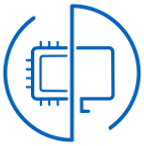


Střední průmyslová škola a Vyšší odborná škola, Písek, Karla Čapka 402, Písek

18-20-M/01 Informační technologie

Maturitní práce

Elektromechanická hra -> samořídící šachovnice

Téma číslo 4.

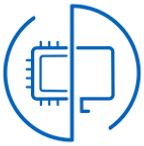
autor:

Václav Zíka, B4.I

vedoucí maturitní práce:

Mgr. Milan Janoušek

Písek 2024/2025



Střední průmyslová škola a Vyšší odborná škola, Písek, Karla Čapka 402, Písek

18-20-M/01 Informační technologie

Zadání maturitní práce

Elektromechanická hra -> samořídící šachovnice

Téma číslo 4.

Termín odevzdání:

31. 3. 2025

student/ka:

Václav Zíka, B4.I

vedoucí maturitní práce:

Mgr. Milan Janoušek

Písek 2024/2025

Zadání

1. Proveďte teoretický úvod k problematice samořídící šachovnice řešící realizaci šachové desky ovládané mikrokontrolérem, uživatelské ovládání šachovnice, vhodné komponenty (součástky, šachové figurky, hrací pole).

2. Realizujete vlastní řešení v následujících bodech

   (a) Navrhněte všechna potřebná schémata pro realizaci samořídící šachovnice.

   (b) Vyberte vhodný mikrokontrolér pro řízení šachovnice.

   (c) Vytvořte program pro zvolený mikrořadič.

   (d) Vyřešte pohyb figurek po šachovnici a sestrojte řešení.

   (e) Navrhněte způsob detekce obsazených polí na šachovnici.

   (f) Vyřešte komunikaci mezi řízením šachovnice a systémem, který bude simulovat tahy protihráče (např. serverem, algoritmem...).

   (g) Navrhněte a potřebnými součástkami osaďte desku šachovnice.

   (h) Realizujte konstrukci šachovnice.

   (i) Výsledné řešení prakticky ověřte.

3. Zpracujte dokumentaci dle metodického návrhu a ppt prezentaci pro účely obhajoby.

4. Propagujte výsledky své práce - např. vyhotovením posteru, účastí na SOČ, zhotovení informační www stránky, natočení promo videa apod.

Kritéria hodnocení maturitní práce

(nutné parametry práce, které musí být splněny, aby práce byla uznána a byla hodnocena)

1. splněn požadovaný minimální rozsah vlastního textu práce v rozsahu 15 stran textu s přiměřeným množstvím obrázků a tabulek nezbytně nutných k popisu/výkladu problému řešeného v textu

2. splněna struktura práce:

   (a) teoretický úvod k problematice řešené v práci v rozsahu max. 4 strany

(b) popis autorského řešení zadaného úkolu, doplněného výpočty, výkladem algoritmů, obrázky, které jsou nezbytně nutné k vyřešení částí zadání, v rozsahu min. 10 stran

(c) závěr hodnotící dosažené výsledky v rozsahu min. 1 normované strany

3. pokud práce nesplňuje předchozí dvě kritéria, je hodnocena: nedostatečně
   pokud jsou předchozí kritéria splněna, je práce hodnocena:

   (a) odpovědnost a přístup žáka v průběhu řešení zadání:
   vedoucí 0–10 %, oponent 0 %

   (b) dodržení obsahové a grafické struktury maturitní práce:
   vedoucí 0–10 %, oponent 0–10 %

   (c) originalita a vhodnost řešení:
   vedoucí 0–25 %, oponent 0–35 %
   (konkretizuje vedoucí práce ve 2 až 5 bodech podle požadovaných výstupů práce)

      i. výběr řídící jednotky
      ii. mechanické provedení šachovnice s políčky a figurkami
      iii. obsazenost detekujících se políček
      iv. možnost hry s protihráčem
      v. možnosti programového kódu

   (d) funkčnost řešení:
   vedoucí 0–30 %, oponent 0–30 %
   (vedoucí práce ve 2 až 5 bodech konkretizuje podle požadovaných kritérií funkčnosti)

      i. pohyb figurek na hracím poli
      ii. detekce obsazenosti políček
      iii. algoritmus vytvořeného programu
      iv. použitelnost k šachové hře

   (e) vlastní obhajoba:
   vedoucí 0–25 %, oponent 0–25 %

Klasifikační stupnice

1. výborný 84–100 %
2. chvalitebný 66–83 %
3. dobrý 48–65 %
4. dostatečný 31–47%
5. nedostatečný 0–30 %

Způsob zpracování a pokyny k obsahu a rozsahu maturitní práce

Práce bude zpracována podle platného metodického pokynu dostupného na n:\!maturita\MetodickýPokyn\...).

Kompletní práce se odevzdává do informačního střediska školy v jednom tištěném exemplář doplněném elektronickým nosičem dat (CD, DVD, USB flash disk, SD karta), na kterém bude uvedena kompletně zpracovaná práce včetně příloh. V případě tvorby software, také zdrojový kód navrženého software. V případě projektu, také projektová dokumentace (podrobná technická zpráva, úplná výkresová dokumentace, podrobný rozpočet).

Náklady na materiál bude hradit žák.
Funkční vzorek bude majetkem žáka.

V Písku 15. 11. 2024

Ing. Jiří Uhlík
ředitel SPŠ a VOŠ Písek

Anotace

Maturitní práce se zabývala tvorbou samořídící šachovnice, jejíž cílem bylo kompletně simulovat protihráče. Ať už se jedná o vymýšlení protitahu či o samotný manuální posun figurky. Šachovnice je ovládána mikrokontrolérem Arduino UNO. Pomocí magnetických spínačů umístěných na PCB desce detekujeme pozici figurek na hracím poli. Tyto informace následně zpracováváme Arduinem a skrze dva krokové motory a elektromagnet realizujeme tahy figurek. Pro zjištění ideálního příštího tahu využíváme Minimax algoritmus, jenž je schopný zevaluovat danou situaci a skrze programovou logiku udat příkazy pro pohyb figurek. Celá samořídící šachovnice je vyrobena z dřevěné konstrukce, do které je umístěna deska plošného spoje s elektronikou. V rámci projektu jsem si pomocí 3D tisku zhotovil magnetické šachové figurky. Šachovnice také umožňuje zvolení si obtížnosti a volbu hry buď za černé či bílé.

Anotace

Maturitní práce se zabývala tvorbou samořídící šachovnice, jejíž cílem bylo kompletně simulovat protihráče. Ať už se jedná o vymýšlení protitahu či o samotný manuální posun figurky. Šachovnice je ovládána mikrokontrolérem Arduino UNO. Pomocí magnetických spínačů umístěných na PCB desce detekujeme pozici figurek na hracím poli. Tyto informace následně zpracováváme Arduinem a skrze dva krokové motory a elektromagnet realizujeme tahy figurek. Pro zjištění ideálního příštího tahu využíváme Minimax algoritmus, jenž je schopný zevaluovat danou situaci a skrze programovou logiku udat příkazy pro pohyb figurek. Celá samořídící šachovnice je vyrobena z dřevěné konstrukce, do které je umístěna deska plošného spoje s elektronikou. V rámci projektu jsem si pomocí 3D tisku zhotovil magnetické šachové figurky. Šachovnice také umožňuje zvolení si obtížnosti a volbu hry buď za černé či bílé.

