Ultimate Tic-Tac-Toe - Roșu Mihai Cosmin - 333CA

Demo

Introducere

Ideea principală

Proiectul constă în implementarea jocului Ultimate Tic-Tac-Toe folosind drept interfață grafică un mic ecran LCD. Jocul oferă două moduri de joc:

1. **Singleplayer**: Utilizatorul joacă împotriva unui AI care dispune de două dificultăți de joc: *Easy* și *Hard*.
2. **Multiplayer**: Pentru acest mod sunt necesari doi utilizatori care își vor juca turele pe rând.

Motivație

Mereu am fost pasionat de jocuri și întotdeauna mi s-a părut interesantă ideea de a folosi un microprocesor/microcontroller pentru a crea ceva de la zero, așa că acest proiect a fost oportunitatea perfectă de a recrea unul dintre jocurile copilăriei (X și 0), într-un format mai dificil (Ultimate Tic-Tac-Toe).

Descriere generală

Explicarea proiectului

- Pentru început, jucătorul poate folosi butoanele pentru a-și alege modul de joc dorit, iar apoi, în cazul în care a fost ales modul de Singleplayer, dificultatea dorită și cu ce simbol vrea să joace (X sau 0). Odată alese, jocul începe.
- Pentru a selecta căsuța dorită pentru plasarea unui X (sau 0) este folosit potențiometrul, care oferă posibilitatea parcurgerii tablei de joc linie cu linie, de la stânga la dreapta. După ce este aleasă

casuța dorită, este folosit unul dintre butoane pentru a definitiva alegerea. Dacă alegerea făcută nu este permisă, acest lucru este anunțat de buzzer, iar jucătorul trebuie sa aleagă din nou o căsuță.
- În final, la terminarea jocului, buzzer-ul va face un sunet pentru a semnala încheierea jocului.

Laboratoare folosite

Laboratoarele folosite pentru realizarea proiectului sunt:

- Laborator 2. Întreruperi hardware
- Laborator 3. Timere. PWM
- Laborator 4. ADC
- Laborator 5. SPI

Schema bloc

Modul în care interacționează componentele este următorul: Arduino primește întreruperi de la butoane și de la potențiometru și trimite date către ecran și către buzzer.

Hardware Design

Lista componentelor

Lista componentelor folosite în cadrul proiectului:

- Arduino UNO - ATmega328P
- Ecran ST7789v
- Translator de nivel logic 5V - 3.3V
- Potențiometru 10K
- Buzzer
- 2 x Butoane
- 2 x Rezistențe 10K
- Breadboard
- Fire de legătură

Schema circuitului

Software Design

Detalii generale

- Pentru dezvoltarea proiectului am folosit **Arduino IDE**, în cadrul căruia am importat următoarele biblioteci:
  1. **Adafruit GFX**, bibliotecă ce se poate importa direct din IDE, necesară pentru a desena/scrie pe ecran
  2. **Arduino ST7789 Fast**, bibliotecă externă, folosită pentru a comunica cu ecranul prin **SPI**, folosește **Adafruit GFX** și **SPI.h**
- În cadrul implementării am folosit și operații cu registre pentru:
  1. **ADC**: Am ales să configurez și să folosesc ADC astfel, fiindcă am avut nevoie să controlez prescalerul folosit de acesta. Acest lucru era necesar fiindcă pentru o citire de ADC, de fapt au loc 100 de citiri, pentru a reduce noise-ul.
  2. **PWM**: Pentru a trimite un semnal PWM buzzer-ului, am ales să folosesc registre pentru a configura **Timer1** în modul **FastPWM**. De asemenea, pentru a determina când trebuie oprit semnalul PWM, folosesc întreruperea de overflow a timer-ului.
  3. **Întreruperi**: Pentru întreruperile externe **INT0**, **INT1** și întreruperea de timer **TIMER1_OVF**.

Implementare generală

- Jocul are **5 stări**, care coordonează ce este afișat pe ecran: Meniu principal, alegerea dificultății pentru Singleplayer, alegerea simbolului, joc și final.
- În funcție de starea în care se află jocul, întreruperile fac lucruri diferite. De asemenea, în starea de Joc se citește constant din ADC pentru a actualiza căsuța selectată.
- Tabla de joc este reprezentată prin două matrice, una de 9×9 (tabla completă) și una de 3×3 (tabla mare/tabla ce conține rezultatele tablelor mici) și o variabila care indică tabla în care trebuie plasată mutarea.
- Timer-ul folosit pentru PWM stă oprit până când trebuie ca buzzer-ul să scoată un sunet. Atunci, timer-ul este pornit în modul FastPWM, și oprit (registrele sunt resetate) după un număr de overflow-uri.

