VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ELEKTRONICKÉ ŠACHY

ELECTRONIC CHESS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

Matúš Kočka

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Ondřej Baštán

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Matúš Kočka ***ID:*** 230269

Ročník: 3 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Elektronické šachy

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a vytvořit elektronické šachy.

1. Zpracujte literární rešerši na téma elektronické šachy.
2. Definujte požadavky na elektronické šachy.
3. Navrhněte koncepci řešení elektronických šachů.
4. Navrhněte a realizujte hardware pro elektronické šachy.
5. Implementujte řídicí software pro elektronické šachy.
6. Vytvořte dokumentaci a zhodnoťte dosažené výsledky.

DOPORUČENÁ LITERATURA:

GAJSKI, Daniel D., et al. Embedded system design: modeling, synthesis and verification. Springer Science & Business Media, 2009.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 22.5.2023

Vedoucí práce: Ing. Ondřej Baštán

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

ABSTRAKT

Táto bakalárska práca je zameraná na funkciu kapacitných senzorov na doske plošných spojov a radu meraní, z ktorých bolo následne možné vytvoriť senzory pre šachovnicu. Ďalej sa zaoberá technikou CoreXY, podľa ktorej je zhotovený návrh pohybového mechanizmu v programe Siemens NX a jeho následná realizácia. Zariadenie obsahuje mikrokontrolér, pre ktorý je vytvorený software pre spracovanie dát zo senzorov a ovládanie krokových motorov.

KĽÚČOVÉ SLOVÁ

šachy, kapacitný senzor, ESP32, MicroPython, AD7147, NEMA17, CoreXY, šachový software, TCA9548A, Siemens NX

ABSTRACT

This bachelor thesis is focused on function of capacitive sensors on a printed circuit board and a series of measurements from which it was possible to create a subsequent sensors for the chess board. It also deals with the CoreXY technique, according to which a design of a motion mechanism is made in the Siemens NX program, and its subsequent implementation. The device includes a microcontroller for which software is created to process data from sensors and control stepper motors.

KEYWORDS

chess, capacitive sensor, ESP32, MicroPython, AD7147, NEMA17, CoreXY, chess engine, TCA9548A, Siemens NX

KOČKA, Matúš. *Elektronické šachy*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2023, 64 s. Bakalárska práca. Vedúci práce: Ing. Ondřej Baštán

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Matúš Kočka
VUT ID autora:	230269
Typ práce:	Bakalárska práca
Akademický rok:	2022/23
Téma závěrečnej práce:	Elektronické šachy

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rád by som poďakoval vedúcemu bakalárskej práce Ing. Ondřejovi Baštánovi za odborné vedenie, konzultácie, a podnetné návrhy k práci. Taktiež by som rád poďakoval svojej rodine za podporu. A najmä svojmu otcovi za veľkú pomoc.

Obsah

Úvod **12**

1 Teoretický úvod **13**
1.1 Kapacitné senzory na doske plošných spojov 13
1.1.1 Princíp senzora priblíženia 13
1.1.2 Tienenie elektrického poľa 14
1.1.3 Topológie kapacitných senzorov 15
1.2 Operácia snímača pomocou kontroléra AD7147 [4] 17
1.2.1 Zaznamenanie a aktivácia senzoru 17
1.2.2 Operačné módy 17
1.2.3 Environmentálna kalibrácia 18
1.3 CoreXY [6] 18
1.3.1 Výhody techniky CoreXY 19
1.3.2 Nevýhody techniky CoreXY 20
1.3.3 Alternatíva H-Bot 20

2 Koncepcia riešenia práce **21**
2.1 Snímanie figúrok 21
2.1.1 Snímanie pomocou kapacitného senzora 22
2.2 Mikrokontrolér 22
2.3 Pohybový mechanizmus 23

3 Návrh Hardvéru pre Elektronické šachy **25**
3.1 Návrh kapacitných senzorov 25
3.1.1 Výsledky meraní rôznych variácií kapacitných senzorov 25
3.1.2 Otestovanie funkčnosti AD7147 [4] 30
3.1.3 Návrh Šachovej dosky so senzormi 34
3.2 Prenosný zdroj napájania 36
3.3 Hardvér Elektronických šachov 37
3.3.1 Návrh mechanizmu elektronických šachov 37
3.3.2 Zapojenie riadiacej jednotky s komponentami 39

4 Riadiaci softvér elektronických šachov **41**
4.1 Program pre spracovanie dát zo senzorov 41
4.2 Šachový softvér 43
4.3 Program pohybového mechanizmu 44

Závěr **48**

Literatúra **50**

Zoznam symbolov a skratiek **52**

Zoznam príloh **53**

A Návrh skúšobnej DPS kapacitných senzorov **54**

B Návrh skúšobnej DPS pre AD7147 **56**

C Návrh šachovej dosky so senzormi **57**

D Kód pre Elektronické šachy **63**

Zoznam obrázkov

1.1 Elektrické pole medzi dvomi elektródami [2] 13
1.2 Ako GND elektróda ovplivňuje elektrické pole senzora [3] 14
1.3 Princíp tieniacej elektródy [2] 15
1.4 Topológie senzorov [5] 15
1.5 Centrálna zem,centrálny senzor symetria [5] 16
1.6 Hrebeňový dizajn senzora [5] 16
1.7 Prahy aktivácie senzora [4] 17
1.8 Referenčný mechanizmus CoreXY [6] 19
1.9 mechanizmus H-Bot 20
2.1 ESP32 DEVKIT V1 [7] 23
2.2 Karteziánsky portálový mechanizmus [12] 24
2.3 CoreXY mechanizmus [6] 24
3.1 Schéma zapojenia kruhového senzora 26
3.2 Hrebeňový a obdĺžnikový senzor 27
3.3 Testovanie senzorov na DPS 29
3.4 Schéma DPS pre odskúšanie AD7147 32
3.5 Testovanie funkčnosti AD7147 33
3.6 ID kontroléra AD7147 [4] 33
3.7 Zvolenia kanálu na multiplexeri TCA9548A [13] 35
3.8 Navrhnutá šachovnica pre snímanie figúrok 35
3.9 Zapojenie Batérií k ochrannému obvodu BMS-40A-3S [15] 36
3.10 Zdroj napájania pre Elektronické šachy 37
3.11 Model elektrických šachov 37
3.12 Model mechanizmu 38
3.13 Mechanizmus Elektronických šachov 39
3.14 Schéma zapojenia riadiacej jednotky s výstupnými perifériami 40
4.1 Nastavenie citlivosti,offsetu a kalibrácie prahov v BANK 2 42
4.2 Zápis statusu senzorov do poľa prvkov pre 1. kontrolér AD7147 43
4.3 Počiatočná pozícia šachového softvéru 43
4.4 Funkcia EngineMove 44
4.5 Obyčajný ťah 44
4.6 Vyhadzovanie 45
4.7 Rošáda 46
4.8 Elektronické šachy ako celok 47
4.9 Elektronické šachy z vnútra 47
A.1 Schéma skúšobnej DPS kapacitných senzorov 54
A.2 DPS skúšobnej dosky TOP 54

A.3 DPS skúšobnej dosky BOTTOM 55
A.4 DPS skúšobnej dosky Tsilkscreen 55
A.5 TOP,BOTTOM DPS skúšobná doska 55
B.1 skúšobná DPS pre AD7147 (TOP) 56
B.2 skúšobná DPS pre AD7147 (BOTTOM) 56
B.3 skúšobná DPS pre AD7147 (FSILK) 56
C.1 Návrh Šachovnice so senzormi Schéma 57
C.2 Návrh Šachovnice so senzormi TOP 58
C.3 Návrh Šachovnice so senzormi BOTTOM 59
C.4 Návrh Šachovnice so senzormi FSILK 60
C.5 DPS Šachovnica so senzormi TOP 61
C.6 DPS Šachovnica so senzormi BOTTOM 62
D.1 Ťah jazdca 63
D.2 While cyklus 1.časť 64
D.3 While cyklus 2.časť 64

Zoznam tabuliek

3.1 Namerané dáta pre kruhový senzor 26
3.2 Dielektrické konštanty materiálov 27
3.3 Dáta namerané pre **obdĺžnikový senzor** 28
3.4 Dáta namerané pre **hrebeňový senzor** 28
3.5 Dáta namerané pre rôzne konfigurácie tieniacej elektródy 29
3.6 Dáta namerané pre 2mm vrstvu dreva medzi senzorom a predmetom 29
3.7 Rozdiely v (%) od nečinného stavu po priblížení predmetu z tabuľky 3.5 29
3.8 Rozdiely v (%) od nečinného stavu po priblížení predmetu pre 2mm vrstvu dreva z tabuľky 3.6 30
3.9 Výsledok kapacitno-digitálneho prevodníka po položení figúrky 33
3.10 Výsledok prevodníka prevedený na zmenu kapacity po položení figúrky 34

Úvod

Šach je jedna z najpopulárnejších doskových hier na svete, vyžaduje veľa plánovania a strategického myslenia, avšak v dnešnej dobe sa človek nevie vyrovnať výpočetnej sile šachových algoritmov. Dá sa povedať, že šach bol pre počítače ako stvorený a práve preto v posledných rokoch rastie záujem o vývoj systémov na báze umelej inteligencie, ktoré dokážu hrať šach na veľmi vysokej úrovni. Avšak pre človeka majú tieto systémy jednu nevýhodu a tou je, že sú čisto založené na softvéri a neposkytujú fyzickú reprezentáciu hry, čo je dôležitým aspektom zážitku zo šachovej partie. A tak keď si chce človek porovnať sily s počítačom musí hľadieť buď do monitora alebo do telefónu. Touto bakalárskou prácou som sa rozhodol tento problém vyriešiť a priniesť trocha fyzického aspektu do hry proti počítaču. Elektronické šachy budú teda fungovať tak, že ťah užívateľa bude zaznamenaný kapacitnými senzormi, ktoré budú umiestnené pod každým políčkom, dáta o tomto ťahu budú spracované mikrokontrolérom a zaznamenané šachovým softvérom, po tom čo softvér vygeneruje svoj vlastný ťah, pošle mikrokontrolér príkaz krokovým motorom aby vykonali ťah počítača a keďže každá figúrka bude mať v sebe zabudovaný magnet, bude možné pomocou elektromagnetu umiestneného pod šachovnicou túto figúrku presunúť na požadovanú pozíciu. A teda cieľom tejto práce bude navrhnúť vhodné kapacitné senzory, ktoré budú schopné figúrku snímať, ďalej bude treba navrhnúť a skonštruovať pohybový mechanizmus, ktorý bude skrytý pod šachovnicou. A nakoniec návrh softvéru pre spracovanie údajov zo senzorov a pre ovládanie pohybového mechanizmu.

1 Teoretický úvod

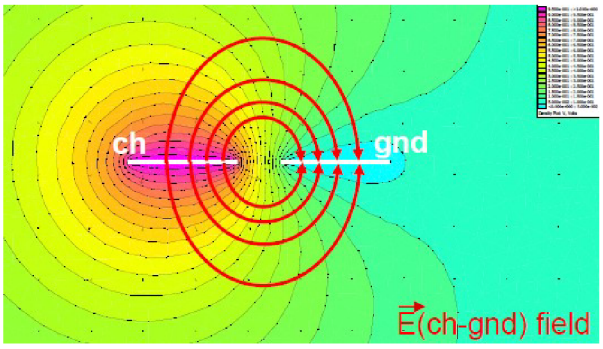
V tomto teoretickom úvode si priblížime fungovanie kapacitných senzorov na DPS, ďalej si povieme niečo o funkciách kapacitno-digitálneho prevodníku (CDC), ktorý pre tento projekt použijeme a nakoniec preberieme metódu pohybového mechanizmu CoreXY.

1.1 Kapacitné senzory na doske plošných spojov

Kapacitné snímanie je spôsob, ktorým môžeme určiť prítomnosť predmetu vďaka zmene kapacity, ktorá nastane pri jeho priblížení. Táto kapitola nám vysvetľuje fungovanie a implementácie kapacitných senzorov na DPS.

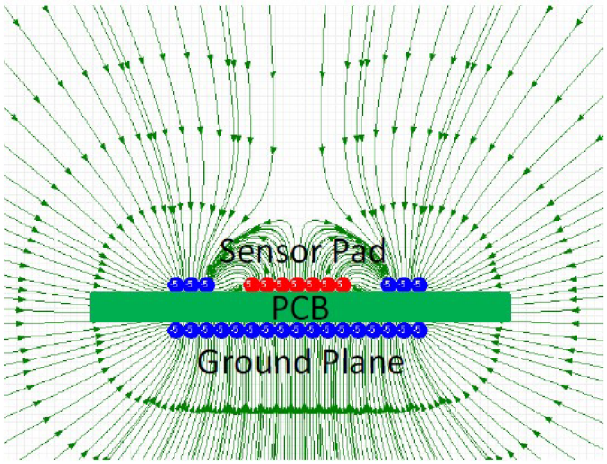
1.1.1 Princíp senzora priblíženia

Keď zoberieme elektródy kondenzátora, medzi ktorými je elektrické pole a položíme ich na ležato vedľa seba, tak elektrické pole bude stále prítomné, ale v inom tvare. Viz obrázok: 1.1.



Obr. 1.1: Elektrické pole medzi dvomi elektródami [2]

Keď priblížime neuzemnený predmet s permitivitou väčšou ako je permitivita vzduchu do blízkosti elektrického poľa, ako napríklad šachovú figúrku, tak týmto zmeníme dielektrikum medzi elektródami a môžeme sledovať nárast kapacity podľa vzorca: 1.1. Túto zmenu v kapacite senzora je možné následne registrovať pomocou prevodníka (Kapacitno-Digitálny prevodník).



Obr. 1.2: Ako GND elektróda ovplivňuje elektrické pole senzora [3]

$$C = \frac{\varepsilon_r \times \varepsilon_0 \times A}{d} \tag{1.1}$$

kde je:
A plocha dvoch elektród [m]
ε_r je dielektrická konštanta materiálu
ε_0 je permitivita voľného priestoru $8,85 \times 10^{-12}[F/m]$
d je vzdialenosť medzi elektródami [m]

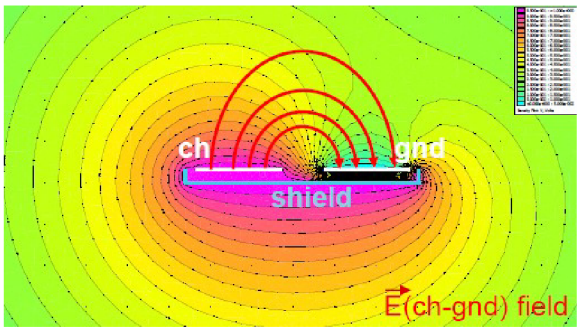
Vo voľnom priestore môže elektróda senzora emitovať elektrické pole voľne do všetkých smerov s malým útlmom. Ale keď k nej priblížime do určitej vzdialenosti zemniacu elektródu (GND), Vektory elektrického poľa emitujúce zo senzora majú tendenciu skončiť na GND elektróde.[3] Na obrázku 1.2 môžeme vidieť, ako sa chová elektrické pole senzora, keď k nemu priblížime zemniacu elektródu.

Zmena v kapacite pri senzore priblíženia bude o veľa menšia, ako pri senzore dotyku a preto si vyžaduje oveľa dôkladnejší ohľad na jeho dizajn.

1.1.2 Tienenie elektrického poľa

Pri návrhu kapacitného senzora musíme dbať aj na nasmerovanie elektrického poľa do priestoru, v ktorom chceme danú zmenu kapacity merať, ale aj na tienenie nechcených vonkajších vplyvov, ktoré by mohli mať zlý dopad na snímanie predmetu.

Toto je možné dosiahnuť umiestnením **tieniacej elektródy**, jej princíp je taký, že na tejto elektróde je vytvorený rovnaký signálový priebeh, akým je excitovaná snímacia elektróda a je tiež privedená na rovnaký potenciál. Tým pádom medzi týmito elektródami nie je žiadny prúd a akákoľvek kapacita vytvorená medzi nimi



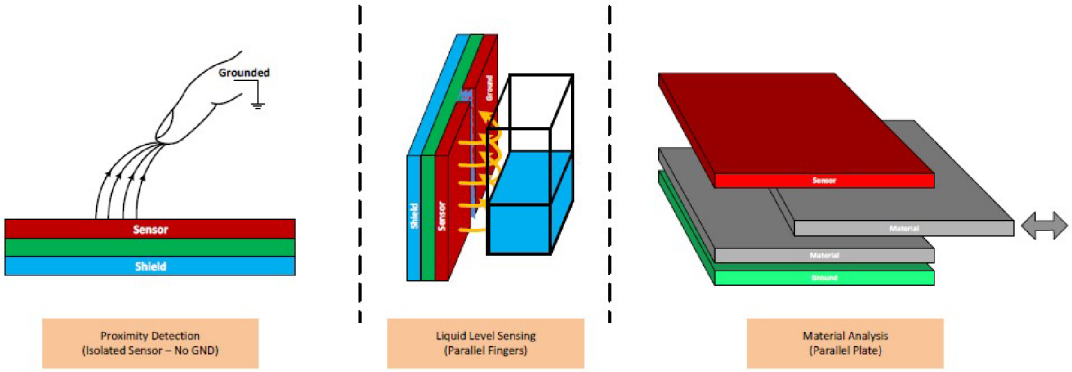
Obr. 1.3: Princíp tieniacej elektródy [2]

neovplyvní prenos elektrického náboja snímacej elektródy.[4] Na obrázku č.1.3 vidíme, ako vplýva tieniaca elektróda na elektrické pole snímača.

1.1.3 Topológie kapacitných senzorov

Podľa správy o aplikácii [5] existujú 3 hlavné typy topológií kapacitných snímačov viz. 1.4:

- Izolovaný senzor
- "Paralelných prstov"(Parallel Fingers)
- Paralelná doska



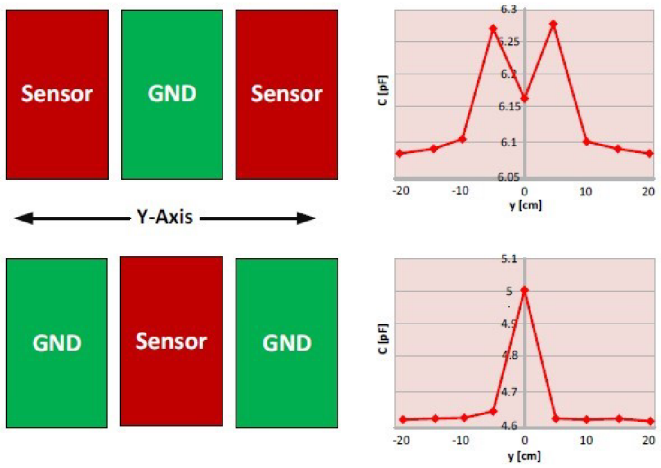
Obr. 1.4: Topológie senzorov [5]

V tejto práci, keďže sa jedná o snímanie neuzemnených predmetov, použijeme topológiu Paralelných prstov.

