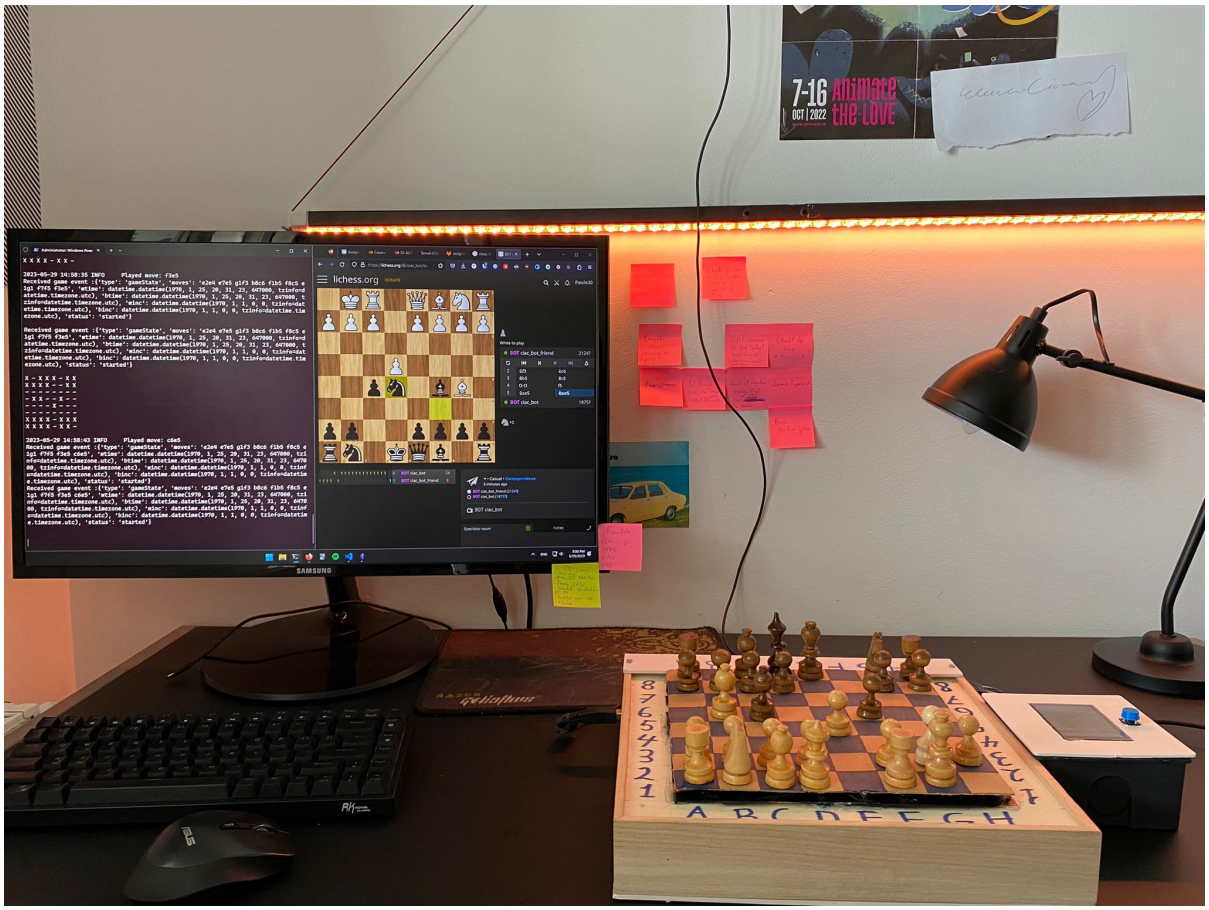
C.I.A.C - (Li)Chess Interface with Arduino Control