Multiplayer

- Ambii jucători folosesc aceleași controale. După ce un jucător face o mutare validă, este rândul celuilalt.

Singleplayer

- Modul de singleplayer se bazează pe rularea unui algoritm pentru a determina cea mai bună mutare, luând în calcul diferiți factori.
- Algoritmii folosiți iau în calcul **1-2 ture** în față.
- Cele două dificultăți diferă prin factorii folosiți pentru determinarea mutării. Pentru ambele dificultăți,

dacă sunt mai multe mutări care sunt considerate la fel de bune, se alege prima găsită.

Singleplayer Easy

- Algoritmul folosit pentru dificultatea **Easy** se bazează pe **prioritatea** pe care o are mutarea în tabla curentă în care se află. Acesta verifică prioritatea pentru fiecare posibilă mutare din tabla curentă.
- Dacă următoarea mutare permite plasarea în orice tablă, se rulează algoritmul în tabla mare pentru a determina în ce tabla mică va avea loc mutarea.
- Prioritățile folosite sunt (jucător curent = AI):
  - **0.** Jucătorul curent câștigă tabla curentă
  - **1.** Jucătorul curent blochează adversarul din a câștiga tabla
  - **2.** Dacă tabla este goală, mutare în colț
  - **3.** Dacă jucătorul curent nu a mutat deloc în tabla curentă, iar adversarul a mutat în centru, mutare în colț
  - **3.** Dacă jucătorul curent nu a mutat deloc în tabla curentă, iar adversarul nu a mutat în centru, mutare în colț
  - **4.** Mutare care duce tabla într-o stare din care jucătorul curent poate câștiga cu o mutare
  - **5.** Mutare în colț
  - **6.** Altfel

Singleplayer Hard

- Algoritmul folosit pentru dificultatea **Hard** se bazează pe **prioritatea** pe care o are mutarea la nivelul întregului joc. Din nou, algoritmul calculează această prioritate pentru posibilele mutări din tabla curentă.
- Dacă următoarea mutare permite plasarea în orice tablă, se rulează mai întâi algoritmul folosit de dificultatea Easy în tabla mare pentru a determina în ce tabla mică va avea loc mutarea.
- Prioritățile folosite sunt (jucător curent = AI):
  - **0.** Mutare care face ca jucătorul curent să câștige întregul joc
  - **9.** Mutare care duce inamicul într-o tablă finalizată
  - **10.** Mutare care duce inamicul într-o tablă pe care o poate câștiga, caz în care câștigă întregul joc
  - **6.** Mutare care duce inamicul într-o tablă pe care jucătorul curent o poate câștiga
  - **7.** Mutare care duce inamicul într-o tablă pe care o poate câștiga, iar dacă o câștigă duce jucătorul curent într-o tablă terminată sau pe care o poate câștiga
  - **8.** Mutare care duce inamicul într-o tablă pe care o poate câștiga, iar dacă o câștigă duce jucătorul curent într-o tablă pe care nu o poate câștiga
  - **1.** Mutare care duce inamicul într-o tablă în care nu a mutat deloc
  - **2.** Mutare care duce inamicul într-o tablă în care a mutat o singură dată
  - **3.** Mutare care duce inamicul într-o tablă în care a mutat de minim două ori, dar jucătorul curent a mutat de mai multe ori
  - **4.**Mutare care duce inamicul într-o tablă în care a mutat de minim două ori, dar jucătorul curent a mutat de cel mult atâtea ori cât inamicul
  - **5.** Altfel
- **Ordinea** priorităților nu este întâmplătoare, întrucât este important în ce ordine sunt verificate posibilele priorități. Acest lucru este datorat faptului că necesitățile unor priorități sunt îndeplinite și de altele, care adaugă restricții suplimentare.