Topológia Paralelné prsty

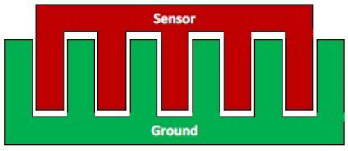
Tento návrh senzora sa väčšinou používa pri aplikáciách, kde sa meria hladina vody. Elektrické pole je viacej dominantné v blízkosti okrajov medzi senzorom a uzemnenou elektródov. Vypočítanie kapacity nie je jednoduché tak, ako to je v prípade paralelných dosiek viz. 1.1, ale senzitivita senzora sa nelineárne zväčšuje so zväčšujúcou sa veľkosťou senzora.

Existuje viacej variácií implementácie senzorov podla tejto topológie. Viaceré senzorové a zemniace elektródy môžu byť vystriedané, tak aby bola v strede zemniaca alebo senzorová elktróda viz.1.5



Obr. 1.5: Centrálna zem,centrálny senzor symetria [5]

Centrálna zem je vhodná použiť, keď je treba mať čo najširšiu odozvu naopak centrálny senzor sa použije v prípade, kedy potrebujeme ostrú úzku odozvu. Kompromisom týchto dvoch implementácií je tzv. **Hrebeňová konfigurácia (Comb configuration)**, táto konfigurácia obsahuje to najlepšie z obidvoch svetov poskytuje veľkú plochu snímania spolu s veľkou senzitivitou viz. 1.6.



Obr. 1.6: Hrebeňový dizajn senzora [5]

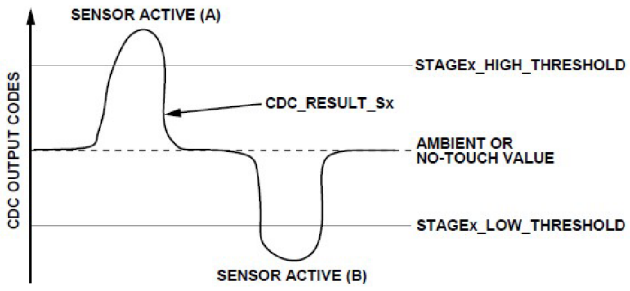
1.2 Operácia snímača pomocou kontroléra AD7147 [4]

AD7147 je kapacitno-digitálny prevodník (CDC), ktorého vnútorný obvod pozostáva z 16-bitového Σ-Δ prevodníka, ktorý mení kapacitný vstup na digitálnu hodnotu. Má 13 vstupov CIN0-CIN12 tieto vstupy sú vedené na prepínaciu maticu, ktorá pripája vstupné signály do CDC a výsledok každého prevodu je uložený v integrovaných registroch. Následne je možné čítať výsledok cez sériové rozhranie SPI. AD7147 má výstup prerušenia (ïnterrupt output") $\overline{INT}$, ten indikuje, keď nové dáta boli vložené do registrov.

AD7147 najprv nabije elektródu senzoru 250kHz excitačným signálom. Keď užívateľ priblíži prst do blízkosti senzora vytvorí virtuálny kapacitor, kde sa prst chová ako druhá doska kapacitora. Počas prevodu je na elektródu senzora aplikovaný budiaci signál štvorcového priebehu a modulátor nepretržite vzorkuje náboj idúci cez elektródu. Výstup modulátora je následne spracovaný cez digitálny filter a výsledné digitálne dáta sú uložené v CDC_RESULT_Sx registroch.

1.2.1 Zaznamenanie a aktivácia senzoru

Keď sa užívateľ priblíži k senzoru, tak sa kapacita asociovaná so senzorom zmení a je odmeraná s AD7147. Ak táto zmena spôsobí prekonanie nastaveného prahu, tak kontrolér si to bude interpretovať ako aktiváciu senzoru. Integrované limity prahov sú použité na určenie toho, kedy nastane aktivácia senzora viz. obrázok 1.7.



Obr. 1.7: Prahy aktivácie senzora [4]

1.2.2 Operačné módy

AD7147 má tri operačné módy:
Plný výkon (Full power), kde sú všetky časti AD7147 neustále plne napájané a prevodník pracuje neustále. Keď je senzor aktivovaný, AD7147 spracováva dáta z tohto senzora. Ak nie je žiadny senzor aktívny, AD7147 meria úroveň okolitej kapacity

a používa tieto údaje pre rutiny kompenzácie na čipe. V režime plného napájania AD7147 prevádza signály konštantnou rýchlosťou.
Nízky výkon (Low power) Ak externé senzory nie sú aktivované, AD7147 znižuje frekvenciu konverzie, čo výrazne znižuje spotrebu energie. Súčiastka sa nachádza v zníženom energetickom stave, kým senzory nie sú aktivované. AD7147 si automaticky zníži odber elektriny, keď sú všetky senzory neaktívne, je navrhnutý tak, aby poskytoval významné úspory energie v porovnaní s režimom plného napájania a je vhodný pre mobilné aplikácie, kde je potrebné šetriť energiu. A **vypínací mód** (Shutdown mode), kde sa zariadenie kompletne vypne.

1.2.3 Environmentálna kalibrácia

AD7147 poskytuje kalibráciu senzora kapacity na automatické prispôsobovanie sa environmentálnym podmienkam, ktoré ovplyvňujú hodnotu kapacity vytvorenú na neaktívnom senzore. Výstupná úroveň kapacitného senzora je citlivá na teplotu, vlhkosť a v niektorých prípadoch aj nečistoty. AD7147 dosahuje optimálny a spoľahlivý chod senzora tým, že neustále monitoruje úrovne kapacity na neaktívnych senzoroch (to znamená na senzoroch, ktoré nie sú aktivované napríklad dotykom) a kompenzuje akékoľvek environmentálne zmeny prostredníctvom úpravy hodnôt registrov, ktoré udávajú prah, pri ktorom sa senzor aktivuje. Po nakonfigurovaní, keď nie je AD7147 aktivovaný sa kompenzačná logika automaticky spúšťa pri každej konverzii. To umožňuje AD7147 kompenzovať rýchlo sa meniace environmentálne podmienky.

1.3 CoreXY [6]

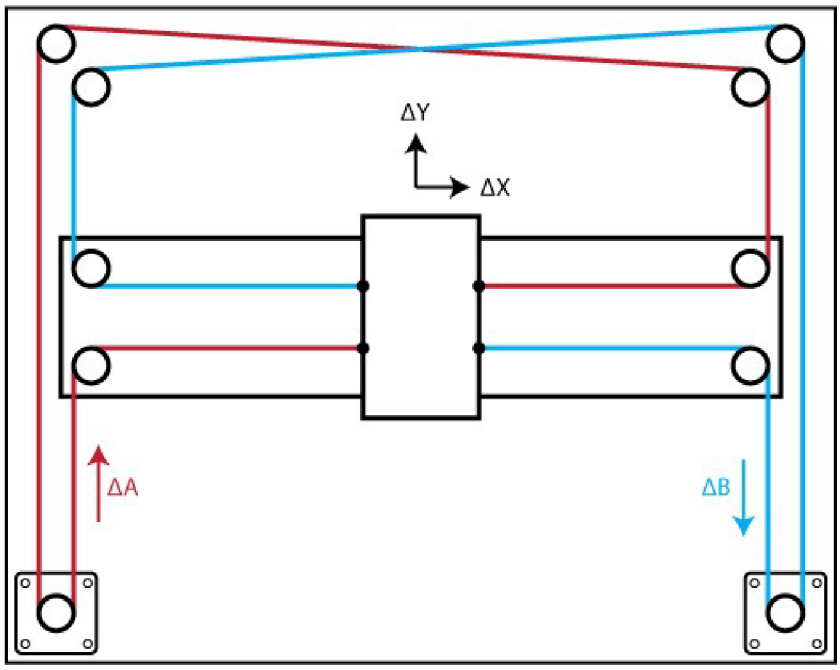
Je metóda, ktorá sa používa na pohyb tlačovej hlavy u 3D tlačiarne, alebo nástrojovej hlavy pri CNC strojoch po horizontálnej rovine. Výhodou tejto metódy je, že dva motory, ktoré sprostredkúvajú pohyb po horizontálnej rovine sú stacionárne a nemusia sa hýbať tým pádom jedna osa nenesie zbytočne váhu jedného motora, čo má za dôsledok presnejší a rýchlejší pohyb. Na obrázku 1.8 môžeme vidieť ako vyzerá mechanizmus metódy CoreXY.

Rovnice pohybu sú popísané v následovnom tvare:

$$\Delta X = \frac{\Delta A + \Delta B}{2} \tag{1.2}$$

$$\Delta Y = \frac{\Delta A - \Delta B}{2} \tag{1.3}$$

$$\Delta A = \Delta X + \Delta Y, \quad \Delta B = \Delta X - \Delta Y \tag{1.4}$$



Obr. 1.8: Referenčný mechanizmus CoreXY [6]

Z týchto rovníc nám vyplýva, že pohyb iba jedného motora spôsobí diagonálny pohyb vozíka po rovine. Keď oba motory hýbeme v rovnakom smere rovnakou rýchlosťou, tak vozík bude jazdiť horizontálne po ose X. A keď sa motory hýbu v opačných smeroch rovnakou rýchlosťou, tak vozík jazdí vertikálne po ose Y.

1.3.1 Výhody techniky CoreXY

Rozdiel medzi systémom CoreXY a klasickým kartézskym pohybovým systémom spočíva v tom, že CoreXY redukuje inerciu z pohybu motora zo statickej polohy, zatiaľ čo kartézsky systém používa aspoň jeden motor na pohyb po každej osi. Hmotnosť motora zvyšuje inerciu a robí zmeny smeru obtiažnejšími. Výsledkom je, že teoreticky je CoreXY rýchlejší a presnejší ako kartézsky systém. Z obrázku 1.8 vidíme, že CoreXY sa skladá z dvoch remeňov čo má za dôsledok hladší pohyb posuvnej hlavice.

Vysoká presnosť: Kvôli presnému a koordinovanému pohybu pomocou dvoch nezávislých motorov dosahuje technika CoreXY vysokej presnosti.

Stabilita: Pevné upevnenie krokových motorov a správne napnutie remeňov znižuje vibrácie a poskytuje vynikajúcu stabilitu počas pohybu posuvnej hlavice.

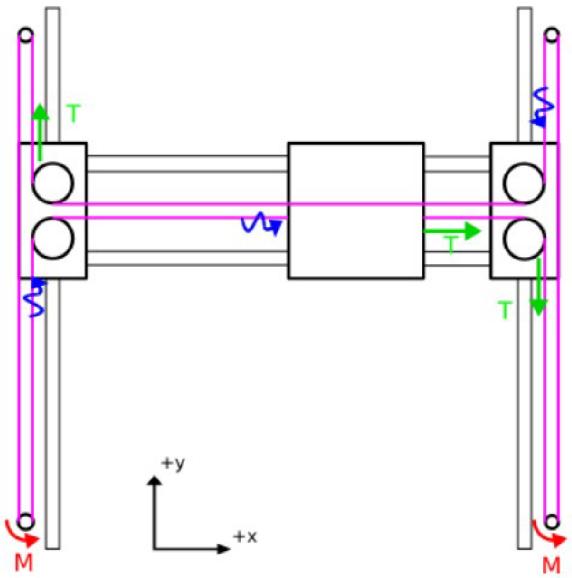
Kompaktnosť: CoreXY je známe tým, že zaberie menej miesta čo je vhodné pre stolové tlačiarne alebo prostredia s obmedzeným priestorom

1.3.2 Nevýhody techniky CoreXY

Dĺžka remeňov: Táto metóda si vyžaduje relatívne **veľkú dĺžku remeňov**, čo môže spôsobovať problémy pri ich napnutí a tak na toto treba brať ohľad pri konštrukcii, pretože zle napnuté remene majú za dôsledok zhoršenie presnosti posúvania hlavice.
Zložitosť konštrukcie: V porovnaní s inými konštrukčnými systémami je o niečo komplexnejší, vyžaduje presné napnutie a zarovnanie remeňov taktiež správne nastavenie krokových motorov a ich synchronizáciu, čo môže byť pre začiatočníkov náročnejšie.
Väčšia hlučnosť: Vzhľadom na väčší počet pohybujúcich sa častí môže systém CoreXY počas prevádzky produkovať vyšší hluk, najmä počas väčších rýchlostí.

1.3.3 Alternatíva H-Bot

Alternatívou pre CoreXY je metóda H-Bot. Ako je vidieť na obrázku 1.9 je pomenovaná podľa tvaru remeňov, ktoré pripomínajú písmeno "H". Pracuje na podobnom princípe ako CoreXY ale využíva iba jeden dlhý remeň, čo rieši nepekné kríženie remeňov ako je vidieť na obrázku 1.8. Výhodou je menší počet remeníc, čo má za následok kompaktnejšie riešenie. H-Bot môže byť vhodný pre určité aplikácie avšak ak ide o rýchlosť a presnosť CoreXY je jasný víťaz.



Obr. 1.9: mechanizmus H-Bot

2 Koncepcia riešenia práce

V tejto kapitole sú popísané moje dôvody na výber vhodných súčiastok a metód pre skonštruovanie a vytvorenie Elektronických šachov. Hlavnou zložkou projektu bude snímanie pozície jednotlivých figúrok na šachovnici, uvediem niekoľko spôsobov, akými by sa dal tento problém vyriešiť a následne aký spôsob som zvolil a prečo. Ďalej bude treba vymyslieť metódu, ktorou bude šachový software hýbať figúrkami a zvoliť mikrokontrolér, ktorý bude celé zariadenie ovládať. Bude nutné zvoliť prenosný zdroj napätia ktorý bude napájať logickú časť ale aj krokové motory a pôjde zabudovať do zariadenia. A nakoniec naprogramovať software pre spracovanie dát zo senzorov a pre ovládanie pohybového mechanizmu. A cieľom teda bude aby celé zariadenie fungovalo tak, že po zapnutí zariadenia môže užívateľ začať ťahom ako prvý, po urobení ťahu sa ťah zaregistruje senzormi a spracuje v šachovom softvéri, následne vygeneruje svoj vlastný ťah, ktorý mikrokontrolér prevedie na počet krokov, ktoré má každý motor vykonať, tak aby bol schopný pomocou elektromagnetu daný ťah zrealizovať. Po dokončení ťahu počítača bude na rade zasa užívateľ.

2.1 Snímanie figúrok

Kritickým aspektom práce je zaznamenávanie pohybov šachových figúrok na šachovnici. S cieľom splniť túto požiadavku som zvážil niekoľko riešení. Táto sekcia poskytuje prehľad o rôznych možnostiach riešenia snímania figúrok, zdôrazňuje ich výhody a obmedzenia. Nakoniec sa ako optimálne riešenie ukázalo snímanie pomocou kapacitných senzorov.

Snímanie pomocou Hallovho senzora

Hallove senzory využívajú magnetické pole pre detekciu prítomnosti objektov, využívajú Hallovho javu, čo je vytvorenie bočného elektrického potenciálu po umiestnení vodiča, ktorým prechádza prúd do magnetického poľa, tento potenciál sa nazýva Hallovo napätie a Hallov senzor teda detekuje zmenu tohto potenciálu. Ako sa dozvieme v sekcii o pohybovom mechanizme 2.3, tak každá figúrka bude mať v sebe zo spodku zabudovaný magnet, ktorý by teda umožňoval detekciu Hallovým senzorom. Výhodou hallových senzorov je ich spoľahlivosť pri detekcii objektov, ktoré vytvárajú magnetické pole a taktiež majú veľmi nízku náchylnosť voči vonkajším faktorom. Nápad vyriešiť snímanie figúrok týmto senzorom ma napadol ako prvý a plán bol taký, že by sa hallove senzory umiestnili jednotlivo pod každé políčko a tým by teda boli schopné určiť prítomnosť figúrky s magnetom. Ale ako som neskôr zistil, umiestnenie senzorov by bránilo elektromagnetu tesne priľahnúť zo spodku ku

šachovnici, tým pádom by musela byť medzera medzi šachovnicou a elektromagnetom, čo by mohlo spôsobiť to, že by magnet nebol dostatočne silný na to aby na takú diaľku pohol figúrkou. A práve kvôli tomuto dôvodu som sa rozhodol nepoužiť Hallove senzory.

Snímanie pomocou kamery

Určenie pozície figúrok by sa dalo vyhotoviť aj pomocou strojového videnia. Tento spôsob mi prišiel ako komplikované riešenie, ktoré by zabralo veľa času na zostrojenie a naprogramovanie, ale hlavne nepraktické pri prenášaní šachovnice.

Snímanie pomocou optických senzorov

Na snímanie by sa mohli použiť aj optické senzory ako napríklad fotodetektory, ktoré by dokázali zaznamenať prítomnosť alebo absenciu figúrky na základe prerušenia svetla, ktoré by na senzor dopadalo. Avšak problémy by mohlo spôsobovať to, že by figúrky museli byť presne položené na senzor a navyše by presné umiestnenie senzorov na políčko zväčšovalo zložitosť konštrukcie.

2.1.1 Snímanie pomocou kapacitného senzora

Tento spôsob snímania som sa rozhodol vybrať najmä preto, lebo spĺňa všetky moje požiadavky, ktoré sú spojené so snímaním figúrok tj. praktický transport spolu so šachovnicou, nie veľmi časovo náročné na zostrojenie a predovšetkým cenovo najviac vyhovujúce. Šachovnica, na ktorej sa bude hrať bude mať **32x32 cm**, preto som sa rozhodoval medzi dvomi možnosťami, či navrhnúť viacej dosiek plošných spojov a následne ich vzájomne prepojiť, táto možnosť by bola z hľadiska ceny najviac vyhovujúca, ale robila by problémy pri následnom hraní šachu, pretože ako sa dozvieme v kapitole 3.1, tak senzory majú obmedzenú senzitivitu a položením čo i len tenkej (drevenej) šachovej dosky sa senzitivita senzorov značne zníži, čo by mohlo mať za dôsledok neregistrovanie šachovej figúrky. Takže som sa rozhodol pre druhú možnosť a tou je návrh jednej dosky plošných spojov so 64-mi kapacitnými senzormi, ktorá bude plniť aj úlohu šachovnice, na ktorej sa bude hrať šach.

2.2 Mikrokontrolér

Pri otázke výberu vhodných SoC (System on chip) mikrokontrolérov som rozmýšlal o viacerých možnostiach no nakoniec som sa rozhodol pre **ESP32** [7]. A to kvôli nasledujúcim dôvodom:

Obr. 2.1: ESP32 DEVKIT V1 [7]

Výpočtový výkon: ESP32 je vybavený dvojjadrovým 32-bitovým procesorom, ktorý poskytuje dostatočný výkon na vykonanie komplexných úloh, ako je beh šachového softvéru spracovanie údajov zo snímačov a mnoho ďalších.

GPIO piny: Má dostupných až 34 programovateľných GPIO pinov, čo je dostatočný počet pre väčšinu projektov, u ktorých je potreba ovládať veľa zariadení, alebo spracúvať veľké množstvo informácií.

Vývojové prostredie: ESP32 má veľkú komunitu vývojárov, ktorá poskytuje mnoho tútoriálov a príklady kódu, ktoré môžu veľmi uľahčiť riešenie rôznych problémov. Taktiež podporuje viacero programovacích jazykov medzi najčastejšie patria: Arduino IDE (C/C++) a **Micropython**, v ktorom sa bude vyvýjať software pre tento projekt.