Klíčová slova: ....

Annotation

Aj...

Key words: ...

Poděkování

....

Licenční smlouva o podmínkách užití školního díla

ve smyslu zákona č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění (dále jen „AZ“), uzavřená mezi smluvními stranami:

1. **Autor práce: Václav Zíka**
   bytem Na Spravedlnosti 974/36, Písek 39701, Česká republika
   datum narození: 29. 12. 2005
   (dále jen „autor“)

   a

2. **Nabyvatel: Střední průmyslová škola a Vyšší odborná škola, Písek, Karla Čapka 402, Písek**
   397 11 Písek, Karla Čapka 402
   zastoupená ředitelem školy: Ing. Jiří Uhlík
   (dále jen SPŠ a VOŠ Písek)

Článek 1
Vymezení pojmů

1.1 Školním dílem dle §60 AZ se pro účely této smlouvy rozumí dílo vytvořené žákem/studentem ke splnění školních nebo studijních povinností vyplývajících z jeho právního postavení ke škole.

1.2 Licencí se pro účely této smlouvy rozumí oprávnění k výkonu práva školní dílo užít v rozsahu a za podmínek dále stanovených.

Článek 2
Dílo

2.1 Předmětem této smlouvy je poskytnutí licence k užití školního díla – maturitní práce.

Název práce (dále jen „dílo“): Elektromechanická hra -> samořídící šachovnice
vedoucí práce: Mgr. Milan Janoušek
odevzdané nabyvateli v tištěné a elektronické formě dne 23. 3. 2025.

2.2 Autor prohlašuje, že:

- vytvořil dílo, specifikované touto smlouvou, samostatnou vlastní tvůrčí činností;
- při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími;
- dílo je dílem původním;
- neposkytl třetí osobě výhradní oprávnění k užití díla v rozsahu licence poskytnuté nabyvateli dle této smlouvy před podpisem této smlouvy;
- je si vědom, že před zamýšleným poskytnutím výhradního oprávnění k užití díla v rozsahu licence poskytnuté nabyvateli dle této smlouvy třetí osobě, je povinen informovat tuto třetí osobu o skutečnosti, že již poskytl nevýhradní licenci k užití díla nabyvateli.

2.3 Dílo je chráněno jako dílo dle autorského zákona v platném znění.

Článek 3
Poskytnutí licence

3.1 Licenční smlouvou autor poskytuje nabyvateli oprávnění k výkonu práva dílo užít pro účely výuky na SPŠ a VOŠ, Písek a pro vnitřní potřebu školy, ze které neplyne škole hospodářský výsledek.

3.2 Licence je poskytována pro celou dobu trvání autorských a majetkových práv k dílu.

3.3 Autor poskytuje nabyvateli oprávnění užít dílo způsoby podle 3.1 neomezeně.

3.4 Autor poskytuje nabyvateli oprávnění užít dílo bezúplatně za splnění podmínky, že nabyvatel nebude užívat dílo za účelem dosažení zisku a nebude-li v budoucnu dohodnuto písemně jinak.

Článek 4

Údaje o autorství

4.1 Nabyvatel se zavazuje, že uvede údaje o autorství autora dle Autorského zákona.

Článek 5

Poskytnutí licence

5.1 Pokud to není v rozporu s oprávněnými zájmy nabyvatele, licence je poskytována jako nevýhradní. Nabyvatel je oprávněn postoupit tuto licenci třetí osobě a udělovat podlicence za splnění podmínek uvedených v § 48 zákona.

5.2 Autor může své dílo užít či poskytnout jinému licenci, není-li to v rozporu s oprávněnými zájmy nabyvatele, za podmínky, že nabyvatel (dle této licenční smlouvy) je oprávněn po autoru školního díla požadovat, aby přiměřeně přispěl na úhradu nákladů, tak, jak je stanoveno v § 60 odst. 3 zákona.

5.3 Nabyvatel není povinen dílo užít.

5.4 Nabyvatel je oprávněn dílo spojovat s jinými díly i zařadit dílo do díla souborného. Autor dává svolení k tomu, aby nabyvatel pořídil pro účely užití uvedené v této smlouvě překlad díla.

5.5 V případě, že z díla plyne hospodářský výsledek autorovi nebo nabyvateli, rozdělení zisku bude řešeno dodatkem k této smlouvě.

Článek 6

Výpověď smlouvy

6.1 Každá smluvní strana může smlouvu kdykoliv písemně vypovědět bez udání úvodu.

6.2 Výpověď musí být učiněna doporučeným dopisem doručeným druhé smluvní straně. Výpovědní lhůta je stanovena na dva měsíce a začíná běžet prvním dnem kalendářního měsíce následujícího po měsíci, v němž byla výpověď doručena druhé smluvní straně.

Článek 7

Závěrečná ustanovení

7.1 Smlouva je sepsána ve dvou vyhotoveních s platností originálu, která budou vložena do dvou výtisků díla (práce), z toho nabyvatel i autor obdrží po jednom vyhotovení.

7.2 Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem a občanským zákoníkem v platném znění, popř. dalšími právními předpisy.

7.3 Smlouva byla uzavřena podle svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoli v tísni a za nápadně nevýhodných podmínek.

7.4 Smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Písku dne 23. 3. 2025

Autor: _____

Nabyvatel: _____

Obsah

1 Úvod **14**

2 Využité technologie **16**

2.1 Arduino 16

2.2 Deska plošného spoje 16

2.3 Elektronické součástky 17

2.3.1 Magnetický spínač 17

2.3.2 Multiplexor 17

2.3.3 Stabilizátor 17

2.3.4 Elektromagnet 18

2.3.5 Krokový motor 18

2.4 Mini-Max algoritmus 18

3 Elektrické zapojení **20**

3.1 PCB deska 20

3.2 Řídící jednotka 22

4 Detekce pozice figurek **23**

4.1 Zapojení a princip 23

4.2 Programový kód 25

4.2.1 Čtení hodnot pomocí getReedValues() 25

4.2.2 Ukládání všech pozic 27

4.2.3 Detekce přemístění figurek 27

5 Pohyb figurek **30**

5.1 Elektromagnet 30

5.2 Krokové motory 31

5.2.1 Konstrukce 32

5.2.2 Programový kód 32

6 Šachová hra **35**

7 Konstrukční provedení **36**

8 Závěr **37**

Přílohy **39**

A schema_sachovnice.pdf **40**

Kapitola 1

Úvod

Nápad pro automatickou šachovnici jsem dostal při hledání nového projektu, který bych si doma zvládl sestrojit. Chtěl jsem, aby projekt obsahoval, jak část softwarovou, kterou jsem se do té doby primárně zabýval, ale také část mechanickou, kterou by bylo nutné vyrobit.

