Sudoku

Calciu Alexandru, 331CC

Introducere

Voi implementa faimosul joc Sudoku pe un ecran LCD. Acest joc consta in completarea unei matrici 9×9 cu cifre de la 1 la 9 astfel incat sa se respecte urmatoarele constrangeri:

- Fiecare rand trebuie sa aiba toate cifrele de la 1 la 9
- Fiecare coloana trebuie sa aiba toate cifrele de la 1 la 9
- Fiecare matrice 3×3 in care se imparte matricea mare trebuie sa aiba toate cifrele de la 1 la 9

Scopul acestui proiect este acela de gasi un mod de a ma juca Sudoku la cursuri fara reclame.

Descriere generală

Tabla va fi reprezentata pe ecranul LCD de 3.5 inch in mod natural. Navigarea pe tabla se va realiza folosind 5 butoane pentru directiile sus-jos, stanga-dreapta si pentru selectarea cifrei la pozitia curenta. Atunci cand cursorul este pe o anumita cifra se va putea folosi butonul din centru pentru a itera prin cifre. Atunci cand se va muta jucatorul de pe patratica la care a selectat cifra, se va considera ca a completat acea casuta. Atunci cand se va completa o cifra gresit se va emite o avertizare sonora folosind buzzer-ul. Acestea vor fi asezate in cruce pentru a fi cat mai intuitiv. Tabelele de sudoku vor fi selectate aleator dintr-o colectia preincarcata pe placuta.

De asemenea va exista o limita de timp pentru fiecare joc de sudoku, jucatorul putand observa cat timp mai are pe un display LED separat.

Schema bloc este urmatoarea:

Hardware Design

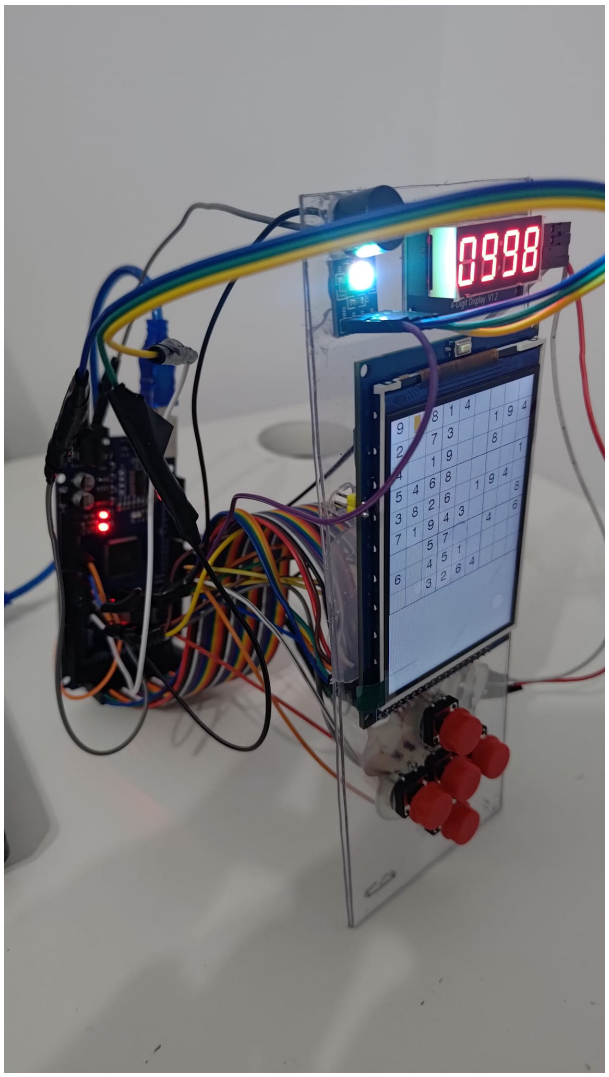
Lista de piese:

- Arduino Mega
- Ecran LCD 3.5 inch cu controller-ul ILI9486: jocul efectiv va fi pe ecran
- Modul cu LED RGB: ledul va fi rosu la inceput si va deveni progresiv mai verde pe masura ce tabela

e completata
- Modul cu Buzzer activ: avertizeaza jucatorul ca a pus o cifra gresita
- Modul display LED TM1637: va fi folosit pentru timer
- Butoane: navigarea pe tabela + iterat prin cifre
- Breadboard: testat chestii
- Fire: era prea simplu fara