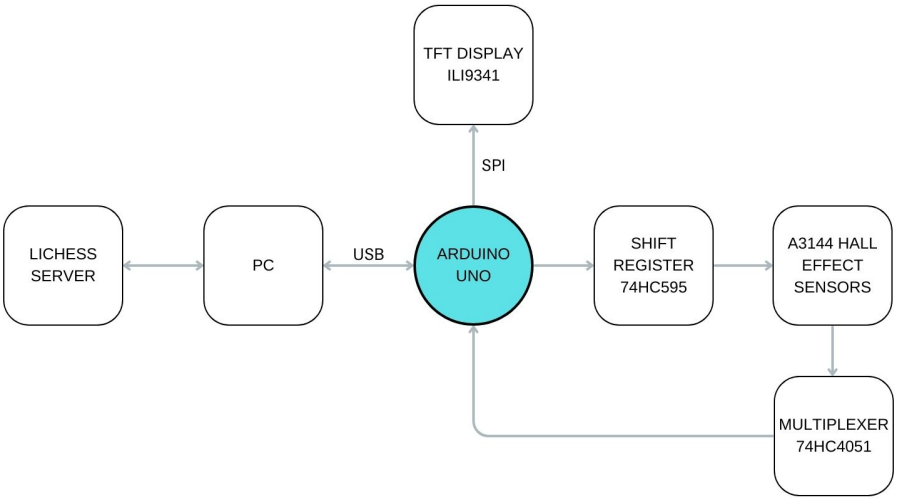
Introducere



Proiectul presupune realizarea unei table de șah care poate fi conectată printr-un port USB la calculator pentru a juca meciuri online sau OTB(over the board) cu un prieten folosind API-ul oferit de Lichess.

Am vrut să fac un proiect care să se alinieze cu unul din hobby-urile mele la momentul de față, ceva pe care să-l pot folosi ulterior ca să exersez sau ca să pierd timpul. Întamplarea a făcut ca semestrul acesta să mă prindă in *febra șahului*, iar asta mi-a dat ideea de a face un proiect care să combine satisfacția de a juca șah pe o tablă fizică cu utilitatea unui **chess engine**.

Descriere generală



Una din funcționalitățile principale ale tablei este detectarea pieselor folosind **senzori magnetici** și transmiterea datelor de la Arduino la calculator pentru procesarea mutărilor. Sub fiecare pătrat al tablei de șah se află un **senzor hall A3144** așezat pe o bucată mică de metal, mai exact ~~o moneda de 10 bani~~ o piulita. În momentul în care o piesă este așezată pe tablă, magnetul de la baza ei magnetizează moneda, activând astfel senzorul. Citirea tablei se realizează prin activarea pe rând a fiecărei coloane cu un **shift register** și citirea datelor de pe fiecare linie cu un **multiplexor** (vezi **schematic**).

Interacțiunea cu utilizatorul se va realiza cu ajutorul unui ecran TFT cu touchscreen în felul următor:

1. Utilizatorul poate selecta modul de joc (vs Stockfish, online sau OTB)
2. Se va afișa timpului rămas fiecărui jucător
3. În cazul în care se joacă online, ecranul va afișa mutarea făcută de oponent în notație algebrică (ex. E4, NxF6, etc.)
4. În cazul în care mutarea efectuată de un jucător nu este legală, se va afișa configurația anterioară a tablei

Hardware Design

Hardware Schematic

Materiale folosite

- Senzori Hall A3144
- Arduino Uno

- Shift register 74HC595
- Magneti neodim 10×2
- Ecran touchscreen TFT ILI9341
- Multiplexor 8 canale 74HC4051
- Test wiring
- Male to female cables
- Cablu
- Condensatoare, rezistente, etc.
- 8 tranzistoare BC337

Software Design

Toata partea de dezvoltare software pentru Arduino a fost facuta in Visual Studio Code folosind extensia PlatformIO, iar pentru partea de comunicare cu serverul de Lichess am folosit Python intr-un environment de Conda.

Biblioteci folosite pe Arduino

- Adafruit_GFX
- SPI
- Adafruit_ILI9341
- UCGLIB

Module folosite in script-ul de Python

```
python==3.11.3
berserk==0.12.4
CairoSVG==2.7.0
chess==1.9.4
ipython==8.13.2
opencv_python==4.7.0.72
Pillow==9.5.0
pyserial==3.5
```

Functionalitati implementate pe Arduino

Am incercat sa impart logica programului in cat mai multe fisiere sursa ca sa am un program cat de cat modular. Deoarece am scris aproape 1000 de linii de cod(si INCA nu e gata), din respect pentru timpul persoanei care va citi asta si din ratiuni de economie pentru timpul meu, o sa ofer mai jos doar un overview asupra functionalitatilor implementate in fiecare fisier sursa, fara sa detaliez explicit fiecare functie.