Integrovaná Wi-Fi: Ktorá by mohla slúžiť v budúcich verziách Elektronického šachu na komunikáciu medzi jednotlivými šachovnicami.

Komunikačné rozhranie: ESP32 poskytuje viacero komunikačných protokolov jedným z nich je I^2C (Inter-Integrated Circuit) je to dvojvodičové obojsmerné synchrónne sériové rozhranie, ktoré sa použije na komunikáciu s prevodníkmi AD7147 [4].

2.3 Pohybový mechanizmus

Pri otázke akým spôsobom bude počítač manipulovať s figúrkami som zvažoval viacero mechanizmov, ale najviac ma zaujalo riešenie, kde bude celé ústrojenstvo schované v krabici pod šachovnicou a bude figúrkami hýbať pomocou elektromagnetu, ktorý bude pripevnený na posuvnej hlavici mechanizmu zostrojeného podľa už spomenutej CoreXY 1.3 metódy.

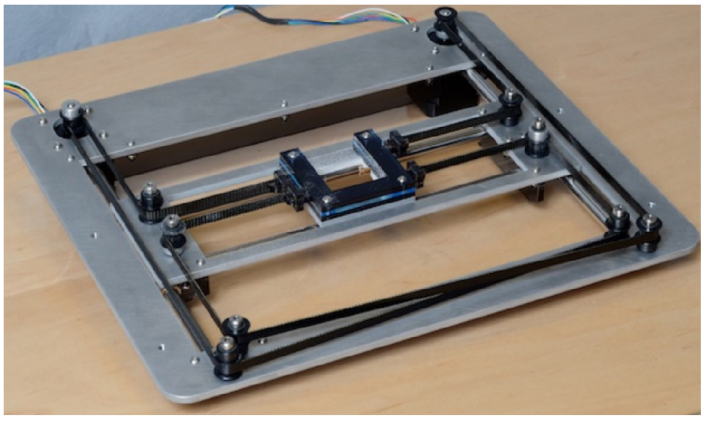
Tu sú niektoré z mechanizmov, ktoré som zvážil a dôvod pre zvolenie CoreXY:

Karteziánsky portálový systém: Je mechanizmus, ktorý sa často používa v rôznych aplikáciách ako napríklad CNC stroje alebo 3D tlačiarne. Funguje tak, že motor č.1 hýbe posuvnou hlavou po ose Y a motor č.2 sa posúva po ose Y a zároveň hýbe posuvnou hlavou po ose X. Takže nevýhodou je, že jeden motor musí niesť váhu druhého. Z dôvodu obmedzeného priestoru a potreby jednoduchšieho dizajnu sa tento spôsob ukázal ako menej vhodný.

Obr. 2.2: Karteziánsky portálový mechanizmus [12]

Robotická ruka: Robotická ruka by mohla slúžiť na manipuláciu a posun šachových figúriek. Tento mechanizmus však prináša ďalšie komplikácie z hľadiska kinematiky a problém pri transporte zariadenia, preto som sa pre túto možnosť nerozhodol.

CoreXY: Mechanizmus CoreXY ponúka zaujímavé riešenie pre presný a synchronizovaný pohyb posuvnej hlavice, skladá sa z dvoch stacionárnych krokových motorov, systému kladiek a remeňov usporiadaných do krížovej konfigurácie. Pre tento mechanizmus som sa rozhodol hlavne z toho dôvodu, že obidva motory budú stacionárne tým pádom nebude pohyb po jednej osi zbytočne zaťažený váhou motora a tak bude mať väčšiu kompaktnosť ústrojenstva.



Obr. 2.3: CoreXY mechanizmus [6]

3 Návrh Hardvéru pre Elektronické šachy

Jedným z kľúčových aspektov návrhu hardvéru je výber vhodných komponentov pre každú funkciu. Na detekciu prítomnosti a pohybu šachových figúrok na šachovnici budú využité kapacitné snímače. Tieto snímače ponúkajú spoľahlivé a nenápadné snímanie umožňujúce presnú detekciu. V tejto kapitole teda zistím optimálnu implementáciu kapacitných senzorov na doske plošných spojov pomocou viacerých meraní, ďalej sa určí najlepší tvar senzora a po odskúšaní funkčnosti kapacitno-digitálneho prevodníka AD7147 [4] sa navrhne už výsledný dizajn snímačov pre šachovnicu. Následne navrhnem model mechanizmu elektronických šachov 3.12. Definujem súčiastky, ktoré budú pri zostrojovaní potreba a postavím ústrojenstvo podľa modelu. Nakoniec navrhnem zdroj napájania pre logickú časť a pre motory.

3.1 Návrh kapacitných senzorov

V našom prípade, keďže figúrky nie sú uzemnené bude jedna elektróda senzora a druhá elektróda zemi. Pre optimálny návrh tohto kapacitného senzora treba brať ohľad na **veľkosť** elektród, **vzdialenosť** medzi nimi a **tienenie** rušenia. Pre vzdialenosť medzi elektródami platí: Čím sú elektródy **bližšie** k sebe, tým je dosah snímania efektívne **znížený**. A tým pádom som pred samotným návrhom a zostrojením 32x32 cm DPS vykonal radu meraní, ktoré sa zaoberali:

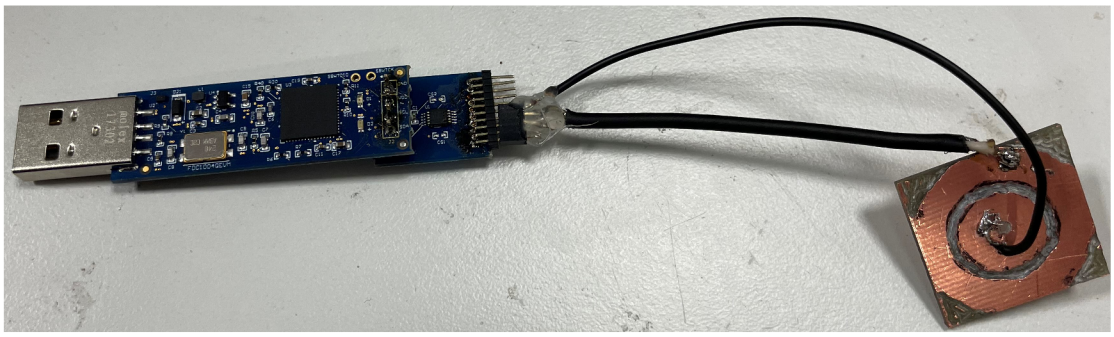
- vzdialenosťou dvoch elektród
- snímaním rôznych materiálov
- vplyvom úzkej vrstvy dreva medzi snímaným predmetom a senzorom a jej vplyv na senzitivitu snímania
- rôznym umiestnením tieniacej elektródy
- odlišnými tvarmi senzora

3.1.1 Výsledky meraní rôznych variácií kapacitných senzorov

Všetky merania boli vykonávané pomocou vyhodnocovacieho nástroja **FDC1004QEVM** [9] je to 4-kanálový kapacitno-digitálny prevodník s možnosťou pripojenia tieniacej elektródy. Výsledky boli spracovávané v programe od Texas Instruments s názvom **Sensing Solutions EVM GUI Tool v1.10.0**.

Meranie č.1

Úloha prvého merania bola zostrojiť kapacitný senzor v tvare kruhu vyfrézovaním kuprexitovej dosky a následne ho pripojiť ku kapacitno-digitálnemu prevodníku



Obr. 3.1: Schéma zapojenia kruhového senzora

FDC1004QEVM [9], pomocou ktorého následne získame výsledné hodnoty kapacít pre tri predmety z rôznehoo materiálu:

- **drevená figúrka** (priemer spodku figúrky = 1,5cm)
- **pliešok z hliníku** (priemer = 2cm)
- **PVC kocka** (strana = 2,5cm)

Senzor má priemer 1 cm , okolo neho je zemniaca elektróda s priemerom 2 cm a medzi nimi je 2 mm medzera. Ďalej je na spodku dosky meď, ktorá slúži ako tieniaca elektróda (SHIELD). Po meraní som vymenil GND a senzorovú elektródu, keďže tá s priemerom 2cm má väčšiu plochu a sledoval ako sa zmenia výsledné kapacity.

Ako môžeme vidieť na obrázku 3.1 tak drôt senzoru je koaxiálny a tienenie tohto drôtu je pripájkované na spodok senzoru.

Tab. 3.1: Namerané dáta pre kruhový senzor

konfigurácia	**nečinný**	**figúrka**	**pliešok**	**kocka**
Senzor v strede [pF]	1,13	1,29	1,55	1,21
GND v strede [pF]	1,5	1,87	2,1	1,72
percentuálna zmena [%] (senzor v strede)	0	14,16	37,16	7,07
percentuálna zmena [%] (GND v strede)	0	24,67	40	14,67

Keďže merania vykonávame z bezprostrednej vzdialenosti k senzoru a detekujeme neuzemnené predmety, tak najväčší vplyv na zmenu kapacity má dielektrikum meraného materiálu. V tabuľke 3.2 môžeme vidieť rôzne dielektrické konštanty materiálov.

Z nameraných hodnôt môžeme vidieť, že zmena v kapacite sa pohybuje v rádoch desatín pF, čo nie je veľa a mohlo to byť spôsobené káblami, ktoré bránia predmetu, aby tesne priľahol k senzoru. Najväčšie zmeny v kapacite vykazoval kovový pliešok z hliníku, čo je dobré znamenie keďže v každej figúrke bude zabudovaný magnet a tým pádom bude lepšie detekovateľná. Následne sme zamenili elektródy tak, že GND je v strede a senzor je okolo nej, pri tomto zapojení vidíme, že zmeny v kapacite spolu s

Tab. 3.2: Dielektrické konštanty materiálov

materiál	**Dielektrická konštanta** ε_r
Vzduch	1
PVC	3
Voda (20 °C)	80
Drevo	(2-6)
Kremík	(11-12)
Alkohol	(16-31)
Plexisklo	3,2
Papier	2,3

nečinnou kapacitou narástli, toto je spôsobené tým, že elektróda senzoru má väčšiu plochu a tým pádom aj väčšiu senzitivitu.

Meranie č.2

V tomto meraní som zisťoval ako vplýva veľkosť medzery medzi elektródami a dielektrikum medzi meraným predmetom a senzorom na výslednú zmenu kapacity a citlivosť senzorov, ktoré mali obdĺžnikový a hrebeňový tvar ako je znázornené na obrázku 3.2 Zapojenie je rovnaké ako v meraní č.1 3.1.1.

Postup: Meranie som vykonával tak, že som zobral senzor s určenou šírkou medzery položil som naň dielektrikum v podobe 3 mm vrstvy kartóna a naň som následne položil snímaný objekt (pliešok z hliníka s priemerom 20 mm) ďalej som odčítal hodnoty a zväčšil veľkosť medzery medzi elektródami, ktorú som odmeral posuvným meradlom a proces som pre väčšiu medzeru opakoval. Následne som sledoval ako sa zväčšuje/zmenšuje rozdielová kapacita na základe šírky medzery.

Obr. 3.2: Hrebeňový a obdĺžnikový senzor

kde je:
C_{idle} ... základná kapacita nečinného senzora

Tab. 3.3: Dáta namerané pre **obdĺžnikový senzor**

medzera (mm)	C_{idle}**[pF]**	C_d**[pF]**	C_{vysl}**[pF]**	δ_C**[%]**
0,5	1,26	1,48	1,58	6,8
1,35	0,98	1,2	1,29	7,5
2,5	0,86	1,05	1,14	8,6
4,4	0,77	0,88	0,95	8

C_d ... kapacita po priložení dielektrika (kartónu)
C_{vysl} ... kapacita s dielektrikom aj s meraným predmetom (pliešok z hliníka)
δ_C ... percentuálna zmena v kapacite po priblížení meraného predmetu

Tab. 3.4: Dáta namerané pre **hrebeňový senzor**

medzera (mm)	C_{idle}**[pF]**	C_d**[pF]**	C_{vysl}**[pF]**	δ_C**[%]**
0,7	1,55	1,62	1,65	1,85
1,5	1,23	1,31	1,35	3
2	1,1	1,2	1,25	4,2

Po predstavení dielektrika môžeme vidieť, že zmeny v kapacite sú oveľa nižšie ako v meraní č.1. V tabuľke 3.3 vidíme, že pri zväčšovaní medzery do 2,5 mm sa percentuálna zmena o trochu zvyšovala, ale pri zväčšení na 4,4 mm sa zmenšila toto je spôsobené tým, že aj keď zväčšíme medzeru, čo má za dôsledok zväčšenie meraného priestoru, v ktorom je senzor schopný predmet registrovať zmenšujeme tým hustotu elektrického poľa, ale aj plochu samotného senzoru. **Lepší postup** pri tomto meraní by bolo vytvoriť viacej senzorov s rovnakým tvarom a plochou elektród, ale s odlišnou medzerou medzi elektródami, týmto spôsobom by sme boli schopný lepšie zistiť aký má dopad vzdialenosť dvoch elektród na snímanie objektu.

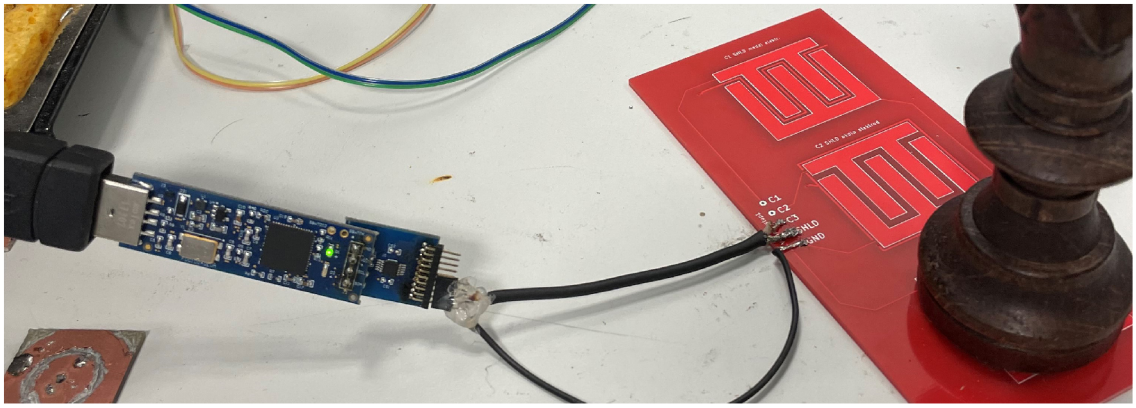
Meranie č.3

V tomto meraní som testoval funkčnosť kapacitného senzoru už na samostatnej DPS (viz. A) pre rôzne konfigurácie tieniacej elektródy, ale aj pre vrstvu dreva medzi figúrkou a senzorom. (Pod každým senzorom je tieniaca elektróda.)

Merania som vykonával pre rôzne predmety:

- **pešiak** (materiál: drevo, priemer základne: 3 cm)
- **figúrka s magnetom** (priemer základne: 1,5 cm)
- **dáma** (materiál: drevo, priemer základne: 4 cm)
- **kruhový pliešok č.1** (materiál: oceľ, priemer: 1,7 cm)

- **štvorcový pliešok** (materiál: meď, strana: 3 cm)
- **kruhový pliešok č.2** (materiál: oceľ, priemer: 3 cm)



Obr. 3.3: Testovanie senzorov na DPS

Tab. 3.5: Dáta namerané pre rôzne konfigurácie tieniacej elektródy

konf.	**nečinný**	**pešiak**	**dáma**	**figúrka**	**k. p. č.1**	**k. p. č.2**	**meď**
shld iba zo spodu [pF]	2,13	2,44	2,54	3,5	10,33	16	6,12
shld okolo senzora [pF]	1,73	2,02	2,09	3,38	9,84	16	5,73
shld medzi elektródami [pF]	1,39	1,66	1,74	3,11	10,12	16	5,35

Tab. 3.6: Dáta namerané pre 2mm vrstvu dreva medzi senzorom a predmetom

konf.	**nečinný**	**pešiak**	**dáma**	**figúrka**	**k. p. č.1**	**k. p. č.2**	**meď**
shld iba zo spodu [pF]	2,74	2,87	2,92	2,83	2,9	3,03	2,91
shld okolo senzora [pF]	2,36	2,47	2,48	2,48	2,53	2,64	2,61
shld medzi elektródami [pF]	1,9	1,98	2,02	1,98	2,02	2,15	2,08

Tab. 3.7: Rozdiely v (%) od nečinného stavu po priblížení predmetu z tabuľky 3.5

konf.	**nečinný**	**pešiak**	**dáma**	**figúrka**	**k. p. č.1**	**k. p. č.2**	**meď**
shld iba zo spodu [%]	0	14,6	19,24	64,3	384	651	187,8
shld okolo senzora [%]	0	16,8	20,8	95,4	468	824	231
shld medzi elektródami [%]	0	19,4	25,2	123,7	628	1051	285

(Kde shld je tieniaca elektróda). Najmenšiu kapacitu v nečinnom stave má senzor s tieniacou (shield) elektródou medzi elektródami GND a senzor toto je spôsobené tým, že najhustejšie elektrické pole, ktoré sa nachádza presne medzi elektródami bolo

Tab. 3.8: Rozdiely v (%) od nečinného stavu po priblížení predmetu pre 2mm vrstvu dreva z tabuľky 3.6

konf.	**nečinný**	**pešiak**	**dáma**	**figúrka**	**k. p. č.1**	**k. p. č.2**	**meď**
shld iba zo spodu [%]	0	4,7	6,6	3,3	5,8	10,6	6,2
shld okolo senzora [%]	0	4,7	5,1	5,1	7,2	11,9	10,6
shld medzi elektródami [%]	0	4,2	6,3	4,2	6,3	13,2	9,5

narušené. Tým pádom u tejto konfigurácie nastali najväčšie percentuálne rozdiely po priblížení predmetu. Z hodnôt možno usúdiť, že najvyššie rozdiely vykazovali kovové predmety a z figúrok nastali najväčšie zmeny u figúrky s magnetom, aj napriek tomu že bola oveľa menšia ako figúrky bez magnetu. Aj keď niektoré konfigurácie tieniacej elektródy zaznamenávajú väčšie rozdiely, dalo by sa povedať že pokiaľ sa jedná o detekciu predmetu v bezprostrednej vzdialenosti všetky by boli vyhovujúce. Ako môžeme vidieť z tabuľky 3.6 po položení 2 mm vrstvy dreva medzi snímaný predmet a senzor hodnoty kapacít a tak aj rozdielov po priblížení predmetu sa značne znížili a bolo by ťažšie stanoviť prahovú hodnotu senzora pre zaregistrovanie predmetu. Z tohto dôvodu som sa rozhodol tak, že samotné **šachy sa budú hrať rovno na doske plošných spojov**.