Napadlo mě vytvořit nějakou deskovou hru pro více hračů. V té době jsem měl ve velké oblibě šach a tím vznikla idea automatická šachovnice. Začal jsem nákresem na papír a tím jsem získal základní představu o projektu. Šachovnice se skládá ze tří hlavní částí:

Konstrukční část Tou jest samotná dřevěná konstrukce šachovnice, do které bude nutné umístít mechanismus pro pohybování s figurky. V rámci toho také její opracování a nadesignování projektu. Tvorba šachových políček a rozhraní pro jednoduché ovládání.

Elektrotechnická část Ta obsahuje systém pro detekci figurek na šachovnici. Ten by se dal vytvořit pomocí mnoha způsobů, ale v práci budu popisovat řešení pomocí desky plošného spoje. Dále vytvoření pohybové soustavy pomocí krokových motorů a umístění elektromagnetu, který bude s figurky pohybovat. Posledním krokem je sestrojení systému, jenž umožní uživateli variabilovat hru pomocí volby barvy a obtížnosti.

Softwarová část Z hlediska softwaru je nutné vytvořit kód, který spojí všechny části dohromady. Je nutné transformovat signály, tak aby s nimi bylo možné pracovat. Z hlediska kódu je nutné vytvořit rozhraní pro komunikaci mezi mikrokontrolérem a součástky. Je zapotřebí umožnit evaluaci dat o pozicích figurek a předat je algoritmu, který nám bude schopen vymyslet další tah. Údáje o dalším tahu ovšem musíme opět transformovat, tak aby z něj byli čitelné pokyny pro krokové motory. Nesmíme

opomenou ani řídící signály a minimalizovat zpoždění a zatížení na hardware.

Poté, co jsem si udělal průzkum o projektu automatické šachovnice zjistil jsem několik věcí. Překvapilo mě očividné. Automatickou šachovnici bylo v dnešní době možné zakoupit na internetu. Ovšem cena této šachovnice se pohybovala od necelé desítky tisíc až po stovky tisíců. Tyto ohromné částky jsem se rozhodl zminimalizovat a svůj projekt vytvořit cenově dostupný ačkoli zachovám kvalitu šachovnice a nevyměním dřevo za plasty a jiné méně vhodné materiály pro klasickou šachovnici.

Kapitola 2

Využité technologie

V této kapitole si popíšeme technologie, které byli zapotřebí při tvorbě projektu. Získáme k nim teoretický základ nutný k porozumění šachovnice.

2.1 Arduino

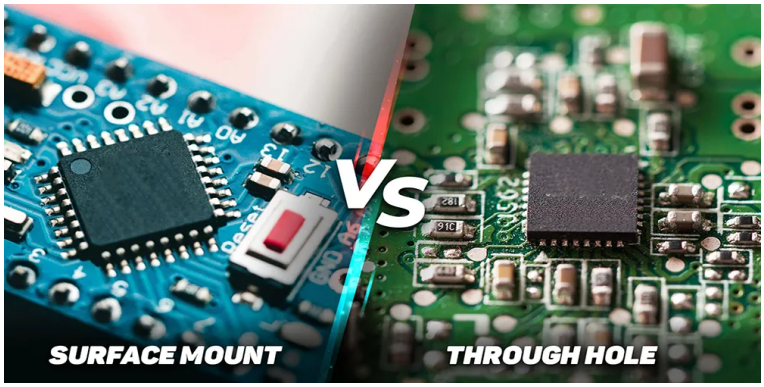
Arduino je platforma pro tvorbu různých elektronických projektů. Zabývá se hardwarem a softwarem a obě tyto části vytváří jako open-source. To je jeden z důvodů velkého rozšíření mikrokontrolerů Arduino.

Mikrokontrolér Arduino je malý počítač založen na jednom čipu. Arduino vyrábí mnoho různých řad jejich projektů, ale mezi nejznámější se řadí Arduino UNO či Mega. Arduino desky na sobě mají široké spektrum možných vstupů, výstupů a senzorůviz. obr. **??**. Velkou výhodou těchto desek je cena. Ta se pohybuje v řádech stovek korun.

Arduino desky se primárně programují pomocí platformy Arduino IDE. To je další open-source program, který má na základě popularity mezi lidmi i velkou podporu knihoven s různými zaměřeními.

2.2 Deska plošného spoje

PCB deska, jak je zkratkou nazývána, nebo také DPS je velmi rozšířená elektronická součástka. Desky jsou vyrobené z izolačních materiálů s pájecími poli, kterým se říká podložky a elektrickými spoji nazývanými stopy. Na PCB desky se připájí potřebné součástky. Tyto součástky jsou spolu pak propojeny na základě schématu pro tvorbu PCB. Jsou dvě primární technologie montáže součástek na desku. Je tady technologie průchozích otvorů (THT, Through-Hole Technology), nebo povrchové montáže (SMD, Surface-Mounted Devices) viz. obr. 2.1. Při povrchové montáži se napájí součástky přímo na desku s tím, že



Obrázek 2.1: Rozlišení SMT a THT na PCB desce

jsou na ní připravené jednotlivé stopy. U THT jsou v desce připravené dírky, do kterých součástky umísťujeme.

2.3 Elektronické součástky

2.3.1 Magnetický spínač

Tento typ spínače je elektronický obvod aktivován pomocí magntického pole. Nejběžnější je konstrukce pomocí dvou feromagnetických kovů umístěných kousek od sebe ve skleněné kapsli. V momentě, kdy je ke spínači přiložen magnet se kovy spojí a začnou vodit. Magnetické spínače jsou například využívány pro detekci uzavření dveří.

2.3.2 Multiplexor

Tato elektronická součástka slouží k přepínání vstupů na jeden výstup na základě řídících signálů. Multiplexor je realizován integrovanými obvody. Pomocí této součástky jsme například schopni zvětšit počet vstupů na určitém zařízení o počet vstupů na multiplexou, s tím že do zařízení zapojíme pouze multiplexor. Tento postup je ideální v momentě, kdy máme zařízení s nedostatkem vstupů pro dané využití.

2.3.3 Stabilizátor

Stabilizátor nám zajištuje výstupní nápětí bez ohledu na změny výstupního proudu či vstupního napětí. Stabilázátory se dělí na lineární paremetrický, lineární zpětnovazební

a spínací zpětnovazební. Stabilizátory fungují na principu regulace odporu pomocí tranzistoru na základě referečního napětí a napětí žádaného. Ke stabilizátoru se přidávají kondenzátory, který eliminuje šum a stabilizují výstup. Lineární stabilizátory regulují napětí, tím že fungují jako proměnný odpor mezi vstupem a výstupem.

