TicTacToe++

Introducere

TicTacToe++ simuleaza o consola handheld care faciliteaza jucarea jocului X si 0 (TicTacToe). Acesta simuleaza mai multe gamemoduri posibile cu grid variabil. Ofera feedback vizual prin intermediul unui ecran si vizual prin intermediul unui buzzer. Proiectul are incarcare prin baterii pentru putea fi folosit si “handheld”, fara a fi conectat la un calculator. Interactiunea cu jocul se face prin 5 butoane tactile de miscare (WASD) si unul de Select.

Descriere generală

Componentele principale pe care le utilizez în proiect sunt:

Microcontroler ATmega328P (Xplained Mini)

Este unitatea centrala de procesare a consolei, responsabila pentru rularea logicii jocului și coordonarea tuturor perifericelor. Acesta citește semnalele de la butoane, procesează mutarile jucatorilor si transmite datele vizuale către ecranul TFT prin protocolul SPI, precum și semnalele audio catre buzzer.

Display TFT LCD 1.44 inch

Am utilizat un ecran TFT color cu rezoluție de 128×128 pixeli pentru a oferi o interfata grafica moderna jocului TicTacToe++. Acesta comunica cu microcontrolerul prin protocolul SPI. Ecranul este compatibil cu logica de 5V și afișează grila de joc, simbolurile “X” si “0” în culori diferite, cursorul de selecție si animații pentru mesajele de victorie sau remiză.

Butoane Push-Button (Input Control)

Sistemul utilizează un set de 6 butoane tactile configurate pentru un control optim: patru butoane așezate în format “D-Pad” pentru navigarea pe grilă (sus, jos, stânga, dreapta), un buton pentru confirmarea mutării (Select) și un buton pentru funcții suplimentare (Reset). Fiecare buton este configurat folosind rezistențele interne de tip pull-up ale microcontrolerului pentru a asigura citiri digitale stabile și pentru a simplifica circuitul electronic.

Buzzer Pasiv

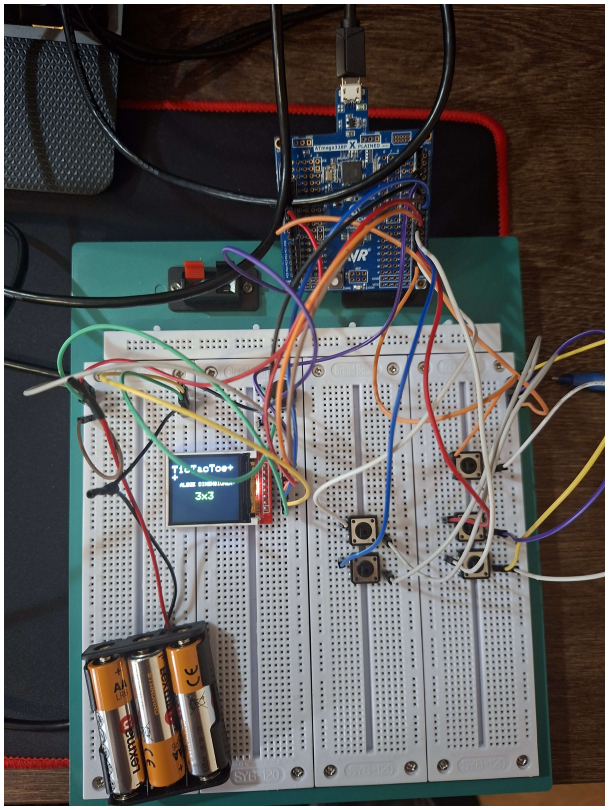
Modulul buzzer pasiv este utilizat pentru a oferi feedback audio în timp real, imbunatatind experiența de utilizare.

Suport de baterii (3xAA)

Asigura alimentarea autonoma a consolei. Doar 3 baterii de aproximativ 1,6 V fiind suficiente.