3.1.2 Otestovanie funkčnosti AD7147 [4]

Aby som zabezpečil funkčnosť všetkých kapacitných senzorov som sa rozhodol použiť kapacitno-digitálny prevodník s názvom AD7147. Rozhodol som sa preň najmä kvôli tomu, že má 13 vstupov na pripojenie senzorov, čo by znamenalo, že ich bude potreba 5 kusov, čo by stačilo na všetkých 64 senzorov.
AD7147 má 5 vlastnosti vďaka ktorým som si zvolil práve tento prevodník:

Sériová komunikácia: AD7147 podporuje priemyselný štandard 2-vodičového sériového rozhrania I^2C. Dva vodiče asociované s I^2C časovaním sú SCLK a SDA vstupy. SDA je vstupno-výstupný pin, ktorý umožňuje operácie zápisu a čítania registrov. Na sériovom rozhraní je AD7147-1 vždy to, ktoré je ovládané, takzvaný typ "slave". Ja budem v mojej práci používať AD7147-1 s I^2C komunikáciou. AD7147 má aj variantu s SPI komunikáciou.

Viacero vstupných kanálov: AD7147 poskytuje viacero vstupných kanálov, čo umožňuje jednoduché pripojenie veľkého počtu kapacitných snímačov, ako bolo už povedané AD7147 má 13 vstupov na kapacitné snímače čo znamená, že stačí 5 kusov pre celú šachovnicu. To je jeden z hlavných dôvodov prečo som tento kontrolér zvolil.

Cenovo výhodné: Keď porovnáme pomer (cena)/(počet vstupov) tak AD7147 je

jedna z výhodnejších možností na trhu oproti iným možnostiam. (Analog Devices mi taktiež poskytlo 3 vzorky zadarmo aby som mohol overiť funkčnosť)

Adaptívny prah a citlivosť: AD7147 poskytuje integrovaný algoritmus s automatickým nastavovaním adaptívneho prahu aktivácie senzora a citlivosti. Tento algoritmus neustále monitoruje výstupné úrovne každého snímača a automaticky prispôsobuje prahy aktivácie senzorov vzhľadom na pokrytú plochu snímača. Výsledkom je udržiavanie optimálnych prahov a citlivosti.

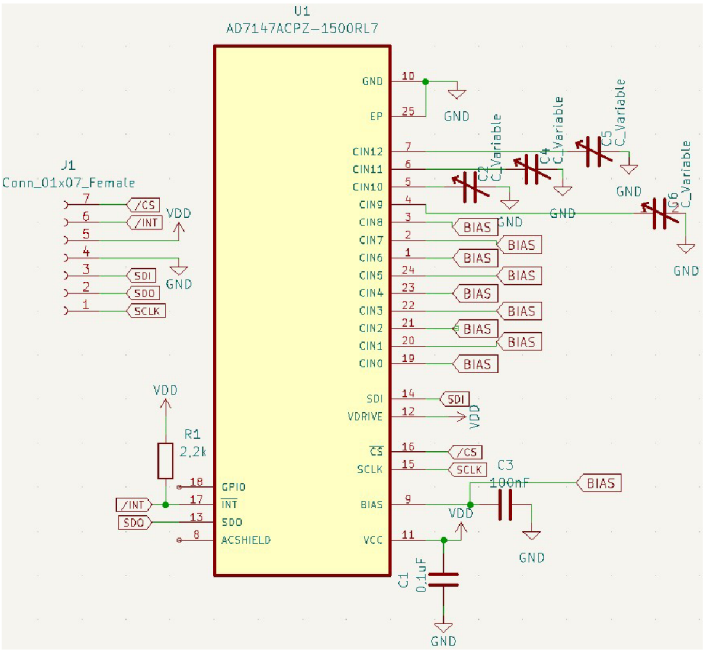
Vysoká citlivosť: AD7147 ponúka vysokú citlivosť pri detekcii zmeny kapacity, rozlíšenie je v rádoch Femtofaradov. Čo je vhodné pre presné snímanie prítomnosti a pohybu šachových figúrok na šachovnici. Toto zabezpečuje spoľahlivé a presné určenie pozícií šachových figúrok.

AD7147 bude mať kľúčovú úlohu pri detekcii prítomnosti a pohybu šachových figúrok na šachovnici. Pred implementáciou tohto komponentu do finálneho návrhu je nevyhnutné dôkladne otestovať jeho funkčnosť a spoľahlivosť. Testovacia fáza zahŕňa vytvorenie malého prototypu so štyrmi kapacitnými snímačmi pripojenými k AD7147. Tento prototyp sa použije na overenie schopnosti detekovať figúrky a dostatočnej citlivosti. Tiež budem overovať, či zariadenie správne komunikuje s mikrokontrolérom ESP32, tak aby sa zabezpečil bezproblémový chod snímačov. Na základe výsledkov testovania sa vypracuje konečný návrh snímačov pre šachovnicu. Tento návrh bude brať do úvahy faktory, ako sú rozmery šachových figúrok, požadovaný rozostup medzi snímačmi a estetiku šachovnice.

Meranie č.4

Toto meranie bude teda spočívať v overení správnej funkčnosti kapacitno-digitálneho prevodníka AD7147 a správnej komunikácie s mikrokontrolérom, tiež budem merať veľkosť zmeny kapacity na štyroch rôznych tvaroch senzora, z ktorých následne vyberiem jeden tvar a ten použijem vo finálnom dizajne. Návrh dosky je možné vidieť v prílohe B

Avšak, keď som navrhoval dosku plošných spojov na vykonanie tohto merania bol som v tom, že som objednal variantu s SPI komunikáciou, ale ako som neskorej zistil kúpil som omylom AD7147ACPZ-1500RL7, čo je varianta s I^2C komunikáciou. A keďže som už mal tento typ kontroléra nakúpený rozhodol som sa ho použiť. Avšak to, že budem používať protokol I^2C bude pre finálny návrh komplikáciou, keďže kontrolér má iba dva adresové vstupy, čo umožňuje pripojiť na I^2C zbernicu iba 4 AD7147 zariadenia, ale je ich potreba 5. Tento problém budem riešiť v sekcii 3.1.3.



Obr. 3.4: Schéma DPS pre odskúšanie AD7147

Ako môžeme vidieť v schéme zapojenia 3.4 nepoužité vstupy pre snímače sú pripojené na BIAS pin. Je taktiež zjavné, že dizajn bol určený pre SPI rozhranie pretože môžeme vidieť $\overline{CS}$ (chip select), čo je vstup pre zvolenie daného čipu pri komunikácii skrz SPI. Ale keďže používam I^2C tak pin SDI je na tomto čipe pin pre určenie prvého bitu adresy **ADD0** a pin $\overline{CS}$ bude pin pre určenie druhého bitu adresy **ADD1**.

DPS som k mikrokontroléru ESP32 pripojil následovne:

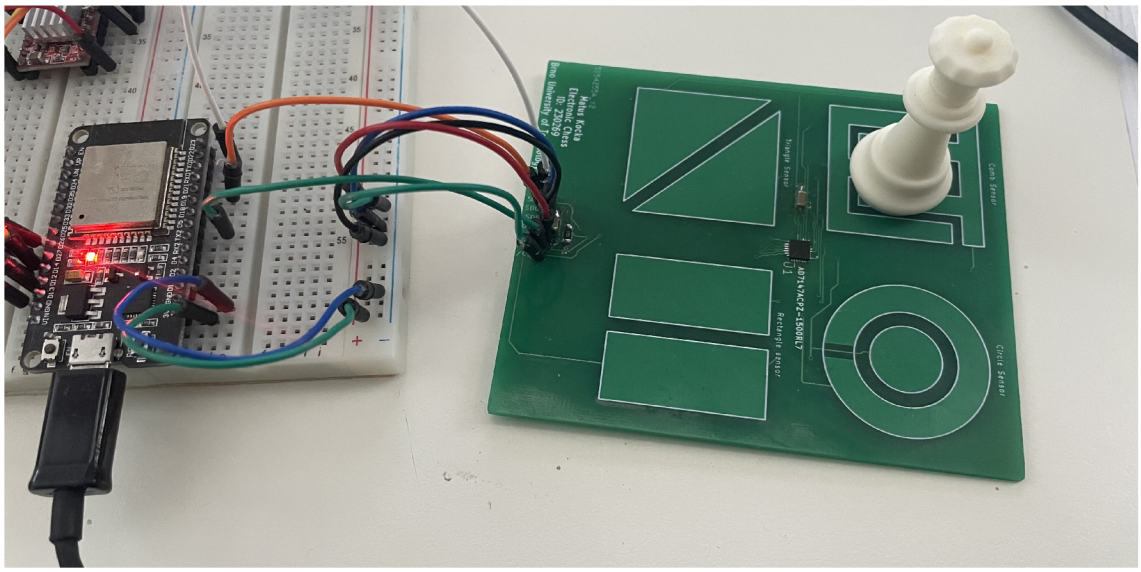
- SCLK - GPIO 22
- SDO (SDA) - GPIO 21
- SDI (ADD0) - GND
- $\overline{INT}$ - GPIO 5
- $\overline{CS}$ (ADD1) - GND

Keďže sú obidva adresové piny pripojené na zem tak podľa technického listu [4] bude mať čip adresu 0x2C.

Pred tým než sa bude komunikovať s AD7147 sa najprv treba uistiť že I^2C protokol beží bez problémov a to tak, že sa prečíta ID čipu, pokiaľ sa bude zhodovať s ID v technickom liste kontroléra.

Ako vidíme na obrázku 3.6, keď prečítame dáta z registra 0x017 mali by sme dostať hodnotu 0x147, tú sme aj dostali to znamená, že zariadenie komunikuje tak ako má a môžeme začať meranie.

Rozsah jedného vstupu u AD7147 je : $\pm$ **8 pF**
Avšak na zmenu citlivosti snímača môžeme použiť takzvaný kapacitný offset, kde sa



Obr. 3.5: Testovanie funkčnosti AD7147

Address	Data Bit	Default Value	Type	Name	Description
0x017	[3:0]	0	R	REVISION_CODE	Revision code
	[15:4]	147	R	DEVID	Device ID = 0001 0100 0111

Obr. 3.6: ID kontroléra AD7147 [4]

Tab. 3.9: Výsledok kapacitno-digitálneho prevodníka po položení figúrky

tvar snímača	**IDLE**	**Δ Xf**	**Δ Xk**
trojuholníkový	45055	1145	4505
obdĺžnikový	46550	1150	3843
kruhový	48750	1050	3950
hrebeňový	48450	1200	5750

± 8 pF rozsah merania dá posunúť až o 20 pF s rozlišením 0.32 pF Toto posunutie sa dá dosiahnuť zapísaním do STAGEx_AFE_OFFSET registrov, kde sa dá nastaviť 6-bitový POS_AFE_OFFSET alebo NEG_AFE_OFFSET (závisí či je vstup pripojený na pozitívny alebo negatívny vstup prevodníka). Zmena kapacity sa ale vždy bude meniť iba o max ±8 pF.

Počas tohto merania boli všetky vstupy pripojené na pozitívny vstup prevodníka. A v POS_AFE_OFFSET bola zapísaná hodnota: 0b000111 (kde 1 najnižší platný bit = 0,32 pF). Čo znamená že merací rozsah sme posunuli o 2,24 pF a tak meriame v rozsahu <2,24 ; 10,24> pF.

Teraz môžeme jednoducho prepočítať výsledky prevodníka z tabuľky 3.9 na fa-

rady. (jedná sa o 16-bitový prevodník)

$$C = \frac{8}{2^{16}} \cdot CDC_{result} + 2,24 \tag{3.1}$$

Tab. 3.10: Výsledok prevodníka prevedený na zmenu kapacity po položení figúrky

tvar snímača	C_{IDLE} **[pF]**	**Δ Cf [pF]**	**Δ Ck [pF]**
trojuholníkový	7,740	0,140	0,550
obdĺžnikový	7,922	0,140	0,469
kruhový	8,191	0,128	0,482
hrebeňový	8,154	0,146	0,702

kde je:
C_{IDLE} ... je kapacita neaktívneho senzora
Δ Cf ... zmena kapacity po položení figúrky bez magnetu
Δ Ck ... zmena kapacity po položení figúrky s magnetom

Zmeny kapacity boli malé, čo bolo spôsobené zlou kalibráciou snímačov, ale aj napriek tomu bolo možné zistiť aktiváciu senzora. Najväčšia zmena nastala u hrebeňového senzora, ktorý má ešte jednu výhodu a to takú, že má veľkú snímanú plochu tým pádom nebude musieť byť figúrka položená presne do stredu pre aktiváciu senzora a preto som sa rozhodol zakomponovať tento tvar do finálneho dizajnu senzorov.

3.1.3 Návrh Šachovej dosky so senzormi

Po viacerých meraní a otestovaní kontroléra AD7147 som získal cenné poznatky a údaje, ktoré mi pomôžu pri konečnom návrhu snímačov pre Elektronické šachy. Fáza testovania mi pomohla s určením veľkosti medzere medzi elektródami, s určením tvaru elektród a s porozumením ako funguje kontrolér AD7147. Tieto cenné informácie teraz budú slúžiť ako základ pre konečný návrh, a umožňujú nám riešiť obmedzenia a problémy, s ktorými som sa stretol počas testovania.

Avšak, keďže som omylom objednal AD7147ACPZ-1500RL7 čo je varianta s I^2C protokolom musel som riešiť problém, že boli dostupné iba štyri odlišné adresy, ktoré sa dali určiť bitmi ADD0 a ADD1, ale ja som potreboval na pokrytie všetkých 64 políčok týchto zariadení 5, keďže jedno má 13 dostupných vstupov na senzory. Toto by spôsobovalo taký problém, že dve zariadenia by mali rovnakú adresu vznikal by takzvaný adresový konflikt a nevedelo by sa z ktorého sa čítajú/zapisujú aké dáta. Našťastie som však pre tento problém našiel riešenie v podobe 8-kanálového

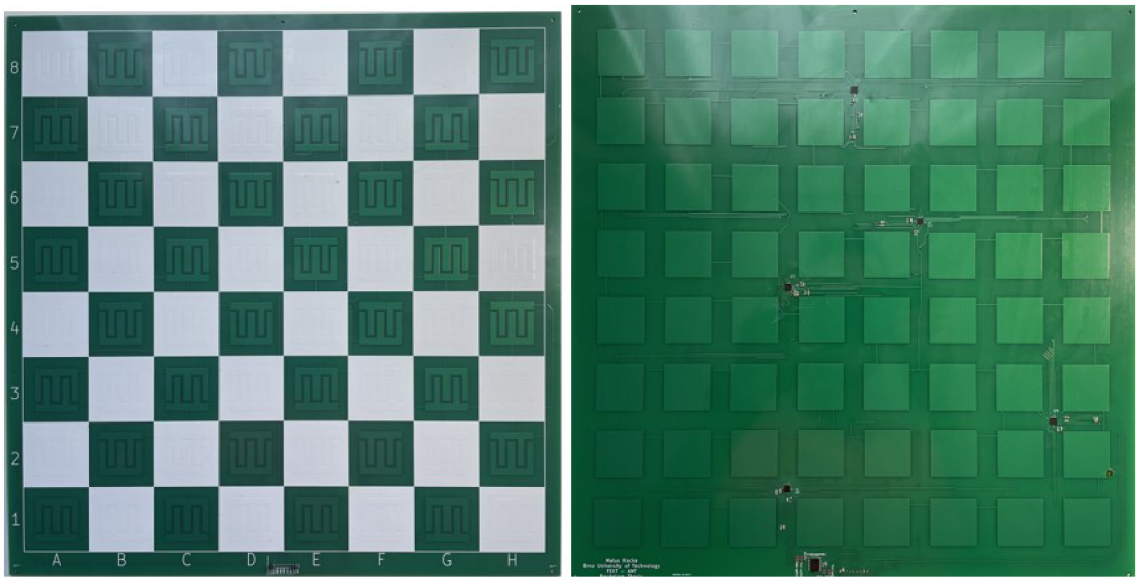
I^2C multiplexera s názvom **TCA9548A** [13]. Pomocou ktorého sa dá prepínať medzi ôsmimi zariadeniami s takou istou I^2C adresou a tento multiplexer sám má 3 adresové piny, takže by sa ich mohlo pripojiť 8 na zbernicu čo by umožňovalo komunikovať s 64-mi zariadeniami s adresovým konfliktom.

V mojom prípade som všetky tri adresové piny pripojil na zem, takže multiplexer bude mať podľa technického katalógu adresu **0x70**

CONTROL REGISTER BITS								COMMAND
B7	B6	B5	B4	B3	B2	B1	B0	
X	X	X	X	X	X	X	0	Channel 0 disabled
							1	Channel 0 enabled
X	X	X	X	X	X	0	X	Channel 1 disabled
						1		Channel 1 enabled
X	X	X	X	X	0	X	X	Channel 2 disabled
					1			Channel 2 enabled
X	X	X	X	0	X	X	X	Channel 3 disabled
				1				Channel 3 enabled
X	X	X	0	X	X	X	X	Channel 4 disabled
			1					Channel 4 enabled
X	X	0	X	X	X	X	X	Channel 5 disabled
		1						Channel 5 enabled
X	0	X	X	X	X	X	X	Channel 6 disabled
	1							Channel 6 enabled
0	X	X	X	X	X	X	X	Channel 7 disabled
1								Channel 7 enabled
0	0	0	0	0	0	0	0	No channel selected, power-up/reset default state

Obr. 3.7: Zvolenia kanálu na multiplexeri TCA9548A [13]

Pre zvolenie zariadenia na komunikáciu pomocou multiplexera TCA9548A zapíšeme na jeho adresu (teda 0x70) číslo vstupu, na ktoré je zariadenie pripojené, tak ako je vyobrazené na obrázku 3.7. A po tomto zvolení už môžeme normálne do zariadenia zapisovať, alebo respektíve z neho čítať dáta.



Obr. 3.8: Navrhnutá šachovnica pre snímanie figúrok

Na obrázku č. 3.8 už môžeme vidieť, ako vyzerá finálna DPS so senzormi. Rozmery šachovnice sú 34 cm na šírku a 35 cm na výšku, každé políčko má 4x4 cm. Celý dizajn bol navrhnutý v programe KiCad 6.0, môžeme ho vidieť v prílohe C. Ako je vidieť navrhol som to spôsobom, kde všetky súčiastky sú pripojené zo spodku dosky aby to nerobilo problém pri pohybe figúrok a tým pádom sa bude musieť elektromagnet pohybovať tesne pod týmito súčiastkami, aby s nimi nekolidoval.