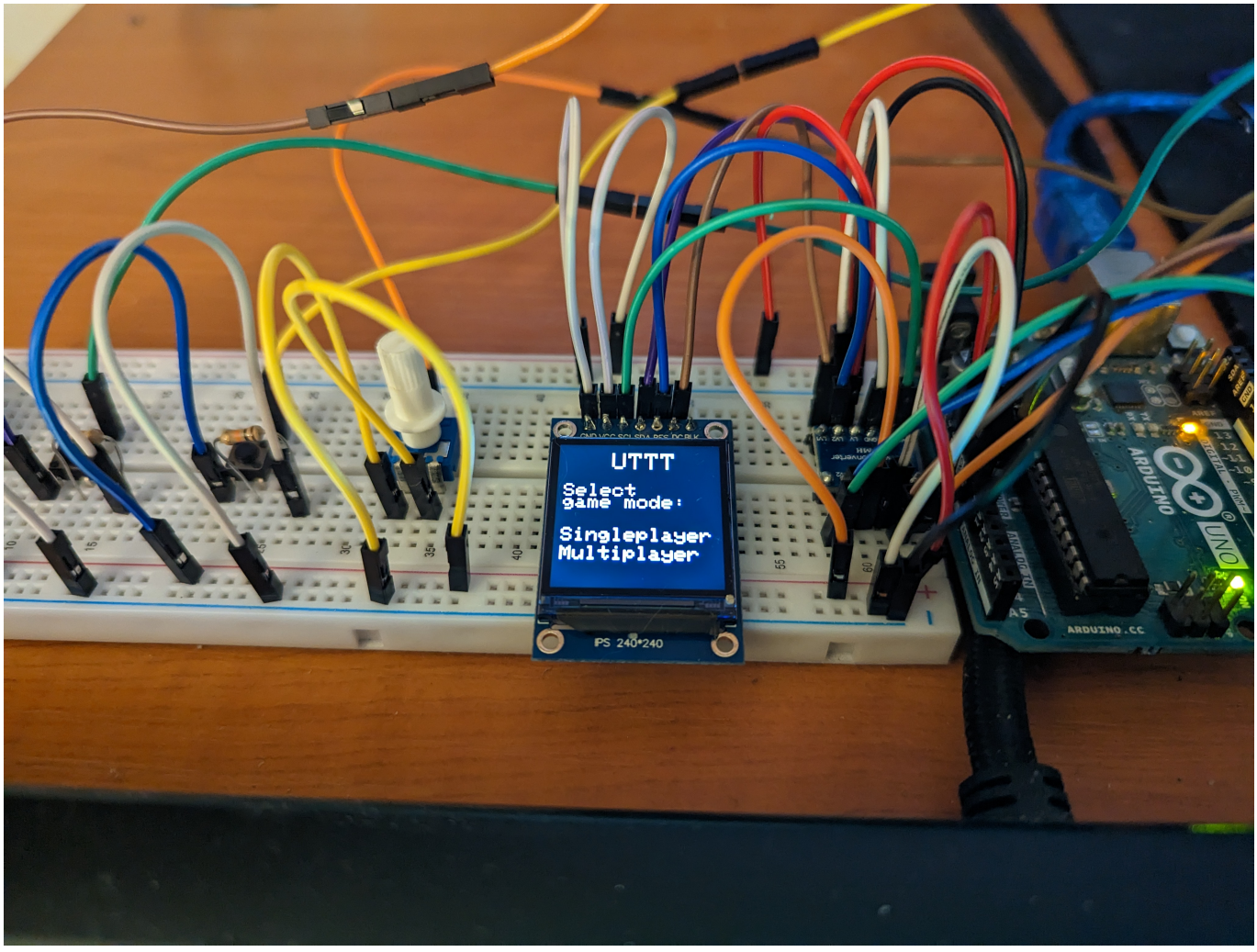
Rezultate obținute

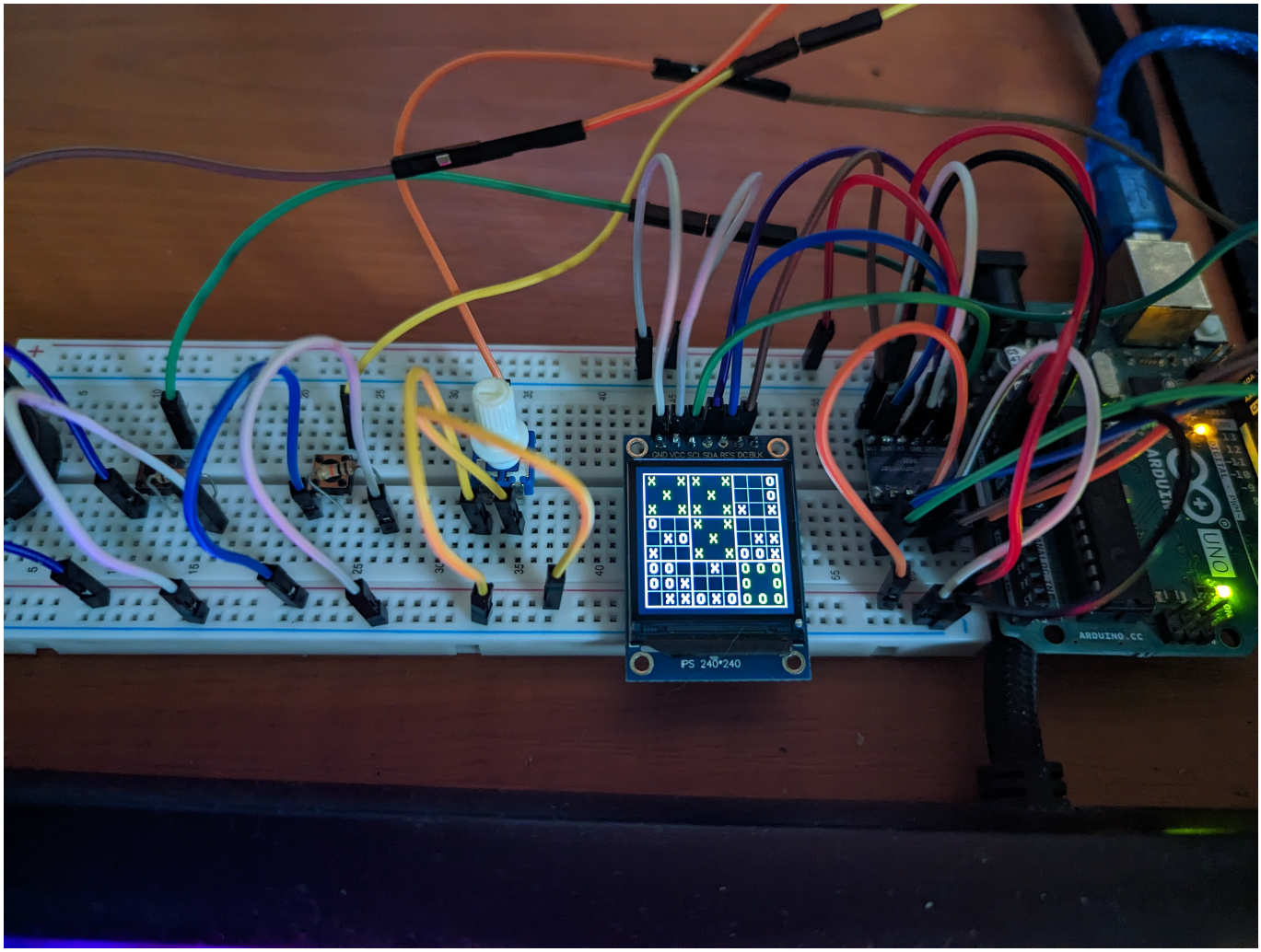
Părere personală

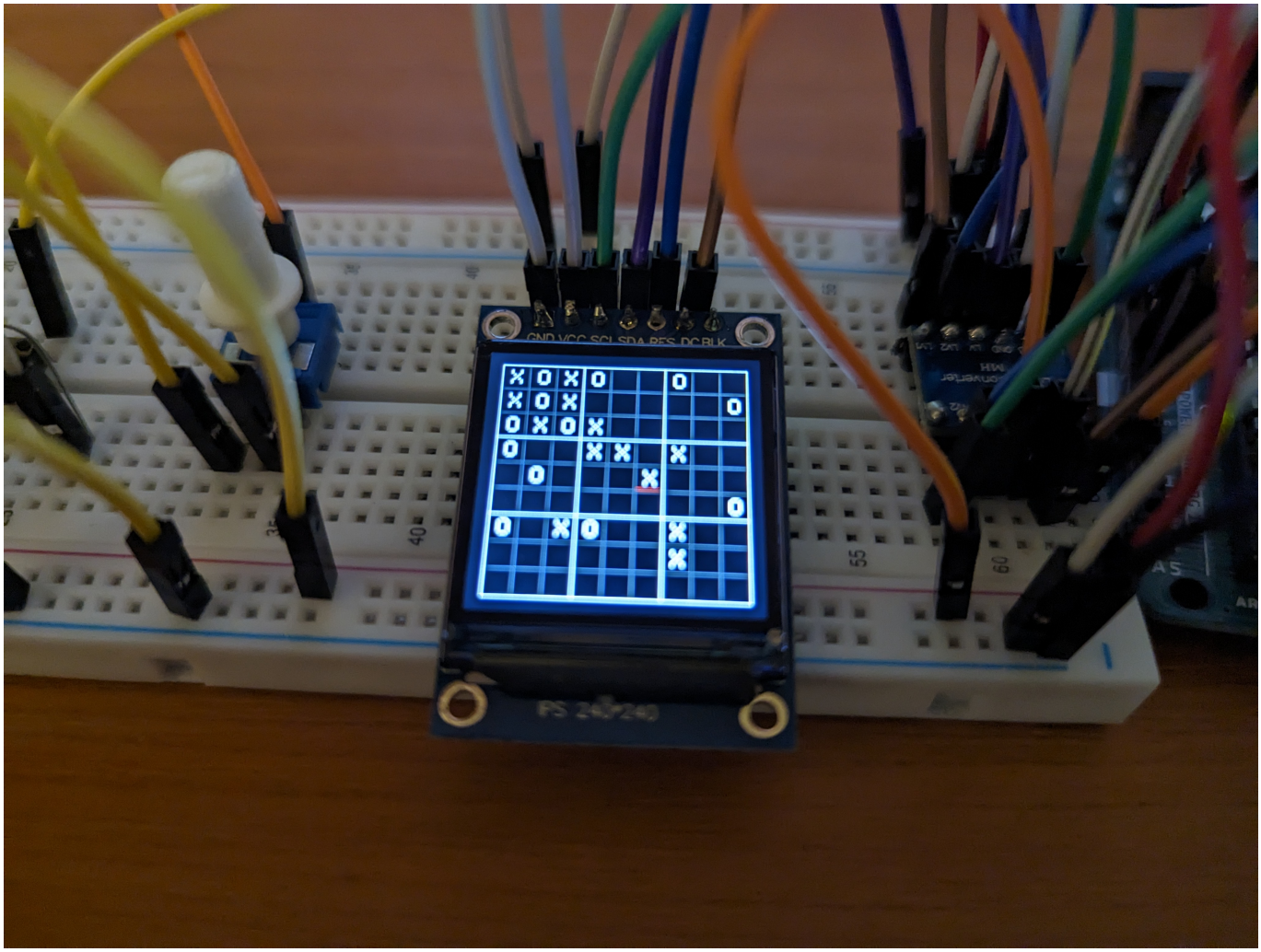
Rezultatul obținut mi-a depășit cu mult așteptările. Acesta fiind **primul proiect** pe care l-am făcut folosind **Arduino**, mă așteptam ca din ce mi-am propus inițial să trebuiască să renunț la unele funcționalități.

Din fericire, nu a fost nevoie, deoarece, deși m-am lovit de diferite dificultăți pe parcursul implementării proiectului, am reușit să rezolv problemele, astfel încat proiectul să funcționeze în modul în care mi-l imaginasem la început.

Imagini proiect




GND VCC SCL SDA RES DC BLK
IPS 240*240
ARDUINO.CC



Concluzii

În primul rand, personal **mă bucur că am avut de făcut acest proiect**, procesul de implementare fiind complet diferit față de o temă obișnuită. Datorită