

# Minesweeper

## Introducere

□ **Minesweeper** – un joc clasic de logică, reinventat pentru platforme embedded!

□ **Scopul proiectului:** Jucătorul explorează o matrice de celule, evitând bombele □ și marcându-le corect cu steaguri □. Am pornit de la ideea jocului original de pe PC și l-am adaptat pe un sistem embedded, pentru a-l face mai interactiv. Este un joc care antrenează logica și atenția, fiind în același timp o demonstrație practică a integrării între hardware și software.

## Descriere generală

□ Implementare joc Minesweeper pe un LCD grafic.



□ Schema bloc a fost realizată în: [Draw.io](https://draw.io).

Placa de dezvoltare utilizată este compatibilă cu **Arduino UNO R3** și controlează un ecran **LCD TFT ST7735S** pentru afișarea grafică a jocului. Navigarea în matricea 8×8 se face cu ajutorul **joystick-ului analogic**, iar cele 2 butoane permit plasarea steagurilor, opțiunea de selecție a numelui și opțiunea de a pune pe pauză/a ieși din joc. **Buzzer-ul** oferă feedback auditiv în cazul declanșării unei bombe sau a terminării jocului. Logica jocului este implementată în software pe **Arduino**, care actualizează afișajul și răspunde în timp real la interacțiunile jucătorului.