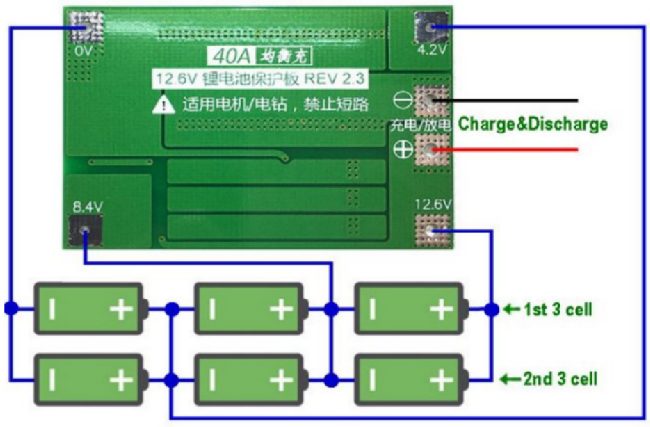
3.2 Prenosný zdroj napájania

Táto sekcia sa bude zaoberať návrhom a implementáciou prenosného napájania. ; Prenosný zdroj napätia sa skladá z:

Batérie: Ako zdroj napätia pre logickú a mechanickú časť použijem 3x 3,6 V Li-ion batérie s kapacitou 2800 mAh

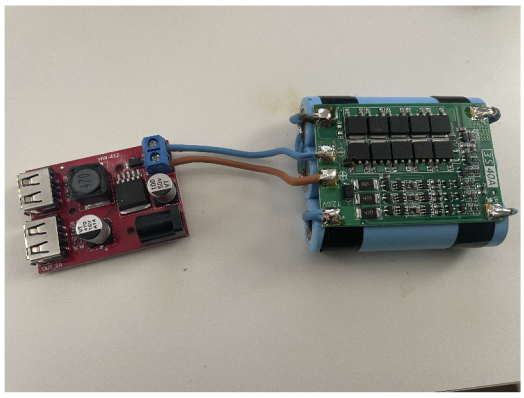
Menič na 5V: 3-A Step-Down napäťový regulátor LM2596 [14] na module HW-412 sa použije ako menič na 5V pre logickú časť projektu.

Ochranný obvod pre batérie: BMS-40A-3S ochranný obvod pre batérie [15], ktorý slúži na ochranu voči nadprúdom a nadmernému vybitiu batérií



Obr. 3.9: Zapojenie Batérií k ochrannému obvodu BMS-40A-3S [15]

Na obrázku č.3.10 môžeme vidieť už vyhotovený prenosný zdroj, ktorý bude poháňať Elektronické šachy.



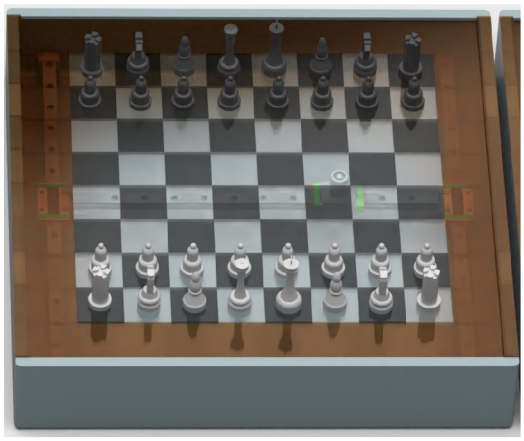
Obr. 3.10: Zdroj napájania pre Elektronické šachy

3.3 Hardvér Elektronických šachov

Návrh správneho hardvéru je kľúčovým prvkom, ktorý ovplyvňuje funkčnosť a celkový zážitok zo hry. V tejto časti sa budem venovať návrhu pohybového mechanizmu a spomenú sa taktiež súčiastky, ktoré boli potrebné na konštrukciu a nakoniec popíšem zapojenie riadiacej jednotky s jednotlivými komponentami.

3.3.1 Návrh mechanizmu elektronických šachov

V tejto kapitole predstavím model elektrických šachov a jeho približný mechanizmus. Všetky modely sú vytvorené v programe NX 1899 [10]. A následne ukážem výsledný mechanizmus skonštruovaný na základe vytvoreného modelu.

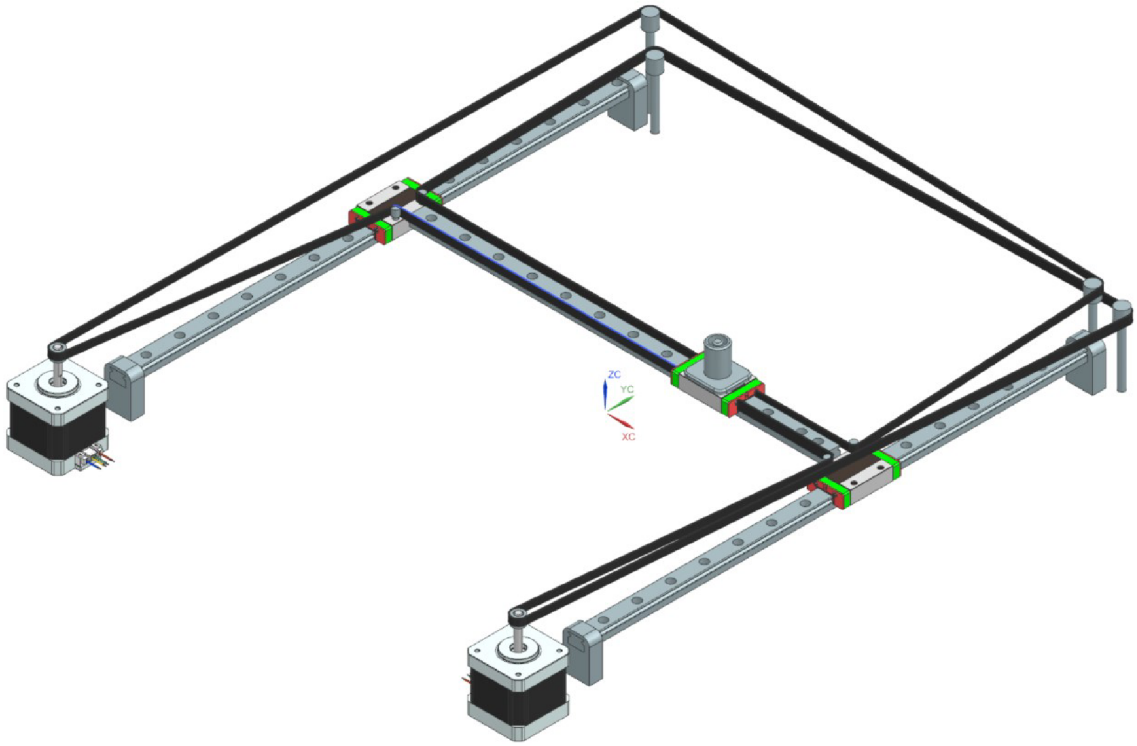


Obr. 3.11: Model elektrických šachov

Mechanizmus sa bude skladať z:

- 2x krokový motor Nema 17 (17HS4401S)
- 3x lineárny vozík MGN12H

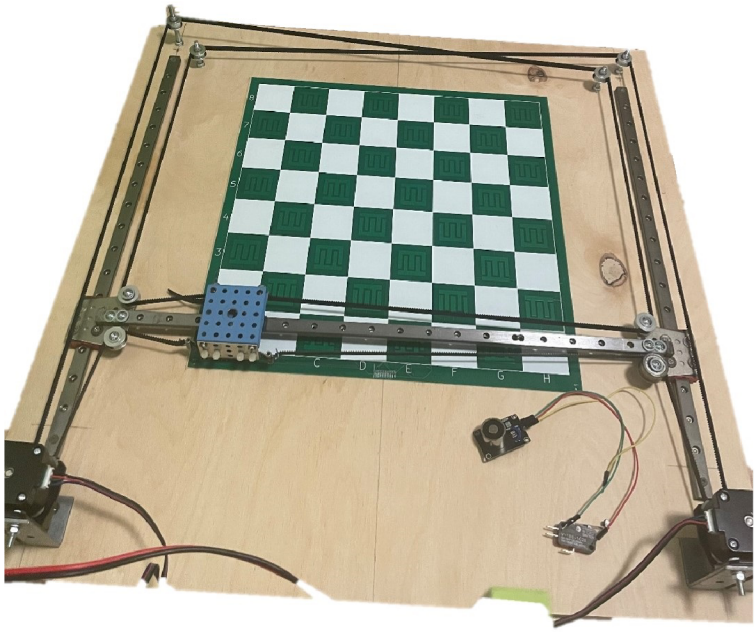
- 3x lineárne vedenie na MGN12H (40cm)
- remeň GT2 6mm (4 m)
- 6x remenica GT2 6 mm (20 zubov)
- 2x remenica GT2 6 mm bez zubov



Obr. 3.12: Model mechanizmu

Mechanizmus je ako bolo už spomenuté inšpirovaný systémom CoreXY 1.8. Hlavnou zložkou sú 2 krokové motory (NEMA 17), ktoré pohybujú elektromagnet umiestnený na vozíku MGN12H. Pohyb do určenej pozície bude závisieť na tom, akým smerom a rýchlosťou sa jednotlivé motory pohybujú v závislosti na sebe. Ako môžeme vidieť motory sú stacionárne a tak nebude pohybový mechanizmus zbytočne zaťažený váhou jedného motora, tým pádom by mohli byť použité aj slabšie krokové motorý, ako napríklad NEMA 14. Malou nevýhodou je iba zväčšená komplikovanosť ovládania mechanizmu.

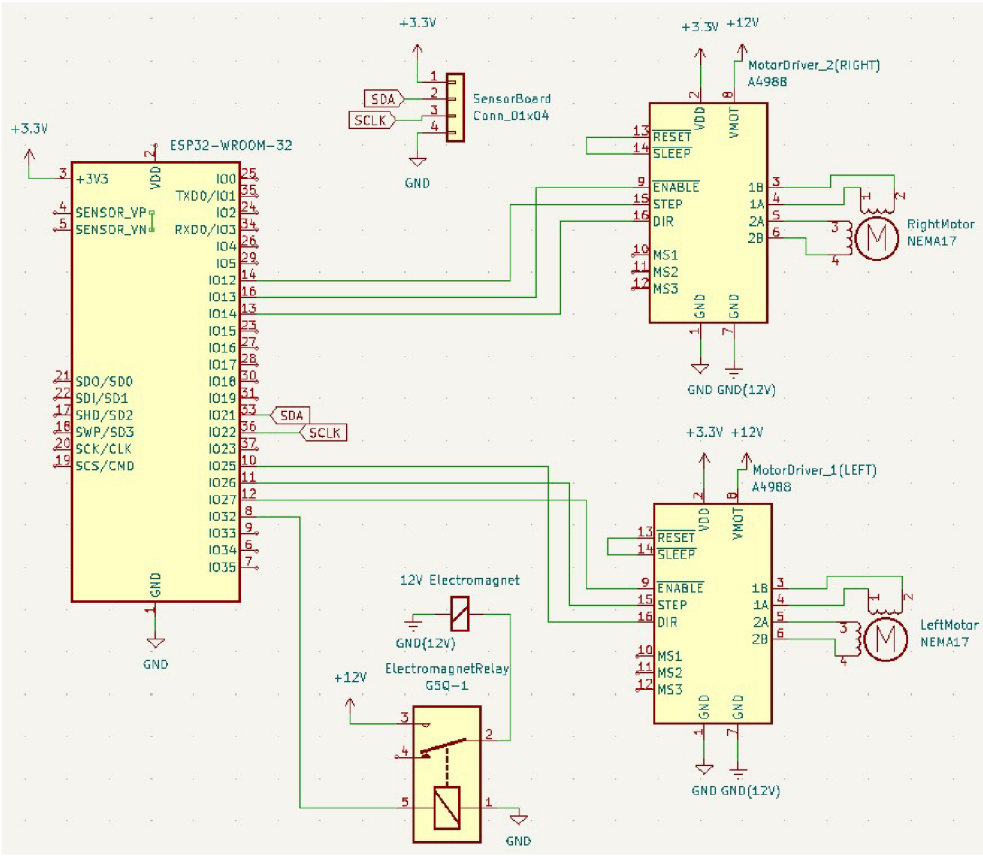
Napriek tomu, že súčiastky, ktoré sa používajú pri konštrukcii sú kvalitné, tak presnosť pohybu posuvnej hlavice by mohlo ovplyvňovať zlé napnutie remeňov. Správne napnutie remeňov je veľmi dôležité najmä kvôli dĺžke remeňov. Je isté, že remene sa budú časom povoľovať a tak tento problém by sa dal riešiť meneným pozície motorov, to znamená, že ak by boli remene veľmi povolené posunuli by sme krokové motory o kúsok dozadu a tým by sa napol aj remeň. Na obrázku 3.13 môžeme vidieť pohybový mechanizmus inšpirovaný modelom.



Obr. 3.13: Mechanizmus Elektronických šachov

3.3.2 Zapojenie riadiacej jednotky s komponentami

Z obrázka č.3.14 vidíme, že na ľavej strane sa nachádza riadiaca jednotka ESP32 bude napájaná zo Step-Down napäťového regulátora LM2596 [14], a bude ovládať ovládače krokového motora A4988 [11] na pravej strane, kde cez piny GPIO(12,26) bude posielať krokové pulzy na STEP vstup ovládača A4988 pravého a ľavého motora. GPIO(14,25) budú ovládať smer točenia motorov a GPIO(13,27) budú povoľovať točenie motorov. Ovládače sú napájané z mikrokontroléra 3,3 Voltami a motory 12 V z batérií. Na piny 21 a 22 sú zapojené SDA a SCLK pre I^2C komunikáciu so senzormi. A nakoniec bude GPIO32 ovládať relé ktoré bude spínať 12 V elektromagnet, ktorý bude hýbať figúrkami.



Obr. 3.14: Schéma zapojenia riadiacej jednotky s výstupnými perifériami

4 Riadiaci softvér elektronických šachov

V tejto sekcii popíšem softvérové riešenie Spracovania dát zo senzorov, ovládanie pohybového mechanizmu a nakoniec popíšem princíp šachového softvéra, ktorý v tomto projekte používam. ESP32 je možné programovať vo viacerých jazykoch, ja som sa rozhodol pre Micropython práve pre jeho jednoduchosť a prehľadnosť.

4.1 Program pre spracovanie dát zo senzorov

Pre snímanie prítomnosti figúrok je najprv dôležité vhodne kalibrovať všetkých 5 AD7147 [4] kontrolérov, to znamená správne určenie offsetu a senzitivity jednotlivých vstupov. Po správnej kalibrácii treba každý vstup pre senzor prečítať a usporiadať do poľa prvkov na takú pozíciu, akej zodpovedá na šachovnici.

AD7147 používa 16-bitové registry a má funkciu automaticky inkrementujúcich sa registrov po každom zápise, to znamená, že stačí vytvoriť pole bajtov, ktoré chceme za sebou zapísať s tým, že nám stačí celé pole vpísať na prvú adresu registra do ktorého chceme vpísať prvé dva bajty v poli a potom po každých dvoch vpísaných bajtoch (1 WORD) sa adresa registra automaticky inkrementuje. Čo znamená, že sa nemusí jednotlivo pre každý jeden WORD dát udávať adresa registra, na ktorý chceme tieto dáta napísať. AD7147 má 3 banky registrov pre túto aplikáciu budú stačiť iba prvé dve. Banka 1 je pre nakonfigurovanie operačných módov, povolenie kalibrácie na jednotlivých senzoroch, aktiváciu prerušení ale taktiež pre čítanie statusu prerušenia, čo budem využívať na určenie prítomnosti figúrky.

Banka 2 slúži na pripojenie jednotlivých senzorov k pozitívnemu alebo negatívnemu vstupu prevodníka, ale taktiež na nastavenie citlivosti daných senzorov a ich offsetov.

Dáta v registroch pre citlivosť, offset a kalibráciu prahov som nastavil pre všetky senzory rovnaké:

Kde:

AFE_OFFSET register určuje posunutie rozsahu, na ktorom sa meria kapacita, tak ako je spomenuté v časti Meranie č.4 3.1.2

SENSITIVITY register je na nastavenie senzitivity jednotlivých senzorov. Na obrázku č.4.1 sú nastavené samé nuly, čo znamená, že aktivácia senzora sa zaznamená pri prekročení 25% nastaveného prahu.(najvyššia senzitivita)

HIGH/LOW_CLAMP register tento register je používaný iba internými algoritmami na prahovú kalibráciu. Hodnota v tomto registri bráni tomu, aby používateľ spôsobil prekročenie očakávanej nominálnej hodnoty výstupu snímača.

```
'''VALUES FOR BANK 2'''
AFE_OFFSET_15_8     = 0x2F
AFE_OFFSET_7_0      = 0x2F
SENSITIVITY_15_8    = 0b00000000
SENSITIVITY_7_0     = 0b00000000

HIGH_CLAMP_15_8     = 0x40
HIGH_CLAMP_7_0      = 0x00
LOW_CLAMP_15_8      = 0x40
LOW_CLAMP_7_0       = 0x00

offLow = int(((LOW_CLAMP_15_8 << 8) | LOW_CLAMP_7_0) * 0.8)
offHigh = int(((HIGH_CLAMP_15_8 << 8) | HIGH_CLAMP_7_0) * 0.8)

#set to 80% of the H/L_CLAMP register
OFFSET_LOW_15_8     = (offLow >> 8)
OFFSET_LOW_7_0      = 0xFF & offLow
OFFSET_HIGH_15_8    = (offHigh >> 8)
OFFSET_HIGH_7_0     = 0xFF & offHigh
```

Obr. 4.1: Nastavenie citlivosti,offsetu a kalibrácie prahov v BANK 2

OFFSET_HIGH/LOW je nastavený na 80% HIGH/LOW_CLAMP registra.

Nastavenie registrov tak, ako je na obrázku 4.1 sa osvedčilo ako najlepšia konfigurácia na základe viacerých pokusov.

Keďže každý kontrolér má iba 12 prevodných stupňov ale 13 vstupov musel som to vyriešiť tak, že u štyroch kontrolérov AD7147 som vždy pripojil na jeden z dvanástich stupňov dva senzory, jeden na negatívny vstup prevodníka a jeden na pozitívny, to má za následok potom to, že iba jeden z týchto dvoch senzorov môže byť aktivovaný v rovnakom čase. Riešiť by sa to dalo navrhnutím novej dosky so šiestimi kontrolérmi AD7147. Ale keďže by to bolo finančne náročné nezaoberal som sa týmto problémom.

Pre spracovanie dát zo senzorov a určenie ich pozície som urobil funkciu, ktorá bude čítať status aktivácie každého senzora (prekročenie prahu) a bude tieto dáta zapisovať do dvoj dimenzionálneho poľa na takú pozíciu, na akej sú na šachovnici, to znamená, že napríklad senzor, ktorý sa nachádza na políčku E4 bude zapísaný do poľa na pozíciu [5][4]. Na obrázku č.4.2 je znázornená funkcia, ktorá zápis z registrov do poľa vykonáva.