acestuia, acum consider că am văzut măcar o parte din câte lucruri se pot face folosint o simplă placuță Arduino și sper să nu fie ultima dată când interacționez cu una.

În al doilea rând, aș vrea să menționez câteva probleme ale proiectului:

- Deoarece ecranul are nevoie de un timp scurt, dar care nu este neglijabil, pentru a desena ceva, este posibil ca atunci când jocul se termina să se apese un buton, care te duce înapoi în meniu, înainte ca textul “X WINS” sau “0 WINS” să se afișeze, ceea ce duce programul în starea de meniu, deși pe ecran este afișat textul de sfârșit de joc.
- Uneori, o singură apăsare de buton declanșează de două ori întreruperea. Acest lucru poate fi observat ușor încercând o mutare invalidă, fiindcă se va auzi buzzer-ul de două ori.
- Pe dificultatea Hard, la începutul jocului, când tabla este în mare parte goală, o mutare a AI-ului durează aproximativ 10 secunde, ceea ce este mult. Deși nu este o problemă la fel de gravă ca celelalte două, este deranjant pentru utilizator. Timpul poate fi redus prin eficientizarea codului.

Download

Codul sursă, biblioteca Arduino_ST7789_Fast, schemele și imaginile din această documentație pot fi găsite în următoarea arhivă: pm_rosu_mihai_cosmin_333ca_2023.zip.

De asemenea, proiectul poate fi găsit și pe GitHub: Ultimate Tic-Tac-Toe

Bibliografie/Resurse

Resurse documentație

- Tool folosit la realizarea schemei bloc
- Tool folosit la realizarea schemei circuitului

Resurse hardware și software

- Documentație ATMega328P
- Tutorial conectare buton Arduino
- Tutorial folosire ecran ST7789
- Tutorial PWM cu registre
- Biblioteca Arduino_ST7789_Fast
- Sursă inspirație pentru prioritățile folosite în modul Singleplayer

Export to PDF