## Hardware Design

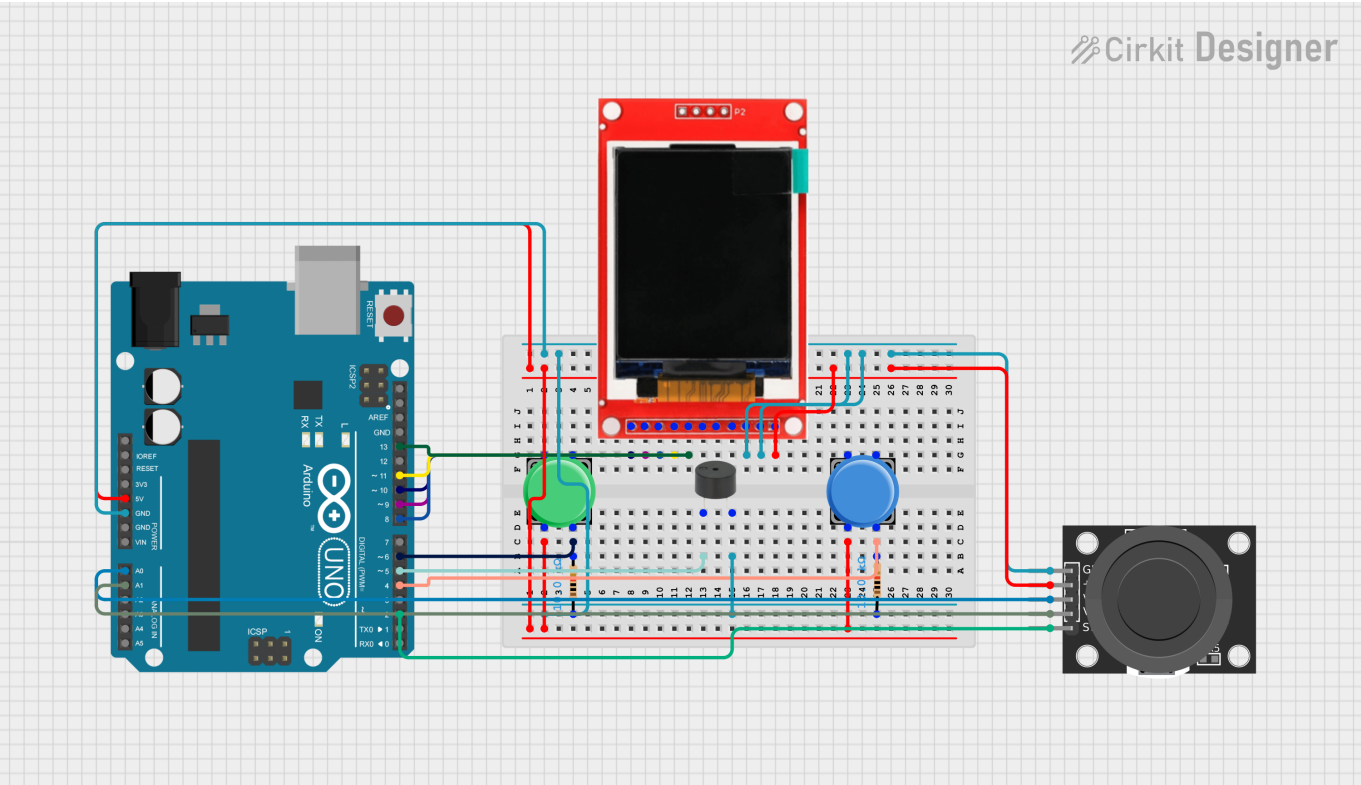
□ **Bill of Materials**

| Componenta                         | Link achiziție                      | Cantitate | Preț      |
|------------------------------------|-------------------------------------|-----------|-----------|
| Buton Alb                          | <a href="#">Buton</a>               | 2         | 1,99 lei  |
| Buton Albastru                     | <a href="#">Buton</a>               | 2         | 1,99 lei  |
| Joystick analogic                  | <a href="#">Joystick</a>            | 1         | 4,96 lei  |
| LCD, 1.8", 128×160 TFT, SPI Serial | <a href="#">Display</a>             | 1         | 54,99 lei |
| Arduino UNO R3 și cablu 50cm       | <a href="#">Placă de dezvoltare</a> | 1         | 39,37 lei |
| Buzzer Pasiv 5V                    | <a href="#">Buzzer</a>              | 1         | 1,40 lei  |

|                                   |                    |   |           |
|-----------------------------------|--------------------|---|-----------|
| Fire rigide                       | Set fire rigide    | 1 | 12,49 lei |
| Fire tată-tată                    | Set fire tată-tată | 4 | 2,85 lei  |
| Breadboard HQ (400 Points)        | Breadboard         | 1 | 4,56 lei  |
| Cost total componente: 137,13 lei |                    |   |           |

Toate componentele sunt conectate pe un breadboard pentru testare ușoară, iar afișajul comunică cu placa prin interfața **SPI**.

Schema circuit



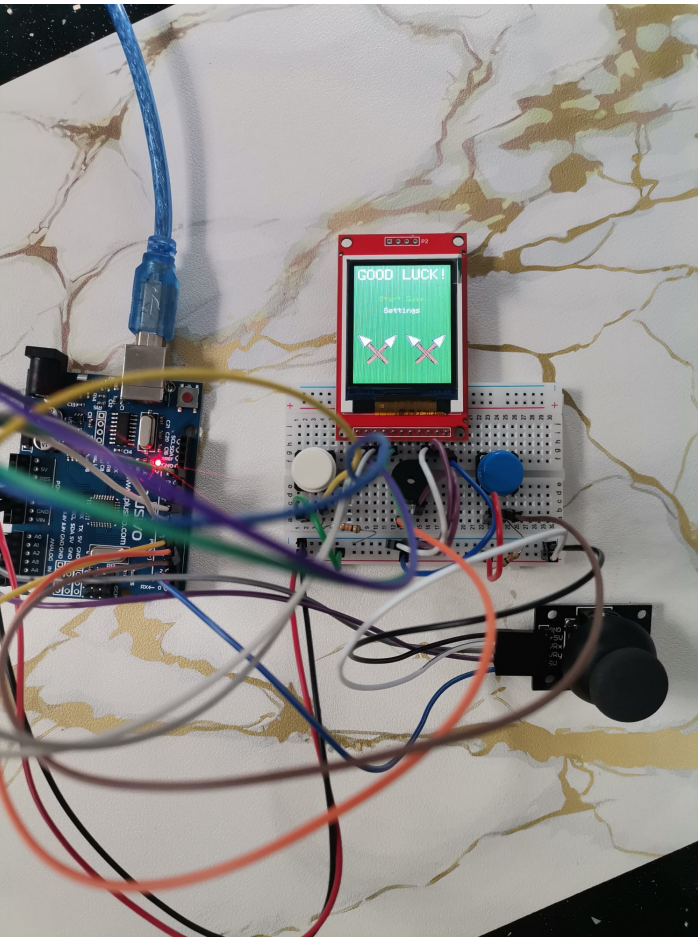
Schema circuitului a fost realizată folosind [Cirkit Designer](#).