Kde **HIGH/LOW_INT_STATUS** sa skladá zo 16-bitov, z ktorých prvých 12 udáva aktiváciu senzora v každom prevodnom stupni (ak sa bit rovná 1, to znamená, že senzor je aktivovaný). Jednotlivé bity získavam vďaka pomocnej funkcie GetBit() ktorá prečíta bit na danej pozícii.

```
'''TRANSFER INTERRUPT DATA FROM REGISTERS TO ARRAY'''
# Interrupt bits will be saved in array in the same position-
# -as they are on the chess board where first indexes will portray-
# -the number of collumn and second number of row array[column][row]
# for examp. array[5][4] will be E4 position on the board
def RegToArray(self, array):
    #For Slave1
    INT_data1H = self.reg_read(self.Slave1_ADD_AD7147, self.STAGE_HIGH_INT_STATUS,1)
    INT_data1L = self.reg_read(self.Slave1_ADD_AD7147, self.STAGE_LOW_INT_STATUS,1)
    Hstatus1 = INT_data1H[0] << 8 | INT_data1H[1] # zkombinuj dva byty
    Lstatus1 = INT_data1L[0] << 8 | INT_data1L[1]
    array[1][8] = self.GetBit(Hstatus1, 0)
    array[2][8] = self.GetBit(Hstatus1, 1)
    array[5][6] = self.GetBit(Lstatus1, 1) # CIN12 Prepnut na LOW treshold
    array[3][8] = self.GetBit(Hstatus1, 2)
    array[4][8] = self.GetBit(Hstatus1, 3)
    array[5][8] = self.GetBit(Hstatus1, 4)
    array[6][8] = self.GetBit(Hstatus1, 5)
    array[7][8] = self.GetBit(Hstatus1, 6)
    array[8][8] = self.GetBit(Hstatus1, 7)
    array[8][7] = self.GetBit(Hstatus1, 8)
    array[7][7] = self.GetBit(Hstatus1, 9)
    array[6][7] = self.GetBit(Hstatus1, 10)
    array[5][7] = self.GetBit(Hstatus1, 11)
```

Obr. 4.2: Zápis statusu senzorov do poľa prvkov pre 1. kontrolér AD7147

4.2 Šachový softvér

Ako šachový softvér je v tomto projekte použitý softvér s názvom SunFish Chess Engine, ktorý bol prerobený a optimalizovaný pre Micropython aby sa zmestil do obmedzenej pamäti mikrokontroléra ESP32 (túto modifikáciu som nerobil ja). Je to king-capture engine, čo znamená, že hra končí až keď užívateľ alebo engine zoberie kráľa oponentovi. Funguje na bázy **alpha-beta prerezávaní** s možnosťou pamätania si predchádzajúcich pozícií s cieľom, že pokiaľ v minulosti našiel engine dobrý ťah, tak ho pri hľadaní uprednostní pred ostatnými, lebo je veľká šanca, že bude dobrý aj v prítomnosti. Oproti tomuto šachovému softvéru sa dá zatiaľ hrať iba za bieleho. Na obrázku č.4.3 je vidieť reprezentácia počiatočnej šachovej pozície počítača na začiatku hry, ktorá sa počas hry mení.

```
# Our board is represented as a 120 character string. The padding allows for
# fast detection of moves that don't stay within the board.
A1, H1, A8, H8 = 91, 98, 21, 28
initial = (
    "          \n"  #   0 -  9
    "          \n"  #  10 - 19
    " rnbqkbnr\n"  #  20 - 29
    " pppppppp\n"  #  30 - 39
    " ........\n"  #  40 - 49
    " ........\n"  #  50 - 59
    " ........\n"  #  60 - 69
    " ........\n"  #  70 - 79
    " PPPPPPPP\n"  #  80 - 89
    " RNBQKBNR\n"  #  90 - 99
    "          \n"  # 100 -109
    "          \n"  # 110 -119
)
```

Obr. 4.3: Počiatočná pozícia šachového softvéru

4.3 Program pohybového mechanizmu

Táto kapitola sa bude zaoberať ovládaním pohybového mechanizmu na základe pokynov šachového softvéru, ukážem spôsob akým bude riešené **vyhadzovanie** , **ťahy koňa** a **rošáda**.

Celý mechanizmus bude ovládať funkcia EngineMove, ktorá bude obsahovať všetky následujúce kúsky kódu.

```
# transfers data from move generated by engine and tells-
# motors what to do ...
# Inputs: move - move generated by search() method
#         capture - True if engine is capturing a piece, False if regular move
#         KingMove - if True move is KingMove
#         KnightMove - if True move is a KnightMove and need to prevent collision with other pieces
def EngineMove(self, move, capture = False, KingMove = False, KnightMove = False):
    self.render(move)
    wait = 300  # wait after moving a piece
    offset = 20 # offset a piece by 20 mm
```

Obr. 4.4: Funkcia EngineMove

Ako môžeme vidieť z obrázku č.4.4 funkcia EngineMove má 4 vstupné parametre, z ktorých prvý je parameter **move** je to ťah počítača, ktorý je na začiatku funkcie prevedený pomocou funkcie render na 4 čísla, z ktorých prvé dve uloží do fil[0] a rank[0], čo predstavuje číslo stĺpca a číslo riadku figúrky, ktorou sa hýbe a do fil[1] a rank[1] sa uložia súradnice na šachovnici kam sa má táto figúrka pohnúť. Ďalej ako druhý parameter je **capture**, ktorý ak je True, tak dáva funkcii najavo že sa jedná o vyhadzovanie figúrky. Tretí parameter je **KingMove** ak je True znamená, že ide o ťah kráľom, tento parameter sám o sebe nenesie žiadnu váhu ale pokiaľ sa jedná o ťah kráľom a zároveň o posunutie o dve políčka ide o rošádu. A ako posledný parameter je **KnightMove**, ktorý signalizuje že ide o ťah koňom.

```
# NORMAL MOVE
if (not KnightMove and not (KingMove and (abs(self.fil[1]-self.fil[0]) == 2 ))):
    self.setTarget(self.fil[0], self.rank[0])
    self.Transfer(self.TargetPos)
    magnetPin.on()
    sleep_ms(wait)
    self.setTarget(self.fil[1], self.rank[1])
    self.Transfer(self.TargetPos)
    magnetPin.off()
    sleep_ms(wait)
    self.GoHome()
```

Obr. 4.5: Obyčajný ťah

Obyčajný ťah figúrkou: Ovládanie pozície elektromagnetu sa vykonáva pomocou funkcie Transfer, ktorá hýbe magnetom vertikálne a horizontálne, ale ak sa pri

ťahu zmena počtu riadkov rovná zmene počtu stĺpcov, tak sa vždy bude hýbať diagonálne. Kód 4.5 vykoná normálny ťah (ak sa nejedná o rošádu alebo o ťah koňom) kde neaktívny magnet najprv presunie z domovskej pozície pod políčkom A1 na pozíciu figúrky, ktorá robí ťah zapne magnet a následne ju presunie na žiadanú pozíciu a vypne magnet, nakoniec sa vráti späť na svoju domovskú pozíciu.

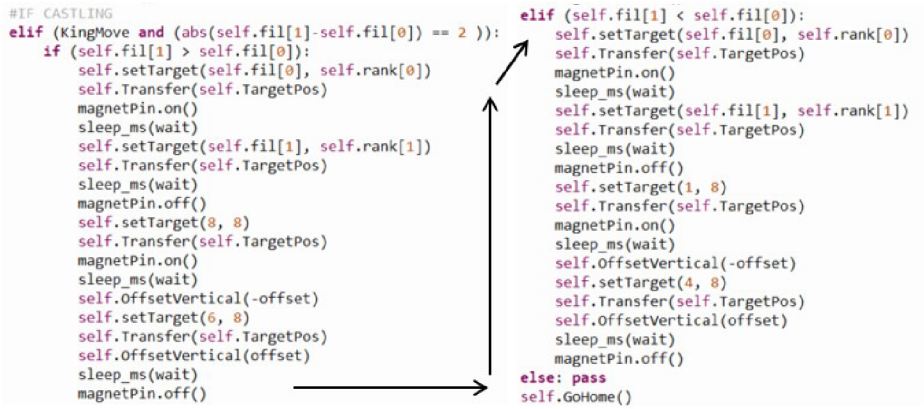
```
#TAKING A PIECE
if(capture):
    self.setTarget(self.fil[1], self.rank[1])
    self.Transfer(self.TargetPos)
    sleep_ms(wait)
    magnetPin.on()
    if(self.rank[1] >= 5):
        self.OffsetVertical(-offset)
    else:
        self.OffsetVertical(offset)

    sleep_ms(wait)
    self.setTarget( 1, self.rank[1])
    self.Transfer(self.TargetPos)
    self.OffsetHorizontal(-2*offset)
    magnetPin.off()
    sleep_ms(wait)
    self.OffsetHorizontal(2*offset)
    if(self.rank[1] >= 5):
        self.OffsetVertical(offset)
    else:
        self.OffsetVertical(-offset)
    self.setActual(1 , self.rank[1])
```

Obr. 4.6: Vyhadzovanie

Vyhadzovanie: Vyhadzovanie bude prebiehať takým spôsobom, že sa najprv magnet presunie pod vyhadzovanú figúrku a vertikálne ju posunie o 20 mm ku stredu medzi políčka následne ju vysunie na vyhradené miesto na vyhadzovanie, ktoré sa nachádza na kraji šachovnice a keďže sa figúrka počas vyhadzovania bude posúvať medzi políčkami nebude vrážať do ostatných, po ukončení vyhadzovania začne obyčajný ťah a figúrku, ktorá vyhadzuje presunie na žiadanú pozíciu.

Rošáda: Rošádu zistíme tak, že sa jedná o ťah kráľa a zároveň o posunutie o dve políčka, rieši sa spôsobom, že kráľ sa posunie o dve políčka doprava alebo doľava (záleží na tom, či ide o veľkú alebo malú rošádu) potom sa veža posunie medzi políčka (o 20 mm) smerom ku stredu dosky a presunie sa vedľa kráľa.

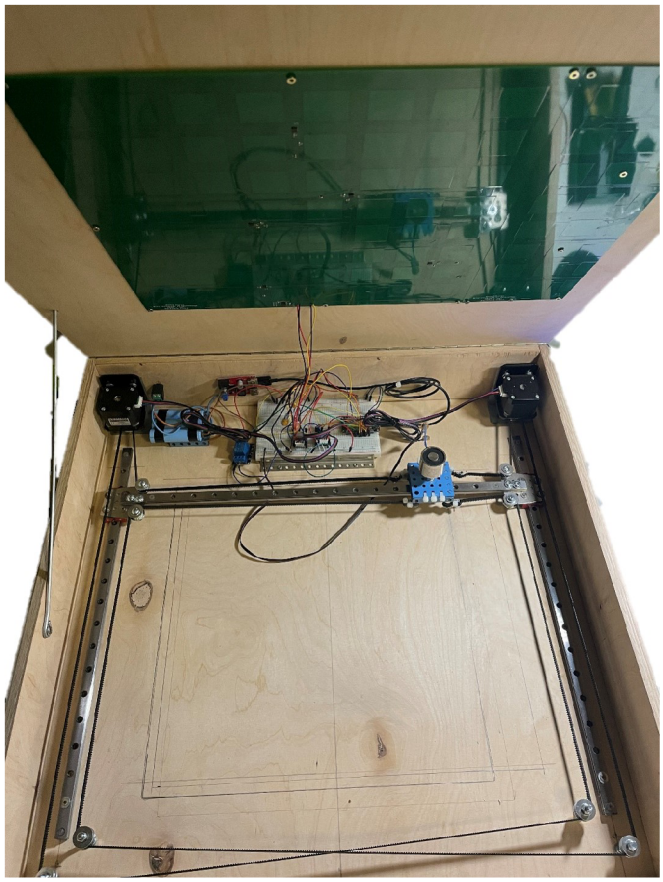
Ťah jazdca: Ťah jazdca je vyriešený tak, že sa najprv posunie medzi políčka vzhľadom na to akým smerom sa hýbe, vykoná ťah medzi políčkami tak aby nenarazil do ostatných figúrok a potom sa späť zasunie na svoje finálne políčko. Toto je znázornené na obrázku č.D.1.



Obr. 4.7: Rošáda

Nekonečný (While) cyklus: V prílohe D.2 vidíme, že v cykle voláme už spomenutú funkciu RegToArray, ktorá zapisuje dáta zo senzorov do dvoj-dimenzionálneho poľa prvkov ActualPos toto pole prvkom kontrolujeme každý krát čo prebehne cyklus porovnaním s minulou pozíciou. Ak sa zaznamená zmena, uložia sa súradnice políčka na ktorom nastala prvá zmena do poľa UserMove, pri vytvorení ďalšej zmeny je už ťah kompletný a na základe toho, či u druhej zmeny je daný senzor aktívny alebo nie určíme či sa jednalo o normálny ťah alebo o vyhadzovanie a na základe toho uložíme súradnice ťahu do UserMove. Indexy poľa ActualPos pred zapísaním do UserMove parsujeme na súradnicu, ktorej šachový softvér pochopí. Potom čo sme vykonali kompletný ťah sa zapne šachový softvér, do ktorého sa zapíše ťah užívateľa (UserMove) spustí sa vyhľadávací algoritmus a po vyhľadaní ťahu počítača sa skontroluje o aký ťah ide po kontrole sa zavolá funkcia EngineMove, do ktorej sa predá ťah počítača spolu s premennými, ktoré udávajú o aký ťah ide a ťah sa vykoná.

Obr. 4.8: Elektronické šachy ako celok



Obr. 4.9: Elektronické šachy z vnútra

Záver

V tejto práci som sa v úvode zoznámil s princípmi fungovania kapacitných senzorov na doske plošných spojov, tieto princípy mi ďalej pomohli pri následnom dizajne kapacitných senzorov na doske plošných spojov. Následne bolo taktiež spísané fungovanie a princíp CoreXY mechanizmu a aj jeho výhody a nevýhody. V kapitole o koncepcii riešení práce som diskutoval o postupe podľa, ktorého som navrhol a vyhotovil celý projekt, taktiež boli spomenuté dôvody na vybranie jednotlivých komponentov a zvolenie určitých metód.

V nasledujúcej kapitole o návrhu hardvéru som pomocou viacerých meraní navrhol a overil funkčnosť kapacitného senzora, ktorý bude použitý vo výslednom dizajne. V meraní č.1 3.1.1 sa ukázalo, že ak sa zväčší plocha senzorovej elektródy, tak sa zlepší aj senzitivita a dosah senzoru, čo vytvorí lepšie podmienky pre detekovateľnosť predmetov. V meraní č.2 3.1.1 som sa snažil zistiť, ako vplýva medzera medzi elektródami kapacitného senzora na výslednú zmenu kapacity, toto meranie nebolo optimálne prevedené, pretože pri zväčšovaní medzery sa zmenšovala plocha senzora a tak sa nedal presne určiť vplyv veľkosti medzery na výsledné snímanie predmetu. Následne som sa v meraní č.3 3.1.1 dostal k záveru, že pri kapacitných senzoroch na doske plošných spojov nie je senzitivita ani dosah natoľko veľký aby sa naň mohla položiť šachovnicová doska (napr. z dreva), tým pádom som sa rozhodol, že sa bude hrať rovno na doske plošných spojov. Oboznámil som sa s tienením elektrického poľa senzoru a urobil som viaceré merania pre odlišné konfigurácie umiestnenia tienenia. A nakoniec som zistili že pre tento projekt bude dostačujúce umiestnenie tienenia iba zo spodku senzorov. Z materiálov, ktoré sme merali je jasne vidieť, že najlepšie detekovateľné predmety pomocou kapacitného senzora sú predmety, ktoré sa skladajú z kovu. Následne som pred finálnym dizajnom navrhol DPS na odskúšanie funkčnosti AD7147 kontroléra, ale hlavne aby som sa ho naučil programovať. V meraní č.4 3.1.2 som teda overoval správnu funkčnosť a to či vôbec dokáže snímať neuzemnené predmety. Nakoniec som zistil, že som objednal AD7147 s I^2C protokolom namiesto SPI, ale ukázalo sa, že kontrolér dokázal snímať neuzemnené predmety veľmi dobre. A preto som mohol s chladnou hlavou navrhnúť finálny dizajn senzorov. Ďalej som navrhol prenosný zdroj napájania, ktorý poskytuje 12V pre motory a elektromagnet a 5V pre mikrokontrolér ESP32. Pri návrhu mechanizmu som ukázal metódu, ktorou bude riešený mechanizmus s názvom CoreXY 1.3, podľa ktorého bol následne vytvorený približný model 3.12 v programe NX. Potom bol spísaný zoznam súčiastok, ktorých bolo pri zhotovení mechanizmu potreba a následne bol podľa modelu mechanizmus vyhotovený. Dôležitým aspektom pre presný pohyb mechanizmu je dobre napnutý remeň.

V kapitole o softvéri bolo popísané ovládanie a správne zapisovanie a čítanie z

registrov kontroléra AD7147 a následne bol popísaný spôsob akým sa čítali a ukladali jednotlivé statusy senzorov pod políčkami. Ďalej bol popísaný stručný princíp a chovanie šachového softvéra a spôsob akým si ukladá pozíciu. Nakoniec bolo riešené ovládanie pohybového mechanizmu na základe pokynov z šachového softvéra, kde sa riešili ťahy koňom, rošáda, a vyhadzovanie ku koncu bola predstavená while slučka, ktorá zlúčila všetky časti programu do jedného. Elektronické šachy by sa dali vylepšiť sprostredkovaním rozhrania medzi užívateľom a počítačom v podobe LCD alebo OLED displeja, ktorý by hráčovi hovoril čo sa deje počas hry. Ďalej by išli rozšíriť o funkciu vybrania si strany za akú chceme hrať, keďže sa používa mikrokontrolér ESP32 s možnosťou pripojenia na Wi-Fi išlo by pri väčšom počte Elektronických šachov sprostredkovať hru medzi hráčmi na väčšie vzdialenosti.

Literatúra

[1] Texas Instruments – SNOA928A – Application Report – [cit. 2022-11-13] *Capacitive Proximity Sensing Using the FDC1004.* [online]. Dostupné z URL: <https://www.ti.com/lit/an/snoa928a/snoa928a.pdf?ts=1668380371360&ref_url=https%253A%252F%252Fwww.google.com%252F>.

[2] Texas Instruments – SNOA926A – Application Report – [cit. 2022-11-13] *Capacitive Sensing: Ins and Outs of Active Shielding* [online]. Dostupné z URL: <https://www.ti.com/lit/an/snoa926a/snoa926a.pdf?ts=1668380558269&ref_url=https%253A%252F%252Fwww.google.com%252F>.

[3] Microchip Technology Inc. – AN1492 – Application Report – [cit. 2022-11-13] *Microchip Capacitive Proximity Design Guide* [online]. – ISBN: 9781620770283 Dostupné z URL: <https://ww1.microchip.com/downloads/en/Appnotes/01492A.pdf>.

[4] *CapTouch Programmable Controller for Single-Electrode Capacitance Sensors: AD7147* [online]. Analog Devices, 2007 [cit. 2022-11-22]. Dostupné z URL: <https://www.analog.com/media/en/technical-documentation/data-sheets/AD7147.pdf>

[5] Texas Instruments – SNOA927A – Application Report – [cit. 2022-11-13] *Basics of Capacitive Sensing and Applications* [online]. Dostupné z URL: <https://www.ti.com/lit/an/snoa927a/snoa927a.pdf?ts=1671818536631&ref_url=https%253A%252F%252Fwww.google.com%252F>.