2.3.4 Elektromagnet

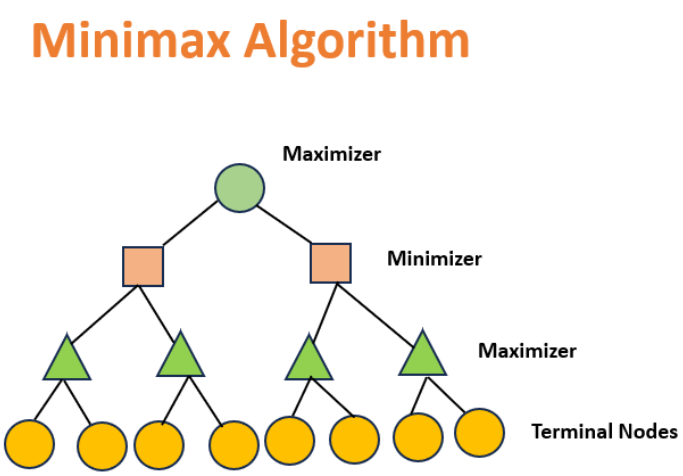
Elektromagnetem se rozumí cívka pomocí, které jsme schopni vytvořit dočasné magnetické pole. Jádro této cívky je z magneticky měkké oceli. V momentě, kdy začneme cívkou provádět proud, začne se okolo ní vytvářet magnetické pole. Velikost a polaritu tohoto pole jsme schopni ovládat.

2.3.5 Krokový motor

Krokové motor je zařízení, který nám dovolí na základě elektromagnetických pulsů přesně rotovat. V motoru je ozubené kolo, které můžeme otočát na základě polových dvojic. Pomocí pulsování jednotlivá pole přitáhnout ozubené kolo a na základě tohoto jevu, můžeme motor otočit o přesný počet kroků. Rychlost rotování jsme schopni korigovat na základě změn frekvence aktivace polí. To realizujeme skrze ovladače krokových motorů.

2.4 Mini-Max algoritmus

Tento typ algorithmu je fundamentálním konceptem umělé inteligence a teorie her. Mini-Max má za cíl zminimalizovat možnou ztrátu na základě analýzy nejhorších možných scénářu, neboli min, a těch nejlepších, max. V hře pro dva hráče si algoritmus vytvoří dvě interní entity. Ta první se jmenuje Maximizer a cílí získat nejvyšší možné skóre. Oproti ní druhá, Minimizer, má za cíl, co nejvíce skóre Maximizer snížit. Toho docílí způsobem, že evaluuje všechny možné tahy obou hráčů zároveň.



Obrázek 2.2: Minimax algorithmus

Z počátku si Mini-Max vytvoří tzv. herní strom. Účelem tohoto stromu je reprezentovat všechny možné tahy, které mohou v daný moment ve hře nastat. Dalším krokem je, že u tzv. závěrečných stavů (terminal nodes) určí skóre. Skoŕe se určuje na základě kritériích daných v kódu a postupně ve stromu postupuje výše a určuje skóre dál. Poté, co Mini-Max určí skóre pro všechny možné stavy, projde jednotlivé scénáře. V kole Maximizera volí variantu s nejvyšším možným skórem. Naopak v kole minimizera tu s nejnižším. Podle nejvyššího možného celkového skóre algoritmus určí příští zvolený blok.

Kapitola 3

Elektrické zapojení

Pro projekt bylo zapotřebí vymyslet efektivní a spolehlivý způsob propojení. Zároveň však způsob nesměl bránit detekci figurek pomocí spínačů. Nabízela se spoustu možností, ale z důvodů velkého množství součástek jsem se rozhodl pro dle mě nejčistčí řešení a to desku plošného spoje.

3.1 PCB deska

Deska plošného spoje bylo řešení, které se nabízelo kvůli spolehlivosti a zároveň, tím že deska je velmi tenké a její materiál nijak nebrání průchodu magnetického pole. Na základě mého schématu v programu Easy ADA jsem si desku nechal sestrojit. Schéma šachovnice najdete v přílohách pod názvem schema_sachovnice.pdf. Z důvodu výrobních možností a rozměrů desky byla deska sestrojena na dvě poloviny. Tyto poloviny jsem spájel dohromady. Desku byla vyrobena technologií SMT, takže jsem na ní povrchově připájel piny pro všechny potřebné součástky. A to

- 69 jazýčkový magnetických kontaktů
- 5 analogových multiplexorů
- Mikrokontroler Arduino UNO
- Dva ovladače pro krokové motorů A4988
- Stabilizátor a piny pro vstupní napětí
- Napájení pro motory a elektromagnety
- Transistor a různé drobné zařízení

Obrázek 3.1: porovnání PCB desky z výroby a hotové

Desku jsem si následně nechal pocínovat za účelem lepší vodivosti. Dvě části jsem pomocí propojek spojil dohromady. Během práce jsem posléze při detekci figurek musel řešit velký problém a to se spolehlivostí desky. Magnetické spínače nefungovali spolehlivě. Důvodem byli narušené cesty a také křekhost spínačů. Ty během pájení velmi snadno praskli.

Tyto problémy jsem řešil za pomocí multimetru a posléze detekcí pomocí počítače. Na desce jsem postupně zkontroloval všechny cesty a v momentě, kdy špatně vodili jsem na ně napájel vrstvu cínu. Pokud problém spočíval v propojení dvou vedlejších cest, vzal jsem nůž a cín manuálně odstranil. Největším úskalím bylo manuální propojení dvou desek dohromady. Tato část desky vyžadovalo časově velmi úpornou manuální péči.

Desky se mi po opravách podařilo dostat do spolehlivého stavu. Jednou věcí, která

mi však projekt stížila bylo, že nespolehlivost dráh a spínačů jsem objevil až v momentě, kdy jsem programoval detekci figurek. V tuto chvíli jsem však již, alespoň mohl využít napojení na multiplexery a pomocí kódu je kontrolovat skrze konzoly na počítači. To usnadilo proces oprav.

Deska je napájena externím 18V zdrojem. Na desce se pomocí stabilizátoru pro většinu součástek ustálí proud 5V, avšak pro motory a elektromagnet zůstává proud 18V.

3.2 Řídící jednotka

Mikrokontrolér, který jsem si vybral pro samořídící šachovnici je Arduino UNO. Toto Arduino splňovalo veškeré nutné požadavky pro ovládání a zároveň to byl mikrokontroler, který jsem měl z počátku projektu doma a mohl jsem na něm zprvu testovat. Arduino je napájeno 5V z PCB desky a ovládá celou šachovnici. Arduino je naprogramováno variantou jazyku C++, ale o tom budu psát podrobněji až dále v práci. Do mikrokontroleru jsou skrze multiplexory zapojeny všechny magnetické spínače. Arduino si pomocí programové logiky přepíná, jaký z multiplexorů bude v dané chvíli aktivní, jelikož data multiplexorů proudí po stejných cestách. Dále jsou do Arduina zapojené ovladače motorů a transistor, který slouží k ovládání k elektromagnetu.