Conexiuni pini

| Arduino Pin | Tip            | Conectat la                            | Note                |
|-------------|----------------|----------------------------------------|---------------------|
| D10         | Digital        | LCD CS                                 | Chip-Select SPI     |
| D9          | Digital        | LCD RST                                | Reset display       |
| D8          | Digital        | LCD RS/DC                              | Data/Command select |
| D11         | Digital (MOSI) | LCD SDA                                | SPI MOSI            |
| D13         | Digital (SCK)  | LCD SCL                                | SPI Clock           |
| 5V          | Power          | LCD VCC, Joystick VCC (+)              | +5 V alimentare     |
| GND         | Power          | LCD GND , Joystick GND (-), buzzer “-” | Masă comună         |
| A0          | Analog         | Joystick VRx                           | X-axis voltage      |
| A1          | Analog         | Joystick VRy                           | Y-axis voltage      |

|    |               |             |                                                                                       |
|----|---------------|-------------|---------------------------------------------------------------------------------------|
| D2 | Digital       | Joystick SW | Switch intern joystick → când apeși conectează D2 la GND; se folosește `INPUT_PULLUP` |
| D4 | Digital       | Buton 1     | Buton la +5 V, rezistor extern de pull-down (~10 kΩ) la GND                           |
| D6 | Digital       | Buton 2     | Buton la +5 V, rezistor extern de pull-down (~10 kΩ) la GND                           |
| D3 | Digital / PWM | Buzzer      | Buzzer pasiv 5 V, minus la GND; poate fi modulat cu PWM                               |

▣ Configurația finală realizată fizic



## Project Planning

Planificarea etapelor:

| ID | Activitate                  | Descriere                                                                                                 |
|----|-----------------------------|-----------------------------------------------------------------------------------------------------------|
| A  | Specificare cerințe         | Stabilirea cerințelor hardware/software, interfață cu joystick, afișaj TFT, butoane, buzzer               |
| B  | Design arhitectură          | Împărțirea codului pe module: input (joystick/butoane), game loop, generare hartă, afișare grafică, sunet |
| C  | Implementare module de bază | Scrierea funcțiilor de citire joystick, debounce, tone buzzer                                             |

|   |                                 |                                                                                                                        |
|---|---------------------------------|------------------------------------------------------------------------------------------------------------------------|
| D | Generare hartă și logică de joc | Algoritmi pentru plasarea bombardelor, flood-fill, verificare câștig/pierdere                                          |
| E | Grafică și UI                   | Funcții drawGrid(), drawHeader(), ecran de selecție nume, splash-screen, elemente decorative (bombe, steaguri, lopate) |
| F | Testare și optimizare           | Măsurarea timpilor de răspuns și optimizarea redraw-ului                                                               |
| G | Documentație și prezentare      | Redactarea README, diagrame, grafice Gantt, concluzii                                                                  |

### Relații de dependență:

- B → C, D, E: înainte de codul efectiv trebuie finalizat designul.
- C → D: logica de joc se bazează pe input-ul debounced.
- D → E: afișarea grafică și interfața necesită funcționalitatea de bază a jocului.
- C, D, E → F: testarea și optimizarea vin după ce modulele principale sunt implementate.

## Software Design

Proiectul a fost dezvoltat în mediul **Arduino IDE**, utilizând limbajul C/C++. Codul a fost încărcat pe placa Arduino UNO R3 și testat pe hardware real. Pentru afișarea grafică pe **LCD-ul TFT ST7735S**, au fost utilizate următoarele biblioteci:

- ``Adafruit_GFX.h`` – funcționalități grafice generale (afișare text, forme, linii)
- ``Adafruit_ST7735.h`` – controlul afișajului ST7735 prin SPI
- ``SPI.h`` – comunicație SPI între Arduino și display

Jocul rulează pe o matrice 8×8 în care fiecare celulă stochează starea proprie: bombă, descoperită, steag sau număr de vecini cu bombe.