[6] MOYER, Ilan. CoreXY: Cartesian Motion Platform. *CoreXY* [online]. 2012, [cit. 2022-12-25]. Dostupné z: <https://corexy.com/>

[7] *ESP32 Series datasheet* [online]. Verzia 4.1. Shanghai: Espressif Systems, 2022 [cit. 2022-12-25]. Dostupné z: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>

[8] *SEN-MAG25N: Electromagnet module with 25 N adhesive force* [online]. Joy-IT, 2020 [cit. 2022-12-26]. Dostupné z: <https://joy-it.net/files/files/Produkte/SEN-MAG25N/Datasheet%20SEN-MAG25N.pdf>

[9] *FDC1004QEVM: User Guide* [online]. Dallas: Texas Instruments, 2015 [cit. 2022-12-27]. SNAU178. Dostupné z: <https://www.ti.com/lit/ug/snau178/snau178.pdf?ts=1672097113779&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FFDC1004QEVM>

[10] *NX: Siemens Software* [online]. Yerevan: Siemens [cit. 2022-12-29]. Dostupné z: <https://www.plm.automation.siemens.com/global/en/products/nx/

[11] *A4988: DMOS Microstepping Driver with Translator And Overcurrent Protection* [online]. Massachusetts: Allegro MicroSystems, 2009 [cit. 2022-12-30]. Dostupné z: <https://www.pololu.com/file/download/a4988_DMOS_microstepping_driver_with_translator.pdf?file_id=0J450

[12] *CS GANTRY SERIES* [online]. Los Angeles: Newmark Systems, 2018 [cit. 2023-05-11]. Dostupné z: <http://newmarksystems.com/datasheets/CS-Series-XY-Gantry-Datasheet.pdf

[13] *TCA9548A: Low-Voltage 8-Channel I2C Switch with Reset* [online]. Dallas, Texas: Texas Instruments, 2012 [cit. 2023-05-12]. Dostupné z: <https://www.ti.com/lit/ds/symlink/tca9548a.pdf?ts=1683893899144&ref_url=https%253A%252F%252Fwww.google.com%252F

[14] *LM2596: 3-A Step-Down Voltage Regulator* [online]. Dallas: Texas Instruments, 1999 [cit. 2023-05-13]. Dostupné z: <https://www.ti.com/lit/ds/symlink/lm2596.pdf?ts=1683959486059

[15] *BMS-40A-3S: 3 cell 12.6V 40A 18650 lithium battery protection board* [online]. Shenzhen: Shenzhen Global technology Co., 2010 [cit. 2023-05-13]. Dostupné z: <http://www.mantech.co.za/Datasheets/Products/BMS-40A-3S_SGT.pdf

Zoznam symbolov a skratiek

Skratky:

CDC	Capacitance-to-Digital Converter
CNC	Computerized Numerical Control
CS	Chip Select
DPS	Doska Plošných Spojov
GPIO	General Purpose Input/Output
I^2C	Inter-Integrated Circuit
INT	Interrupt
LCD	Liquid Crystal Display
OLED	organic light-emitting diode
PVC	Polyvinyl Chloride
SCLK	Serial Clock
SDA	Serial Data
SDI	Serial Data In
SDO	Serial Data Out
SoC	System on Chip
SPI	Serial Peripheral Interface

Zoznam príloh

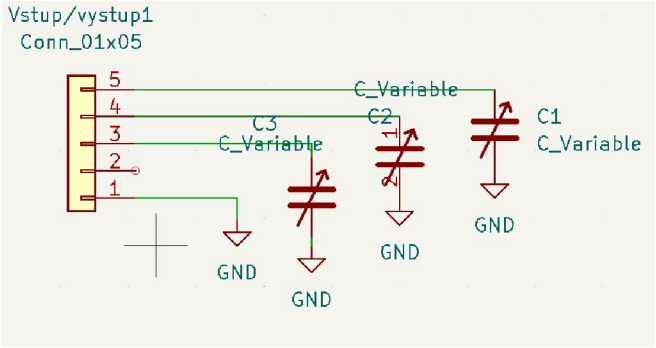
A Návrh skúšobnej DPS kapacitných senzorov **54**

B Návrh skúšobnej DPS pre AD7147 **56**

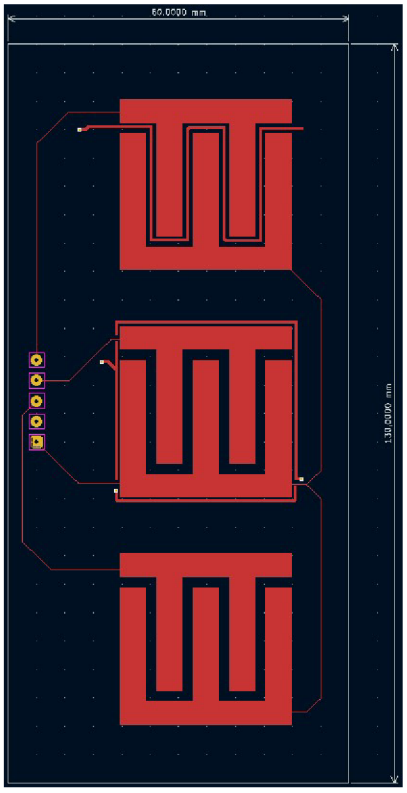
C Návrh šachovej dosky so senzormi **57**

D Kód pre Elektronické šachy **63**

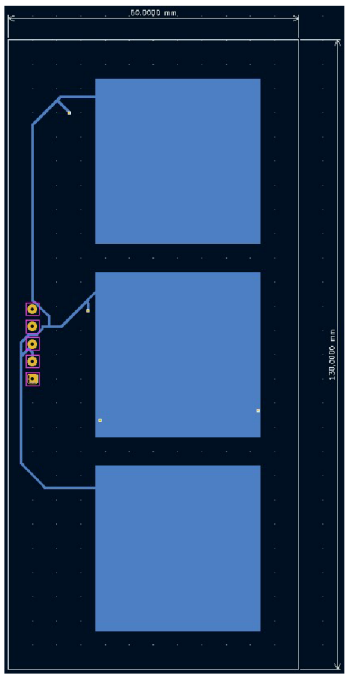
A Návrh skúšobnej DPS kapacitných senzorov



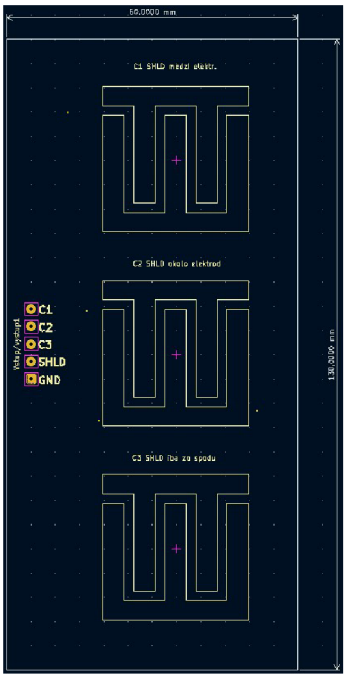
Obr. A.1: Schéma skúšobnej DPS kapacitných senzorov



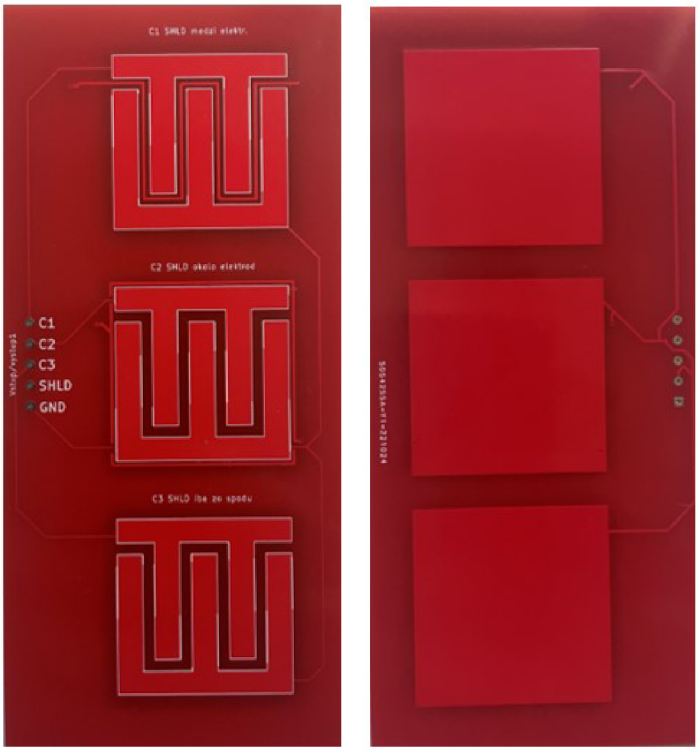
Obr. A.2: DPS skúšobnej dosky TOP



Obr. A.3: DPS skúšobnej dosky BOTTOM

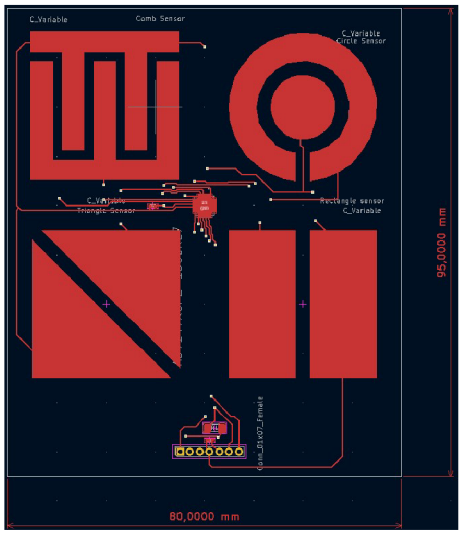


Obr. A.4: DPS skúšobnej dosky Tsilkscreen

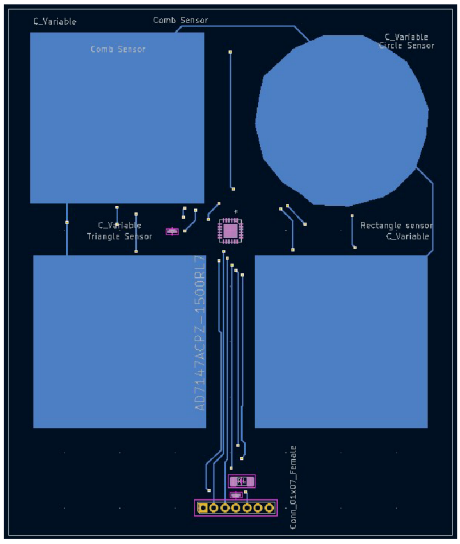


Obr. A.5: TOP,BOTTOM DPS skúšobná doska

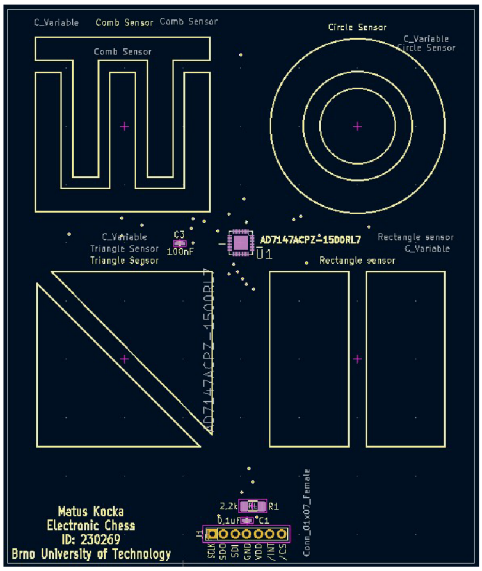
B Návrh skúšobnej DPS pre AD7147



Obr. B.1: skúšobná DPS pre AD7147 (TOP)

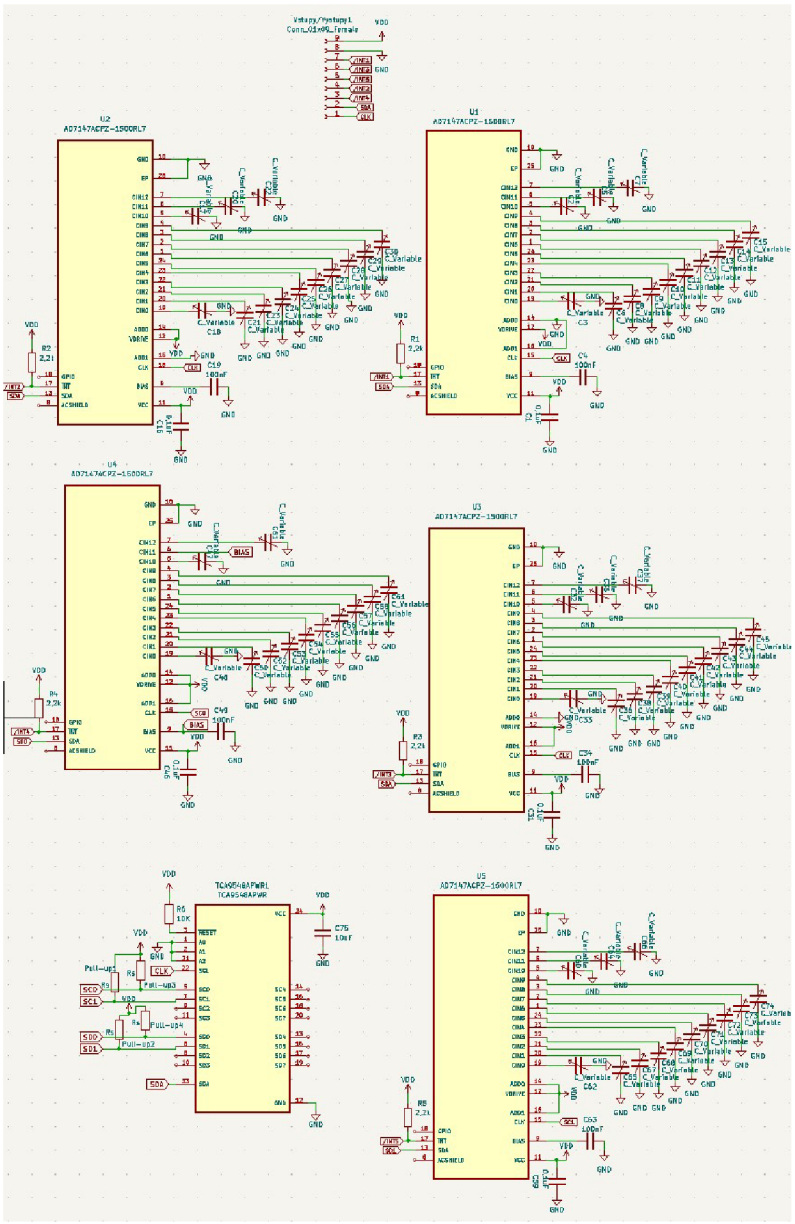


Obr. B.2: skúšobná DPS pre AD7147 (BOTTOM)

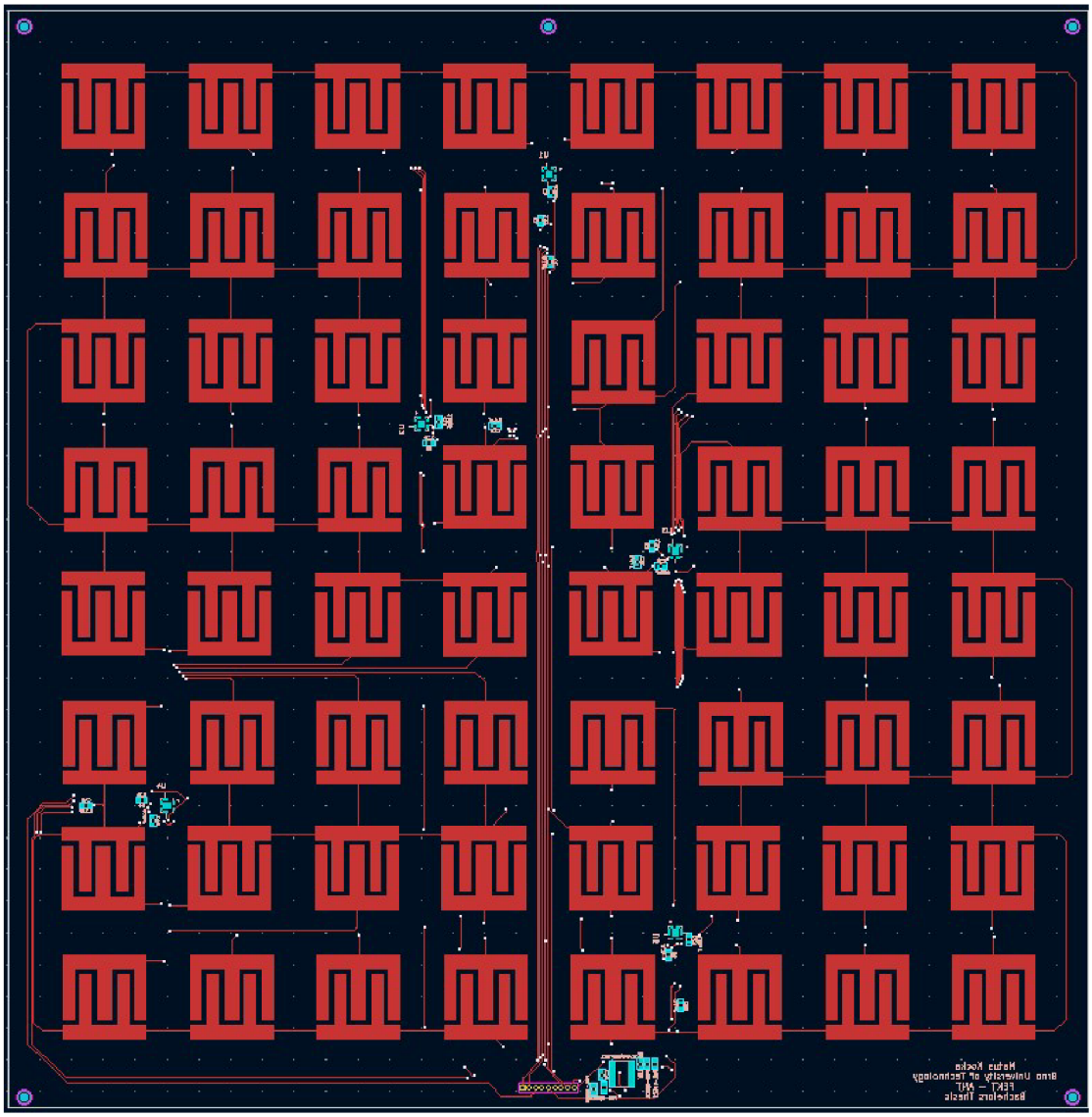


Obr. B.3: skúšobná DPS pre AD7147 (FSILK)

