ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra elektrických pohonů a trakce

Řízení krokového motoru pomocí platformy Arduino

Control of stepper motor by Arduino platform

Bakalářská práce

Studijní program: Elektrotechnika, energetika a management, Bakalářský

Obor: Aplikovaná elektrotechnika

Vedoucí práce: Ing. Hlinovský Vít CSc.

Petr Maňák

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Vítu Hlinovskému, CSc. za vedení, cenné rady a připomínky k bakalářské práci.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne: ....

podpis

Abstrakt:

Tato bakalářská práce se zabývá nejprve teorií řízení krokových motorů a poté i prakticky řízením krokového motoru pomocí vývojové platformy Arduino a H-můstků. Cílem práce bylo uvést do chodu jednoduchý řídicí systém, umožňující ovládání krokového motoru z PC.

Abstract

This thesis deals with theory of stepper motor control and practical control of the stepper motor by development platform Arduino and H-bridges. The goal was to create simple control system capable of controling stepper motor from PC.

Klíčová slova:

Krokový motor, H-můstek, Arduino

Keywords:

Stepper motor, H-bridge, Arduino

Obsah

1 Úvod

Krokový motor je speciální typ elektrického synchronního motoru, vyznačující se vysokou přesností nastavování polohy a velmi vysokou opakovatelností kroků. Pro účely tohoto projektu byl použit dvoufázový krokový motor s aktivním rotorem od firmy Microcon.

Arduino je open source vývojová platforma určená pro rychlé navrhování prototypů jednoduchých zařízení. Vyznačuje se jednodušším programovacím jazykem, vysokou flexibilitou a možností používat velké množství periferního hardwaru. Pro účely tohoto projektu byla použita vývojová deska Arduino Mega 2560.

Jako silové obvody pro spínání proudů cívkami krokového motoru byl použit modul s osmi IGBT tranzistory. Tranzistory jsou spínány externí TTL logikou, kterou obstarává výše zmíněná platforma Arduino. Osm tranzistorů umožňuje zapojit krokový motor se dvěma fázemi bipolárně, což znamená, že proud může téci cívkami v obou směrech. Tranzistory jsou zde tedy využity jako H-můstky.

Poslední součástí projektu je obslužný program na PC, který komunikuje s platformou Arduino po sériové lince a lze jím ovládat pozici, směr a rychlost krokového motoru. Řídící program je psán v jazyce Object Pascal. Umožňuje s motorem provádět základní úkony a nastavovat požadovaný úhel natočení hřídele motoru.

2 Krokový motor

Krokový motor lze zařadit jako speciální typ synchronního motoru. Vyznačuje se nespojitým pohybem a v závislosti na konstrukci může zaujímat konečný počet definovaných poloh. Jeho výhodou je vysoká přesnost a hlavně vysoká opakovatelnost. Tyto parametry ho předurčují k použití v aplikacích, kde je zapotřebí vysoká přesnost polohy, jako jsou CNC stroje. Nevýhodou tohoto typu motoru je jeho složitější řízení. Neobejde se bez ovladače (driveru). Tento ovladač také ve velké míře ovlivňuje parametry krokového motoru. Profesionální ovladače dokážou zajistit vyšší rychlosti, větší rozlišení motoru, tlumí vibrace, atd. Krokový motor je tedy řízen impulsně a ovladač krokového motoru musí zajistit přepínání proudu fázemi v definovaných posloupnostech, aby docházelo k otáčení rotoru motoru.

2.1 Základní rozdělení krokových motorů

Krokové motory lze rozdělovat podle různých kritérií. Jedno z nejzákladnějších rozdělení je podle typu pohybu jaký motor vykonává. Existují krokové motory s rotačním pohybem, kterými se bude tato práce výhradně zabývat a s lineárním pohybem. Tyto typy motorů se dále rozdělují podle konstrukce a principu činnosti na motory s proměnnou reluktancí, motory s permanentními magnety a motory hybridní [2]. Motory s proměnnou reluktancí se také nazývají motory s pasivním rotorem, jelikož se rotor skládá pouze z plechů s vhodně tvarovanými póly. Motorům s permanentními magnety se také říká motory s aktivním rotorem, jelikož mají na rotoru permanentní magnet. U motorů s aktivním rotorem se dále rozlišují motory s radiálně polarizovaným magnetem a s axiálně polarizovaným magnetem. Motory s axiálně polarizovaným magnetem jsou označovány jako hybridní, protože obsahují konstrukční prvky typické pro motory s pasivním rotorem, ale obsahují i permanentní magnet. Existují také motory s pružným rotorem, kde se rotor v několika místech dotýká statoru. Krokové motory lze také rozdělovat podle počtu fází na dvou, tří a vícefázové. Podle způsobu buzení fází na unipolární a bipolární (bude podrobněji popsáno dále) [1].

2.2 Definice pojmů krokového pohonu

V této části textu jsou definovány pojmy obecně používané v oblasti pohonů s krokovými motory. Definice byly převzaty z [1].

Krokový pohon je zařízení, které se skládá z ovladače a krokového motoru, mezi nimiž je elektrické spojení.

Elektronický ovladač krokového motoru je elektronický přístroj, který řídí funkční pohyb a režimy chodu krokového motoru v závislosti na přivedené vstupní informaci. Hlavní funkční části ovladače jsou zpravidla elektronický komutátor a výkonový spínací zesilovač. Další částí

může být řídící logika, jejíž rozsah je velice variabilní, závislý na konkrétním použití krokového pohonu. Může to být jen generátor řídícího či reverzačního signálu ale i mikropočítač.

