Politechnika Lubelska
Wydział Elektrotechniki i Informatyki
Inżynierskie Zastosowania Informatyki w Elektrotechnice

Imię i nazwisko autora:
Marek Rowiński
nr 93342

Szachownica elektroniczna SmartBoard

Promotor:
dr inż. Jacek Majcher

Lublin, 2024

Spis treści:

1. **Wprowadzenie.....1**
2. **Wymagania systemowe.....1**
3. **Sposób użytkowania.....2**
4. **Środki ostrożności zestawu elektronicznego.....2**
5. **Projekt hardware'u**
    - **5.1. Schematy elektryczne.....3**
    - **5.2. Lista oraz opis poszczególnych komponentów.....8**
    - **5.3. Kosztorys.....10**
6. **Projekt software'u**
    - **6.1. Odczytywanie figur z szachownicy.....11**
    - **6.2. Tworzenie ruchów w formacie PGN.....14**
    - **6.3. Logika stanu gry.....14**
    - **6.4. Obsługa interakcji z użytkownikiem.....15**
    - **6.5. Zarządzanie czasem i zegarem.....15**
    - **6.6. Potencjalne ruchy i bicie.....15**
7. **Analiza i wnioski**
    - **7.1. Ocena osiągniętych rezultatów.....16**
    - **7.2. Analiza napotkanych problemów i ich rozwiązania.....16**
    - **7.3. Możliwe usprawnienia i przyszłe kierunki rozwoju projektu.....17**
8. **Bibliografia.....17**

1. Wprowadzenie

Elektroniczne szachownice to nowoczesne narzędzia, które rewolucjonizują tradycyjną grę w szachy poprzez integrację zaawansowanych technologii. Moim celem jest stworzenie innowacyjnej elektronicznej szachownicy, która automatycznie rozpoznaje ruchy, umożliwia grę online i ułatwi analizę i archiwizację partii.

Motywacją do realizacji projektu jest rosnące zainteresowanie grą w szachy oraz potrzeba dostarczenia narzędzia, które umożliwi graczom śledzenie rozgrywek w sposób nowoczesny i efektywny. Dzięki elektronicznym szachownicom, gra staje się bardziej dostępna, interaktywna i inspirująca.

2. Wymagania systemowe

System ma realizować szereg funkcji, które mają na celu ułatwienie i udoskonalenie doświadczenia użytkownika. Główne funkcje obejmują:

- Gra online na platformie Lichess: Użytkownik będzie mógł grać partie szachów online poprzez integrację z platformą Lichess. System umożliwi logowanie się na konto Lichess, wyszukiwanie przeciwników oraz rozgrywanie partii w czasie rzeczywistym.
- Archiwizowanie partii na koncie Lichess: Podczas rozgrywki system będzie automatycznie zapisywał przebieg partii w formacie UCI na platformie Lichess, umożliwiając użytkownikowi późniejszą analizę i archiwizację rozgrywki.
- Interaktywny interfejs użytkownika: System będzie posiadał intuicyjny interfejs użytkownika, który umożliwi łatwe korzystanie z funkcji, takich jak rozpoczęcie nowej gry, wybór przeciwnika online oraz dostęp do zapisanych partii.

Dodatkowo, celem projektu jest stworzenie elektronicznej szachownicy łączącej funkcjonalność, estetykę i ergonomię, umożliwiającej grę w szachy w sposób wygodny i nowoczesny. Szachownica automatycznie rejestruje ruchy podczas gry, przesyła je do platformy Lichess, gdzie są zapisywane w formacie UCI, co eliminuje konieczność ręcznego wprowadzania posunięć. Dzięki temu

użytkownik może analizować partie, korzystać z programów szachowych lub grać online – zarówno przeciwko komputerowi, jak i innym graczom na Lichess. Szachownica działa także jako urządzenie wejściowe, przenosząc ruchy wykonane na planszy bezpośrednio do aplikacji lub stron internetowych.

3. Sposób użytkowania

Elektroniczna szachownica jest wyposażona w wyświetlacz z interfejsem, który pokazuje czas trwania partii oraz oferuje podstawowe funkcje, takie jak gra online lub z komputerem (Stockfish 5). Aby rozpocząć grę, należy poprawnie ułożyć figury na szachownicy zgodnie z zasadami szachowymi. Aby sprawdzić czy położenie figur pokrywa się z czujnikami należy wejść do debug mode klikając boczny przycisk i wybrać odpowiednią opcję (Testowanie kontraktonów). Po ustawieniu figur, użytkownik może wybrać odpowiednią opcję na ekranie dotykowym, aby zagrać online lub z silnikiem.

Podczas gry, po wykonaniu ruchu przez gracza, należy delikatnie nacisnąć odpowiedni przycisk na ekranie dotykowym, aby wysłać ruch do serwerów Lichess. Wszystkie partie są zapisywane i dostępne na stronie lichess.org na koncie marekr93342. Ważne jest, aby figury były umieszczane równo i dokładnie na polach szachownicy, co zapewni prawidłowe działanie systemu detekcji ruchu.

W celu zapewnienia optymalnej wydajności i dokładności, należy dbać o regularne czyszczenie powierzchni szachownicy i wyświetlacza.

4. Środki ostrożności zestawu elektronicznego

Do wykrywania elektronicznych figur szachowych na szachownicy używany jest rezonans elektromagnetyczny. Silny sygnał wygenerowany przez różnego rodzaju nadajniki w pobliżu urządzenia może spowodować samoistne i wadliwe wykrycie nieistniejących figur na danych polach szachownicy, tzw. zjawisko "figur duchów".