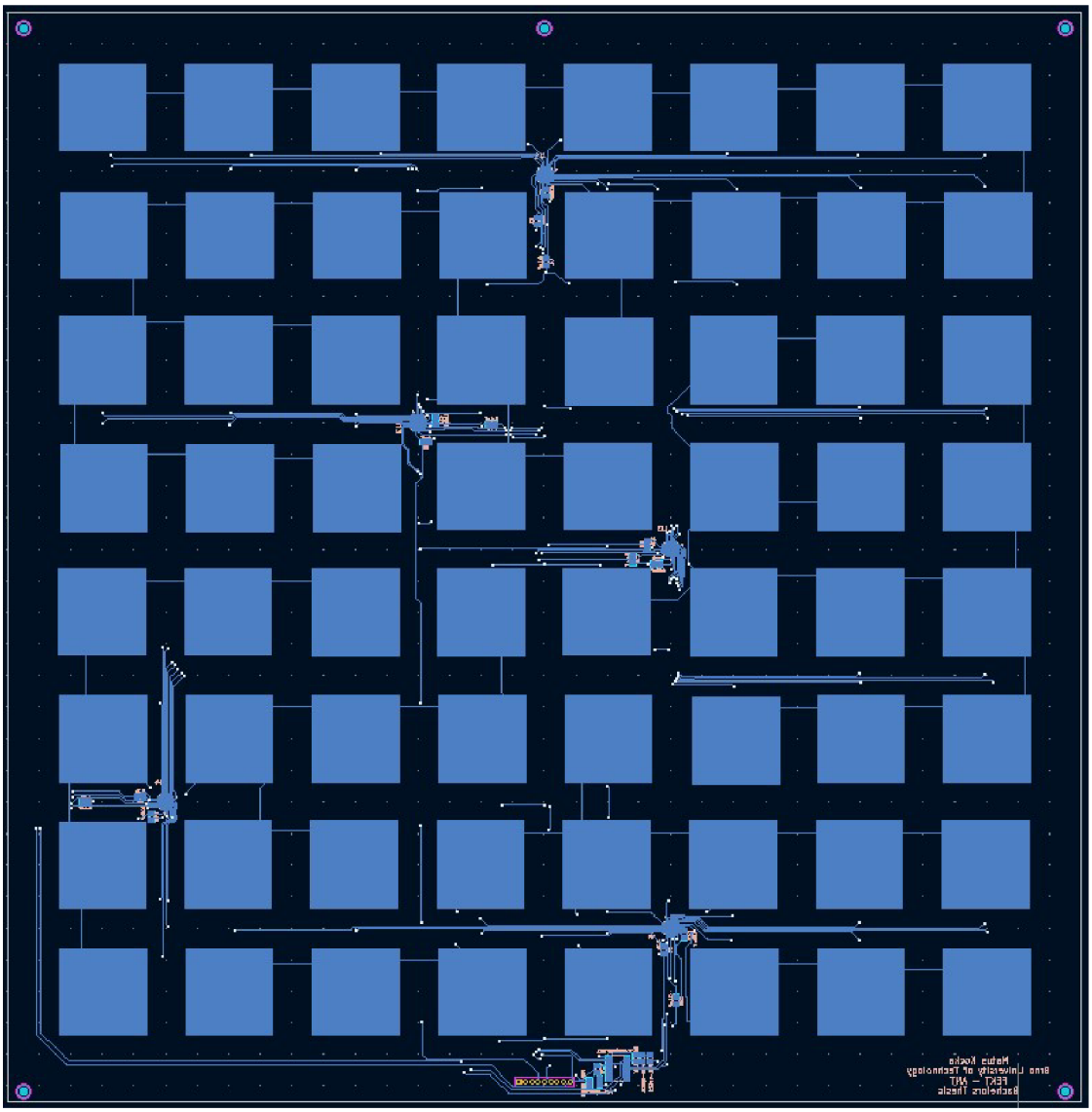
C Návrh šachovej dosky so senzormi



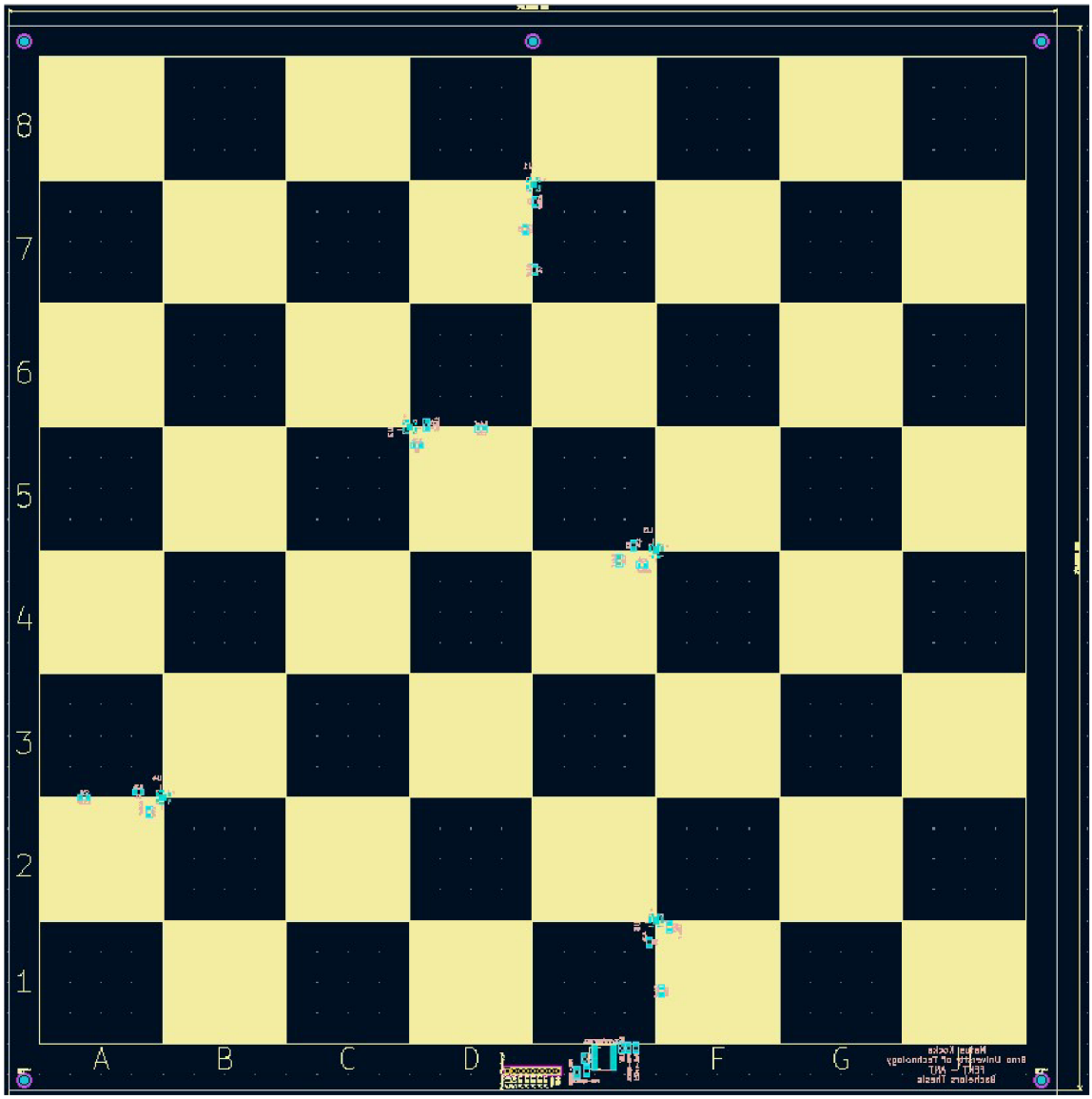
Obr. C.1: Návrh Šachovnice so senzormi Schéma



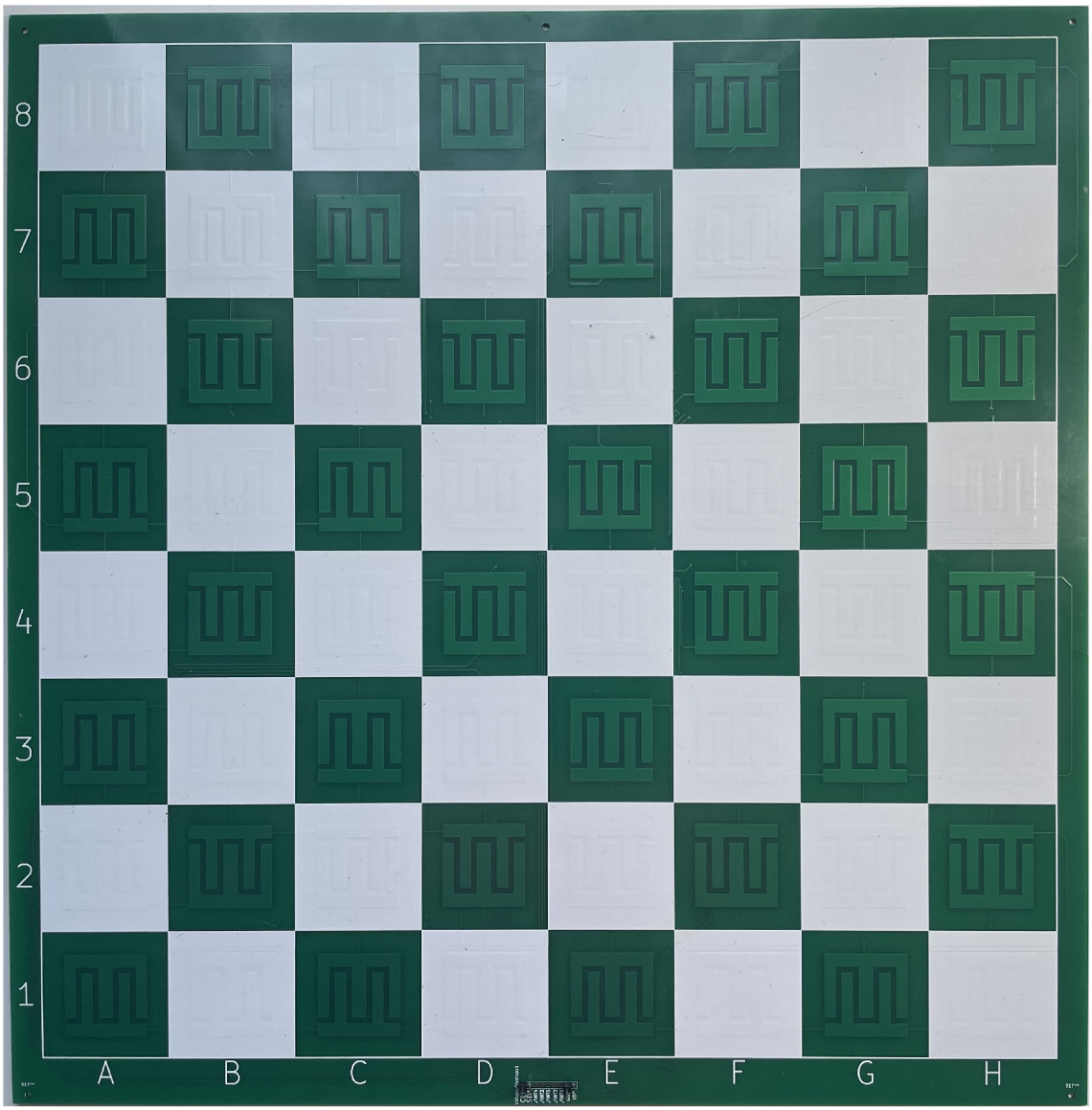
Obr. C.2: Návrh Šachovnice so senzormi TOP



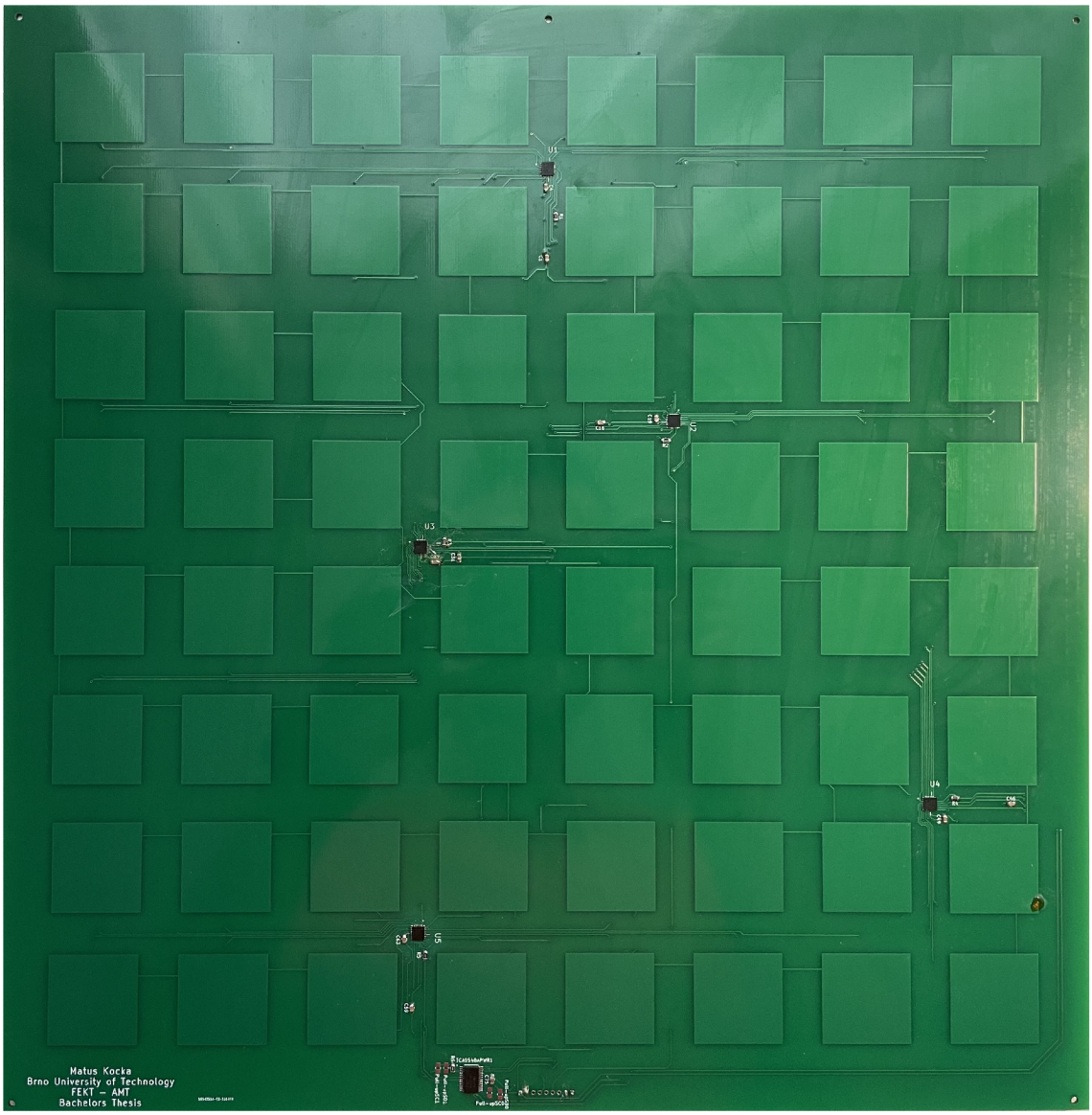
Obr. C.3: Návrh Šachovnice so senzormi BOTTOM



Obr. C.4: Návrh Šachovnice so senzormi FSILK



Obr. C.5: DPS Šachovnica so senzormi TOP



Obr. C.6: DPS Šachovnica so senzormi BOTTOM

D Kód pre Elektronické šachy

```
elif(KnightMove):
    self.setTarget(self.fil[0], self.rank[0])
    self.Transfer(self.TargetPos)
    sleep_ms(wait)
    #if knight moving two to the side and one up
    if ((abs(self.fil[1] - self.fil[0]) > abs(self.rank[1] - self.rank[0])) and self.rank[1]>self.rank[0]):
        self.OffsetVertical(offset)
        sleep_ms(wait)
        self.setTarget(self.fil[1], self.rank[0])
        self.Transfer(self.TargetPos)
        sleep_ms(wait)
        self.OffsetVertical(offset)
        sleep_ms(wait)
        self.setActual(self.fil[1], self.rank[1])
    #if knight moving two to the side and one down
    elif ((abs(self.fil[1] - self.fil[0]) > abs(self.rank[1] - self.rank[0])) and self.rank[1]<self.rank[0]):
        self.OffsetVertical(-offset)
        sleep_ms(wait)
        self.setTarget(self.fil[1], self.rank[0])
        self.Transfer(self.TargetPos)
        sleep_ms(wait)
        self.OffsetVertical(-offset)
        sleep_ms(wait)
        self.setActual(self.fil[1], self.rank[1])
    #if knight moving two vertically and one to the right
    elif ((abs(self.fil[1] - self.fil[0]) < abs(self.rank[1] - self.rank[0])) and self.fil[1]>self.fil[0]):
        self.OffsetHorizontal(offset)
        sleep_ms(wait)
        self.setTarget(self.fil[0], self.rank[1])
        self.Transfer(self.TargetPos)
        sleep_ms(wait)
        self.OffsetHorizontal(offset)
        sleep_ms(wait)
        self.setActual(self.fil[1], self.rank[1])
    #if knight moving two vertically and one to the left
    elif ((abs(self.fil[1] - self.fil[0]) < abs(self.rank[1] - self.rank[0])) and self.fil[1]<self.fil[0]):
        self.OffsetHorizontal(-offset)
        sleep_ms(wait)
        self.setTarget(self.fil[0], self.rank[1])
        self.Transfer(self.TargetPos)
        sleep_ms(wait)
        self.OffsetHorizontal(-offset)
        sleep_ms(wait)
        self.setActual(self.fil[1], self.rank[1])
    else: pass
    self.GoHome()
```

Obr. D.1: Ťah jazdca

```
while(True):
    CAPTURE = False
    KNIGHTMOVE = False
    KINGMOVE = False
    gc.collect()
    AD.RegToArray(ActualPos) # READ ALL CHESS SQUARES TO THIS 2D ARRAY every cycle

    #read move from user and convert to move readable by engine
    for x in range(1,fil):
        for y in range(1,rank):
            if not (ActualPos[x][y] == LastPos[x][y]):
                CNT += 1
                if CNT == 1:
                    print("CNT 1")
                    UserMove[0] = AD.parseSensorToEngine(x,y)
                    UserMove[1] = AD.parseSensorToEngine(x,y)
                    LastPos[x][y] = ActualPos[x][y]
                else:
                    print("CNT 2")
                    if not (ActualPos[x][y]):
                        UserMove[0] = AD.parseSensorToEngine(x,y)
                        LastPos = ActualPos
                    else:
                        UserMove[1] = AD.parseSensorToEngine(x,y)
                        LastPos = ActualPos
    if (CNT == 2):
        CNT = 0
        hist.append(hist[-1].move(UserMove))
        # Limit the history size. (QL)
        hist_remained = hist[-8:]
        hist.clear()
        hist.extend(hist_remained)

        #if hist[-1].score <= -MATE_LOWER:
            #print("You won")
            #break
        # Fire up the engine to look for a move.
        start = time.ticks_ms()
        for _depth, CompMove, score in searcher.search(hist[-1], hist):
            if time.ticks_diff(time.ticks_ms(), start) > 1000:
                break
        CompMoveArr = [0,0]
        CompMoveArr[0] = CompMove[0]
        CompMoveArr[1] = CompMove[1]
        #print(CompMoveArr)
```

Obr. D.2: While cyklus 1.časť

```
    if hist[-1].board[CompMove[0]] == "N":# or hist[-1].board[move[0]] == "n":
        KNIGHTMOVE = True
    if not(hist[-1].board[CompMove[1]] == "."):
        CAPTURE = True
    if hist[-1].board[CompMove[0]] == "K":# or hist[-1].board[move[0]] == "n":
        KINGMOVE = True
    hist.append(hist[-1].move(CompMove))

    mechanism.EngineMove(move = CompMoveArr, capture = CAPTURE, KingMove = KINGMOVE, KnightMove = KNIGHTMOVE)
    sleep_ms(200)
    AD.RegToArray(ActualPos)
    LastPos = ActualPos

sleep_ms(300)
```

Obr. D.3: While cyklus 2.časť