Pozițiile bombelor sunt generate aleator, iar valorile vecinilor sunt calculate automat. Utilizatorul se poate deplasa prin matrice cu joystick-ul analogic și poate interacționa prin butoane:

- **Buton 1 / SW joystick** – descoperă celule (configurat cu **pull-up intern**)
- **Buton 2** – marchează/șterge steag/confirmă numele (configurat cu **rezistență de pull-down**)
- **Buton 3** – intră în **QUIT-MENU** / Go back functionality prin meniuri
- **Buzzer** – semnal sonor la pierdere (explozie bombă) sau la câștigare

Funcții implementate:

- ``isButtonPressedActiveLow_PD2()`` – verifică dacă butonul conectat la pinul 2 (cu semnal activ LOW) a fost apăsat și eliberat, cu debounce intern de ~50 ms.
- ``isButtonPressedActiveHigh_PD4/PD6()`` – verifică dacă butonul conectat la pinul 4/6 (cu semnal activ HIGH) a fost apăsat și eliberat, cu debounce intern de ~50 ms.
- ``fastAnalogReadA0/A1()`` – verifică și citește rapid valoarea tensiunii analogice de pe pinul A0 (ADC0) / A1 (ADC1), utilizând registrele interne ale convertorului analog-digital (ADC) al microcontrollerului.
- ``startTone()`` – activează semnalul PWM pe pinul PD3 (OC2B) cu frecvența specificată în Hz, folosind Timer2 configurat în mod CTC.
- ``stopTone()`` – oprește semnalul PWM generat de Timer2 pe pinul PD3 și setează pinul LOW.

- ``playBombSequence()`` - redă pe buzzer o succesiune de tonuri care indică „explozia” unei bombe (pierdere de joc).
- ``playWinSequence()`` - redă pe buzzer o succesiune de tonuri care indică victoria în joc.
- ``generateBoard()`` - reinițializează matricea de joc: plasează aleator 10 bombe (valori 9) și apoi calculează pentru fiecare celulă ne-bombă numărul de bombe din cele 8 poziții adiacente.
- ``flood()`` - descoperă recursiv (flood-fill) toate celulele cu valoarea 0, pornind din poziția (r,c), și incrementează contorul de celule descoperite.
- ``checkWin()`` - returnează true dacă numărul de celule descoperite + numărul de bombe acoperă întreaga grilă, adică jucătorul a câștigat.
- ``drawGrid()`` - desenează întreaga matrice 8×8 pe TFT: pentru fiecare celulă (x,y): dacă este „descoperită”, afișează fie fundal alb + număr/bombă; dacă este „ascunsă”, afișează fundal albastru + steag; apoi trasează conturul negru al celulei și la final evidențiază poziția cursorului cu un contur verde.
- ``updateScore()`` - șterge zona veche a scorului din header și afișează noul număr de celule descoperite.
- ``updateTimer()`` - șterge zona veche a timer-ului din header, calculează timpul rămas (MM:SS) și îl afișează.
- ``drawLetterGrid()`` - desenează ecranul de selectare nume: un mic header „Select Name” și o grilă 7×4 cu literele A-Z dispuse pe rânduri.
- ``drawHeader()`` - reîncarcă complet banda de sus a ecranului de joc.
- ``drawBomb()`` - desenează o bombă în jurul punctului (cx,cy) cu raza R: un cerc negru conturat cu alb, opt țepi albi pentru aspectul de explozie și un ochi central roșu cu un fir de fitil galben.
- ``drawFlag()`` - trasează un steag roșu într-o celulă definită de colțul (x,y) și dimensiunile date: o tijă neagră verticală și un triunghi roșu cu bordură albă în vârful acesteia.
- ``drawCrossedPickaxes()`` - desenează două târnăcoape încrucișate centrate în jurul punctului (cx,cy), scalate de factorul size: mânere sub formă de dreptunghiuri maro, capete metalice umplute cu gri și conturate în negru, plus un accent alb care sugerează o reflexie pe lamă.
- ``animateBombReveal()`` - animație de descoperire a bombelor atunci când calci pe una. Calculează pozițiile tuturor bombelor, le amestecă într-o ordine aleatoare și animăm apariția acestora.
- ``drawSplash()`` - desenează meniul de start cu un mesaj de 'GOOD LUCK' în partea de sus, urmat de 2 opțiuni “START GAME” și “SETTINGS” și randarea tarnacoapelor folosind funcția `drawCrossedPickaxes()`.
- ``drawDifficultyMenu()`` - desenează meniul de setări, mai exact dificultatea pe care o poți alege în timpul jocului. Easy care va genera doar 4 bombe, Medium cu 7 bombe și Hard cu 10 bombe.
- ``drawConfirmQuitMenu()`` - desenează meniul de **QUIT GAME** din timpul jocului. Acest meniu prezintă 2 opțiuni “No” → continui jocul sau “Yes” → care te duce în meniul principal.