Hardware Design

Pin Arduino	Port Hardware (Mcu)	Componenta Periferica	Rol / Functie in Joc
D3	PD3	Buton **SELECT**	Confirma selectia in meniu si pune piesa pe tabla
D4	PD4	Buton **W** (Sus)	Navigare in sus pe grila / Schimba dimensiunea in meniu
D5	PD5	Buton **S** (Jos)	Navigare in jos pe grila / Schimba dimensiunea in meniu
D6	PD6	Buton **A** (Stanga)	Navigare la stanga pe grila de joc
D7	PD7	Buton **D** (Dreapta)	Navigare la dreapta pe grila de joc
D8	PB0	Ecran TFT **DC** (Data/Command)	Spune ecranului daca primeste pixeli sau comenzi
D9	PB1	Ecran TFT **RST** (Reset)	Reseteaza ecranul la pornirea sistemului
D10	PB2	Ecran TFT **CS** (Chip Select)	Activeaza/dezactiveaza comunicarea SPI cu ecranul
D11	PB3 (MOSI)	Ecran TFT **SDA** / **MOSI**	Linia principala de date prin care trimitem grafica
D13	PB5 (SCK)	Ecran TFT **SCL** / **SCK**	Semnalul de ceas (Clock) pentru sincronizarea SPI
A5	PC5	**Buzzer Pasiv**	Generare audio low-level pentru efecte si alerte
5V / **VCC**	VCC	Ecran TFT **VCC**	Alimentarea cu energie a ecranului (Unde e conectat si suportul de baterii)
GND	GND	Ecran / Buzzer / Butoane	Masa comuna a intregului circuit electric



Software Design

Mediu de dezvoltare, librarii si surse

Proiectul este scris in **PlatformIO (in VS Code)** cu compilerul standard *avr-gcc*. Singura librarie din exterior pe care am folosit-o este **Adafruit_ST7735** (impreuna cu *Adafruit_GFX*), strict pentru a desena pe ecran.

Restul functiilor (partea de SPI, citirea butoanelor, debouncing-ul, sunetele si timpii) sunt facute de noi de la zero, **fara librarii**, umbland direct in registrii de pe ATmega328P.

Algoritmi si structuri de date

1. Masina de stari (State Machine)

Codul se bazeaza pe o masina de stari simpla (folosind un `enum StareJoc`):

- `MENIU`: Ecranul de start unde alegem dimensiunea tablei (3×3, 4×4, 5×5).
- `IN_JOC`: Jocul propriu-zis, unde miscam cursorul, punem piese si scade timpul.
- `GAME_OVER`: Afiseaza cine a castigat si asteapta sa apasam pe butonul de reset.

2. Reprezentarea tablei si Verificarea Castigului

Tabla este salvata intr-o matrice de intregi `int tabla[5][5]`. Logica este asa: `0` pentru patratel gol, 1 pentru Jucatorul X (Albastru) si 2 pentru Jucatorul O (Portocaliu). Pentru a vedea daca a castigat cineva, parcurgem matricea pe linii, coloane si diagonale cu o „fereastra” (Sliding Window) de 3 elemente vecine. Daca gasim 3 piese la fel, jocul se opreste.

3. Periferice si Temporizare Hardware

- **SPI Hardware Manual:** Ca sa mearga ecranul foarte rapid, am configurat manual modulul SPI din procesor. Setați din registrul DDRB pinii de MOSI (PB3), SCK (PB5) si CS (PB2) ca iesiri. Dupa, am pornit SPI-ul din registrul SPCR pe mod Master si am activat viteza maxima de 8MHz din registrul SPSR (folosind bitul SPI2X). Asa ecranul se deseneaza instant.
- **Sistemul cu doua timere (Timer0 si Timer2):** Ca sa nu folosim functii blocante gen delay() care pun tot codul pe pauza, am impartit treaba pe doua timere interne:
  - **Timer2 (Bara de timp):** L-am pus in mod Normal cu prescaler de 64. Ii dam valoarea `TCNT2 = 6` in functia de intrerupere `ISR(TIMER2_OVF_vect)` si asa primim o intrerupere fixa la fiecare 1 milisecunda. O folosim ca sa scadem timpul din tura (limita de 10 secunde).
  - **Timer0 (Delay-uri pentru sunet si butoane):** Este pus tot pe prescaler de 64 (un pas inseamna 4 microsecunde). Citim direct registrii `TCNT0` si `TIFR0` (prin metoda de *polling* / verificare continua) ca sa facem pauze exacte pentru frecventele de la buzzer si debouncing.