- mux.cpp
  - Initializeaza multiplexorul.
  - Defineste functii necesare activarii si citirii intrarilor.
- shift_reg.cpp
  - Initializeaza shift register-ul.
  - Defineste functii necesare setarii de valori in registrele interne ale modului si scoaterii de date la output pe latch.
- timer.cpp
  - Initializeaza timerul 1 pentru a activa intreruperi o data la 1s.
  - Defineste logica de intrerupere si o functie se updateaza timpul jucatorilor.
- tft.cpp
  - Implementeaza metodele definite in clasa numita TFT si initializeaza display-ul
  - Realizeaza afisarea ultimei mutari pe ecran in format **UCI**
  - Afiseaza tabla de sah asa cum apare ea pe Lichess folosind in loc de imagini caractere(P,N,B, etc.). Ca o mica paranteza, am incercat sa afisez bitmap-uri ca sa arate totul mai frumos doar ca dureaza cam mult si am vrut sa pastrez afisarea cat de cat *instantanee* (dar cu ocazia asta am aflat ce face **PROGMEM** si cum se iau datele din memoria Flash).
  - Defineste functii pentru afisarea timpului ramas
- serial.cpp
  - Aici am definit niste functii ajutatoare pentru citirea de date de pe interfata seriala si pentru trimiterea de *pachete* catre calculator.
- main.cpp
  - Defineste si initializeaza toate variabilele globale folosite in program(flag-uri, pini, array-ul cu tabla de sah, buffere, etc.)
  - Parseaza si apeleaza functiile necesare in urma citirii datelor de pe interfata seriala. Fiecare mesaj este identificat printr-un byte de start care descrie ce actiune trebuie sa aiba loc(de exemplu, un mesaj care incepe cu **B** va contine noua configuratie a tablei ce trebuie afisata pe Arduino). Comenzile sunt delimitate folosind '\n'.
  - In urma apasarii unui buton, citeste si trimite configuratia tablei fizice catre script-ul de Python.

Functionalitati implementate in script-ul de Python

- class Reader(threading.Thread)
  - Primeste si trimite date catre Arduino
  - In momentul de fata, deoarece nu am reusit sa fac modulul cu touchscreen XPT2046 sa functioneze, singurele mesaje pe care le poate primi de la Arduino sunt cele cu configuratia tablei fizice.
  - Defineste functiile necesare pentru validarea mutarilor (rocada, **en passant**, mutare catre un spatiu gol, mutare care promoveaza un pion, luarea de piese). De mentionat aici este faptul ca, din cauza naturii imprevizibile a senzorilor magnetici, am luat decizia de a nu citi tabla decat la apasarea unui buton, drept urmare ar fi existat unele cazuri in care mi-ar fi fost imposibil sa determin ce piesa a fost luata de pe tabla. Solutia ideala ar fi fost sa citesc in mod constant tabla si sa ma folosesc de 2 stari pentru a determina ce piesa a disparut. Solutia simpla si realista, care merge in 100% de cazuri, a fost sa adaug o noua regula in conduita nescrisa a sahului: *Atunci cand un jucator vrea sa ia o alta piesa, acesta trebuie sa ridice ambele piese de pe tabla si sa apese pe buton.*
  - Afiseaza mesaje de logging pentru debug (mutare invalida, ce senzori s-au citit, etc.)
  - Defineste functii prin care se trimit date la Arduino(ultima mutare, pozitia curenta a tablei,

timpul jucatorilor, etc.)
- class Game(threading.Thread)
  - Defineste functiile necesare handle-ului minimal de evenimente care pot aparea intr-un meci de Lichess (opponent moved, game ended, game aborted,
  - Foloseste o referinta la un obiect de tip Reader pentru a comunica cu Arduino-ul. Tot aici se realizeaza si trimiterea mutarilor dupa validare la Lichess.
- Main Thread
  - La incepului rularii scriptului se creeaza clientul, numit in continuare **ciac_bot**, si un thread Reader care ruleaza ca **daemon**.
  - Se asculta eventuri de tip new challenge sau gameStart si se creeaza threaduri noi de tip Game in functie de gamemode-ul selectat. Gamemode-ul default e online, dar pot juca si doua persoane simultan pe aceeasi tabla, caz in care se creeaza un challenge intre **ciac_bot** si **ciac_bot_friend** in care se inregistreaza meciul pentru a fi analizat ulterior pe calculator.

Rezultate Obţinute si concluzii

Link catre varianta prezentata la PM fair.

