Do you know..?

Introducere

„Do you know..?” este un joc de tip trivia, multiplayer! Jucatorii trebuie sa raspunda la intrebari de cultura generala, cat mai repede posibil! Primul care apasa butonul sau si raspunde corect este mai aproape de castig!

Ideea proiectului provine din incercarea realizarii unui produs care sa anime si sa destinda utilizatorii! Elementele aprofundate in cadrul laboratoarelor m-au ajutat enorm sa adaug caracteristici interesante jocului (precum intreruperile, in cazul determinarii celui care a apasat primul butonul).

Descriere generală

Schema bloc:

Module:

- **ecran LCD** - are ca scop afisarea intrebarilor, a variantelor de raspuns si a scorului;
- **buzzer** - emite un sunet de tip **„tic-tac”** pe durata rundei, iar la final reda un sunet diferit care indica daca raspunsul jucatorului a fost corect sau gresit; de asemenea, un alt sunet distinct semnalizeaza terminarea timpului in cazul in care nici un jucator nu stie sa raspunda;
- **leds** - indica care dintre cei patru jucatori a apasat primul butonul PRESS si, de asemenea, semnalizeaza daca raspunsul jucatorului a fost corect (verde) sau gresit (rosu);
- **butoane** - sunt de doua tipuri: *PRESS*, care reprezinta butonul principal pe care jucatorul trebuie sa il apese cat mai rapid pentru a putea raspunde, si butoanele *A / B / C*, folosite pentru a alege varianta corecta.

Flow:

O sesiune de joc se realizeaza in felul urmator:

1. **Intro** - pe ecran apare mesajul *“Ready<3”* si pentru a incepe un jucator (oricare) trebuie sa apase

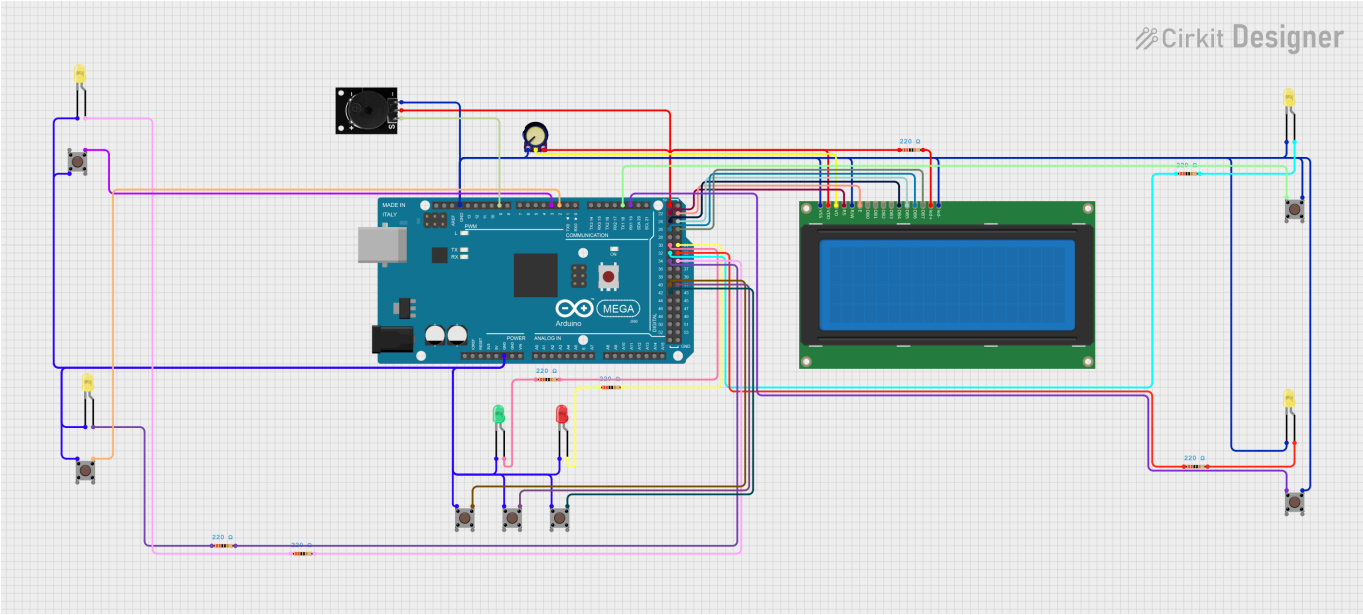
butonul *PRESS*;
2. **Intrebare** - intrebarile apar pe ecran pe rand si jucatorii trebuie sa apase cat mai repede butonul PRESS pentru a putea raspunde; o intrebare va sta pe ecran doar 30 de secunde, iar buzzerul, prin sunetele “tic-tac”, care devin din ce in ce mai rapide, ii face pe jucatori constienti de scurgerea timpului;
3. **Raspuns** - dupa ce cel putin un jucator apasa PRESS, led-ul jucatorului care a apasat primul se va aprinde; dupa aceasta el are ocazia sa raspunda, iar led-urile verde si rosu ii vor oferi feedback, acompaniate de buzzer;
4. **Scor final** - dupa ce jucatorii raspund la 10 intrebari, pe ecran va aparea scorul final al tuturor; folosind butoanele PRESS, jucatorii pot reincepe jocul;

