Tetris

Vlad Andra, 331CC

Introducere

În acest proiect voi implementa un joc de Tetris controlat de un joystick. Jocul va fi afișat pe o matrice de LED-uri 8×8.
Detaliile jocului (scor, highscore, game over etc.) se vor afișa pe un display LCD 20×4.
Buzzer-ul va cânta melodia Tetris. De fiecare dată când se va face un rând în jocul de Tetris, buzzer-ul va întrerupe melodia și va emite un sunet de confirmare, după care rândul va fi eliminat de pe matricea de LED-uri.
Am ales acest proiect nu numai pentru că este un joc retro foarte drăguț, ci și pentru că am putut include multe dintre informațiile învățate la laborator, precum lucrul cu GPIO, SPI, I2C, EEPROM, întreruperi și multe altele. :D

Descriere generală

Jocul funcționează astfel:

- La **pornire**, se afișează o **animație** cu TETRIS pe matricea de LED-uri și un mesaj pe display.
- La fiecare **începere a jocului**, începe să cânte **melodia Tetris** și se deplasează în jos la un interval de câteva secunde **piesa de Tetris** aleasă random. Pe display afișăm **scorul**.
- Putem muta Joystick-ul în stânga/dreapta pentru **mutarea piesei** și în sus pentru **rotirea** ei. Mutăm Joystick-ul în jos ca să forțăm piesa să se ducă direct jos (**force down**), fără a mai aștepta ca să se deplaseze câte un rând la un anumit interval.
- În **momentul în care se creează o linie**, se mărește scorul. Linia dispare din joc, toate piesele deasupra ei se deplasează în jos. Melodia curentă se întrerupe și buzzer-ul **emite un sunet** ce semnalează noul scor. **Scorul se actualizează** pe display.
- Dacă ajungem cu vreo piesă pe rândul de sus al matricii de LED-uri, jocul se încheie. Melodia se oprește și se afișează '**Game Over**'.
- În orice punct al jocului, chiar și după 'Game Over', putem **ține apăsat pe butonul de switch** al Joystick-ului timp de 3 secunde pentru a da **restart**. După acest interval, matricea de LED-uri va avea toate LED-urile aprinse, se va afișa pe display 'Restarting Game...', iar după 5 secunde jocul va reîncepe și melodia va începe să cânte de la capăt.

Pentru înțelegerea interacțiunii dintre joc și jucător, am realizat o schemă simplificată. **Schema bloc** este următoarea:

Hardware Design

Piese utilizate:

- Arduino MEGA 2560
- Modul cu Matrice LED MAX7219 - matrice de LED-uri 8×8
- LCD 2004 cu Backlight Albastru și Interfață I2C
- Modul Joystick Biaxal
- Buzzer pasiv

