

ARDU Chess




ACIOCANESA BIANCA
A1



ARDUchess

Știați că, la nivel global, aproximativ **600 de milioane de persoane** sunt pasionate de șah, iar platformele online atrag peste **100 de milioane de utilizatori**? Totuși, pentru începători, șahul poate părea intimidant, cu reguli complexe și strategii greu de înțeles. Ideea proiectului meu porneste de la aceasta idee și își dorește să ofere o modalitate de a transforma învățarea șahului într-o experiență interactivă, intuitivă și captivantă.

Ce este ARDUchess? Pe scurt, acesta combină tradiția cu tehnologia pentru a oferi o tablă de șah inteligentă, realizată cu ajutorul: **Plăcii Arduino Mega**, **Senzorilor electromagnetici**, **LED-urilor RGB**. Cum funcționează? Simplu! Atunci când ridicați o piesă, tabla luminează toate mutările valide. Este ca și cum tabla devine un antrenor personal de șah, ghidându-vă pas cu pas.

1. IDEI ASEMENATOARE DEJA EXISTENTE PE PIATA

Officially Supported by 



ChessUp 2 Smart Chess Board

1.837,59 lei ~~2.143,56 lei~~ **Save 306,27 lei**

Regular Price: €419.99

Estimated delivery between **Wed, Jan 22** and **Sat, Jan 25**.

Quantity:

Add to cart **Buy with shop Pay**

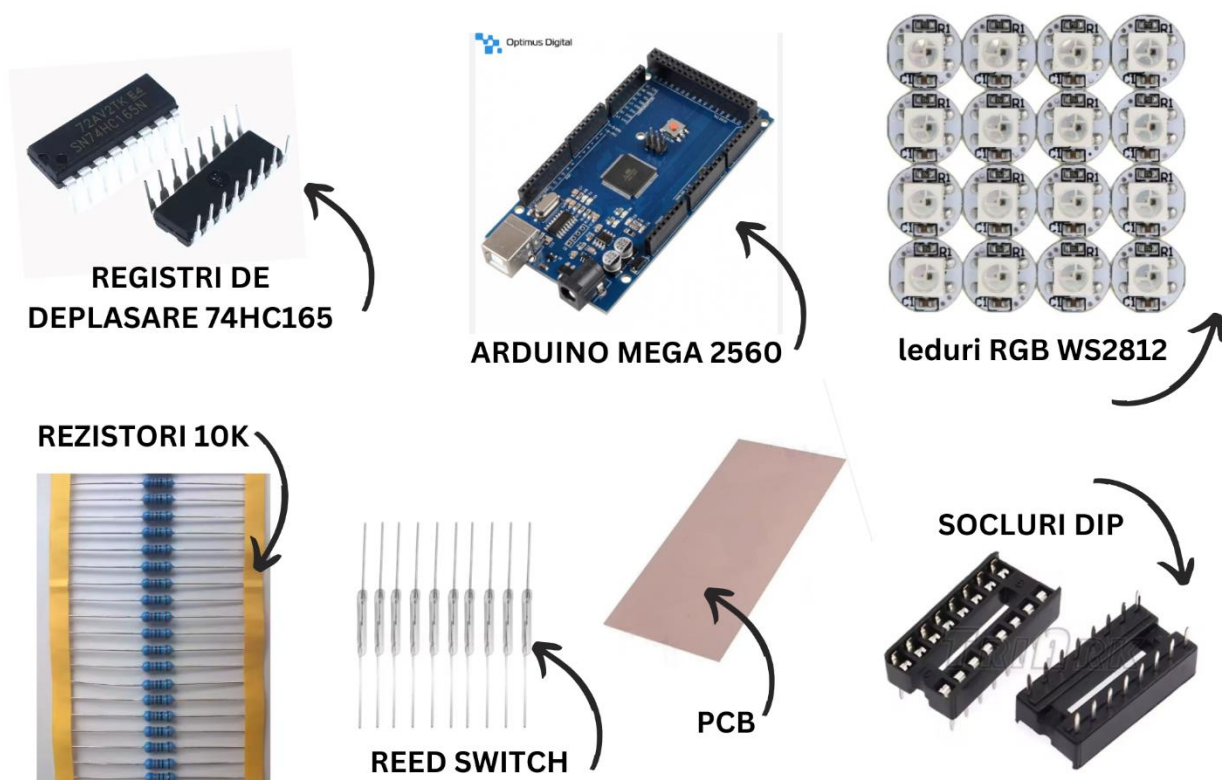
[More payment options](#)

