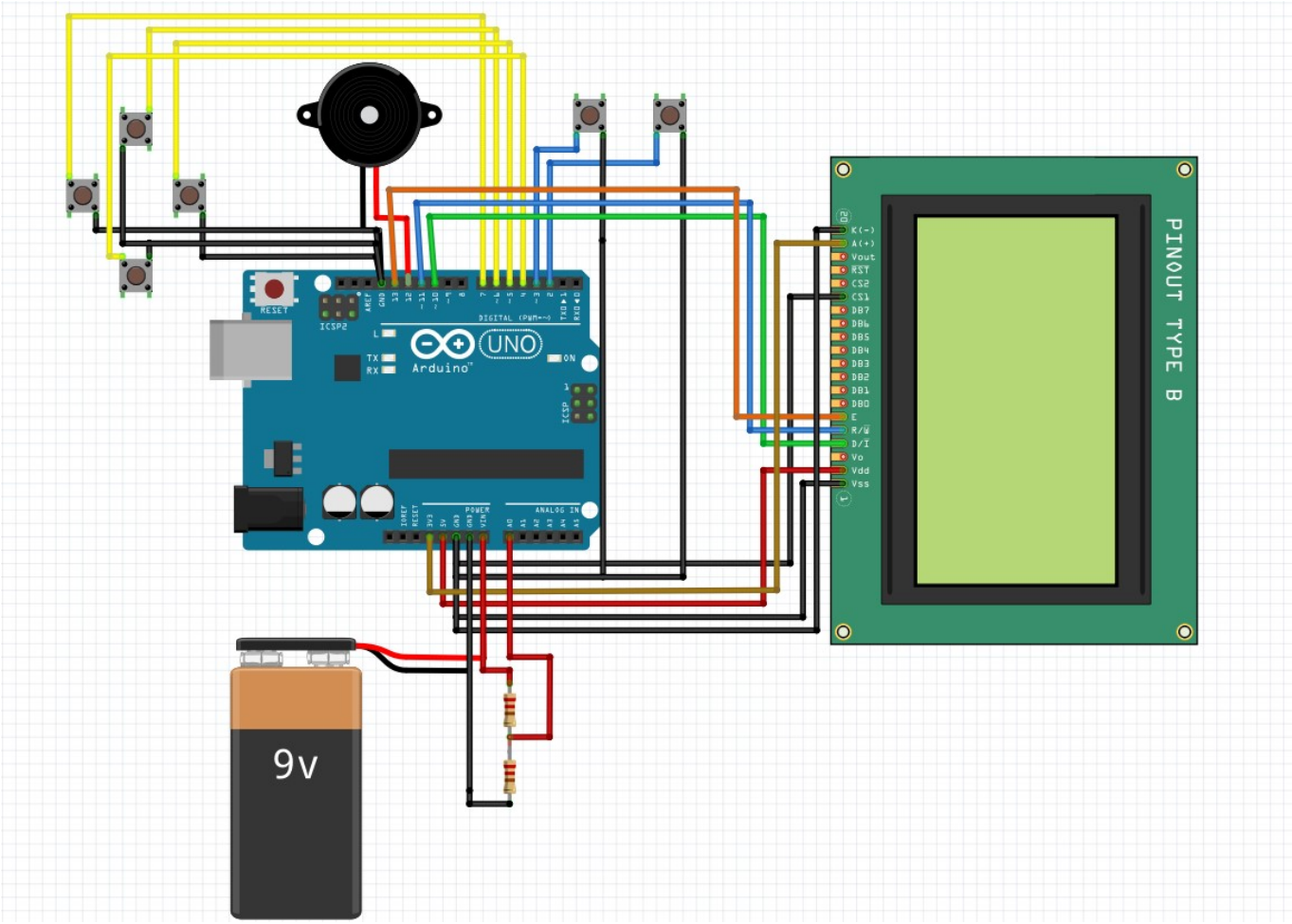
Hardware Design

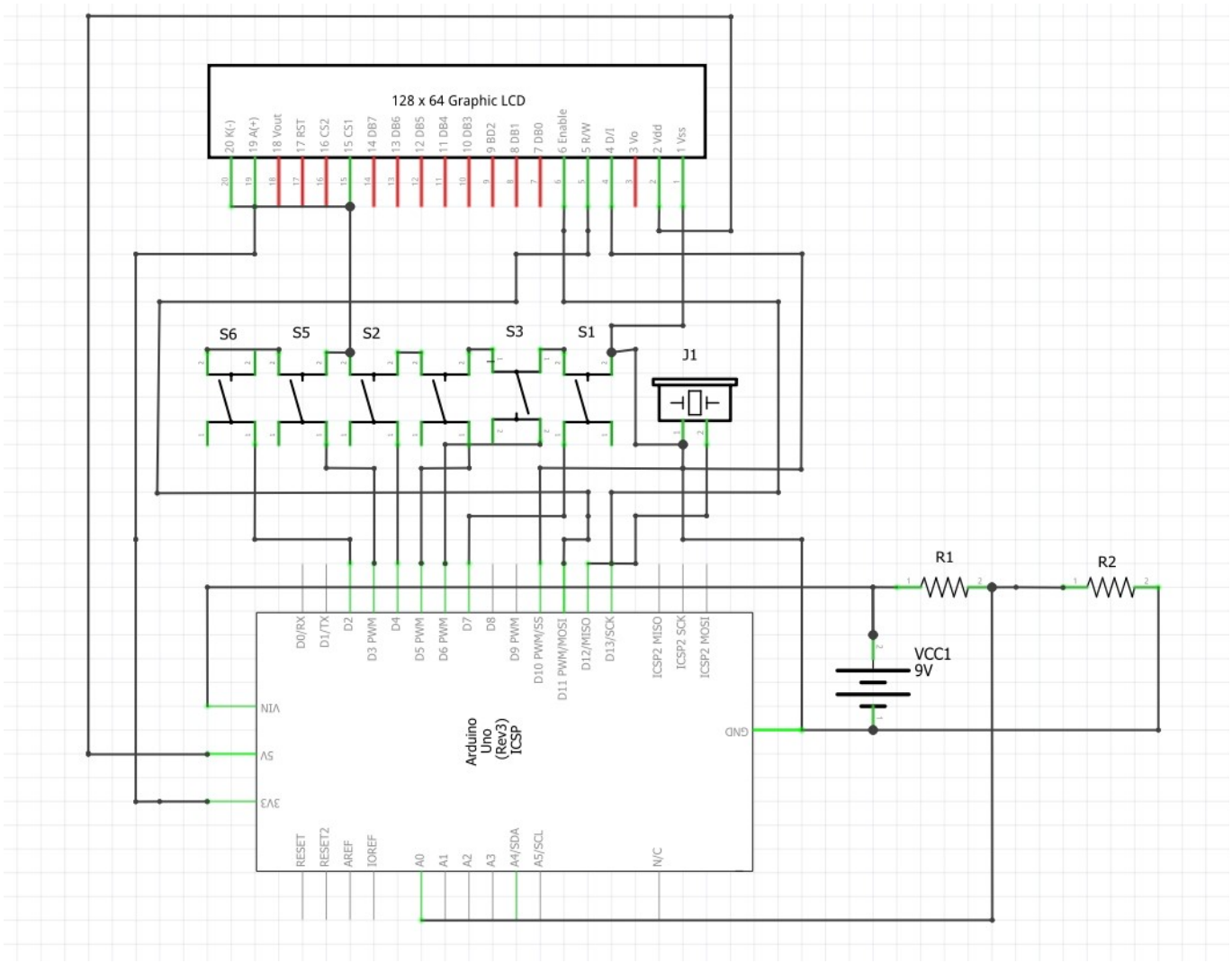
Lista componentelor hardware:

```
1 x Groundstudio Jade Uno+
1 x Display LCD 128*64 (5V iluminat, ST7920 controller)
1 x Modul Buzzer
6 x Buton
1 x Baterie 9V
1 x Conector baterie 9V
2 x Rezistor 22k Ω
```

Una dintre functionalitatile consolei pe care am introdus-o este afisarea **procentului de incarcare curent al bateriei.** Pentru masurarea tensiunii bateriei am utilizat pinul analogic **A0** al placutei. Am redus tensiunea de 9V la o tensiune de sub 5V folosind un **divizor de tensiune**, iar estimand rata de descarcare a bateriei am putut aproxima nivelul de incarcare al bateriei.

Mai jos se poate observa schema electrica pe care am urmat-o in realizarea proiectului.