Hardware Design

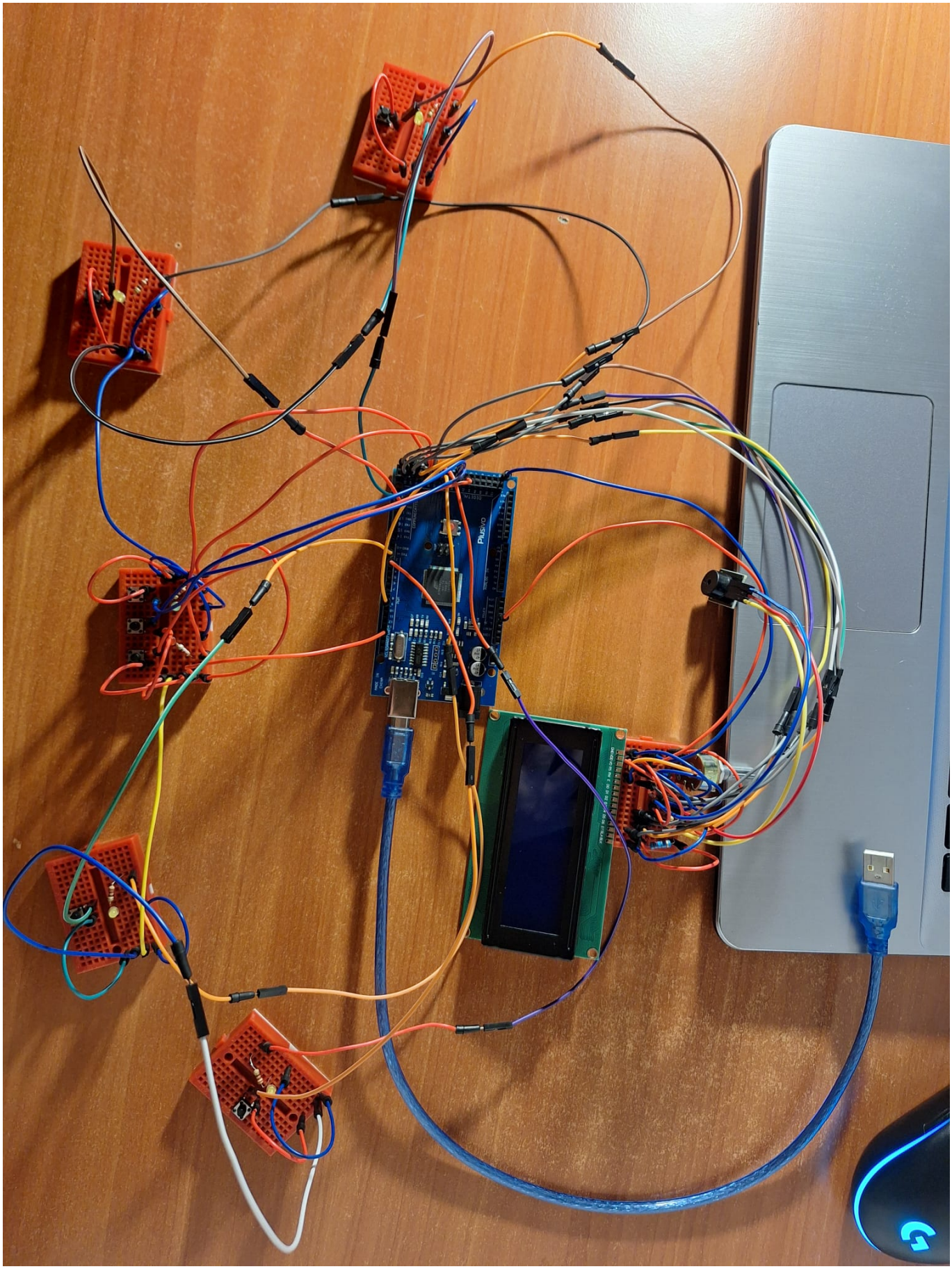
Componente:

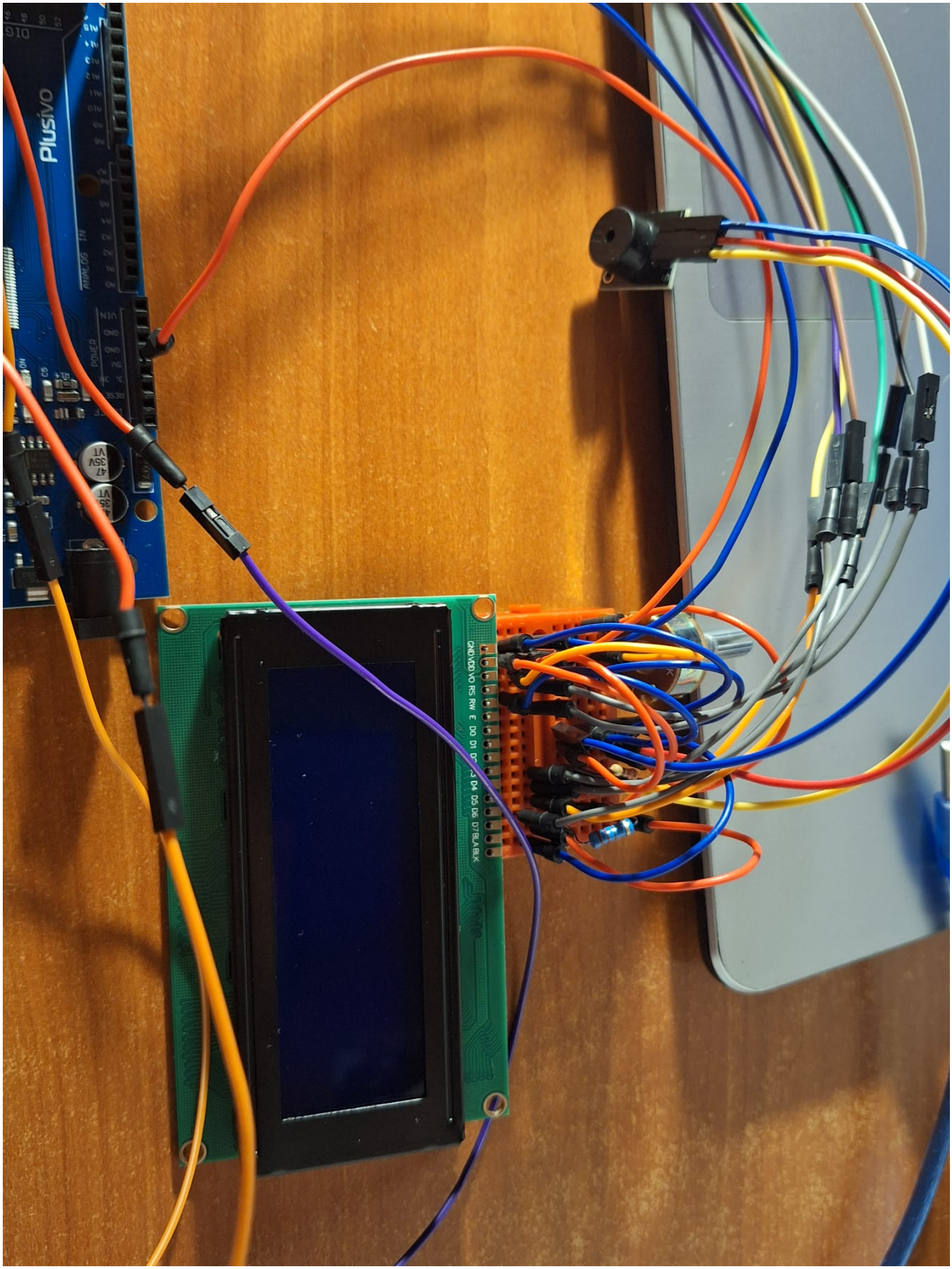
Nume	Cantitate	Link
Arduino Mega 2560	1	link
LCD 2004	1	link
Buzzer pasiv	1	link
Led-uri galbene	4	link
Led rosu	1	link
Led verde	1	link
Butoane 6x6x6	7	link
Rezistor 0.5W 220Ω	1	link
Rezistor 0.25W 220Ω	6	link
Breadboard cu 170 de puncte	6	link
Potentiometru stereo 10k	1	link
Fire mama-mama, mama-tata, tata-tata	numar variat	
Fire jumper	set variat	