Jeśli problemy z wykrywaniem "figur duchów" występują podczas używania systemu, przyczyną może być umieszczenie szachownicy zbyt blisko ekranów plazmowych, monitorów LCD, głośników, zasilaczy i innych podobnych urządzeń. Metalowe powierzchnie lub przedmioty umieszczone obok, pod szachownicą lub pod stołem, na którym się ona znajduje, mogą zmniejszyć czułość systemu, co może spowodować błędy w wykrywaniu figur.

Nie należy umieszczać szachownicy na metalowej tablicy lub stole z metalową konstrukcją pod blatem.

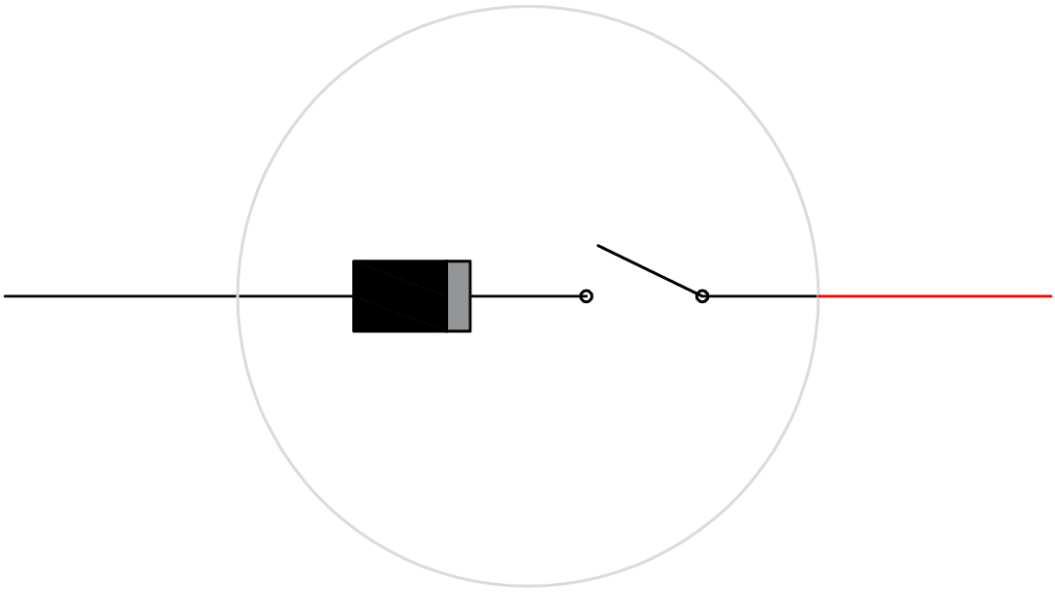
Sprzęt należy chronić przed skrajnymi warunkami atmosferycznymi, takimi jak deszcz, wysoka temperatura czy intensywne nasłonecznienie.

5. Projekt hardware'u

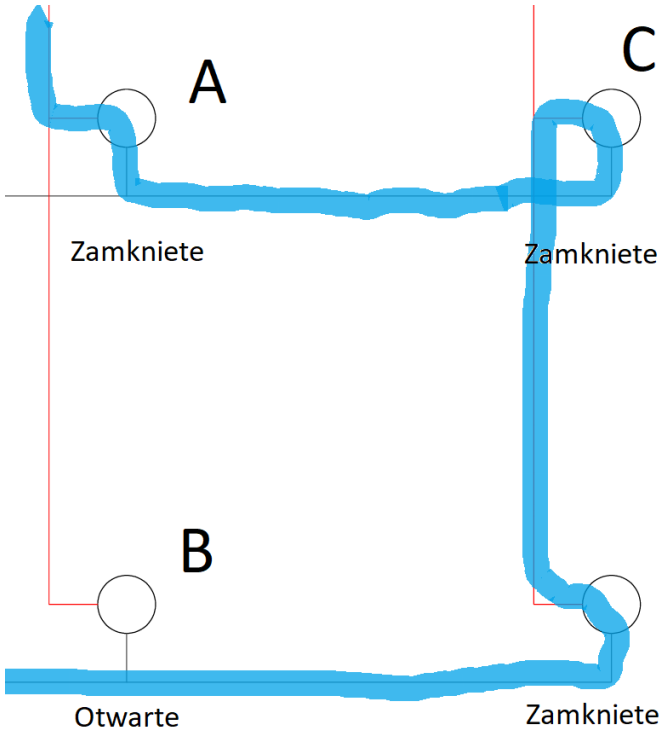
5.1. Schematy elektryczne:

Schematy elektryczne przedstawiają szczegółowy układ połączeń i komponentów zastosowanych w elektronicznej szachownicy. Obejmują one rozmieszczenie czujników detekcji ruchu, układ zasilania, moduły komunikacyjne oraz interfejsy do przesyłania danych.