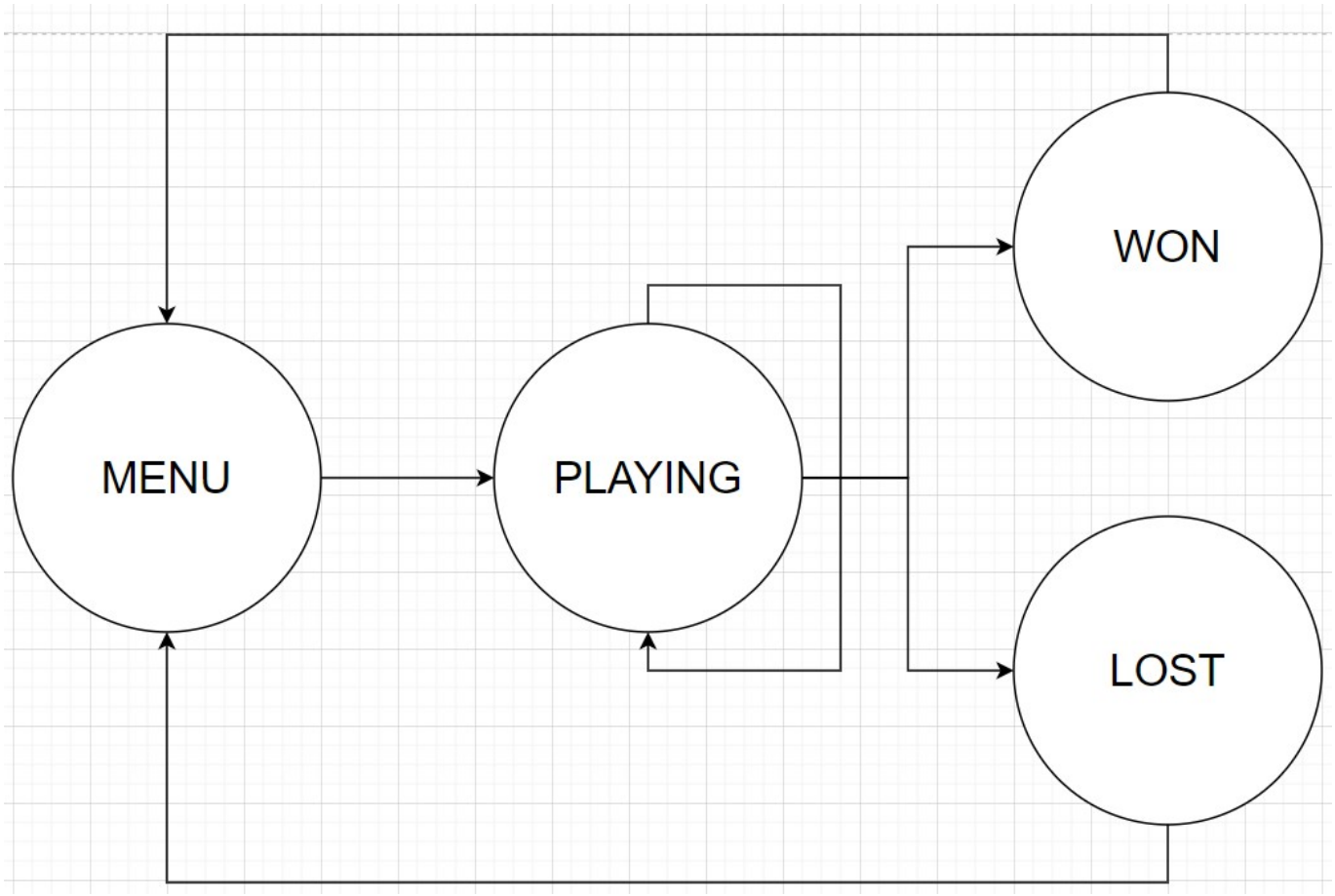

PINOUT TYPE B
K(-)
A(+)
Vout
RST
CS2
CS1
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DB0
E
R/W
D/I
Vo
Vdd
Vss
UNO
Arduino
9v



Software Design

Functionarea consolei poate fi redusa la urmatoarele stari:

- **MENU**
- **PLAYING**
- **WON**
- **LOST**



Starea initiala este cea de **MENU**, in care este afisat nivelul bateriei consolei si in care se poate alege dificultatea jocului. Odata ales nivelul de dificultate, se trece in starea de **PLAYING**. Cheia implementarii sunt matricile **boardState** - retine informatii despre stadiul curent al tablei de joc si **bombs** - retine pozitiile bombelor.