Am invatat ca de multe ori ceea ce pare usor pe hartie se dovedeste a fi un chin in realitate. Pe parcursul proiectului, a trebuit sa-mi schimb de multe ori abordarea pentru ca de multe ori am descoperit ca ideile mele nu functioneaza asa cum m-as fi asteptat. Am improvizat mult si am fost nevoit sa renunt la multe lucruri pe care as fi vrut sa le adaug la proiect din cauza lipsei de timp.

Cele mai multe probleme le-am avut cu senzorii magnetici pentru ca uneori citeau prost, iar alteori nu citeau deloc. De asta, am fost nevoit sa schimb o parte din ei si sa improvizez un capac din carton pentru tabla pentru ca nu reuseam sa citesc intr-un mod consecvent senzorii prin materiale putin mai groase.

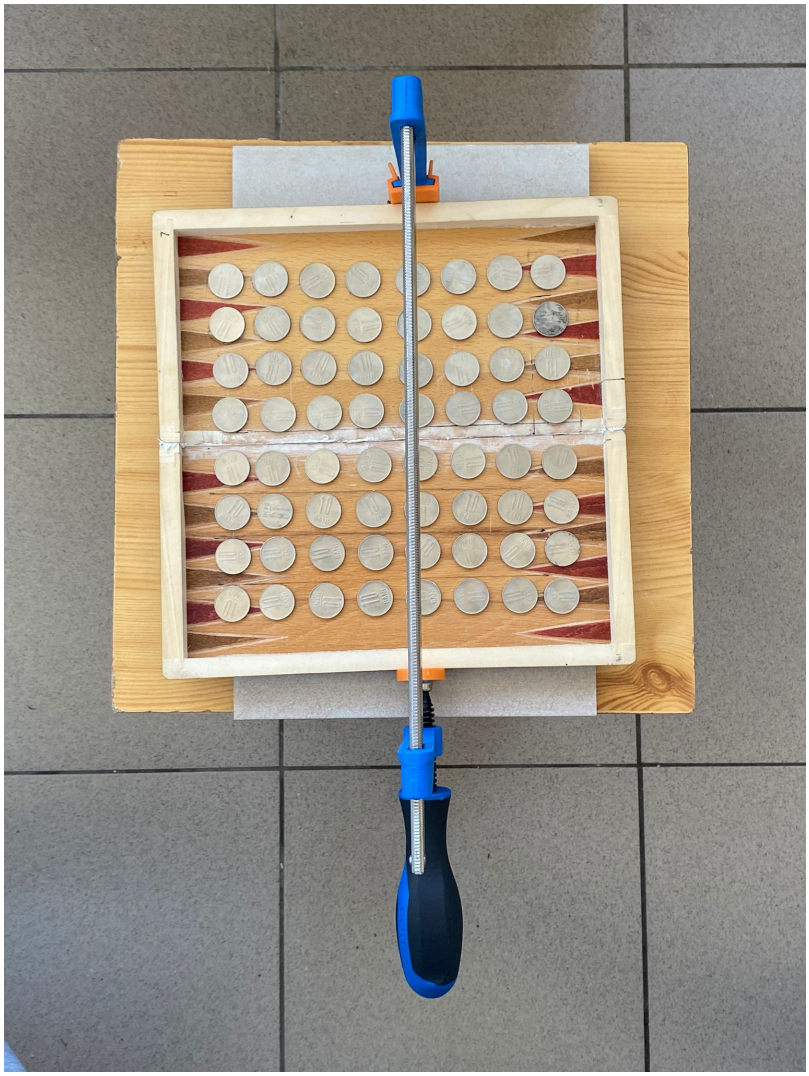
In final, desi am avut parte de multe dezamagiri de-alungul proiectului, ma simt foarte bine ca am reusit sa-l duc la capat. Satisfactia de a realiza ceva functional cu mainile tale e un lucru greu de echivalat.