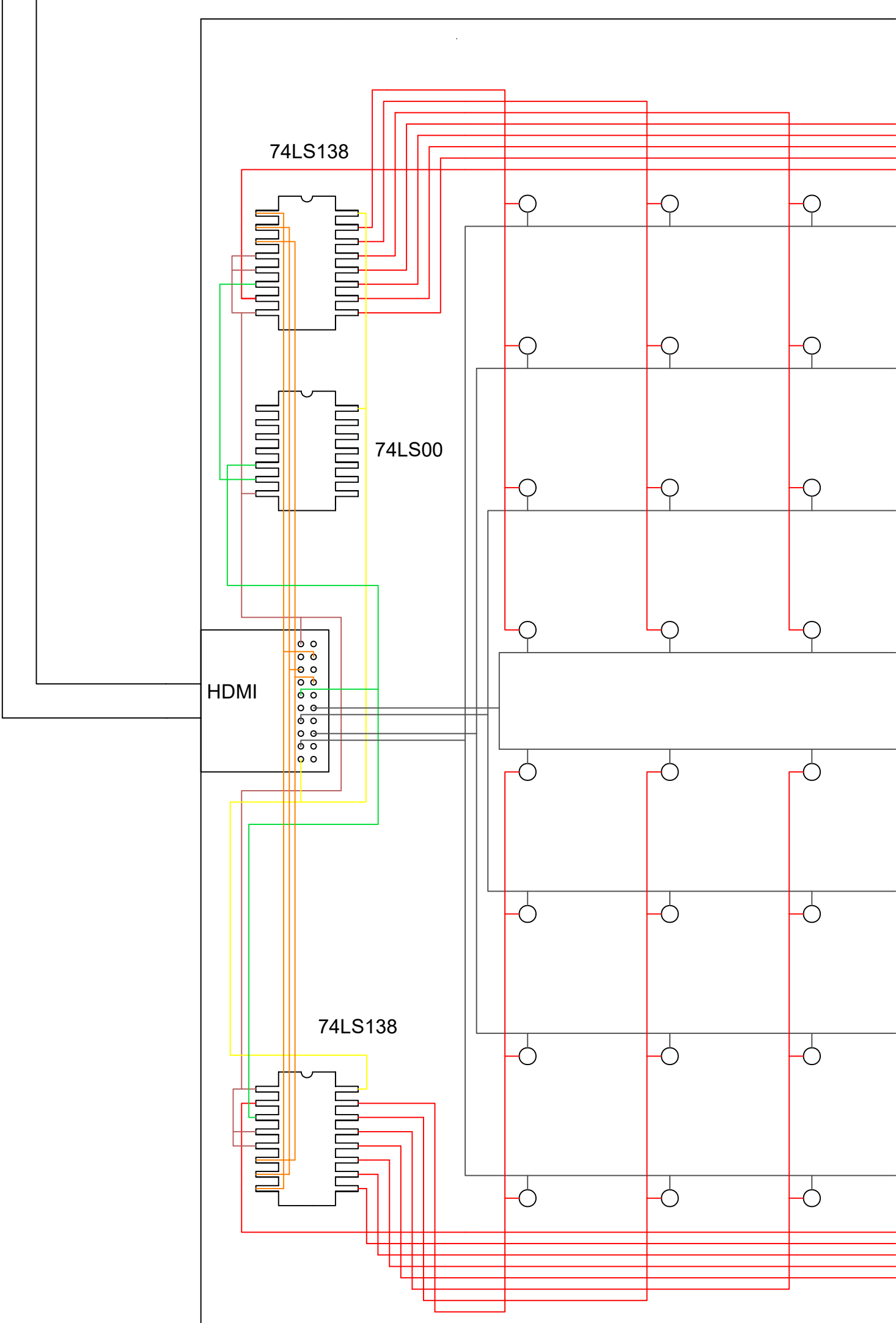
Schemat (Sch. 1.) składa się z diody Schottky'ego oraz kontraktonu. Kontraktony są używane do wykrywania obecności figur z wbudowanymi magnesami, natomiast dioda zapobiega powstawaniu fałszywych sygnałów w układzie. Na schemacie (Sch. 2.) przedstawiono problem fałszywego odczytu sygnału w przypadku braku diod. Jeśli na trzech polach (A, C, D) są figury (zamknięte przełączniki), a na jednym polu (B) jej nie ma (otwarty przełącznik), sygnał mógłby niechcący przejść przez inne zamknięte pola i dać błędny odczyt.