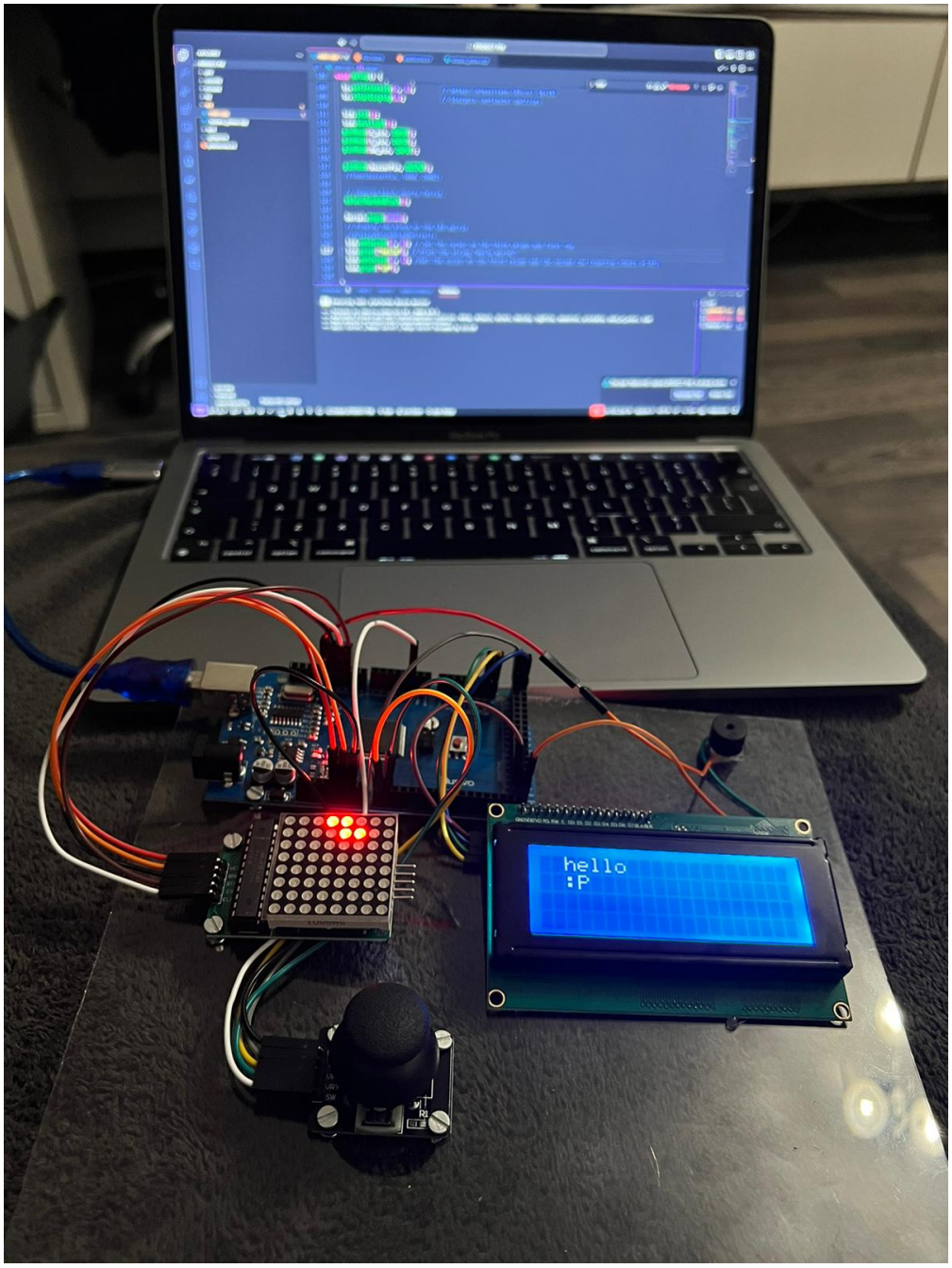
Modulul de comunicație **MAX7219** mi s-a părut interesant, deoarece acesta folosește protocolul SPI pentru comenzi ca și MOSI, fără MISO. Acesta facilitează controlul unei matrice de LED-uri folosind un număr mic de pini. Interesanți sunt pinii DIN și CS:

- **DIN** = Data In, transmite date în mod serial (câte 1 bit) către MAX7219, comenzi precum ce LED-uri să aprindă.
- **CS** = Chip Select, însă nu funcționează chiar ca Slave Select-ul cu care eram obișnuiți. Pinul este folosit să marcheze când ar trebui începute citirea datelor. Este setat la 0 ca să fie începută sesiunea de comunicare (să accepte date pe DIN) și la 1 ca să se încheie comunicarea.

Pinul **DOUT** al modulului este folosit doar dacă dorim să conectăm mai multe matrice de LED-uri între ele.