`int boardState[ROWS][COLUMNS];` 0-8 = number of adjacent bombs, 10 = not revealed, 11 = bomb, 12 = flag

`int bombs[ROWS][COLUMNS] = {0};` 0 = no bomb, 1 = bomb

Matricea de stari este reprezentata in mod continuu pe ecranul **LCD**, iar orice modificare in starile acesteia este vizibila pe ecran. Odata ce se indeplineste conditia de victorie sau conditia de pierdere a jocului, se trece in una din starile **WON** sau **LOST**, din care se poate reveni in starea **MENU** prin apasarea butonului **RESET** de pe placuta. Am implementat diverse functii cu nume sugestive pentru implementarea logicii:

```
// Randomly generates boms.
// Seed for random functions is taken
// from the analog pin A1.
void generateBombs() {
  for (int i = 0; i < ROWS; i++)
    for (int j = 0; j < COLUMNS; j++)
      bombs[i][j] = 0;

  randomSeed(analogRead(A1));

  for (int i = 0; i < numberOfBombs; i++) {
```

```
    int pos_i = random(ROWS);
    int pos_j = random(COLUMNS);
    if (bombs[pos_i][pos_j] == 1) {
      i--;
      continue;
    }
    bombs[pos_i][pos_j] = 1;
  }
}
```

```
// Calculates number of neighbouring bombs.
int getNumberOfNeighbourBombs(int i, int j) {
  int nr = 0;
  if (isPosValid(i - 1, j) && bombs[i - 1][j] == 1)
    nr++;

  if (isPosValid(i - 1, j - 1) && bombs[i - 1][j - 1] == 1)
    nr++;

  if (isPosValid(i - 1, j + 1) && bombs[i - 1][j + 1] == 1)
    nr++;

  if (isPosValid(i, j - 1) && bombs[i][j - 1] == 1)
    nr++;

  if (isPosValid(i, j + 1) && bombs[i][j + 1] == 1)
    nr++;

  if (isPosValid(i + 1, j) && bombs[i + 1][j] == 1)
    nr++;

  if (isPosValid(i + 1, j - 1) && bombs[i + 1][j - 1] == 1)
    nr++;

  if (isPosValid(i + 1, j + 1) && bombs[i + 1][j + 1] == 1)
    nr++;

  return nr;
}
```

```
// Function used to reveal all
// bombs, in case of loss
void revealAllBombs() {
  for (int i = 0; i < ROWS; i++) {
    for (int j = 0; j < COLUMNS; j++) {
      if (bombs[i][j] == 1) {
        boardState[i][j] = BOMB;
      }
    }
  }
```

```
}
```

```
// Check win condition
void checkWin() {
  if (fieldsRevealed == ROWS * COLUMNS - numberOfBombs)
    gameState = 1;
}
```

Pentru realizarea graficii am utilizat libraria **u8g** si functii precum:

- **U8GLIB_ST7920_128X64_4X u8g(10)** - initializarea ecranului
- **u8g.setFont()** - alegerea fontului
- **u8g.drawStr()** - scrierea unui String la o anumita pozitie
- **u8g.drawXBMP()** - afisarea unui bitmap - imagine predefinita, precum cele pentru bomba sau steag

Buzzer-ul este configurat prin intermediul registrului timer oferit de **ATmega328pb** si este configurat pentru a emite sunete la interval de o secunda.

```
// Setting up the timer register for the buzzer
noInterrupts();
TCCR1A = 0;
TCCR1B = 0;
TCNT1  = 0; // initialize counter value to 0
// set compare match register for 1 Hz increments
OCR1A = 62499;
// turn on CTC mode
TCCR1B |= (1 << WGM12);
// Set CS12, CS11 and CS10 bits for 256 prescaler
TCCR1B |= (1 << CS12) | (0 << CS11) | (0 << CS10);
// enable timer compare interrupt
TIMSK1 |= (1 << OCIE1A);
interrupts();
```

Nivelul de incarcare al bateriei este masurat prin intermediul pinului analogic A0 si estimat in functie de rata de descarcare a unei baterii alcaline de 9V.

```
// Reading battery voltage
int inputValue = analogRead(A0);
float batteryVoltage = inputValue * (5.0f / 1024.0f) * (22 + 22) / 22;
// Calculated using aproximate battery discharge rate
float batteryPercentage = 25 * batteryVoltage - 125;
```