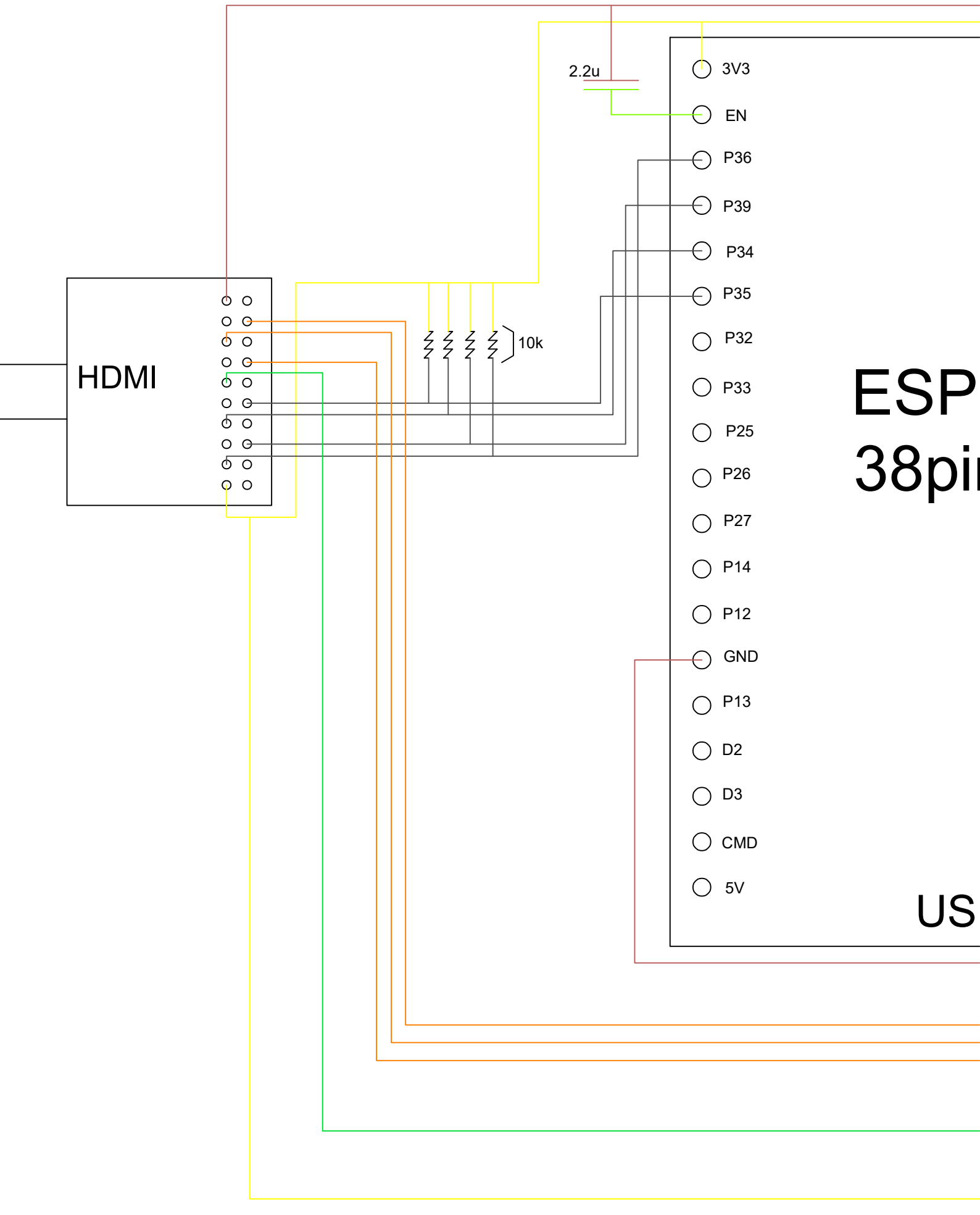


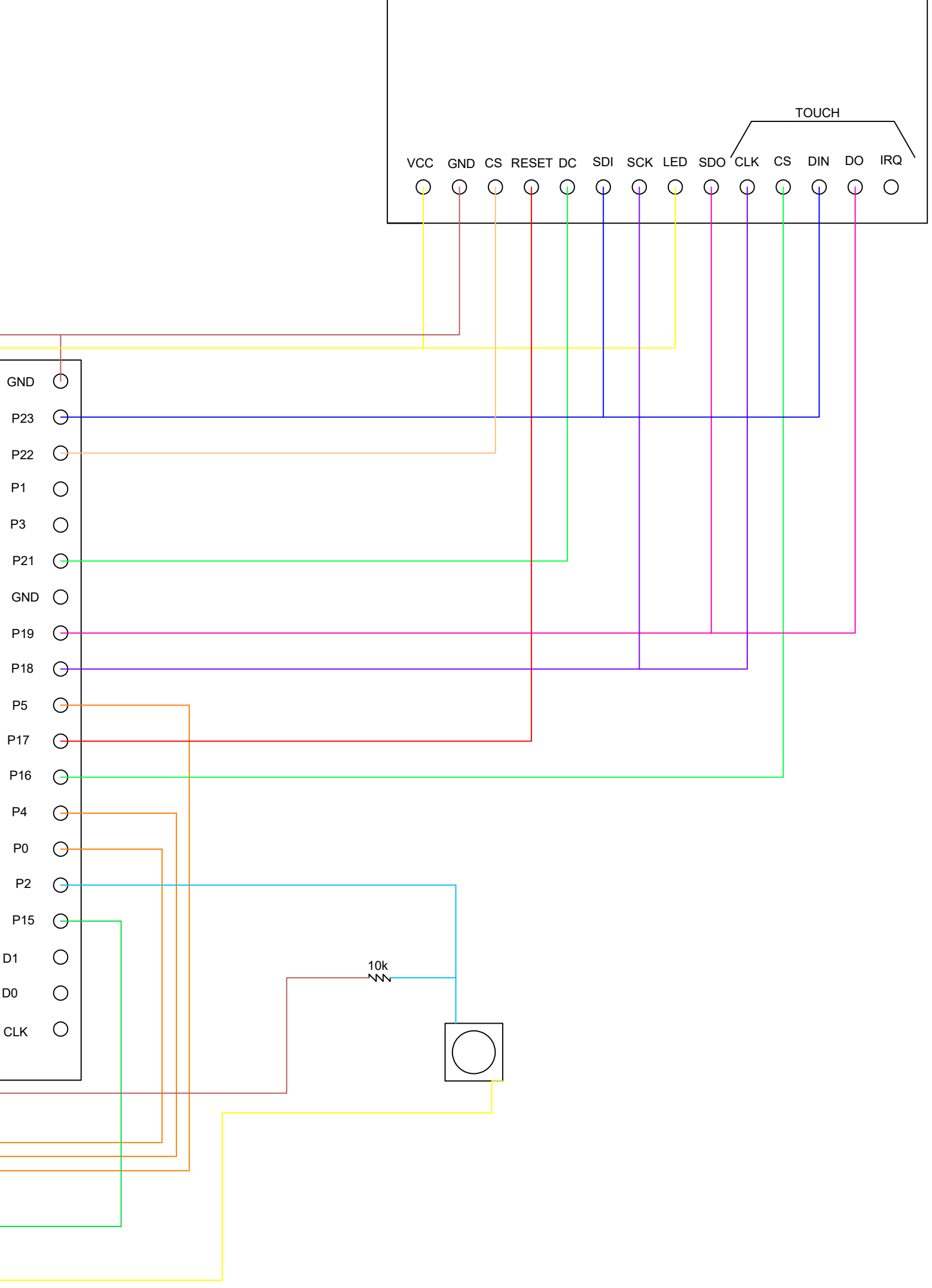
Sch. 1. Element odczytujący jedno pole na szachownicy