Stadiul actual

- Componenetele merg individual.
- Componentele au fost lipite pe plexiglas pentru a face proiectul sa semene cu o mini-consola de jocuri.
- Pe viitor e probabil sa mai schimb din pinii de la componente, deci schema de acum va mai suferi modificari. Codul pe care l-am scris pana acum e in mare parte independent de pini(cu exceptia ecranului unde nu vor exista schimbari).
- Dupa cum se poate observa in poza de mai jos, totul merge si viata e buna.



Schema hardware

In realitate in loc de acel sir lung de pini sunt doua siruri unul peste celalalt dar am preferat sa nu fac asa in Fusion.

Alegerea pinilor

- Ecranul e facut special sa fie pus peste placuta, deci era un singur mod logic de a-l conecta, dupa cum se va vedea si in poze.
- In stadiul asta buzzer-ul si LED-ul RGB sunt pe pini analogici, ii voi conecta ulterior la pini digitali pe care voi face PWM.
- Butoanele sunt conectate fiecare la un pin digital din moment ce am suficienti pini digitali pe placa pentru asta
- Display-ul cu 7 segmente e conectat la pini digitali pentru ca comunica prin I2C
- Pe ecran exista un pin care se conecteaza la pinul de reset de pe placuta, am ales sa nu il conectez pentru ca nu vreau ca utilizatorul sa poata dea reset.

Software Design

Mediu de dezvoltare

VSCode cu extensia PlatformIO

Biblioteci folosite:

- Pentru display-ul cu 7 segmente
- Pentru LCD

Surse:

- Repository

Cod

E foarte mult de explicat. Sunt vreo 1000 de linii de cod, se mai putea reduce din ele dar am preferat sa ma concentrez sa “make it work” si nu mi-am mai lasat timp ca sa “make it pretty”.

La baza programul este un state machine, iar prin apasarea butoanelor se poate schimba starea sistemului.

O sa descriu design-ul software pe componente logice:

- Tabla de sudoku: fiecare tabla de sudoku este un array de 81 de elemente de tipul `struct board_element`. Aceasta structura este un bitfield de dimensiune de 1 byte in care am retinut toate informatiile relevante pentru o celula dintr-un joc(numarul din celula, daca e setata de utilizator sau nu, daca e gresita sau nu si daca cursorul este pe ea).
- Jocurile de sudoku: Aici probabil a fost cea mai mare/amuzanta greseala de design. Mi-am zis sa nu ma complic cu un card SD si sa fac ca jocurile de sudoku sa fie incluse in program. De asemenea m-am gandit sa salvez fiecare sudoku impreuna cu rezolvarea lui in loc sa fac o functie care sa verifice daca o mutare e valida. Pe langa lene, un alt motiv pentru aceasta decizie a fost faptul ca am vrut ca o greseala sa fie sesizata imediat, nu somewhere down the line(presupunem ca utilizatorul a introdus o cifra gresit dar in momentul in care a introdus-o aceasta nu incalca regulile sudoku dar mai tarziu il forteaza pe utilizator sa ramana fara o mutare valida). Am gasit pe github un script in javascript care genera sudoku-uri cu tot cu rezolvari si m-am bucurat. Am folosit script-ul

ca sa exportez sudoku-urile in format JSON si apoi am facut un script pentru a le converti in array-uri de structuri si m-am folosit de preprocesor pentru a aduce suita de jocuri in forma finala. Initial erau 600 de jocuri, acum sunt 30. Ce nu am luat in calcul este cantitatea foarte mica de RAM de pe placuta si ca jocurile acelea trebuiau si ele sa fie memorate undeva. Asa s-a ajuns la un usage al RAM-ului de 360% inainte sa reduc din numarul de jocuri.