Kapitola 4

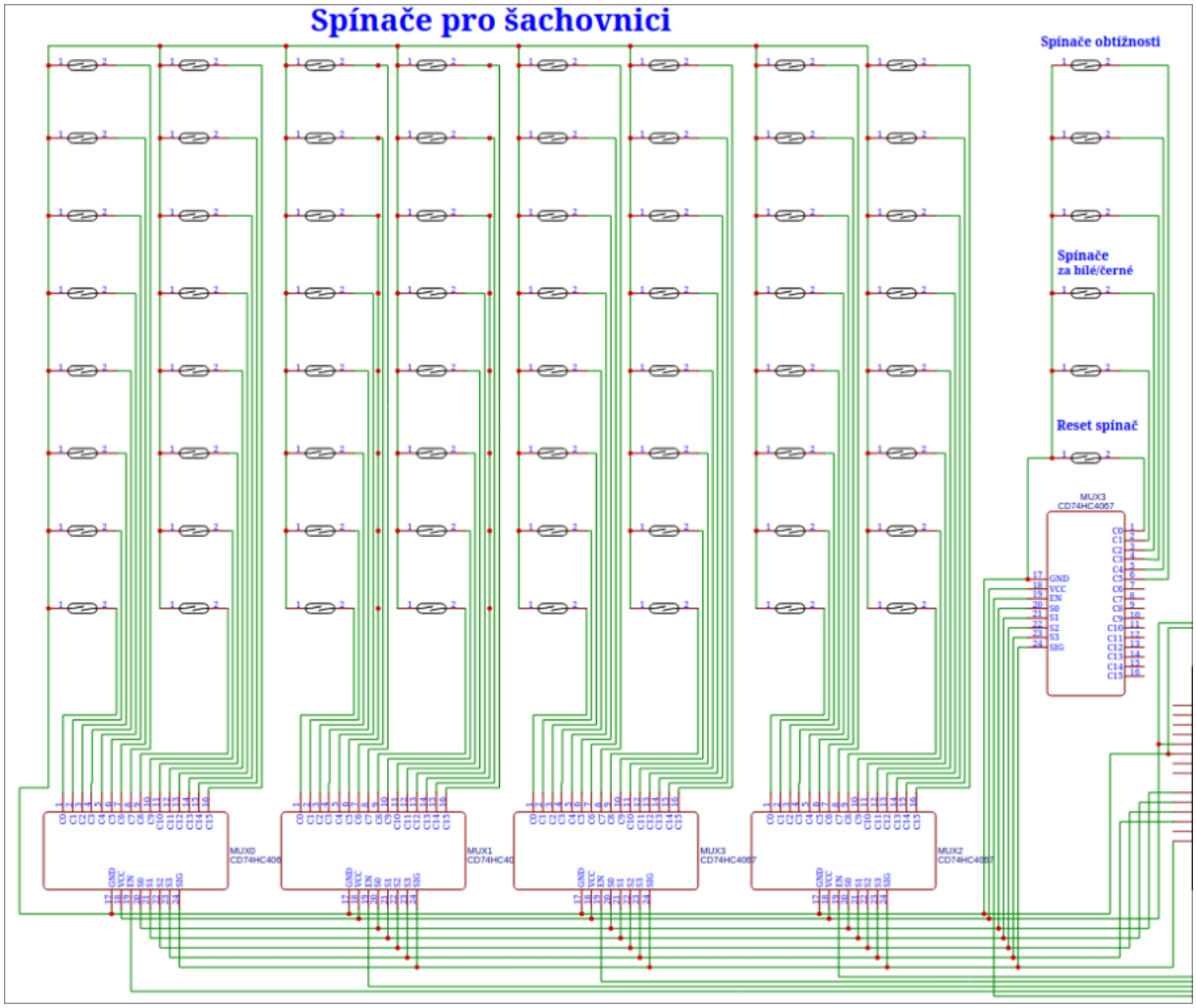
Detekce pozice figurek

V této kapitole popíši, na jakém principu se v šachovnici detekují pozici figurek. Vysvětlím jednotlivé zapojení multiplexorů a kód, který je ovládá. Spolehlivé detekování pozice figurek je stěžejním bodem této prace, jelikož z něho vychází další operace projektu.

4.1 Zapojení a princip

Aby samořídíc šachovnice fungovalo je zapotřebí vědět pozici figurek na herním poli. Nabízí se dvě řešení. Pomocí detekce pomocí kamer a zpracování obrazu a pomocí magnetickýc spínačů. Řešení pomocí kamer jsem zavrhl, již z počátku jelikož by to znamenalo nějaké rameno na šachovnice s kamerou. To mi nepřipadalo ideální. Magnetické spínače nám zachovají čistý vzhled šachovnice. Jelikož však na šachovnici máme 64 políček, potřebujeme 64 spínačů. Tolika piny naše Arduino Uno nedisponuje, a tak je zapotřebí využít nějaký systém pomocí, kterého počet vstupů snížíme.

Po průzkumu a práci na řešení jsem se inspiroval z jednoho projektu dostupném na internetu [9]. V tomto projektu bylo využita multiplexorové řešení. Zapojíme piny spínačů do multiplexoru a ten pak čteme pomocí řídící jednotky. Pomocí tohoto postupu jsme schopni zredukovat 64 pinů na 4 multiplexory. V této práci jsem jich využil pět, jelikož poslední multiplexor se stará o volbu obtížností a barvy.



Obrázek 4.1: Zapojení magnetických spínačů na desce

Multiplexory, které jsem využil mají 16 datových vstupů, 4 vstupy adresové, nápajení a zem, zapínací pin EN a výstup SIG. Výstupy multiplexorů, jak je vidět na schématu viz. obr. 4.1 jsou zapojeny do stejných cest na PCB desce. Stejna tak výstup SIG je pro všechny multiplexory stejný. Toto není problém, jelikož v kódu využivám pin Enable (en). Jestliže je tento pin v logické jedničce, je multiplexor vypnutý. V momentě, kdy je v logické nule, je zapnutý. Na základě tohoto principu jsme schopni přepínat přepínače samotné a tím ještě více zminimalizovat počet nutných vstupů do Arduina UNO.

4.2 Programový kód

Jak, už jsme zmínili dříve v práci, šachovnice je ovládána mikrokontrolerem Arduino UNO, jenž se programuje variantou C++. Kód pro ovládání multiplexorů nám musí zajistit jejich přepínání pomocí Enable pin a stabilní detekci pozic. První věc, kterou musíme v kódu nastavit jsou Enable piny. Po zapnutí nastavím Enable pin jako OUTPUT a pomocí for loopu si procyklujeme všechny Enable piny multiplexorů a dáme na ně logickou jedničku. Tato část kódu je v setup() funkci kódu. Ta se spustí pouze po spuštění mikrokontroleru.

```
// Disabling Multiplexers (When high - does nothing)
pinMode(muxEnable[4], OUTPUT);
for (int i = 0; i < 4; i++) {
  pinMode(muxEnable[i], OUTPUT);
  digitalWrite(muxEnable[i], HIGH);
}

for (int i = 0; i < 4; i++) {
  pinMode(muxAddr[i], OUTPUT);
  digitalWrite(muxAddr[i], LOW);
}
```

4.2.1 Čtení hodnot pomocí getReedValues()