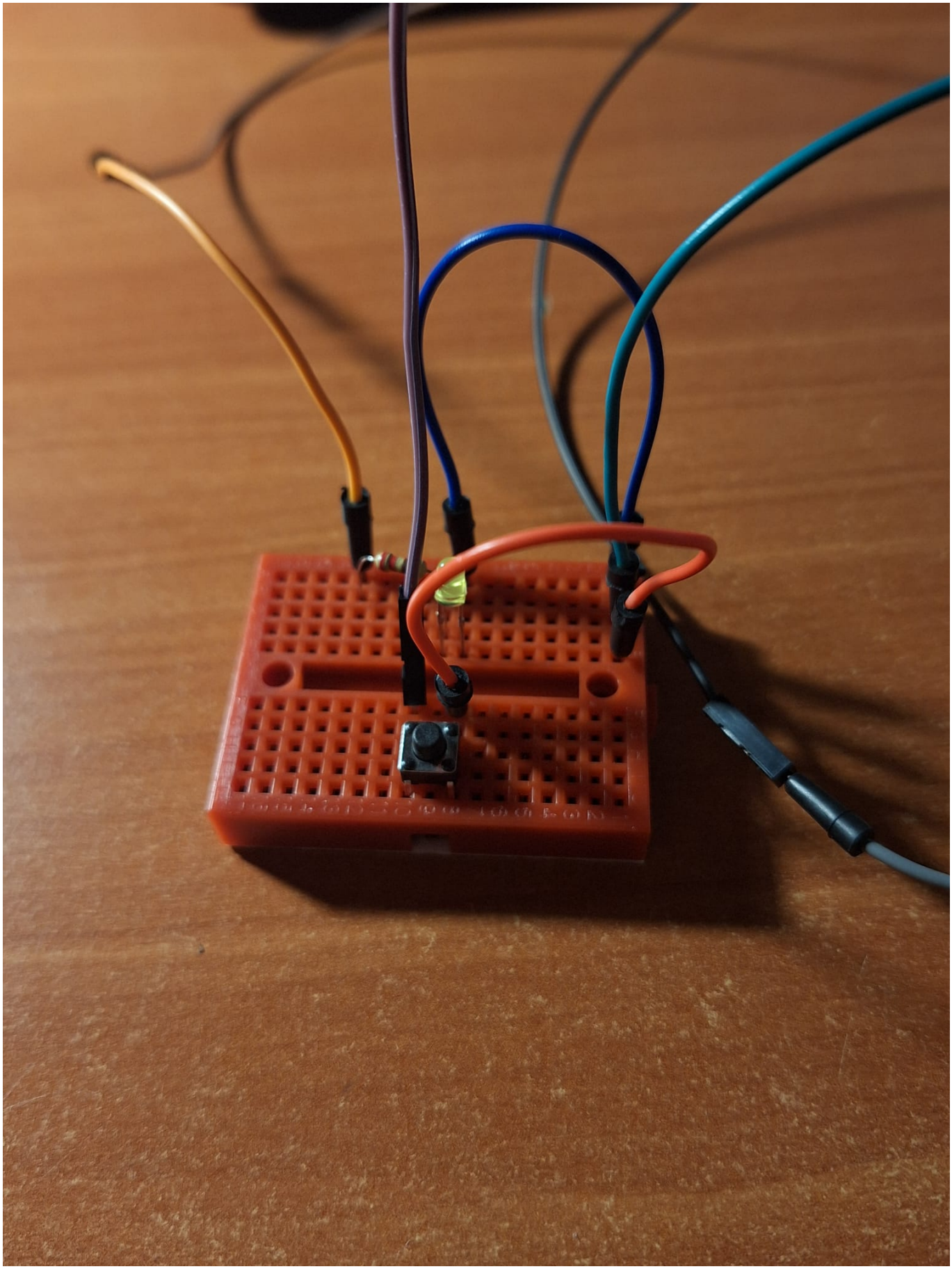
Schema electrica:

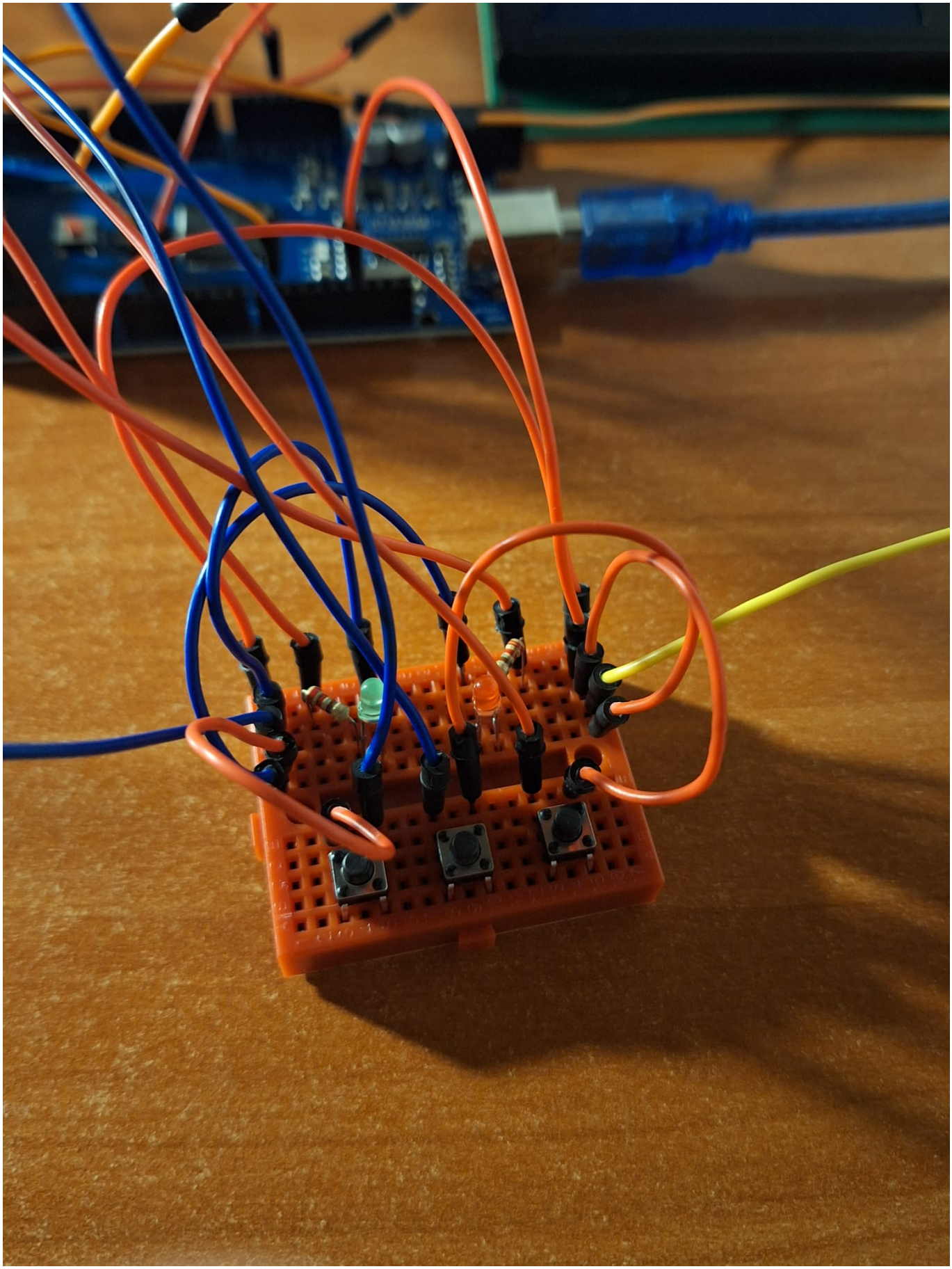


Poze hardware:




Plusivo
GND VDD VO RS RW E D0 D1 D2 D3 D4 D5 D6 D7 BLA BLK





Software Design

Mediu de dezvoltare

- IDE: **Vizual Studio Cod**
- Limbaj: **C++**
- Extensie: **PlatformIO**

Biblioteci:

- standard: <avr/io.h> si <avr/interrupt.h>
- externe: <Arduino.h> si <LiquidCrystal.h>

Functii si organizarea codului:

Din punct de vedere al organizarii, avem fisierul `main.cpp` care contine logica principala a jocului si a setup-ului, iar pe langa acesta avem mai multe headere care contin functionalitati ajutatoare:

- `lcd_text.h`
  - in acest header am declarat toate textele care vor fi afisate pe ecran (*intro*, *intrebari*, *scor final*);
  - textele sunt stocate ca siruri de caractere constante in memoria flash (**PROGMEM**); am ales sa stochez in memoria flash deoarece placuta dispune de o dimensiune mare (**~270KB**), pentru a nu umple RAM-ul si deoarece intrebarile fiind numeroare, ele ocupa un spatiu mai mare;

```
// intro
const char intro[] PROGMEM = "Ready? <3";
// prompt-ul jucatorului 1 pentru scorul final
const char player1Prompt[] PROGMEM = "1: ";
```

- `utils.h`
  - in acest header am definit toate **constantele** - pinii tuturor componentelor, cate intrebari sa fie puse per joc, cat sa dureze o intrebare, cat de rapid sa accelereze “tic-tac”-ul generat de buzzer,etc.;
  - am definit alte **tipuri de date** - un enum pe care l-am folosit pentru a determina in ce stare ma aflu in joc;
  - am declarat toate **variabilele** globale - care tin de player, care tin de alegerea intrebarilor, care tin de printarea textului, care tin de buzzer sau care tin de butoane; de asemenea, variabilele care sunt accesate/modificate in intreruperi sunt declarate de tip `volatile`
- `functions.h`
  - in acest header avem numeroase functii, atat legate de o componenta hardware sau de un concept, cat si functii care sa inglobeze o secventa de instructiuni, cu scopul de a lasa usor vizibila logica in cadrul main-ului;

- pentru **timer**:
  - in primul rand, timerul l-am folosit pentru a cronometra cat dureaza o intrebare dar si pentru a accelera buzzerul;
  - am folosit *timerul 1* si am inceput prin a-l initializa; am calculat OCR1A astfel incat el sa genereze o intrerupere o data la o milisecunda;
  - la fiecare intrerupere, daca cumva au trecut numarul de secunde alocat per intrebare, contorul timerului se reseteaza; de asemenea, timerul se mai ocupa si cu trecerea buzerului prin cele trei stari ale lui: **tic - off - tac - ..**; pe parcursul acelor `TIME_ROUND` secunde alocate intrebarii, distanta dintre aceste stari scade cu `STEP_PAUSE_DURATION`; astfel realizam accelerarea sunetelor scoase de buzzer pentru fiecare intrebare;

- pentru **intreruperi butoane**:
  - butoanele PRESS sunt legate la pinii microcontroler-ului care permit intreruperi externe (`INT2`, `INT3`, `INT4`, `INT5`);
  - pe langa initializarea butoanelor pe modul de `INPUT` si activarea rezistentei de **PULL_UP**, am si activat aceste intreruperi externe si le-am setat pe **falling edge** - intreruperea sa aiba loc cand se trece din `HIGH→LOW`;
  - in cadrul intreruperilor, deoarece avem nevoie sa fie rapide, am folosit numeroase variabile care sunt modificate in functie de scopul butonului in starea respectiva a jocului - trecea de la intro la intrebari, raspunderea la o intrebare sau reinceperea jocului;

- pentru **printarea pe ecran**:
  - avem implementari diferite in functie de ce ne dorim sa printam - intro, intrebare cu variante sau scor;
  - pentru toate acestea am folosit functii din biblioteca externa **LiquidCrystal**:
    - `clear` - sterge tot ecranul
    - `setCursor(coloana, rand)` - pentru a merge pe ecran la pozitia de unde ne dorim sa incepem scrierea textului;
    - `print`

- pentru **buzzer**:
  - pentru buzzer am ales diferite frecvente care sa ma ajute sa formez un sunet sugestiv pentru diferite situatii: tic → **1000 Hz**, tac → **1500 Hz**, raspuns corect → ne trebuie un sunet inalt → **1200 Hz**, raspuns gresit/ terminarea timpului → un sunet mai jos → **400 Hz**
  - pentru a genera sunetul efectiv am folosit `tone` si `noTone` din biblioteca externa **Arduino** in combinatie cu variabilele modificate de timer (care ne spun ce tip de sunet trebuie generat la fiecare moment);