Pentru a vedea cum conectez pinii analogici și digitali am ales să realizez schematicul în **Fusion**. Majoritatea simbolurilor și footprint-urilor sunt făcute de mine, însă am plecat la câteva piese de la unele deja existente [0]. **Schema electrică** este următoarea:

Inițial, am conectat și testat toate componentele pentru a mă asigura că funcționează corespunzător. Am considerat că ar fi mai potrivit pentru un joc să conectez piesele pe o bucată de plexiglas, pentru a avea tot ce am nevoie într-un mod accesibil pentru utilizator. De asemenea, plănuiesc să conectez proiectul la o baterie, pentru a putea fi ținut în mână de jucător. Pentru a testa faptul că funcționează componentele, am folosit următoarele resurse: [1], [2], [3]. Am folosit bibliotecile LiquidCrystal_I2c și LedControl, însă voi detalia mai mult despre acestea la partea de software.



Software Design

Logica jocului

Mediul de dezvoltare al proiectului este **PlatformIO**.
Stadiul jocului de pe matricea de LED-uri 8×8 este reprezentat de o matrice 12×8 în cod. Motivul pentru care mai avem câteva linii în plus este pentru că o piesă 'cade' progresiv în joc, nu apare din start pe matricea de LED-uri. Matricea conține valori de 0, 1, 2 și 3:

- **0** = **nu am nicio piesă plasată** în joc, LED-ul nu este aprins.
- **1** = parte din **piesa curentă**, care se mișcă treptat în jos. LED-ul este aprins.

- **2** = **centrul piesei curente** (pivot), marcat ca să știm cum să rotim piesa. LED-ul este aprins.
- **3** = parte din **piesele deja plasate** în joc. LED-ul este aprins.

Pentru înțelegerea jocului, am realizat un schelet cu mai puține detalii, pe care îl voi aprofunda în următoarele secțiuni.

```
void loop() {
    // Daca butonul de SWITCH de pe JoyStick e apasat de mai mult de 3
secunde, resetam jocul.
    if (restartGame) {
        // Aprinde LED-urile, reseteaza variabilele...
        delay(5000);
        // Restart game
    }
    if (!gameOver) {
        if (intervalJoystick) {
            // ... Citim input-urile de la Joystick ...
            if (X_val < CENTRU - 200) {
                moveLeft();
            } else if (X_val > CENTRU + 200) {
                moveRight();
            } else if (Y_val < CENTRU - 200) {
                rotateLeft();
            } else if (Y_val > 840) {
                forceTetrominoDown();
                selectRandomTetromino();
            }
        }
        // Daca a trecut un anumit interval de timp, se actualizeaza jocul.
        if (intervalUpdate) {
            // Daca ajugem jos, adaugam scor si trecem la o piesa noua.
            if (pieceSettled) {
                int score = addScore();
                selectRandomTetromino();
            }
            // Altfel, mutam piesa in jos.
            else {
                moveDown();
            }
        }
    }
    else {
        // Opreste melodia, afiseaza 'Game Over', reseteaza variabilele...
    }
}
```

Interacțiunea joc-jucător

- Pentru a genera piese random, am folosit un pin analog pentru a genera un seed astfel încât

jucătorul să nu primească aceleași piese la fiecare joc. Am adăugat și un calcul de **probabilități astfel încât șansele de a primi aceeași piesă** de Tetris de 2-3 ori la rând să fie **reduse**, considerând ca matricea noastră de LED-uri este mică și sunt totuși puține piese ce pot fi generate.

```
const int SEED_PIN = A2;

void setup() {
    randomSeed(analogRead(SEED_PIN) + micros());
}

void selectRandomTetromino() {
    int probabilitate[5] = {100, 100, 100, 100, 100};
    if (pieceId != -1) {
    // Reducem probabilitatea unei piese de a aparea de doua ori
        probabilitate[pieceId] = max(probabilitate[pieceId] - 50, 0);
    }
    // Daca apare o piesa de 2 ori, ii scadem din nou probabilitatea de a
aparea.
    if (freqPiece >= 2) {
        probabilitate[pieceId] = max(probabilitate[pieceId] - 30, 0);
    }

    // .... Calcul probabilitati dupa intervale pe baza greutatilor ...
    int randomPiece = selectedPiece;
    if (pieceId != randomPiece) {
        freqPiece = 0;
    }
    else {
        freqPiece++;
    }
    pieceId = randomPiece;
}
```