Samotný proces čtení pozic si ukážeme jako další. Využívám na tou svou funkci getReedValues. V této funkci nejprve zaktivujeme multiplexoru, tím že mu nastavíme pin na 0. Poté začneme na jeho adresové piny vysílat binarní kombinace od 0 do 15. Toho docíleme opět pomocí for cyklu a využití modulo funkce (v kódu jako %). Data následně ukládáme do našeho interního variablu muxValues a poté, co deaktivujeme multiplexor a jeho adresové vstupy, překopírujeme pomocí funkce memcpy hodnoty do proměnné recordedReedValue, která nám slouží pouze pro ukládání nejnovější možné detekce pozic.

```

void getReedValues(int targetMuxAddress, int from, int to) {
  // Splitting to two rows because row of chessboard has 8 pieces
  int muxValues[8];

```

```
  // Activating MUX
  digitalWrite(targetMuxAddress, LOW);

  for (int j = from; j < to; j++) {
    // Checking MUX combinations
    digitalWrite(muxAddr[0], j % 2);
    digitalWrite(muxAddr[1], j / 2 % 2);
    digitalWrite(muxAddr[2], j / 4 % 2);
    digitalWrite(muxAddr[3], j / 8 % 2);

    int reedValue = digitalRead(muxOutput);

    if (j > 7) {
      muxValues[7-(j-8)] = reedValue;
    } else {
      muxValues[j] = reedValue;
    }
  }

  // Resetting MUX
  digitalWrite(muxAddr[0], LOW);
  digitalWrite(muxAddr[1], LOW);
  digitalWrite(muxAddr[2], LOW);
  digitalWrite(muxAddr[3], LOW);

  digitalWrite(targetMuxAddress, HIGH);
  memcpy(recordedReedValue, muxValues, sizeof(recordedReedValue));
}
```

Jelikož je SIG pin multiplexorů nastavený jako INPUT_PULLUP, což je nutné pro detekci zda spínač spíná zem či je rozpojený, dostávám všechny rozpojené spínače jako jedničky. Ty seplé jako nuly. Ve výsledku může výstup v konzoli po zahájení hry vypadat takto.

```
8 - 0 0 0 0 0 0 0 0
7 - 0 0 0 0 0 0 0 0
6 - 1 1 1 1 1 1 1 1
5 - 1 1 1 1 1 1 1 1
4 - 1 1 1 1 1 1 1 1
3 - 1 1 1 1 1 1 1 1
2 - 0 0 0 0 0 0 0 0
1 - 0 0 0 0 0 0 0 0
----A B C D E F G H
```

4.2.2 Ukládání všech pozic

Tato funkce nám zajišťuje přečtení hodnot multiplexoru na základě daných parametrů. V kódu máme další pomocnou funkci setCurrentBoard(). Cílem této funkce je projít všechny řady šachovnice od první až po osmou a uložit je do mikrokontroleru. V mikrokontroleru je opět pomocí funkce memcpy překopíruje do dvouřadého listu boardValues, který obsahuje všechny pozice figurek.

```
1 void setCurrentBoard() {
2   getReedValues(muxEnable[0], 0, 8);
3   memcpy(boardValues[0], recordedReedValue, sizeof(boardValues[0]));
4
5   getReedValues(muxEnable[0], 8, 16);
6   memcpy(boardValues[1], recordedReedValue, sizeof(boardValues[1]));
7   //.....another five reads
8   getReedValues(muxEnable[3], 8, 16);
9   memcpy(boardValues[7], recordedReedValue, sizeof(boardValues[7]));
```

4.2.3 Detekce přemístění figurek

Předchozí funkce v této kapitole nám vysvětlují, jak šachovnice data čte. Nadcházející funkce je však zodpovědné za samotnou detekci změny pozic, když uživatel vezme figurky do ruky a přendá ji na jiné políčko. Tato funkce je napsána bloku loop(). To je opět

platformou Arduino daná funkce, která se po spuštění již zmíněné funkce setup() neustále opakuje.

Abychom zjistili zda se událo přemístění figurky musíme pravidelně číst pozice hodnot a porovnávat je z přechozími. To děláme pomocí while cyklu, který se opakuje do doby než jsou splněné podmínky uvedené v závorkách. Podmínky, které jsem v kódu nastavil jsou, že list move bude obsahove čtyři hodnoty. Neboli hodnotu čísla a písmena odkud se figurka přemístila a hodnoty kam. Jak zjistíme hodnoty, které ukládáme do move, vysvětlím dále v textu. Nejprve tedy uložíme hodnoty pozic do proměnné boardValuesMemory a následně přejdeme do cyklu. V tomto cyklu se neustále volá funkce detectBoardMovement() až do splníme zmíněných podmínek.

```
1  // Player's turn => waiting for movement
2  memcpy(boardValuesMemory, boardValues, sizeof(boardValuesMemory));
3  while (!move[0] || !move[1] || !move[2] || !move[3]) {
4    detectBoardMovement();
5  }
```

Funkce detectBoardMovement()

Pomocí této funkce detekujeme, zda bylo uskutečněno přemístění figurek. Tato funkce je volána cyklicky z while loopu dokud nenajdeme čtyři hodnoty pro proměnnou move. Např. e2e4 by nám značilo pohyb z počátku hry pěšce o dvě místa z e2 na políčko e4. Funkce porovnává současné hodnoty pozic, které nastaví pomocí funkce setCurrentBoard(), boardValues s hodnoty uloženými v hlavním cyklu před zahájením while cyklu, boardValuesMemory.

```
1  void detectBoardMovement() {
2    setCurrentBoard();
3
4    for (int i = 7; i >= 0; i--) {
5      for (int j = 7; j >= 0; j--) {
6        if (boardValuesMemory[i][j] != boardValues[i][j]) {
7          // Position of piece has changes
8          if (boardValues[i][j]) {
9            // Piece was here before
10           move[0] = letterTranslate[j];
```

```
11            move[1] = numberTranslate[i];
12          } else {
13            // New piece on this position
14            move[2] = letterTranslate[j];
15            move[3] = numberTranslate[i];
16          }
17        }
18      }
19    }
20    delay(2000);
21  }
```

Takto pomocí dvou for loopu prochází hodnoty od nejvyšší po nejnižší. To je z důvodu uložení dat a zapojení multiplexorů. Tam postupně porovnává každé políčko s každým dokud nenajde nějakou změnu. V případě, že všechny hodnoty jsou stejné jako předtím, funkce se spustí z hlavního cyklu znovu. V případě změny však pomocí kódu detekujeme zda šlo o změnu z 0 na 1 či naopak.

Změny z 0 na 1 Jelikož logická jednička označuje prázdné políčko, tato zmena značí odkud se figurka přemístila. Zapíše se tedy do move na první a druhé pozici.

Změny z 1 na 0 Značí, že políčko bylo dříve prázné a nyní se na něj umístila nová figurka. Zapíše se tedy do list move na třetí a čtvrtou pozici

V momentě, kdy naplníme všechny čtyři místo v listu move, zruší se while cyklus v hlavní funkci programu a kód může postupovat dále. Ve funkci máme na konci ještě dané zpoždění, které zamezí chybám v detekci při pomalém přemísťování figurek.