Această idee de “tablă de șah inteligentă” există deja și poate fi cumpărată chiar de la cei care oferă cel mai popular site de șah, chess.com. Tabla de șah poate fi găsită sub numele de [ChessUP](https://chess.com/products/chessup-2-smart-chess-board), însă prețul ei este de aproximativ **2000 de lei**. Pe lângă faptul că proiectul **ARDUchess** își propune să ofere o modalitate de a face învățarea șahului o experiență captivantă, de asemenea demonstrează că ne putem crea propria tablă de șah, cu aproximativ aceleași funcționalități ca ChessUP, însă la un **preț semnificativ mai redus**.

2. COMPONENTE NECESARE

Pentru realizarea proiectului au fost necesare urmatoarele componente pentru realizarea partii hardware:

- ☑ Placa de dezvoltare compatibila cu Arduino MEGA 2560 ([Optimus Digital, 72.99 lei](#))
- ☑ placa PCB cu 2 fete de dimensiune ~ 215 x 215 (General System, ~30 lei -> ceva asemanator cu [aceasta](#), doar ca de dimensiune mai mare)
- ☑ 64 de leduri RGB WS2812 ([Aliexpress, 5.92 lei/set de 50 de leduri](#))
- ☑ 64 de comutatori magnetici Reed ([Aliexpress, 23.86 lei/set de 100](#))
- ☑ 64 de rezistori de 1K ohm ([Aliexpress, 4.89 lei/set de 100](#))
- ☑ 8 registri de deplasare 74HC165 ([Aliexpress, 7 lei/set de 10](#))
- ☑ 8 socluri DIP, 16 pini ([Aliexpress, 4.89 lei/set de 10](#))
- ☑ alte elemente necesare pentru imprimarea circuitului pe PCB + lipirea elementelor (letcon, fludor etc.)



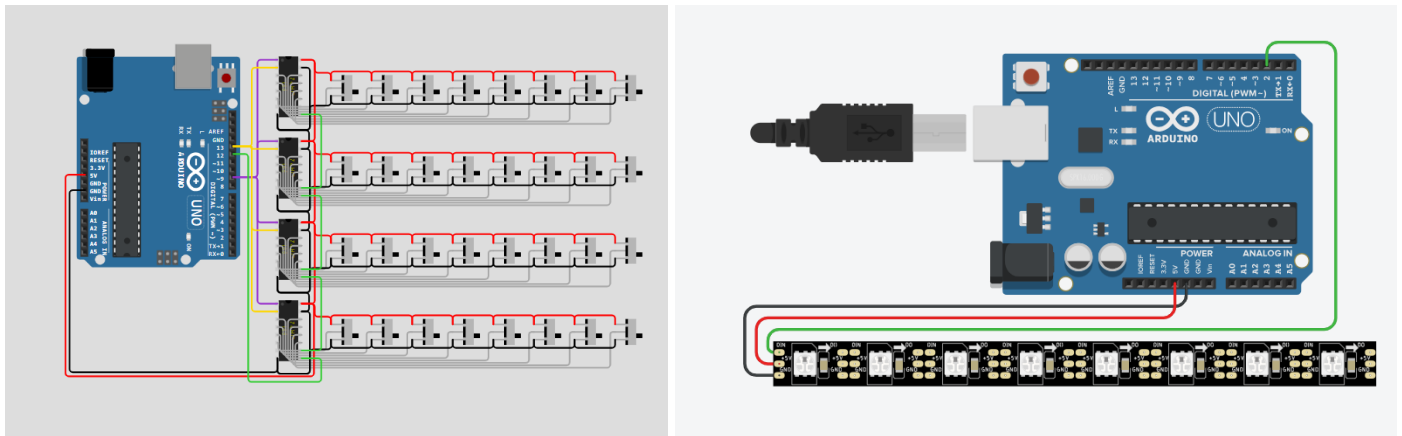
Pentru realizarea partii software am folosit Arduino IDE (pentru logica sahului si a componentelor) + Eagle 7 (pentru circuit).