Sch. 2. Możliwe powstanie nieporzadanego sygnału


74LS138
74LS00
HDMI
74LS138


2.2u
3V3
EN
P36
P39
P34
P35
P32
P33
P25
P26
P27
P14
P12
GND
P13
D2
D3
CMD
5V
10k
HDMI
ESP
38pi
US


TOUCH
VCC
GND
CS
RESET
DC
SDI
SCK
LED
SDO
CLK
CS
DIN
DO
IRQ
GND
P23
P22
P1
P3
P21
GND
P19
P18
P5
P17
P16
P4
P0
P2
P15
D1
D0
CLK
10k

W analizie schematów elektronicznych szachownicy można wyróżnić kluczowe elementy:

1. **Rezystory 10k**
   Rezystory pełnią funkcję pull-up lub pull-down, stabilizując stan logiczny sygnału w przypadku jego braku, lub ograniczać prąd, chroniąc układ przed uszkodzeniem w razie zwarcia.
2. **Kondensator 2,2 µF do masy (GND)**
   Kondensator eliminuje zakłócenia i stabilizuje napięcie, co zapobiega problemom podczas ładowania kodu.

5.2. Lista oraz opis poszczególnych komponentów:

- **ESP32-DevKitC-32E V4 WiFi + BT 4.2**

Platforma mikroprocesorowa wyposażona w moduł ESP-WROOM-32E. Oferuje wbudowane WiFi i Bluetooth 4.2, co umożliwia bezprzewodową komunikację z siecią i innymi urządzeniami.

- **Wyświetlacz dotykowy TFT LCD 2,8″ 240x320px**

Wyświetlacz dotykowy o przekątnej 2,8 cala i rozdzielczości 240x320 pikseli. Służy do interaktywnej obsługi urządzenia, wyświetlania czasu gry, opcji menu oraz innych istotnych informacji. Technologia TFT zapewnia wyraźny obraz i dobre kąty widzenia.

- **Kabel HDMI - HDMI NEDIS 1.5 m**

Standardowy kabel HDMI używany jako zbiór kabli do przesyłu i odbioru sygnałów z szachownicy.

- **Demultiplexer 3 na 8 linii 74LS138**

Układ scalony, który pozwala na wybór jednego z ośmiu wyjść na podstawie trzech sygnałów wejściowych. Umożliwia efektywne sterowanie wieloma liniami sygnałowymi przy użyciu mniejszej liczby przewodów sterujących.

- **Bramka NAND 74LS00**

Służy do negacji sygnału (zielony kabel). Sygnał zerojedynkowy dodany do zestawu sterującego (pomarańczowe kable) sprawia, że program rozróżnia strony szachownicy

- **Szachownica i figury szachowe**

Fizyczna szachownica i figury służące jako sensor. Figury wyposażone są w magnesy neodymowe, które współpracują z kontaktronami umieszczonymi pod szachownicą, pozwalając na precyzyjne wykrywanie ruchów.

- **Dioda Schottky'ego**

Dioda półprzewodnikowa charakteryzująca się niskim napięciem przewodzenia i dużą szybkością działania. W układzie zapobiega przepływowi sygnału w niepożądanym kierunku, eliminując błędy odczytu podczas wykrywania pozycji figur.

- **Konstrukcja przewodowa z wyłącznikiem układu**

Służy do zasilania oraz przesyłania danych między urządzeniami. Może być używany do podłączenia modułu ESP32 do komputera w celu programowania lub zasilania.

- **Zestyk kontaktron prosty 14mm**

Proste zestyki kontaktronowe, które zamykają obwód w obecności pola magnetycznego. Umieszczone pod polami szachownicy, umożliwiają wykrywanie pozycji figur szachowych wyposażonych w magnesy.

- **Rezystor justPi THT CF węglowy 1/4W 10kΩ**

Rezystory węglowe, używane do ograniczania prądu w obwodach. Chronią inne komponenty przed uszkodzeniem przez nadmierny prąd oraz służą do ustawiania odpowiednich poziomów sygnałów.

- **Magnes neodymowy okrągły 6x3mm**

Magnesy neodymowe, które są umieszczane w figurach szachowych. Oddziałują na zestyki kontaktronowe pod szachownicą, umożliwiając precyzyjne wykrywanie pozycji figur.