Butoanele FLAG si **REVEAL** sunt legate la pinii digitali 2 si 3, iar declasarea acestora este realizata prin intermediul **intreruperilor hardware.**

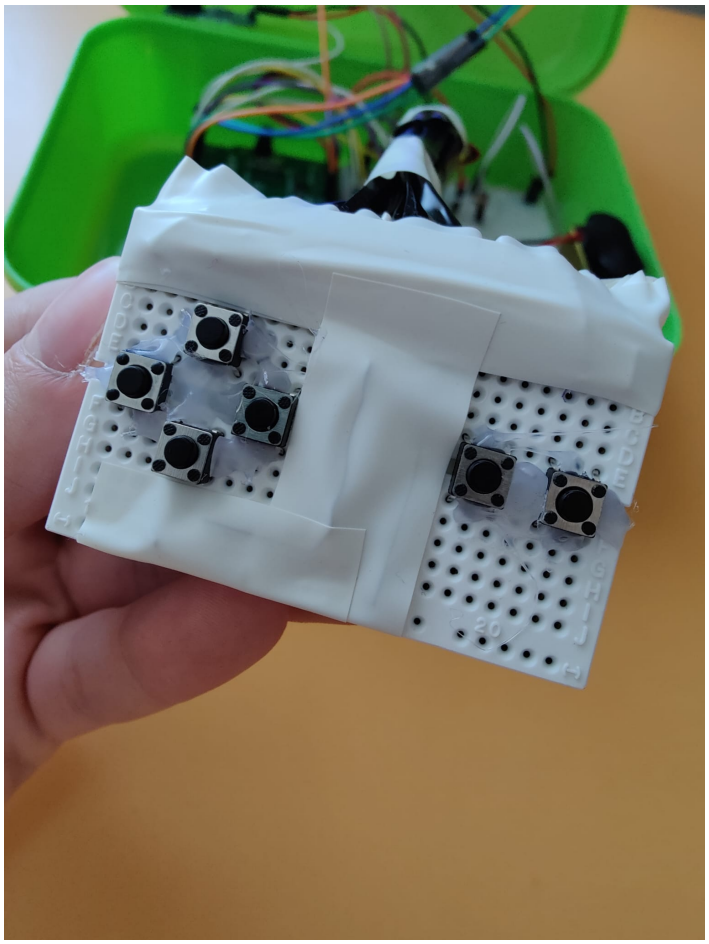
```
// Pentru butonul FLAG
attachInterrupt(digitalPinToInterrupt(flagButton), btnInterruptFlag, FALLING);
```

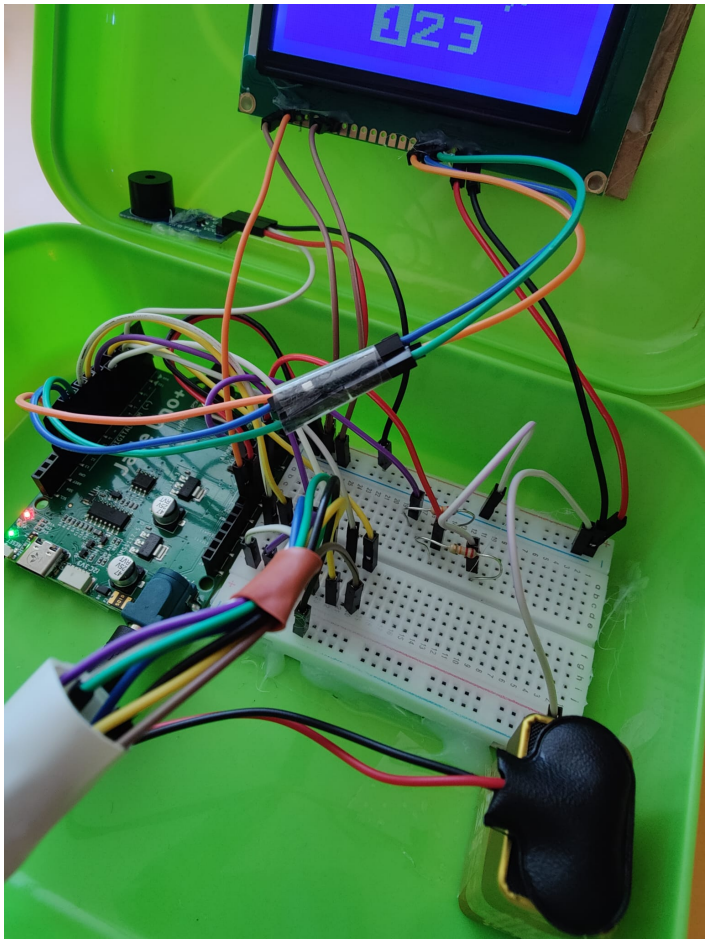
```
void btnInterruptFlag() {
  interruptFlag = true;
}
```