Pentru funcțiile de draw am utilizat diverse **apeluri grafice** printre care:

```
tft.fillRoundRect(x, y, w, h, r, color)
```

Cu parametrii:

- x, y — coordonatele colțului din stânga-sus
- w, h — lățimea și înălțimea dreptunghiului
- r — raza de rotunjire a colțurilor
- color — culoarea de umplere (16-bit RGB565)

```
tft.fillTriangle(x0, y0, x1, y1, x2, y2, color)
```

Cu parametrii:

- (x0,y0), (x1,y1), (x2,y2) — vârfurile triunghiului
- color — culoarea de umplere (16-bit RGB565)

```
tft.drawTriangle(x0, y0, x1, y1, x2, y2, color)
```

Cu parametrii:

- (x0,y0), (x1,y1), (x2,y2) — vârfurile triunghiului
- color — culoarea de umplere (16-bit RGB565)

Funcțiile grafice utilizate în cod:

| Funcție       | Ce face                               |
|---------------|---------------------------------------|
| fillScreen    | curăță tot ecranul într-o culoare     |
| fillRect      | umple un dreptunghi                   |
| drawRect      | contur dreptunghi                     |
| fillCircle    | cerc solid                            |
| drawCircle    | contur cerc                           |
| drawFastHLine | linie orizontală                      |
| drawLine      | linie oblică                          |
| fillTriangle  | triunghi solid                        |
| drawTriangle  | contur triunghi                       |
| fillRoundRect | dreptunghi cu colțuri rotunjite solid |
| setCursor     | poziționează „cursorul” pentru text   |
| setTextSize   | setează mărimea fontului              |
| setTextColor  | setează culoarea textului             |

Logica din funcția de **loop()** constă într-un switch care jongleaza cu stările posibile:

```
enum GameState {STATE_SPLASH, STATE_SETTINGS, STATE_ENTER_NAME, STATE_PLAY, STATE_CONFIRM_QUIT};
```

- **STATE\_SPLASH** → Starea pentru meniul principal
- **STATE\_SETTING** → Starea pentru meniul de setări
- **STATE\_ENTER\_NAME** → Starea pentru meniul de selectare a numelui
- **STATE\_PLAY** → Starea pentru meniul cu jocul propriu-zis
- **STATE\_CONFIRM\_QUIT** → Starea pentru meniul de **QUIT GAME**

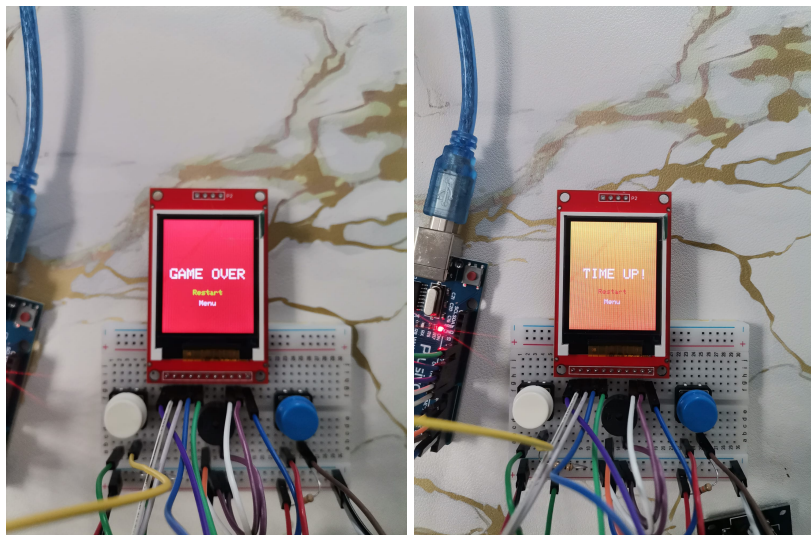
| Laboratoare folosite | Funcționalitate                   | Cod                                                                                                                       |
|----------------------|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| GPIO                 | citire / scriere de pini digitali | <b>configurePinsWithRegisters()</b> folosind regiștrii                                                                    |
| UART                 | comunicație serială               | Nu este explicit în cod, dar am folosit pentru afișajul în consola pentru debug                                           |
| Timere & PWM         | generare ton și temporizări       | <b>startTone(uint16_t freq)</b> si <b>stopTone()</b> folosind Timer0 pe 8 biti pentru PWM                                 |
| ADC                  | conversie analog-digitală         | <b>fastAnalogReadA0()</b> și <b>fastAnalogReadA1()</b> pentru citirea poziției joystick-ului folosind regiștrii           |
| SPI                  | interfațare display TFT           | <b>tft.initR(INITR_BLACKTAB)</b> + toate comenzile <b>tft.xxx</b> folosesc <b>SPI</b> pentru comunicare cu ecranul ST7735 |