3. PASII REALIZarii PROIECTULUI

- a. REALIZAREA CIRCUITULUI
- b. IMPRIMAREA CIRCUITULUI PE PCB
- c. LIPIREA COMPONENTELOR
- d. SCRIEREA CODULUI PENTRU LOGICA MUTARII PIESELOR
- e. TESTAREA FUNCTIONALITATII

● REALIZAREA CIRCUITULUI

Înainte de începerea realizării designului circuitului, am folosit wokwi și tinkercad pentru a verifica cum se realizează conexiuni funcționale între led-uri sau între regiștri si reeds. Astfel, schemele de mai jos au fost orientative pentru realizarea circuitului ulterior (acest simulator nu conține reed switches, de aceea pe schemă se vor observa butoane in locul lor, pentru a simula aproximativ aceeași funcționalitate).



Pentru realizarea circuitului, dacă am face un calcul, ar fi nevoie de aproximativ: 3 conexiuni de la un led la altul, 8 conexiuni de la registrul de deplasare la minusul fiecărui reed, 3 conexiuni de la un registru la altul pentru serializarea acestora ...etc. Ideea este că realizarea proiectului ar fi fost mult mai dificilă dacă nu aș fi imprimat circuitul pe un PCB, deoarece utilizarea unui număr mare de cabluri ar fi dus la un aspect dezordonat și neprofesional, mai ales pe o suprafață mică, așa cum este cea a tablei de șah (aproximativ 20 x 20 cm). În plus, conexiunile realizate manual ar fi fost mai susceptibile la erori mai mari, creând riscuri suplimentare de scurtcircuit. PCB-ul oferă un design compact, clar și organizat, facilitând atât asamblarea, cât și funcționarea fiabilă a proiectului. Așa că am început prin realizarea circuitului în aplicația **Eagle**.

Se începe cu un editor de scheme electrice, unde componentele electronice (rezistențe, condensatori, tranzistori etc.) sunt adăugate dintr-o bibliotecă extinsă. Componentele sunt conectate între ele utilizând trasee, care reprezintă conexiunile electrice. Fiecare componentă are un simbol schematic și un layout fizic corespunzător. După finalizarea schemei, proiectul este transferat în editorul de PCB. Componentele plasate în schematică vor apărea ca simboluri fizice, care pot fi poziționate pe placa de circuit imprimat. Conexiunile electrice definite în schematică apar ca "airwires" (trasee virtuale), indicând unde trebuie să fie conectate pe PCB. După finalizarea design-ului, se generează fișierele care includ informații despre trasee, găuri, straturi și alte detalii esențiale.

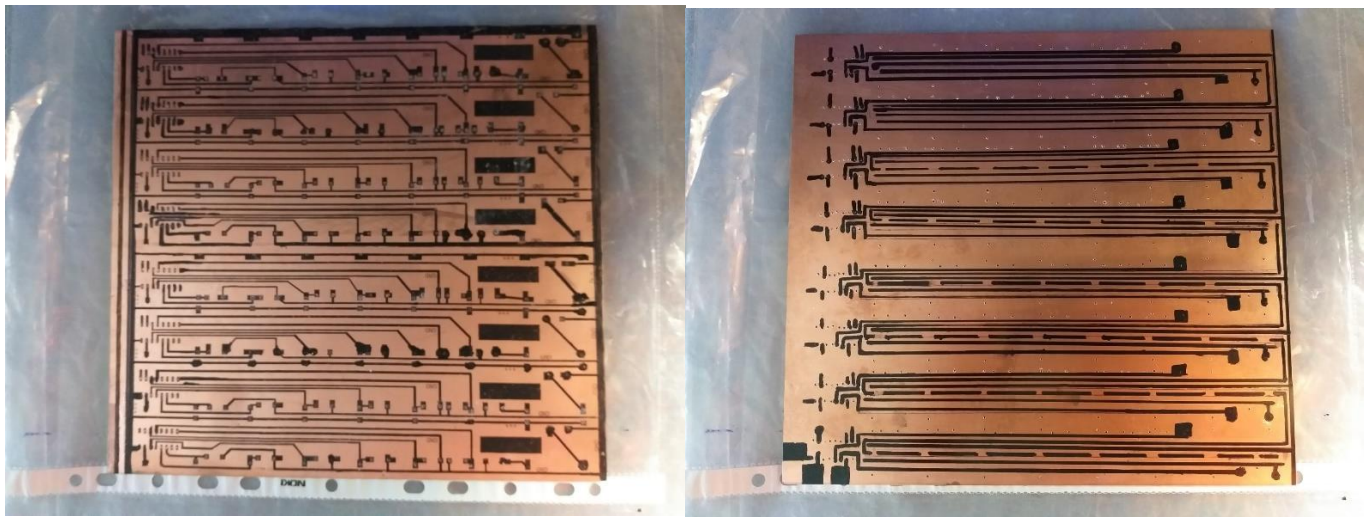
Pentru a realiza un circuit cât mai ușor de înțeles, dar și din cauza spațiului redus al plăcii, design-ul circuitului a fost creat folosind 2 fețe ale plăcii PCB, astfel: pe fața plăcii au fost realizate conexiunile ledurilor, iar pe spatele