Surse si functii implementate

Control Hardware Low-Level & Audio

- `void initSPIHardware()`: Seteaza pinii de pe portul B si porneste modulul SPI din procesor.
- `void initTimer0Delay()`: Porneste Timer0 ca sa il avem gata pentru functiile de delay.
- `void asteaptaMicrosecunde(unsigned int us)` / `void asteaptaMilsecunde(unsigned int ms)`: Functiile noastre de delay care asteapta dupa timerul hardware (se uita in registrul `TCNT0` si asteapta flag-ul `TOV0`).
- `void pulsHardware(int frecventaMicros, int durataMs)`: Misca rapid pinul PC5 (unde avem buzzerul) modificand registrul `PORTC` ca sa scoata sunete pe diferite frecvente.
- `uint8_t citesteButoaneCuDebounce()`: Citeste pinii din registrul `PIND`, asteapta 5ms si verifica din nou cu un XOR ca sa fie sigur ca apasarea e stabila si nu e doar zgomot (bounce).
- `sunet[Eveniment]()`: Functii simple (ex: `sunetCastig()`, `sunetMiscare()`) care doar apeleaza `pulsHardware` cu valori diferite ca sa sune diferit.

Logica si Randare Vizuala

- `void calculeazaDimensiuni()`: Calculeaza in pixeli cat de mari sa fie patratelele in functie de

modul ales (3×3, 4×4 sau 5×5) ca grila sa stea pe centrul ecranului.
- `void deseneazaMeniu()` / `void deseneazaGameOver()`: Coloreaza ecranele fixe si pune textul la coordonate fixe ca sa nu se incalece literele.
- `void deseneazaGrila()` / `void deseneazaPiesa()` / `void deseneazaCursor()`: Deseneaza efectiv liniile tablei, cercurile/X-urile si patratul de cursor in timpul meciului.
- `void updateTimerGrafic()`: Vede cat timp a trecut din secunda in secunda si redeseneaza bara colorata de sus (verde/rosie). Daca trec cele 10 secunde, schimba rândul automat.

Functionalitati din laborator utilizate

- **SPI:** Configurat manual prin registrii hardware (SPCR, SPSR, SPDR) si pinii de pe Portul B. Trimitem octetii direct in siliciu la viteza maxima de 8MHz, fara functii gata facute.
- **GPIO:** Tot controlul pinilor (butoane, DC/RST ecran, buzzer) se face prin manipularea directa a registrelor de porturi (`DDRB`, `PORTB`, `PORTD`, `PIND`, `DDRC`, `PORTC`).
- **Timere: Timer0** este folosit pentru delay-uri precise prin polling (numara pasi de 4 microsecunde pentru buzzer si debouncing), iar **Timer2** genereaza baza de timp pentru cronometrul jocului.
- **Intreruperi:** Am activat intreruperea de overflow pe **Timer2** (`TIMER2_OVF_vect`) pentru a scadea timpul la fiecare 1ms. Am facut asta ca sa nu avem operatii blocante in loop-ul principal, eliminand complet micro-lag-urile in timpul jocului.

Rezultate Obtinute

In urma testarii pe hardware, proiectul este 100% functional, aspect demonstrat si in clipul video (.mp4) atasat in arhiva.

Performantele obtinute:

- **Grafica fluida:** Ecranul se randeaza instant datorita SPI-ului.
- **Control fara lag:** Cursorul si meniurile raspund imediat la butoane, iar debouncing-ul elimina apasarile duble.
- **Audio stabil:** Buzzerul scoate sunete clare pentru mutari, selectii, victorii sau erori.
- **Timp precis:** Bara de 10 secunde scade uniform prin Timer2, schimband automat jucatorul la expirare fara sa inghete jocul.

Concluzii

In urma multor teste si incercari, proiectul a fost implementat cu succes, respectand toate functionalitatile din planul initial. Pe partea de dezvoltare am intampinat si probleme hardware specifice: am ars controllerul USB al unei placi de test si am reusit sa rup pinii unui buton de pe board (cel configurat initial pentru Start). Totusi in final am reusit sa ajung la ce mi-am propus.

Download

archive.zip

https://github.com/BurtescuEusebiu/TicTacToePM

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

xplainedmini.pdf

st7735.pdf