## Rezultate Obținute

Codul sursă și alte imagini pot fi observate pe [pagina de GitHub](#)

Scurt **demo**  de prezentare poate fi gasit pe [linkul de YouTube](#)





## Concluzii

Proiectul **Minesweeper** realizat pe **Arduino** a fost o experiență foarte reușită și satisfăcătoare. A reușit să aducă laolaltă funcționalitatea completă a jocului, grafică interactivă și control prin joystick și butoane, toate integrate pe un ecran TFT. Faptul că totul rulează în timp real m-a ajutat să înțeleg mai bine ce înseamnă să optimizezi interfața și logica jocului pentru resurse limitate.

Mi-a plăcut în mod special partea de redare continuă a ecranului, care mi-a amintit de temele de la cursul de grafică. M-am bucurat să regăsesc acolo concepte precum bucla de render, actualizarea doar a zonelor modificate și controlul precis asupra afișajului. A fost interesant să văd cum acele noțiuni se aplică și într-un context diferit, pe un microcontroler, cu constrângeri reale de memorie și procesare.

Pe lângă partea grafică, m-a ajutat mult și pe partea de hardware: am lucrat cu pini GPIO, citirea joystick-ului prin ADC, comunicația SPI cu display-ul. Toate astea m-au făcut să înțeleg mai bine ce înseamnă să îmbini partea software cu cea hardware într-un mod funcțional și coerent.

În final, proiectul ăsta mi-a dat un plus de încredere și clar mi-a trezit interesul pentru proiecte embedded mai complexe.

## Download

[Arhivă](#) cu tot conținutul proiectului.

## Jurnal

03.05.2025 – Am început redactarea documentației proiectului: am completat secțiunile **Introducere** și **Descriere generală** și am încărcat schema bloc în pagina wiki, completând lista de

componente pe care le voi utiliza și ideea de start a design-ului software.

□ **05.05.2025** - □ Am cumpărat piesele menționate în lista de componente hardware.

□ **10.05.2025** - □ Componentele au fost livrate și am început montajul hardware.

□ **11.05.2025** - □ Am finalizat implementarea hardware a proiectului. Am realizat **schema circuitului** pe breadboard și am desenat **schema electrică** completă, documentând conexiunile între toate componentele utilizate.

□ **18.05.2025** - □ Am realizat implementarea software-ului.

□ **20.05.2025** - □ Am terminat ultimele retușuri, am finalizat pagina de ocw, am creat repo-ul de GitHub și am încărcat un demo intuitiv pe YouTube

## Bibliografie/Resurse

### Hardware Related

[Datasheet Arduino UNO R3](#)

[Datasheet ATmega328P](#)

[Guide display](#)

[Tutorial display with Arduino](#)

[Guide joystick](#)

[Tutorial button with Arduino](#)

[Tutorial buzzer with Arduino](#)

### Software Related

[Adafruit-GFX-Library](#)

[Adafruit\\_ST7735S](#)

[Adafruit GFX Graphics Library](#)

[Arduino Documentation](#)

[Export to PDF](#)

From:

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

Permanent link:

<http://ocw.cs.pub.ro/courses/pm/prj2025/ajipa/mihai.lemnaru>



Last update: **2025/05/29 20:20**