Download

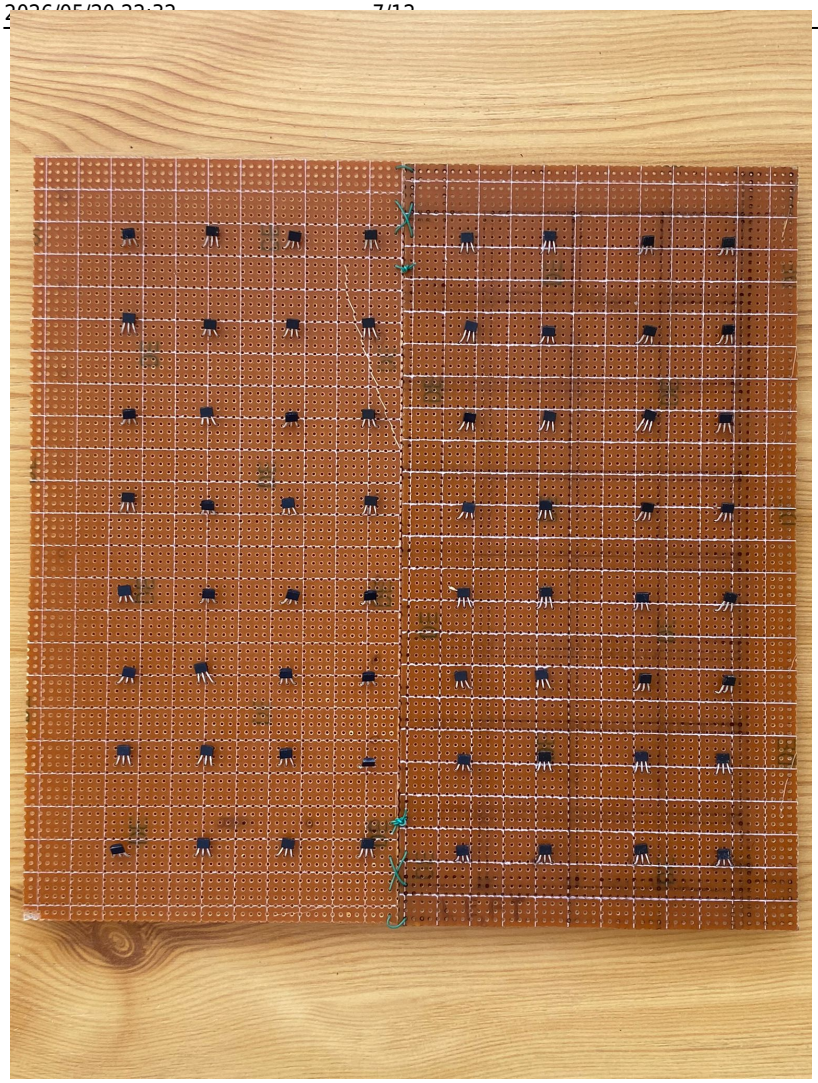
- Link Repo

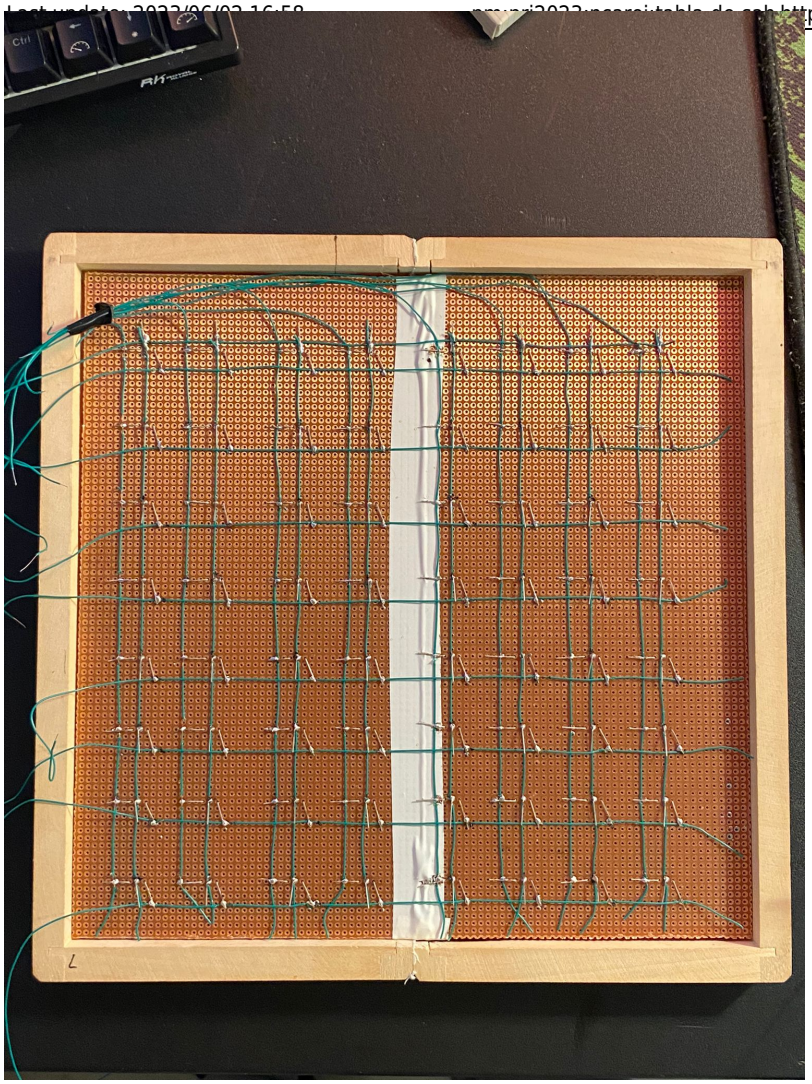
Jurnal

- 29-04-2023: Am testat logica de citire a senzorilor pe un breadboard cu cativa senzori.
- 30-04-2023: Am inceput sa lucrez la capacul pentru tabla de sah si