Memory Game

Introducere

Jocul în care poți testa cine are cea mai bună memorie. Acesta constă în reținerea ordinii în care s-au aprins LED-urile și introducerea acesteia pe keypad. Vor fi 2 moduri de joc:

- șirul existent devine din ce în ce mai lung, până când jucătorul greșește
- dupa fiecare secvență corectă dată de către jucător se dă un șir nou de lungime crescută

Folosind aceste 2 moduri se obțin dificultăți diferite, dar și încă o șansă pentru a demonstra cine are memoria cea mai bună. Scorul maxim pentru modul curent va fi mereu vizibil pentru jucători, pe un ecran.

Oricând e bună puțină competiție!

Descriere generală

Schema bloc:

Jocul de memorie se va desfășura astfel:

- Se vor aprinde o succesiune de LED-uri (9 LED-uri diferite)
- Jucătorul va trebui să apese pe keypad pe cifrele de la 1 la 9 in ordinea corectă, în funcție de LED-urile care au fost aprinse (0 pornește jocul)
- Un buton care va schimba dificultatea jocului
- Ecran LCD pe care va fi afișat scorul maxim și scorul curent al jucătorului
- Buzzer pentru a oferi feedback jucătorului pentru tastele corecte/greșite apăsate
- Baterie de 9V pentru a funcționa fără ajutorul unui laptop

Hardware Design

Componente utilizate:

- Arduino Uno (AtMega328P-AU R3 CH340G)
- Tastatura matriceala 4×4 Keypad
- 9 LED-uri
- Ecran LCD1602 cu modul I2C/IIC
- Modul buzzer pasiv, KY-006

- Buton
- Conector baterii AA

Schema electrica: schema_electrica.pdf

BOM:

Componenta	Cantitate	Locul achizitionarii	Datasheet
Arduino Uno R3	1	Arduino	Datasheet
LCD 16×2 (I2C)	1	Ecran LCD	
Keypad 4×4	1	Keypad	Datasheet
Buton	1	Buton	
Buzzer	1	Buzzer	
Shift Register	2	74HC595	Datasheet
LED	9	LED	
Rezistenta 220Ω	9	Rezistenta	
Suport baterii	1	Suport baterii	

Descrierea componentelor:

- Arduino Uno R3

Controler principal, responsabil cu logica jocului, gestionarea LED-urilor, afisajul si input-ul.

- Keypad 4×4

Folosita de catre jucator pentru introducerea secventei de LED-uri retinute si pentru pornirea jocului;
Pini utilizati: D2-D9, conform datasheet-ului.

- Ecran LCD 16×2 I2C

Afiseaza dificultatea curenta si scorul maxim facut pentru acea dificultate;
Pini utilizati: VCC - 5V, GND - GND, SDA - A4, SCL - A5 (pinii A4 si A5 sunt pinii SDA, respectiv SCL ai placutei).

- LED-uri

Se vor aprinde intr-o secventa generata aleator pe care jucatorul va trebui sa o retina si sa o introduca pe tastele de la 1 la 9 de pe Keypad;
Pini utilizati: folosesc doar 3 pini, nu 9, datorita celor 2 shift registers.

- Shift Register (74HC595)

Permit utilizarea a mai putini pini de pe placuta Arduino, si permit o eventuala scalare pana la 16 LED-uri;
Pini utilizati: D11, D12, D13 (pinii digitali ramasi).

- Buzzer pasiv

Folosit pentru oferirea unui feedback sonor jucatorului atunci cand greseste un LED din secventa sau cand apasa tastele in ordinea corecta pentru intreaga secventa;

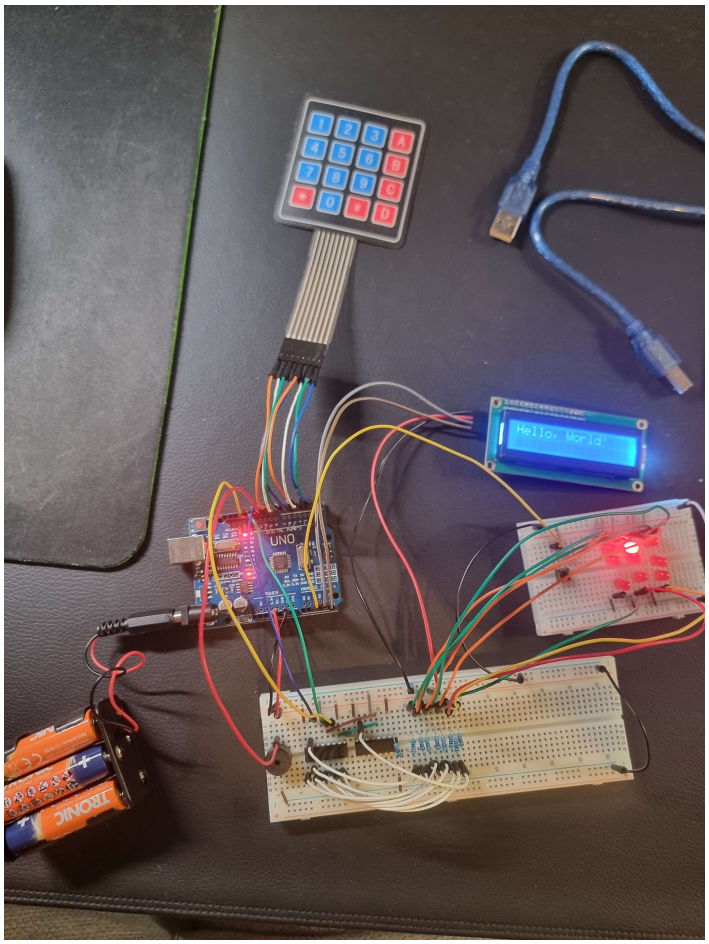
Pini utilizati: GND - GND, D10.