Kapitola 5

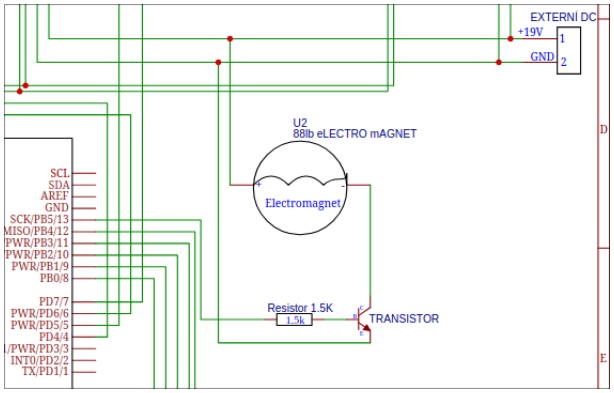
Pohyb figurek

Realizaci šachových tahů nám zajištují dva krokové motory a elektromagnet. Pro účel práce bylo nutné umožnit XY pohyb elektromagnetu uvnitř šachovnice. Soustavu bylo nutné umístit dovnitř konstrukce a zajistit, aby byl elektromagnet velmi blízko PCB desce a tím i figurkám.

5.1 Elektromagnet

Samotné přemístění figurky je realizováno pomocí elektromagnetu. Ten po přivedení proudu, vytvoří elektromagnetické pole, které začne působit na magnetickou šachovou figurku nad ním a tím s ní můžeme hýbat. Během tvorby řešení jsem musel zajistit, aby elektromagnetické pole nebylo příliš velké a neovlivňovalo ostatní figurky. Zprvu ovšem nastal jiný problém, a to že elektromagnet, i když zapnutý, nebyl schopen figurkou pohnout. Tato situace nastala na základě faktu, že pro velkou účinost elektromagnetu je nutné zminimalizovat prostor mezi elektromagnetem a přitahovaným tělesem. V situaci, kdy jsem používal elektromagnet se silou 150N stále to nebylo dostatečné. Problém jsem vyřešil vyvýšením elektromagnetu v konstrukci a zasazením jej blíže k plošnému spoji. Po těchto úpravách mi postačil, elektromagnet se silou 50N. Avšak ani tento elektromagnet by s figurky nepohl, jestliže by do nich nebyli umístěné magnetické prvky. Pokud bu do figurky byla umístěna prostá ocel, nedokázal by jí elektromagnet přemístit.

Elektromagnet, který v práci využívám je v paremetrech u prodejce uváděn na 12V. Já do něj však vysílám 18V z důvodu lepší spolehlivosti. Abychom však elektromagnet mohli ovládat, musel jsem vymyslet způsob, jakým do elektromagnetu těchto 18V budu1 vpouštět pouze po zaslání signálů. Na to využivám transistor. Transistor použitý v práci po přivedení signálu na bázi, přepne vstup do kolektoru na emitor. Na kolektoru mám přivedenou zem a po zaslání signálů na bázi, vyšlu zem skrze emitor do elektromagnetu.



Obrázek 5.1: Ovládání elektromagnetu pomocí transistoru

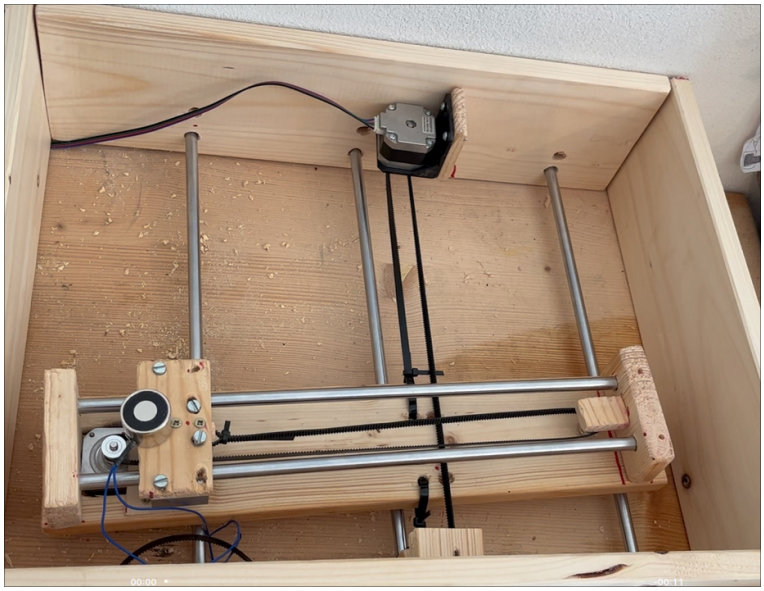
Ten je z jeho druhého vstupu připojen k napětí a po přivedení země, začne působit. Po ukončení signálu na bázi se elektromagnet vypne.

Kód, který transistor ovládá je velmi jednoduchý. Ve fuknci moveMagnet, o které bude v dalších kapitolách hovořit, je boolovská proměnná magnetActivated. V případě, že magnet potřebuji nastavím jí jako true a pomocí podmínky v této funkci vyšlu na pin báze transistoru (magnetPin) logickou jedničku. Na konci funkce po provedení pohyby, magnet vždy vypínám.

```
1 void moveMagnet(int direction, int distance, bool magnetActivated) {
2   // Enabling the electromagnet if neccesary
3   if (magnetActivated) {
4     digitalWrite(magnetPin, HIGH);
5   }
```

5.2 Krokové motory

Pro zajištění pohybu využívám dva krokové motory NEMA17 s krokovými ovladači A4988. Krokový ovladač slouží jako mezikrok mezi programem a kódem a zajištuje pohodlné ovládání motorů. Do těchto ovladačů přivádím po desce, jak 18V, tak 5V. Dále na nich nastavujeme směr pohybu a vysíláme signály pomocí, kterých pak motor hýbe s konstrukcí. Pro zajištění správného pohybu bylo nutné zajistit dostačné napětí. To je dle specifikací 8 až 35V. Po otestovaní jsem zjistil, že 18V bylo pro projekt ideální. Abych zamezil přehřívání ovladčů, nalepil jsem na ně z vrchu na ovladač, který je s nimi prodáván.



Obrázek 5.2: Konstrukce krokových motorů

5.2.1 Konstrukce

První motor je umístěn zevnitř boční stěny šachovnice. Na motor jsem umístil vhodnou řemenici a řemen natáhl na protější stěnu. Zde jsem si ze dřeva vyrobil sestrojil ústrojí pro další řemenici. Pod řemenem vedou tři vodicí tyče, do kterých jsem umístil druhý motor s konstrukcí. Řemen prochází touto konstrukcí a při otáčení řemenice, tak pomocí řemenu pohybujeme na Y ose i s konstrukcí uvnitř šachovnice. Vnitřní konstrukce je umístěna na třech ložiskách, kterými prochází vodící tyče. V této vnitřní dřevěné konstrukci je vyříznut otvor pro druhý krokový motor a zde je zde postupováno podle podobného postupu. Vedou zde dvě vodicí tyče na X ose. Na těchto vodicích tyčí jsou dvě ložiška, na které jsem si sestrojil dřevěnou plošinku připravenou pro umístění magnetu. Magnet je na plošině vyvýšen pro zajištění spolehlivého přenosu figurek a kabely jsou vyvedeny na PCB desku. Na druhé straně je opět ústrojí pro řemenice a řemen je proveden skrze plošinu s elektromagnetem.