- pentru **generarea aleatoare a intrebarilor**:
  - a trebuit sa gasesc o varianta de a genera un *seed* aleator pentru a putea genera o serie aleatoare de intrebari la fiecare joc;
  - pentru asta m-am gandit sa folosesc zgomotul pinilor analogici (mai exact A0) datorita incertitudinii lui;

- pentru a converti zgomotul in informatie folositoare am implementat un **ADC**;
  - am ales un *prescaler* astfel incat conversia sa fie buna (**128**);

Concepte din laboratoare:

- *GPIO* - configurarea pinilor pentru LED-uri si butoane;
- *Intreruperi* - o parte din butoane au configurate intreruperi externe;
- *Timer* - pentru cronometrarea intrebarilor si accelerarea buzzer-ului;

Logica principala:

Pe parcursul jocului el trece prin mai multe stari, stari in care poate asteapta un input sau trebuie sa realizeze anumite operatiuni, o data sau de mai multe ori; motiv pentru care am ales sa implementez un **Finite State Machine (FSM)**; cu aceasta abordare, **loop()** ramane functia principala, iar in fiecare bucla se realizeaza codul corespondent stari in care ne aflam; in functie de modificari si necesitate, se trece de la o stare la alta.

Avantajul este usurinta cu care se poate implementa si extinde, dar si probabilitatea mult mai mica de a avea probleme, inconsistente sau poate chiar blocari.

Pentru a implementa starile am folosit un **enum**, iar in cadrul buclei, folosind un switch ne dam seama in ce stare ne aflam.

```
enum GameState {
    INTRO, // afisam intro-ul
    WAIT_INTRO_DONE, // asteptam input de la jucatori
    QUESTION, // afisam o intrebare
    WAIT_PLAYER_QUESTION, // asteptam input de la jucatori
    ANSWER, // o mica animatie
    WAIT_ANSWER, // asteptam un raspuns de la jucatori
    AFTER_ANSWER, // daca se poate mergem la urmatoarea intrebare, daca nu
afisam scorul final
    WAIT_FOR_RESTART // asteptam input de la jucatori (daca vor sa joace din
nou)
};
```

Un element de implementare pe care l-am folosit a fost **polling** pentru asteptarea unui input in starea de WAIT_ANSWER. Fiind in aceasta stare urmatoare actiune posibila este strict apasarea unui buton din cele trei care reprezinta variantele de raspuns (A, B sau C). In aceasta situatie, verificarea repetata a valorilor ai acestori pini pana cand unul devine **LOW** este suficienta.

```
case WAIT_ANSWER:
 // wait for answer
 stateA = (PING & (1 << PG1)) ? 1 : 0;
 stateB = (PING & (1 << PG0)) ? 1 : 0;
 stateC = (PINL & (1 << PL7)) ? 1 : 0;
```

```
if (stateA == LOW || stateB == LOW || stateC == LOW) {
   if (stateA == LOW) player_answer = 1;
   else if (stateB == LOW) player_answer = 2;
   else if (stateC == LOW) player_answer = 3;
   checkPlayerAnswer();
   state = AFTER_ANSWER;
}
break;
```

Rezultate Obţinute

Ca si rezultat, am reusit sa construiesc jocul sa faca tot ce mi-am dorit initial! Mai jos este atasat un mic demo care prezinta un exemplu de sesiune!

demo!

Concluzii

A fost o experienta placuta. Am realizat ceva foarte nou pentru mine si desi a fost cu peripetii pe alocuri, s-a meritat efortul. Am invatat foarte multe, mai ales pe partea de hardware si pot spune ca am avut noroc sa nu ard nimic!

Jocul consider ca are un potential mare desi acum e o varianta mai simpla, poate fi extins cu tot felul de alte features. Sper ca persoanele care au ocazia sa il testeze sa se simta bine!

Download

Arhiva: do_you_know_game.zip

GitHub

Jurnal

Bibliografie/Resurse

Resurse software:

- Documentatie biblioteca externa *LiquidCrystal* - doc_lcd
- Documentatie biblioteca externa *Arduino* - doc_arduino
- Laboratoarele 1-4

Resurse hardware:

- Datasheet Arduino Mega 2560 - mega-datasheet.pdf
- Conectivitate LCD - lcd
- Tutorial cum sa lipesti un header la LCD - tutorial
- Laboratoarele 1-4