- A fost nevoie de calibrarea senzorilor Joystick-ului și adăugarea unor intervale astfel încât interacțiunea cu jucătorul să fie una funcțională. Pe lângă un **threshold** de 200 al axelor, am adăugat verificări suplimentare astfel încât jucătorul să nu poată da force down decât după intervale mai mari de timp și să nu poată roti/da force down până când piesa nu este vizibilă.

Biblioteci Arduino

Pentru implementarea codului, am folosit următoarele biblioteci și resurse:

- LiquidCrystal_I2C pentru setarea display-ului și afișarea textelor. Display-ul meu comunică prin I2C printr-un modul de comunicație care a venit direct conectat la acesta. Pentru a înțelege cum funcționează I2C, am folosit cunoștințele din laboratorul 6. Am folosit codul de la [2] pentru a determina adresa I2C a modulului.

```
// Argumente: adresa I2C a modulului LCD, numărul de coloane și numărul de
rânduri al LCD-ului.
```

```
LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 20, 4);

lcd.init();
lcd.backlight();
lcd.clear();
// setCursor = setează linia și coloana cursorului.
lcd.setCursor(0, 0);
lcd.print("Tetris Game");
lcd.setCursor(0, 1);
lcd.print("is Starting...");
```

- LedControl pentru folosirea matricei de LED-uri cu ajutorul modulului MAX7219. Modulul folosește SPI, despre care am învățat în laboratorul 5 și am detaliat despre funcționalitățile acestuia în partea de hardware.

```
// Pini folosiți de comunicarea SPI și numărul de module MAX7219.
LedControl lc = LedControl(DATA_IN_PIN, CLK_PIN, LOAD_PIN, NUM_MAX7219);

lc.shutdown(0, false);
lc.setIntensity(0, 1);
lc.clearDisplay(0);

for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 8; j++) {
        lc.setLed(0, i, j, gameGrid[i + 4][j] != 0);
    }
}
```

- TimerOne pentru folosirea întreruperilor. Am avut nevoie de întreruperi pentru a putea avea jocul și melodia simultan. Pentru întreruperi, am folosit cunoștințele din laboratorul 2.

```
void setup() {
    Timer1.initialize(100000); // Initializam timerul cu 1 secunda.
    Timer1.attachInterrupt(urmatoareaNotaCantecel);
}

void loop() {
 // .... Restul jocului ...
  int switch_state = digitalRead(SW_pin);
  if (switch_state == LOW) {
    if (buttonPressTime == 0) {
            buttonPressTime = millis();
        }
    // Daca butonul e apasat timp de 3 secunde, restartam jocul.
    else if (millis() - buttonPressTime >= 3000) {
        restartGame();
        buttonPressTime = 0;
    }
  }
 // ....
}
```

```
void restartGame() {
  Timer1.stop();
  // ... Resetare variabile ...
  gameOver = false;
  playSong = true;
  currentNote = 0;
  Timer1.start();
}
```

- MD_Parola pentru folosirea animațiilor de LED-uri pentru MAX7219. Aceasta folosește de asemenea bibliotecile SPI.h și MD_MAX72xx.h. Pentru utilizare, am folosit urmatorul clip: [4].

```
MD_Parola lc_animatie = MD_Parola(HARDWARE_TYPE, DATA_IN_PIN, CLK_PIN,
LOAD_PIN, NUM_MAX7219);

void setup() {
    lc_animatie.begin();
    // Setam brightness-ul.
    lc_animatie.setIntensity(0);
    // Aliniem textul la centru, cu speed time 100 si pause time 1000, cu
animatia de scroll left.
    lc_animatie.displayText("TETRIS", PA_CENTER, 100, 1000, PA_SCROLL_LEFT,
PA_SCROLL_LEFT);
    while (!lc_animatie.displayAnimate()) {
        // Asteptam sa se termine animatia.
    }
}
```