acestea legăturile dintre registri și comutatorii reed (din lipsa de spațiu, pentru ultimii 2 reeds de pe fiecare linie, traseele au fost trasate tot pe fața circuitului).

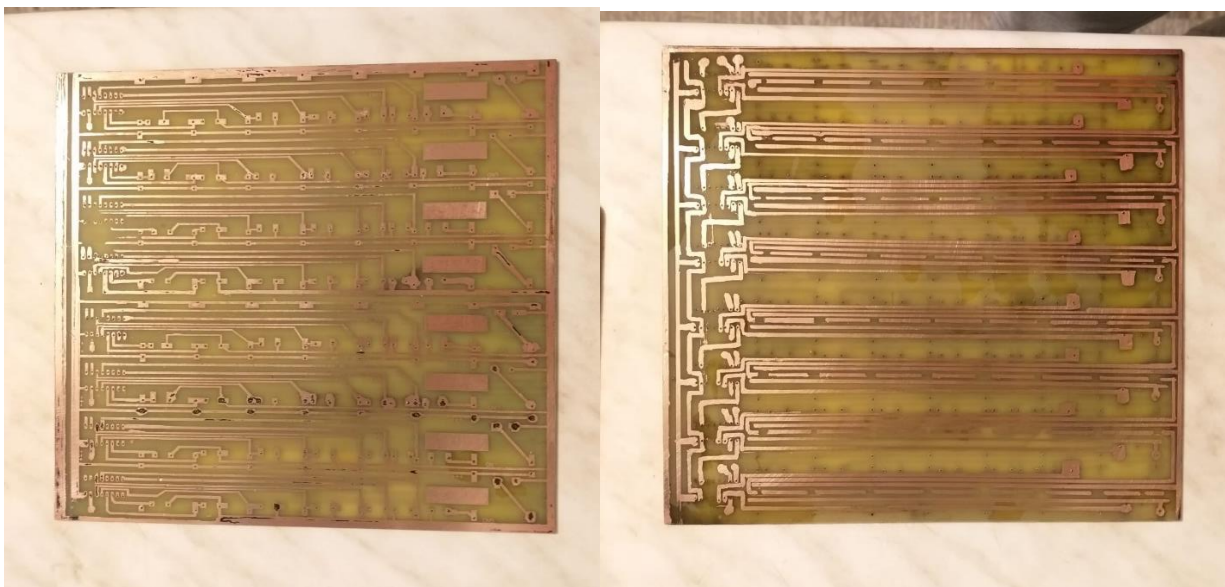
Astfel, circuitul a ajuns la această formă finală.

- **IMPRIMAREA CIRCUITULUI PE PCB**

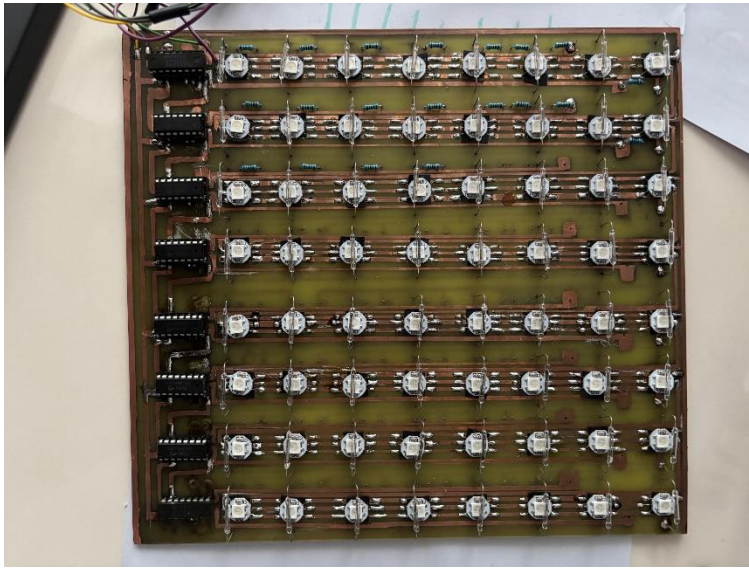
Pentru a imprima un circuit pe un PCB folosind o imprimantă, se începe prin exportarea layout-ului prezentat la pasul de mai sus în format alb-negru și oglindit. Design-ul se imprimă cu o imprimantă laser pe hârtie specială de transfer termic (sau orice hârtie lucioasă). Hârtia imprimată se așază cu cerneala spre cupru și se presează cu un fier de călcat pentru câteva minute, transferând cerneala pe placă. După răcire, hârtia se înmoaie în apă caldă și se îndepărtează, lăsând traseele cernelei pe cupru.