5.2.2 Programový kód

Funkci, která ovládá motory jsem nazval moveMagnet. Tato funkce bere tři vstupní parametry. Směr (direction), vzdálenou (distance) a zda má být magnet aktivován při pohybu

(magnetActivated). Pro přehlednost nerozlišuje směry pomocí if podmínek, ale pomocí switche. V něm nastavím, jaký blok kódu se provedu jestliže směr bude odpovídat hodnote za case. Hodnota směru je dána jako pozice v numerické klávesnici. Tedy 4 jako pohyb vlevo, 2 pohyb směrem dolů a např. 9 jako diagonální pohyb doprava nahoru.

```
void moveMagnet(int direction, int distance, bool magnetActivated) {
  switch (direction) {
    //...
    case 4:
      // Left
      digitalWrite(dirPinX, LOW);
      for (int i = 0; i < boxLength * distance; i++) {
        digitalWrite(stepPinX, HIGH);
        delayMicroseconds(motorDelay);
        digitalWrite(stepPinX, LOW);
        delayMicroseconds(motorDelay);
      }
      magnetX = magnetX - distance;
    break;
    case 9:
      // Diagonal right up
      digitalWrite(dirPinY, HIGH);
      digitalWrite(dirPinX, HIGH);
      for (int i = 0; i < boxLength * distance; i++) {
        digitalWrite(stepPinY, HIGH);
        digitalWrite(stepPinX, HIGH);
        delayMicroseconds(motorDelay);
        digitalWrite(stepPinY, LOW);
        digitalWrite(stepPinX, LOW);
        delayMicroseconds(motorDelay);
      }
      magnetX = magnetX + distance;
      magnetY = magnetY + distance;
    break;
  }
//...
```

Samotný kód, který vysílá signály ovladači je v bloku pod case. Nejprve určíme směr po-

hybu. Buď logickou nulou či jedničkou. Následně začneme pomocí for cyklu vysílat pulzy do motoru. Vždy nejdříve aktivujeme nutný stepPin. Pak zahájíme prodlevu. Tato prodleva nám určuje rychlost otáčení. Čím menší je, tím rychleji se motory točí. Poté stepPin vypneme a znovu zahajíme prodlevu. Tento cyklus se opakuje podle žádané vzdálenosti, kterou násobím délkou jednoho políčka.

Kapitola 6

Šachová hra

Kapitola 7

Konstrukční provedení

Kapitola 8

Závěr

Seznam tabulek

Seznam obrázků

2.1 Rozlišení SMT a THT na PCB desce 17
2.2 Minimax algorithmus 19

3.1 porovnání PCB desky z výroby a hotové 21

4.1 Zapojení magnetických spínačů na desce 24

5.1 Ovládání elektromagnetu pomocí transistoru 31
5.2 Konstrukce krokových motorů 32

Příloha A

schema_sachovnice.pdf

Literatura

[1] CONRAD.CZ. Arduino® » Praktický mikrokontrolér pro individuální spínací a řídicí úlohy. *https://www.conrad.cz* [online]. ©2024 [cit.29. 2. 2024]. Dostupné z: https://www.conrad.cz/cs/clanky/elektromechanika/arduino.html?srsltid=AfmBOoq-DVdYd1lkV0hVguOg5cvqM7O721czoRtYbZBji4hT3SLgxRoY

[2] BOTLAND.CZ. Deska PCB – co to je?. *https://botland.cz/* [online]. ©2023 [cit.13. 3. 2023]. Dostupné z: https://botland.cz/blog/deska-pcb-co-to-je/

[3] UK.RS-ONLINE.COM. reed-switches-guide. *https://uk.rs-online.com* [online]. ©2023 [cit.1. 2. 2023]. Dostupné z: https://uk.rs-online.com/web/content/discovery/ideas-and-advice/reed-switches-guide

[4] DUBNO.CZ. VY_32_INOVACE_CTE_2.MA_13_Multiplexory. *https://dubno.cz/* [online]. ©2012 [cit.1.8.2012]. Dostupné z: https://dubno.cz/images/stories/dum/8.sablona/24/pdf/VY_32,_INOVACE_CTE_2.MA_13

[5] WIKIPEDIA.CZ. Stabilizátory napětí. *https://www.wikipedia.cz* [online]. ©2024 [cit.22.2.2024]. Dostupné z: https://cs.wikipedia.org/wiki/Stabiliz%C3%A1tor_nap%C4%9Bt%C3%AD

[6] LASTMINUTEENGINEERS.COM. 28byj48-stepper-motor-arduino-tutorial/. *https://lastminuteengineers.com* [online]. ©2025 [cit.23. 3. 2025]. Dostupné z: https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/ https://lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial/

[7] CS.WIKIPEDIA.ORG. Elektromagnet. *https://cs.wikipedia.org/* [online]. ©2025 [cit.25. 1. 2025]. Dostupné z: https://cs.wikipedia.org/wiki/Elektromagnet https://cs.wikipedia.org/wiki/Elektromagnet

[8] GEEKSFORGEEKS.COM. mini-max-algorithm-in-artificial-intelligence. *https://www.geeksforgeeks.org/* [online]. ©2024 [cit.27.8.2024]. Dostupné z:

https://www.geeksforgeeks.org/mini-max-algorithm-in-artificial-intelligence/
https://www.geeksforgeeks.org/mini-max-algorithm-in-artificial-intelligence/

[9] HTTPS://WWW.INSTRUCTABLES.COM. Automated-Chessboard. *https://www.instructables.com* [online]. ©2022 [cit.2.1.2022]. Dostupné z: https://www.instructables.com/Automated-Chessboard/ https://www.instructables.com/Automated-Chessboard/

[10] PCBA-MANUFACTURERS.COM. SMT vs THT. *2023* [online]. ©23. 3. 2023 [cit.https://www.pcba-manufacturers.com/smt-vs-tht/]. Dostupné z:

[11] STORE.ARDUINO.CC. Arduino UNO pinout. *2025* [online]. ©19. 3. 2025 [cit.https://store.arduino.cc/en-cz/products/arduino-uno-rev3]. Dostupné z:

[12] HTTPS://WWW.TPOINTTECH.COM. minimax-algorithm-in-python. *https://www.tpointtech.com/* [online]. ©2025 [cit.23.3.2025]. Dostupné z: https://www.tpointtech.com/minimax-algorithm-in-python https://www.tpointtech.com/minimax-algorithm-in-python