- **Powerbank Tracer 10000mAh**

Powerbank służący jako źródło zasilania.

Dodatkowo:

- **Przewody połączeniowe**
- **Obudowa wydrukowana w drukarce 3D**
- **Porty HDMI**
- **Adapter USB Typ C - Micro USB LANBERG AD-UC-UM-01**
- **Tact switch 6x6**

5.3. Kosztorys

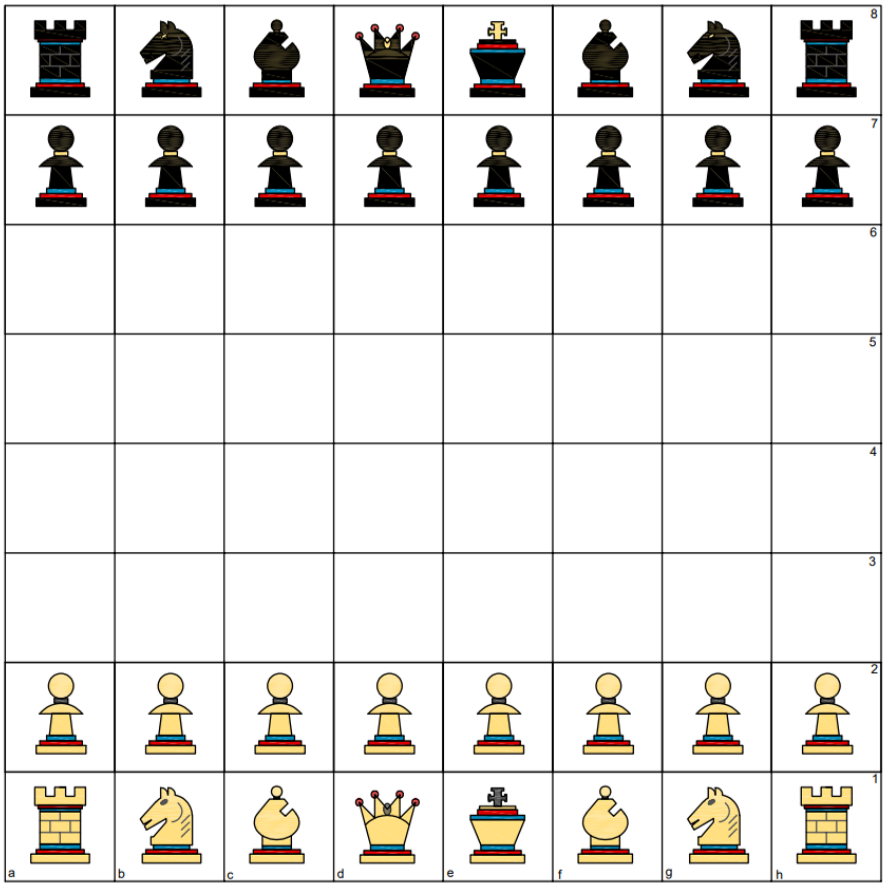
Nazwa:	Cena (w zł):	Ilość:
ESP32 - 38pin	59,99	1x
Wyświetlacz	40,00	1x
74LS138n	1,50	3x
74LS00n	6,00	1x
Zestaw figur	33,00	1x
Szachownica	59,00	1x
Diody	0,15	64x
Magnesy	0,55	34x
Kontraktony	0,85	64x
Wyłącznik układu	16,24	1x
Zestaw kabli	59,90	1x
Adapter do ESP32	16,00	1x
Adapter USB - B	6,99	1x
Obudowa	30,00	1x
Śrubki i nakładki	3,00	1x
Powerbank	50,00	1x
Płytka stykowa 170o.	3,50	2x
Porty HDMI	5,55	2x
SUMA:	501,12	

6. Projekt software'u

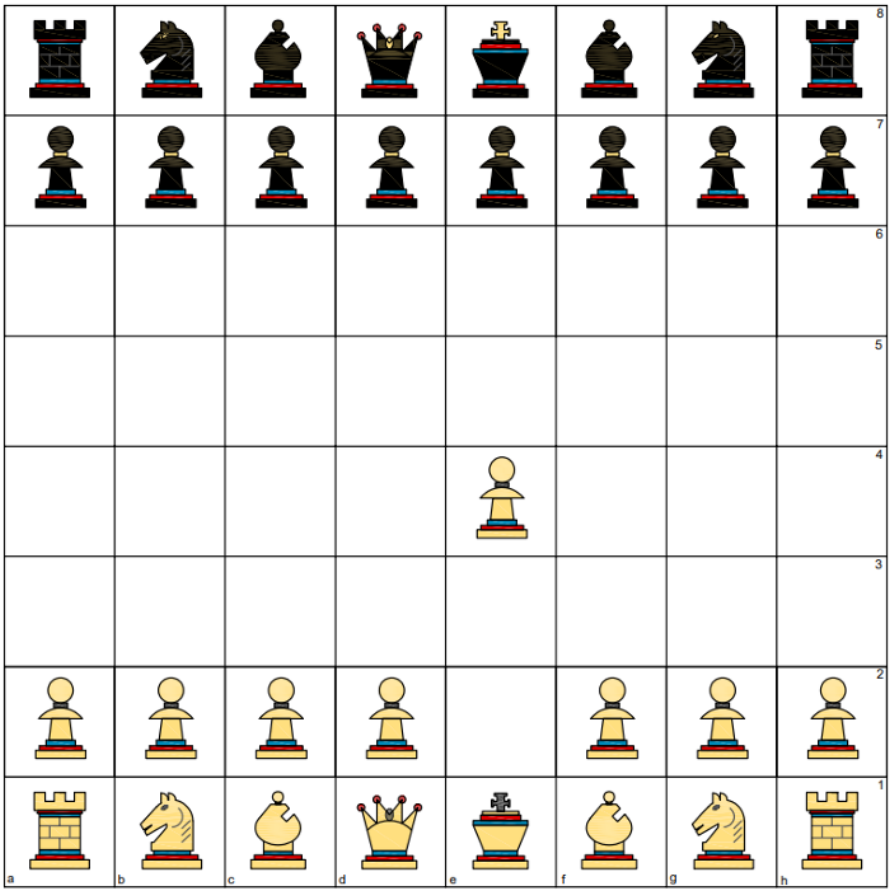
W systemie działa kilka kluczowych mechanizmów, które pozwalają na rozgrywanie partii, śledzenie ruchów, obsługę różnych stanów gry oraz interakcję z użytkownikiem poprzez menu. Poniżej znajduje się bardziej szczegółowy opis poszczególnych elementów i ich logiki:

6.1. Odczytywanie figur z szachownicy

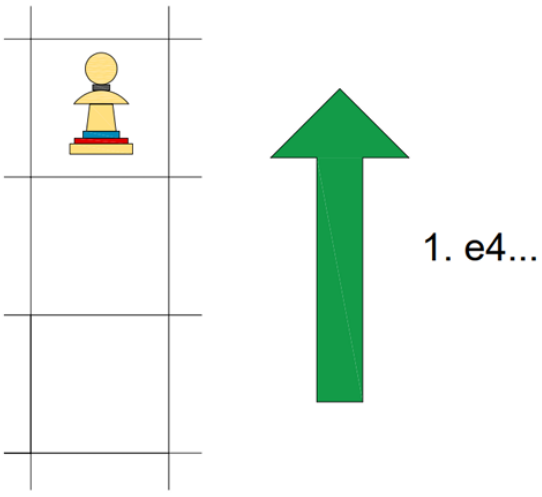
Szachownica jest wyposażona w kontraktony, które umożliwiają wykrywanie obecności figur na poszczególnych polach. Odczytywanie figur odbywa się poprzez skanowanie stanu szachownicy i porównywanie go z oczekiwanym stanem (Rys. 1-3.):



Sch. 5. Pierwotne ustawienie figur



Sch. 6. Ustawenie po wykonanym ruchu



Sch. 7. Porównanie stanów oraz dedukcja ruchu

Kluczowe funkcje związane z tym procesem to:

• **checkUpdatedSquaresForMoves()**: Ta funkcja sprawdza, które pola na szachownicy zostały zmienione od ostatniego odczytu. Na tej podstawie określa, jakie ruchy zostały wykonane. W zależności od liczby zmienionych pól rozpoznaje różne rodzaje ruchów (prosty ruch, bicie, promocja, en passant, roszada).

• **getChangedSquares()**: Funkcja ta zwraca listę pól, które zostały zmienione od ostatniego skanu. Jest to kluczowa funkcja do śledzenia zmian na szachownicy.

• **isPlayerPiece()** i **isOpponentPiece()**: Te funkcje sprawdzają, czy dana figura na określonym polu należy do gracza, czy do przeciwnika.

```
51
52  void checkUpdatedSquaresForMoves() {
53
54
55      int nbChangedSquares = getChangedSquares(modifiedSquares, serverBoardPieces);
56
57      userMoveTentative = Move();
58
59      for(int i = 0; i < nbChangedSquares; i++) {
60          Vector2i sqr = modifiedSquares[i];
61
62          if(isOpponentPiece(serverBoardPieces, sqr) &&
63              serverBoardPieces[sqr.x][sqr.y] > 0)
64          {
65              potentialCapturedSquare = modifiedSquares[i];
66          }
67      }
68
69      bool clearLeds = true;
70      if(isSimpleCaptureMove(nbChangedSquares))
71      {
72          Vector2i playerSquare = modifiedSquares[0];
73
74          char promotion = 0;
75          if(toUpper(getPieceChar(serverBoardPieces, playerSquare)) == 'P' &&
76              (potentialCapturedSquare.y == 0 || potentialCapturedSquare.y == 7))
```

Zdj. 1. Element kodu

6.2. Tworzenie ruchów w formacie UCI

UCI (Universal Chess Interface) to standardowy protokół komunikacyjny dla silników szachowych. System generuje ruchy w formacie UCI na podstawie wykrytych ruchów. Oto kluczowe elementy tego procesu:

• **squaresToMove()**: Ta funkcja tworzy obiekt ruchu (ang. Move) na podstawie pozycji początkowej i końcowej figury oraz ewentualnej promocji. Obiekt Move zawiera informacje potrzebne do zapisu ruchu w formacie UCI.

• **userMoveTentative.toString()**: Konwertuje obiekt ruchu na jego reprezentację tekstową w formacie UCI, która może być wyświetlona użytkownikowi lub wysłana do serwera.

6.3. Logika stanu gry

Stan gry jest zarządzany za pomocą zmiennej **gameState**, która może przyjmować różne wartości określające bieżący etap gry. Kluczowe etapy to:

• **GameState::Connecting**: Łączenie z serwerem lub przeciwnikiem.