Rezultate Obţinute

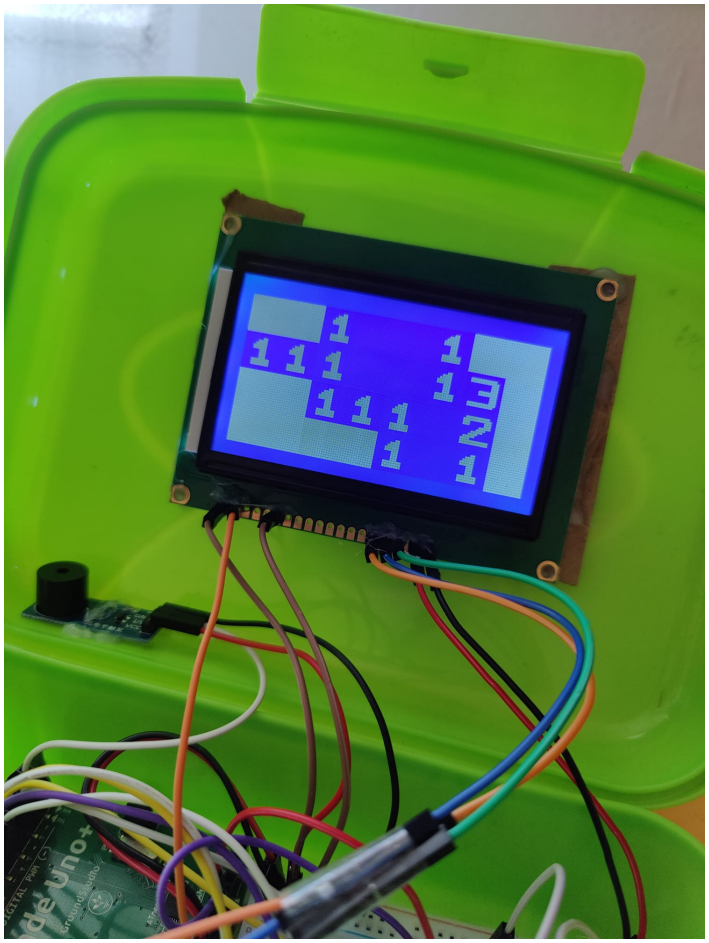
Am lipit toate componentele intr-o cutie si am creat propriul controller, folosind un breadboard de dimensiuni mici.