- Buton

Va schimba dificultatea;
Pini utilizati: A3 (este pin analogic dar poate fi folosit si ca pin digital).

- Alimentare

Suport de 6 baterii AA, conectat folosind jack-ul de pe placuta Arduino.



Software Design

Mediu de dezvoltare

Pentru implementarea software am folosit Visual Studio Code cu extensia PlatformIO.

Librarii

- **Wire.h** - comunicarea dintre Arduino si dispozitivele I2C
- **LiquidCrystal_I2C** - pentru usurarea folosirii ecranului LCD 16×2 (ex functii: init, backlight, setCursor, print)
- **Keypad.h** - gestioneaza input-ul utilizatorului (ex functii: getKey, waitForKey)

Elementul de noutate

Jocurile de memorie sunt de obicei formate din 4 LED-uri de culori diferite. Jocul pe care l-am realizat

eu contine 9 LED-uri de aceeasi culoare pozitionate intr-un patrat 3×3, punand accentul pe retinearea pozitiei, nu a culorii.

Functionalitati din laborator

Functionalitatile din laborator pe care le-am folosit sunt: **GPIO**, **Intreruperi**, **I2C**;

Scheletul proiectului

Proiectul este realizat in fisierul main.cpp si este structurat astfel:

- **setup()** - contine toate initializarile de inceput;
- **loop()**
  - daca programul se afla in starea de joc, se vor aprinde LED-urile in secventa generata random, iar apoi se va astepta input-ul jucatorului. Daca este corect se va adauga un LED la secventa si se va continua jocul, altfel se va reseta secventa pentru urmatorul jucator, fiecare situatie avand un feedback audio corespunzator
  - se verifica daca butonul este apasat, pentru a schimba dificultatea (Easy - Hard)
  - **Easy** - cand jucatorul introduce corect secventa de LED-uri lungimea secventei creste si se adauga inca un LED random
  - **Hard** - cand jucatorul introduce corect secventa de LED-uri lungimea secventei creste si se genereaza o secventa noua de lungime crescuta
  - pentru generarea random a secventei am folosit in setup() functia **'randomSeed(analogRead(A0)'**, pentru a genera un seed random de fiecare data cand se alimenteaza jocul
  - se asteapta input-ul jucatorului pentru a incepe jocul (0 pe Keypad)
  - la finalul loop-ului se afiseaza animatia de Idle (cand nu este niciun joc activ)
- **sendToShiftRegister()** - trimite informatii catre shift register (74hc595) pentru a aprinde un anumit LED;
- functii ajutatoare pentru afisarea pe ecranul LCD (printIdleMessage, printWrongMessage etc.)

Calibrari

- Buton - debouncer
- Animatia idle - evitarea folositii functiei 'delay()' si calcularea timpului trecut de la ultima schimbare a LED-urilor, pentru a lasa jucatorul sa apese pe taste in timp ce se face animatia.

Optimizari

- Pentru a evita nevoia de a conecta jocul la curent/baterie continuu, am adaugat un buton pe care

utilizatorul poate apasa pentru a porni sistemele (LCD, LED-uri, feedback audio).

Animatie idle

```
if (idleState) {
  unsigned long currentTime = millis();
  if (currentTime - lastLedUpdateTime >= ledInterval) {
    uint16_t data = 1 << ledIndex;
    sendToShiftRegister(data);
    ledIndex = (ledIndex + 1) % 9;
    lastLedUpdateTime = currentTime;
  }
}
```

Folosind aceasta logica pentru animatia de idle, ci nu apelusi de delay(), am putut evita blocarea celorlalte dispozitive, tinand cont de resetarea animatiei in cazul in care jucatorul opreste si porneste jocul sau in cazul in care pierde si este intors in starea de idle.

Rezultate Obţinute

Am obtinut un joc de memorie distractiv, dar si competitiv. Este usor de jucat, intuitiv si functioneaza corect, comparativ cu planurile realizate la inceput. Cele doua dificultati sunt diferite si permit testarea a 2 tipuri de memorie (Easy - repetitia LED-urilor din secventele anterioare in toate cele care vor urma; Hard - memoria vizuala la prima vedere prin schimbarea totala a secventei de LED-uri). In plus, circuitul este montat intr-o cutie pentru a putea fi folosit cu usurinta si prin folosirea bateriilor poate fi jucat oriunde fara sa fie conectat la un laptop/PC.

Concluzii

Realizarea proiectului a fost o experienta interesanta, care incepea sa imi placa din ce in ce mai mult cu cat avansam in dezvoltarea acestuia, de la alegerea componentelor hardware, pana la introducerea lor intr-o cutie pentru a arata mai bine.

Bibliografie/Resurse

Resurse

- Github: https://github.com/Kris131/Arduino-Memory-Game

Bibliografie

- https://docs.arduino.cc/tutorials/communication/guide-to-shift-out/#shftout12
- https://docs.arduino.cc/learn/electronics/lcd-displays/
- https://projecthub.arduino.cc/SURYATEJA/use-a-buzzer-module-piezo-speaker-using-arduino-uno-cf4191

Export to PDF