- Butoane: Am folosit intreruperi externe pentru butoane si am incercat sa fac codul din intreruperi cat mai scurt. Practic tot codul relevant din intreruperea pentru un buton(in afara de debouncing) este setarea unui flag intr-o variabile definita de mine(`static volatile int button_flags_register = 0;`). In loop este apelata mereu functia `handle_buttons` care face polling constant pe variabila aceasta. In caz ca flagul aferent butonului este setat, se executa actiunea specifica lui, care depinde si de starea sistemului.
- Buzzer: A fost ultima chestie implementata. Sa fac sa tiuie atunci cand cifra introdusa a fost usor(se poate observa in main.cpp, in board.cpp si in buzzer.cpp). Ce nu a fost usor a fost sa fac cantecele pentru succes si esec. Stiam ca exista biblioteci cu cantece preincarcate dar am decis ca ar fi mai fun sa imi pun propriile cantece pe buzzer. Cantecele pe care le-am ales sunt:succes si esec. Am facut conversia din note muzicale in frecvente si durate si a fost fun overall.
- Led RGB: Am folosit mostly codul din lab pentru a face PWM-ul manual(fara biblioteci/functii predefinite). Singura diferenta majora e ca l-am pus pe non-inverting mode.
- Timer: Am folosit un calculator online pentru prescalar si valoarea lui OCR1A pentru a face timerul sa ruleze la 1 secunda, comparand in acelasi timp cu codul din laborator. In privinta display-ului cu 7 segmente, codul la el este straightforward, in spate este practic un contor de secunde care se decrementeaza in intrerupere si in `handle_timer` se verifica daca valoarea lui s-a modificat de la ultimul apel, caz in care se updateaza display-ul efectiv. Am ales aceasta abordare pentru eficienta, pentru a-mi mentine intreruperea light si pentru a nu updata display-ul prea des.
- LCD: Tabla e facuta folosind apeluri de biblioteca, foarte straightforward, nu foarte multe de explicat. Ce a fost mai interesant a fost ca eu la fiecare modificare(apasare de buton), dadeam clear la ecran si desenam tabla din nou de la 0 cu modificarea noua. Aceasta abordare foarte EGC-like avea foarte mult sens in capul meu dar ecranul nu avea un refresh rate suficient de bun pentru asa ceva(sau nu se transmiteau datele suficient de repede pentru a face tranzitia sa fie seamless). Asta m-a fortat sa schimb abordarea si sa gestionez logica de update prin overwrite-uri(redesenez doar cifra curenta si cea anterioara, acoperind practic ce era inainte pentru a realiza update-ul). Am obtinut rezultate mai bune asa fara prea multa restructurare a codului, ceea ce mi s-a parut nice.
- Starile: Sunt 4 stari, cea de meniu, cea pentru jocul efectiv si cate una pentru succes si esec. Tranzitiile sunt foarte straightforward si logice.

Rezultate Obţinute

Demo:

Concluzii

A fost surprinzator de fun? Exceptand partea de hardware, de aceea nu o sa imi fie dor. Software-ul a

fost fun de facut, bibliotecile pentru ecrane sunt foarte usor de folosit si foarte puternice. Mi s-a parut interesant sa lucrez cu un hardware destul de limitat si sa imi adaptez codul pentru conditiile acestea. Cable management-ul putea fi mai bun dar nu ma pricep suficient la DIY ca sa fac ceva in privinta asta si sincer mi-a fost si frica sa nu stric ceva.

Download

Link repository

Jurnal

- 07.05 - O versiune initiala de cod pentru randarea placii si interactiunea cu ea(doar ecranul era conectat la placa)
- 12.05 - Lipit piese, facut design-ul hardware
- 18.05 - Facut logica pentru timer
- 24.05 - Facut meniul, inceput implementarea state machine-ului
- 25.05 - Finalizat logica pentru joc si facut pwm pentru modulul LED
- 26.05 - Adaugat sunete/cantece prin intermediul buzzer-ului

Bibliografie/Resurse

Resurse Hardware

- Datasheet Arduino Mega
- Pin layout Arduino Mega
- Optimus Digital 5 stelute

Resurse Software

- Datasheet Arduino Mega
- Poza asta si Songsterr pentru facut cantecele cu buzzer-ul in cel mai inutil mod posibil.
- Acest script pentru generarea jocurilor de sudoku
- Bibliotecile mentionate anterior: LCD si display cu 7 segmente
- Thread-uri de stack overflow si arduino forum pe care nu le am salvate
- Un calculator pentru prescalar la timer pe care nu l-am salvat

Export to PDF