m-am taiat atat de tare cu un cutter de la Dexter incat am ajuns la urgente. todo: de adaugat betadina la lista de materiale.



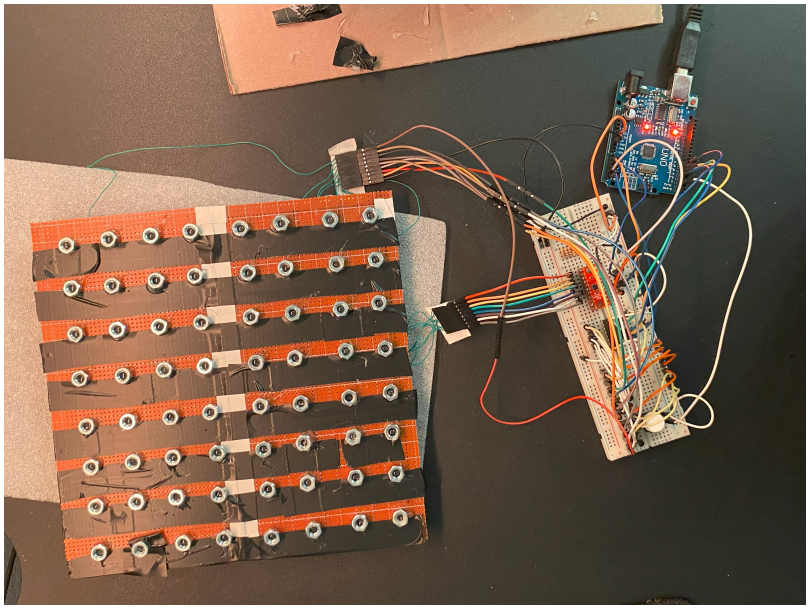
- 02-05-2023 - 05-05-2023: Am terminat de realizat cablajul pentru senzori si am invatat sa lipesc.