- EEPROM este biblioteca folosită pentru a scrie în memoria plăcuței și a stoca datele chiar după ce aceasta a fost închisă. ATMega2560 are 4KB de memorie EEPROM și am învățat cum să o folosim în laboratorul 0. Eu o folosesc pentru a reține highscore-ul.

```
int address1 = 0;

void setup() {
    // Extragem valoarea highscore-ului din jocuri anterioare din memoria
EEPROM.
    EEPROM.get(address1, retrievedValue1);
}

int addScore() {
    // ...
    if (new_score > highscore) {
        highscore = new_score;
        EEPROM.put(address1, highscore);
    }
    return new_score;
}
```

- Pentru **cântecul de Tetris**, am folosit următorul cod de pe GitHub: [5].

Optimizări

- Pentru a nu consuma bateria, după finalizarea jocului LED-urile și LCD-ul se închid, însă dacă activăm opțiunea de Restart ele se vor reaprinde și jocul va reveni la normal.
- Pentru a nu a actualiza continuu LCD-ul în fiecare loop, avem variabila **displayGameOver** care se asigură că mesajul este afișat doar prima oară când intrăm în buclă cu Game Over, dar și **lastScore** și **currentScore** care se asigură că actualizăm scorul pe display doar atunci când scorul diferă între loop-uri.
- Reducerea apelurilor multiple la **millis()**
- Modularizarea codului în funcții ce au funcționalități repetate în verificările noastre pentru mutarea Tetromino-urilor.

Rezultate Obţinute

Concluzii

Deși nu a fost unul foarte complex din punct de vedere al implementării hardware, consider că proiectul m-a ajutat, deoarece am reușit să interacționez cu multe dintre cunoștințele despre care am învățat la laborator și acum știu să lucrez cu ele atât la nivel de regiștri, cât și la nivelul bibliotecilor disponibile pentru lucrul cu Arduino. Mi-aș fi dorit să implementez proiectul pentru mai multe matrice de LED-uri legate între ele, deoarece nu mi-am dat seama la început că e un spațiu atât de mic pentru Tetris. Cu toate acestea, consider că a ieșit bine și mă bucur că am realizat acest proiect și că acum am o jucărie nouă.

Download

proiect_pm_andra_vlad.zip
schematic_pm_andra_vlad.zip

Jurnal

25.04.2024 - descrierea proiectului
03.05.2024 - adăugarea schemei bloc și a componentelor hardware
12.05.2024 - începere realizare hardware
15.05.2024 - realizarea schemei electrice și descrierea stadiului hardware
20.05.2024 - corectare schematic și adăugare updates hardware

23.05.2024 - adăugare implementare software
24.05.2024 - finalizare implementare software și adăugarea fișierelor
25.05.2024 - adăugare demo

Bibliografie/Resurse

[0] Simbol și footprint MAX7219:
https://www.snapeda.com/parts/MAX7219/Analog%20Devices/view-part/
[1] Interfacing Buzzer to Arduino: https://www.instructables.com/Interfacing-Buzzer-to-Arduino/
[2] Arduino - LCD I2C: https://arduinogetstarted.com/tutorials/arduino-lcd-i2c
[3] How to control 8×8 dot Matrix with MAX7219 and Arduino: https://youtu.be/SGjQ-E3UD7A
[4] MD Parola library for the Max7219 & Arduino: https://www.youtube.com/watch?v=_H2v8uDgqps
[5] Tetris Song On Arduino: https://github.com/robsoncouto/arduino-songs/blob/master/tetris/tetris.ino

Export to PDF