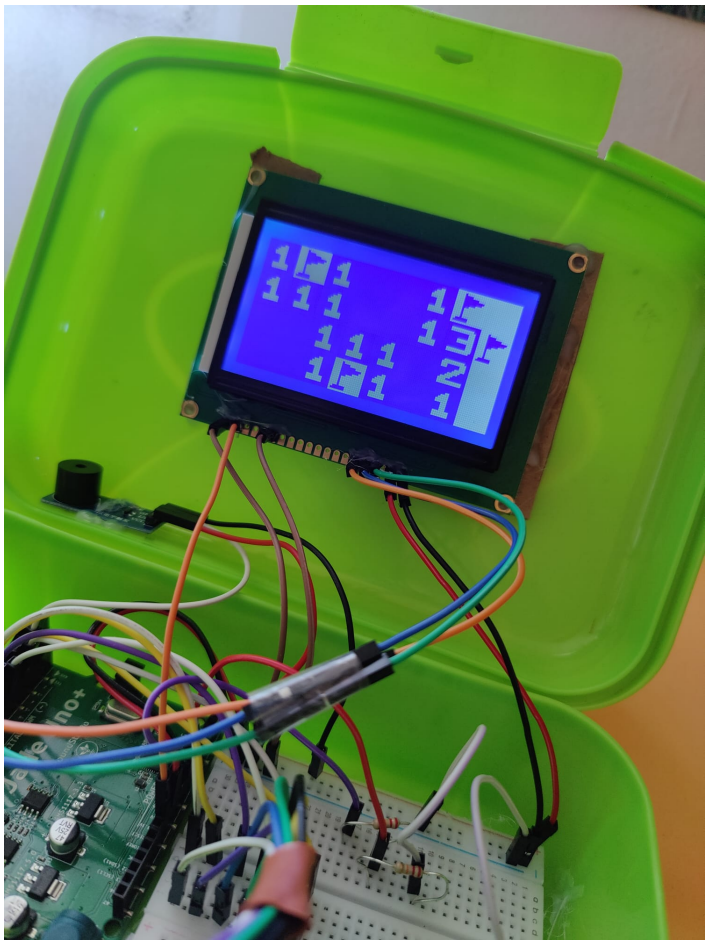

MINESWEEPER
Battery: 61%
Select difficulty:
1 2 3
VARTA





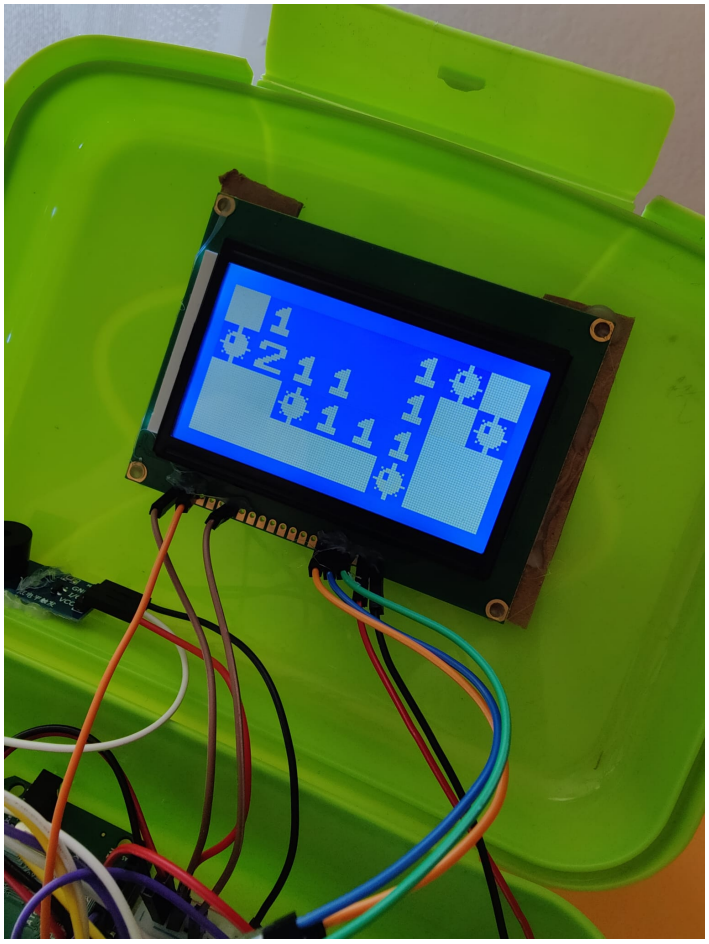
Poze din diverse stadii ale jocului:






YOU WON!
Press 'RESET' to play again.


YOU LOST!
Press 'RESET' to play again.



Concluzii

Realizarea acestui proiect a fost o experienta foarte interesanta, ce a pornit de la pasiunea mea pentru un joc simplist si foarte popular, anume **Minesweeper**. In timpul dezvoltarii proiectului, am venit cu numeroase imbunatatiri si idei suplimentare, ajungand in final la un produs ce reprezinta o consola de jocuri portabila, cu feedback video, dar si audio si un controller cu 6 butoane pentru input. Proiectul poate fi oricand extins si imbunatatit, implementand numeroase alte joculete pentru consola.

Download

src_minesweeper.zip

Jurnal

- Realizarea montajului initial pentru verificarea functionalitatii pieselor

- Implementarea logicii jocului 'Minesweeper'
- Am decis sa alimentez circuitul la o baterie de 9V, pentru a transforma proiectul intr-o consola portabila
- Realizarea noului montaj si construirea controller-ului
- Finalizarea documentatiei

Bibliografie/Resurse

https://arduino-tutorials.net/tutorial/control-graphic-lcd-display-spi-st7920-128x64-with-arduino

https://github.com/olikraus/u8glib

Export to PDF