• **GameState::NoGame**: Brak aktywnej gry.

• **GameState::GameFound**: Znalezienie przeciwnika i rozpoczęcie gry.

• **GameState::WaitingToReceiveOpponentMove**: Oczekiwanie na ruch przeciwnika.

• **GameState::UserMakingUserMove**: Użytkownik wykonuje swój ruch.

• **GameState::WaitingToReceiveUserMoveConfirmation**: Oczekiwanie na potwierdzenie ruchu użytkownika przez serwer.

Każdy stan jest odpowiednio obsługiwany przez wyświetlanie odpowiednich menu i komunikatów na ekranie, np. informowanie użytkownika o ruchu przeciwnika czy konieczności wykonania ruchu.

6.4. Obsługa interakcji z użytkownikiem

System obsługuje interakcje użytkownika poprzez menu wyświetlane na ekranie. Menu zmieniają się dynamicznie w zależności od stanu gry. Funkcja **mainLoop_Game()** jest główną pętlą sterującą grą i obsługującą wszystkie stany oraz interakcje użytkownika.

6.5. Zarządzanie czasem i zegarem

System śledzi czas pozostały dla obu graczy i odpowiednio aktualizuje wyświetlanie zegara. Zmienne takie jak **playerTimeLeft** i **opponentTimeLeft** są aktualizowane na bieżąco w zależności od aktywnego gracza.

6.6. Potencjalne ruchy i bicie

Funkcje takie jak **checkPotentialCaptureSquareWithSentMove()** oraz **checkUpdatedSquaresForMoves()** sprawdzają, czy wykonane ruchy są legalne i czy doszło do bicia figury przeciwnika. Na tej podstawie system może aktualizować stan gry i informować użytkownika o koniecznych działaniach, np. resetowaniu nielegalnego ruchu.

Cała logika sieciowa znajduje się w pliku **NetworkLoop.ino**.

```
void getPlayerStream() {
    bool success = false;
    while(!success) {
        httpClientStream->begin("https://lichess.org/api/stream/event");
        httpClientStream->addHeader("Authorization", (LongString("Bearer ") + lichessToken).c_str());
        int httpCode = httpClientStream->GET();
        Serial.println("player stream httpCode: " + String(httpCode));
        if(httpCode > 0) {
            if(httpCode == HTTP_CODE_OK) {
                // get tcp stream
                wifiClientStream = httpClientStream->getStreamPtr();

                gameState = GameState::NoGame;
                bGameStateDirty = true;

                success = true;
            }
        } else {
            // stream failed
        }

        delay(500);
    }
```

Zdj. 2. Główna pętla odpowiadająca na wyszukiwanie gry w pliku Network.ino

7. Analiza i wnioski:

7.1. Ocena rezultatów

Projekt z sukcesem połączył sprzęt i oprogramowanie do gry w szachy, umożliwiając śledzenie ruchów na fizycznej szachownicy oraz ich wyświetlanie w aplikacji. Interfejs użytkownika jest intuicyjny i wygodny. Niestety, tryb dla dwóch graczy (jeden z głównych celów) nie został zaimplementowany.

7.2. Analiza problemów i rozwiązań

Projekt napotkał trudności z kalibracją czujników oraz interpretacją ruchów, ale te problemy zostały rozwiązane dzięki testom i iteracyjnemu debugowaniu. Użycie ESP32 zewnętrznej anteny mogłoby poprawić stabilność sygnału Wi-Fi, co wpłynęłoby na niezawodność połączenia.

7.3. Możliwe usprawnienia i przyszłe kierunki rozwoju

Projekt może być rozwinięty poprzez dodanie opcji konfiguracji silnika szachowego, umożliwiającej dostosowanie go do potrzeb gracza. Dodatkowo, usprawnienie wyświetlania ruchów w notacji PGN, rozbudowa menu silnika szachowego oraz dodanie funkcji większej personalizacji menu (np. ustawienia partii, analiza statystyk) poprawiłoby doświadczenia użytkownika. Integracja z platformami społecznościowymi do udostępniania partii i prowadzenia rankingów to także możliwy kierunek rozwoju.

8. Bibliografia:

Filmy:

- https://youtu.be/ZDDEnBZZvmo?list=LL - Foldable Electronic Chess Board - an overview
- https://youtu.be/Qv0fvm5B0EM?list=LL - Code CHESS in JavaScript
- https://youtu.be/D84sVPR6g7o - Smart Chess board Arduino project

Artykuły:

- https://projecthub.arduino.cc/maguerero/automated-chess-board-67db6f - Automated Chess Board
- https://electronics.stackexchange.com/questions/551581/detecting-chess-pieces-on the-board - Detecting chess pieces on the board
- https://kcir.pwr.edu.pl/~witold/ip/2019_reports/BoczarJedrzej_IP_final_report.pdf - Chess pieces detecting – Politechnika Wrocławska

Materiały:

- https://github.com/omercier01/Chessboard/tree/main

Elementy CAD:

- https://www.bibliocad.com/en/library/chess-pieces_59883/ - Figury szachowe

Poza materiałami korzystałem z konsultacji mailowych z twórcą oprogramowania. Oprogramowanie zostało przeze mnie przerobione na własny użytek. Oprogramowanie zawiera MIT Licence, co umożliwia dokonanie edycji oraz skorzystania z tego programu zgodnie z prawem. Dodatkowo korzystałem z narzędzia ChatGPT.