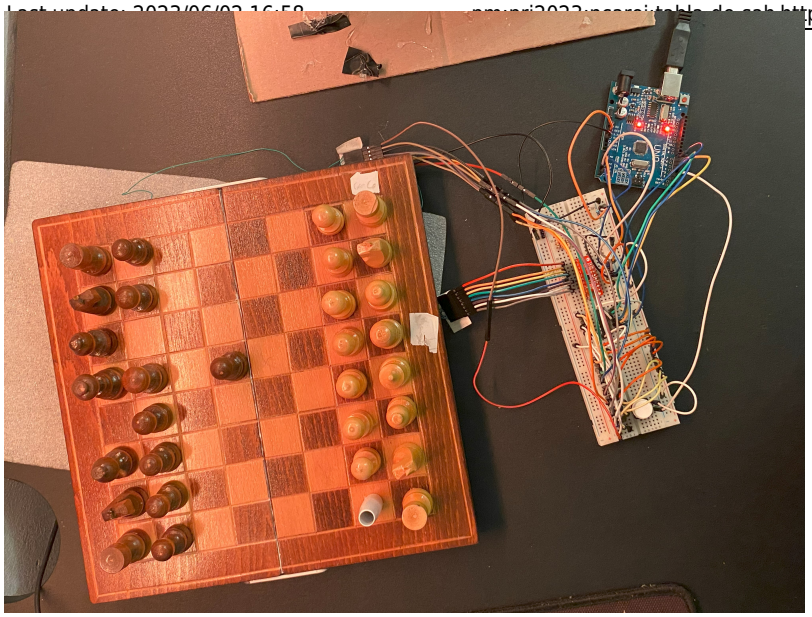




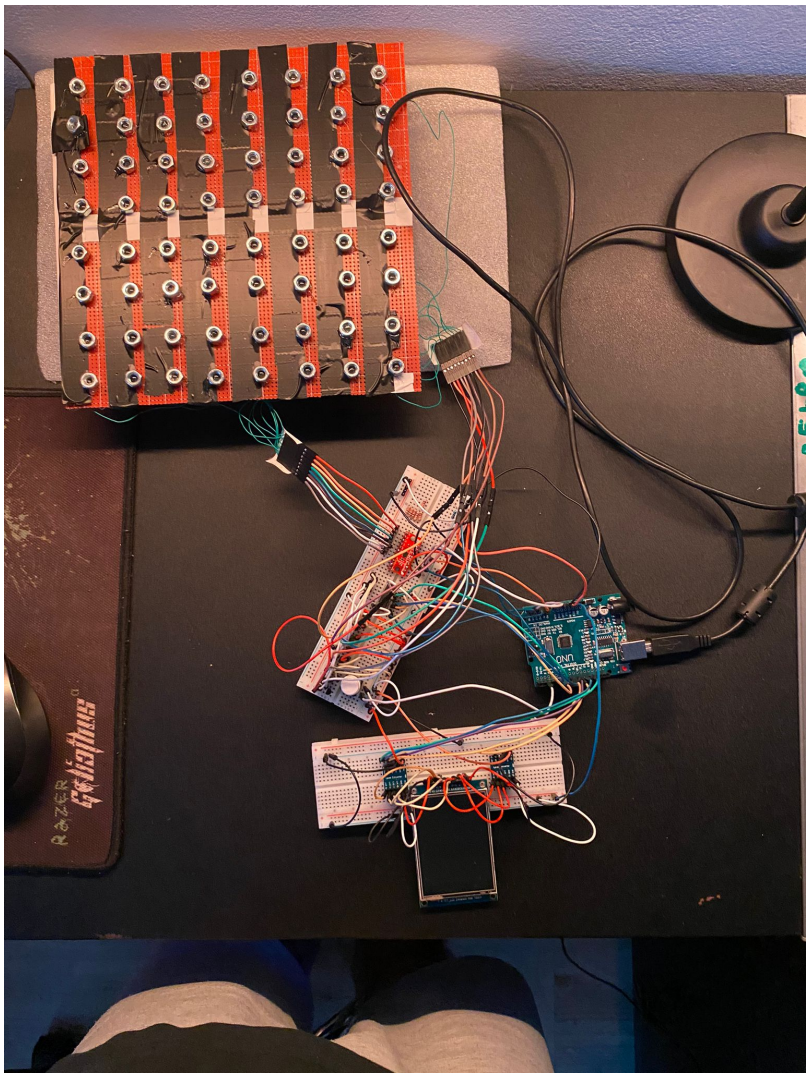
- 16-05-2023: Am mai facut niste DIY ca sa bag magnetii in piese azi si am inlocuit toate monezile cu niste piulite. Uneori se prindeau piesele intre 2 banuti, asa ca a trebuit sa caut o alternativa cu o suprafata mai mica.

- 19-05-2023: Am terminat de realizat partea de hardware dupa mult debuggind la senzori:





- 21-05-2023 - 24-05-2023: Am facut comunicarea cu serverul de Lichess si afisarea pe ecranul TFT.



- 29-05-2023: Nu puteam citi toti senzorii prin tabla de sah pe care voiam sa o folosesc initial, asa ca a trebuit sa-mi confectionez alta cutie. Astazi marchez sfarsitul lucratului la tabla de sah.

Resurse

Resurse Software

- tutorial 74HC595
- tutorial MUX sparkfun
- Lichess API documentation
- Python chess documentation
- Berserk documentation

Resurse Hardware

- datasheet mux
- Documentatie 74HC595
- Datasheet senzori Hall Effect A3144

Export to PDF