Placa se scufundă apoi într-o soluție de corodare (clorură ferică), dizolvând cuprul expus, iar traseele rămân intacte (aproximativ 😊). După corodare, cerneala este îndepărtată cu alcool izopropilic sau acetonă, traseele de cupru fiind astfel vizibile. Urmează găurirea pentru montarea componentelor.



- **LIPIREA COMPONENTELOR**



- **SCRIEREA CODULUI PENTRU LOGICA MUTARII PIESELOR**

Codul este format din 3 părți:

- fișierul chess.hpp, care conține declarațiile pentru clasa de bază Piece, și clasele derivate din aceasta, câte una pentru fiecare piesă, care vor suprascrie funcția valid_moves() în funcție de modul în care piesa respective se poate deplasa pe table de șah. De exemplu:

```
16 class Piece {
17 protected:
18     int x, y;
19     Player player;
20 public:
21     Piece(int coord_x, int coord_y, Player player);
22     void set(int coord_x, int coord_y);
23     int getX();
24     int getY();
25     Player get_player();
26     void move(vector<vector<Piece*>>& board, int new_x, int new_y);
27     virtual vector<vector<int>> valid_moves(vector<vector<Piece*>> board, bool flag) = 0;
28     bool isCheck(vector<vector<Piece*>> board);
29     virtual string getType() const { return "NotKing"; }
30 };
31
```

```
46 class Knight : public Piece {
47 public:
48     Knight(int coord_x, int coord_y, Player player) : Piece(coord_x, coord_y, player) {}
49     vector<vector<int>> valid_moves(vector<vector<Piece*>> board, bool flag) override;
50 };
51
52 class Bishop : public Piece {
53 public:
54     Bishop(int coord_x, int coord_y, Player player) : Piece(coord_x, coord_y, player) {}
55     vector<vector<int>> valid_moves(vector<vector<Piece*>> board, bool flag) override;
56 };
57
```

- fișierul chess.cpp, care conține aceste funcții și logica lor: în general, fiecare funcție va returna un vector de vectori cu 3 poziții (coordonatele x și y reprezentand poziția pe care piesa poate fi mutate, și un 1 sau 2 care

reprezintă culoarea cu care ar trebui luminată acea căsuță – bverde dacă poziția e goală sau roșu dacă este atacată o altă piesă)

- fișierul chess.ino, care va conține logica pentru senzori și leduri, folosindu-se și de funcțiile menționate anterior pentru logica mutării pieselor de șah.

4. CONCLUZII SI IDEI DE VIITOARE ÎMBUNĂTĂȚIRI

Cu toți acești pași parcurși până acum, urmați de mai multe etape de verificare și reparare a erorilor care au apărut atât la scrierea codului, cât și la partea hardware, din cauza zonelor în care PCB-ul nu a fost corodat exact cum trebuie (și existau mai multe întreruperi pe parcursul circuitului), am ajuns la finalul realizării proiectului, care totuși nu este încă în varianta compet finalizată.

Câteva idei de viitoare îmbunătățiri ar fi următoarele:

- finalizarea codului pentru logica jocului și adăugarea componentelor care nu sunt încă lipite pe PCB
- realizarea unui case pentru întregul proiect, pentru a îmbunătăți aspectul vizual al tablei de joc
- adăugarea unui ecran LED cu niște butoane (care ar fi foarte util pentru a afișa mutările jucătorilor sau, cel mai important, pentru a realiza de exemplu pawn promotion, deoarece în momentul de față, fără monitorul serial din care am putea alege în ce dorim să transformăm pionul respective, nu avem o modalitate direct de pe table de joc pentru a face asta)