Elektronický komutátor je funkční část ovladače sestavená z elektronických obvodů, ve které se mění vstupní impulsní řídící signál na sled cyklicky se opakujících napětí na výstupech komutátoru. Výstupní signál komutátoru je nevýkonový. Pořadí kombinací napětí na výstupech komutátoru lze měnit v opačné pomocí elektrického reverzačního signálu. Tím se prakticky dosáhne změny smyslu otáčení krokového motoru.

Výkonový spínací zesilovač je funkční část ovladače, která výkonové zesiluje výstupní signál z elektronického komutátoru a přímo napájí vinutí krokového motoru, které je tak částí koncového obvodu výkonového spínacího zesilovače.

Řídící signál je elektrický signál vhodného tvaru a polarity, přivedený k řídícímu vstupu ovladače.

Reverzační signál je elektrický signál vhodného tvaru a polarity, přivedený k reverzačnímu vstupu ovladače.

Výstupní proud ovladače je proud tekoucí obvodem výkonového spínacího zesilovače, v němž je zapojena jedna fáze vinutí krokového motoru.

Výstupní napětí ovladače je napětí zdroje, kterým je napájen výkonový spínací zesilovač.

Krokový motor je impulsně napájený motor, jehož funkční pohyb je nespojitý a děje se po jednotlivých úsecích (krocích). K řízení krokového motoru slouží ovladač. Krok je pro každý krokový motor konstantou danou tvarem magnetického obvodu motoru.

Krok je mechanická odezva krokového motoru (jeho rotoru) na jeden řídící impuls, při níž rotor vykoná pohyb z výchozí magnetické klidové polohy do nejbližší magnetické klidové polohy.

Velikost kroku je jmenovitý úhel, daný konstrukcí a způsobem ovládání motoru, který odpovídá změně polohy rotoru po zpracování jednoho řídícího impulsu, jestliže motor není zatížen. Značí se α (°).

Magnetická klidová poloha je poloha, kterou zaujímá rotor nabuzeného krokového motoru, jestliže je statický úhel zátěže rovný nule. To znamená, že rotor je ideálně sesouhlasen s polohou statorového magnetického pole.

Řídicí kmitočet je kmitočet řídícího signálu. Značí se f_S (Hz).

Kmitočet kroků je počet kroků za jednu sekundu, které vykonává rotor krokového motoru při konstantním řídicím kmitočtu. Je stejný jako řídicí kmitočet, otáčí-li se rotor bez ztráty kroku. Značí se f_Z (Hz).

Statický moment je moment motoru, který je v rovnováze s kroutícím momentem působícím na hřídel stojícího nabuzeného krokového motoru a vychylujícím rotorem z magnetické klidové polohy o statický úhel zátěže. Značí se M_S (N.m). Důležitým poznatkem je, že nachází-li se rotor krokového motoru v magnetické klidové poloze, je statický moment motoru nulový. Soubor hodnot M_S v závislosti na statickém úhlu zátěže β tvoří statickou charakteristiku.

Statický vazební moment je největší statický moment, který se rovná kroutícímu momentu, jímž lze působit na hřídel stojícího nabuzeného krokového motru, aniž by došlo k roztržení magnetické vazby. Značí se M_{SV} (N.m).

Statický vazební moment nenabuzeného motoru je největší statický moment, který se rovná kroutícímu momentu, jímž lze působit na hřídel stojícího nenabuzeného krokového motoru, aniž by došlo k roztržení magnetické vazby. Značí se M_{SVO}(N.m).

Statický úhel zátěže je úhel, o který se vychýlí rotor nabuzeného krokového motoru z magnetické klidové polohy při dané zátěži na hřídeli krokového motoru. Značí se β (°).

Tolerance kroku je největší statická úhlová odchylka od velikosti kroku, která může nastat, když rotor krokového motoru bez zátěže vykoná jeden krok. Značí se $\Delta\alpha$ (°).

Největší úhlová chyba je největší úhlový rozdíl mezi úhlem odpovídajícím i-té magnetické klidové poloze a i-tým násobkem velikosti kroku, který může vzniknout během jedné otáčky rotoru. Značí se $\Delta\alpha_m$ (°).

Chod naprázdno je stav krokového motoru po připojení na zdroj elektrické energie a zdroj řídícího signálu, při kterém není na hřídeli motoru žádná zátěž.

Rozběhová oblast je oblast možných zátěží krokového motoru a takových kmitočtů kroků, na které se musí motor rozběhnout a z nich zastavit bez ztráty kroku i v případě, že rychlost změny řídícího kmitočtu není omezena.

Zátěž je současné působení vnějšího zátěžného a dynamického momentu na hřídel krokového motoru. Dynamické momenty jsou úměrné momentu setrvačnosti tělesa a změně jeho rychlosti.

Oblast omezené řiditelnosti je oblast možných zátěží krokového motoru a takových kmitočtů kroků, při kterých je motor schopen překonávat zátěž jen bez změny smyslu otáčení a zvyšovat nebo snižovat rychlost otáčení jen do určité hodnoty rychlosti změny řídícího kmitočtu.

Provozní oblast se skládá z rozběhové oblasti a oblasti omezené řiditelnosti.

Nejvyšší rozběhový kmitočet se značí f_{aom} (Hz). Je to nejvyšší řídící kmitočet, při kterém se krokový motor bez zátěže musí rozběhnout a zastavit bez ztráty kroku i v případě, že rychlost změny řídícího kmitočtu není omezena, jinak řečeno okamžitě nabývá hodnoty f_{aom}, nebo také, mění se skokem z nulové hodnoty na hodnotu f_{aom}.

Mezní rozběhový kmitočet je nejvyšší řídící kmitočet, při kterém se krokový motor s určitou zátěží musí rozběhnout i zastavit bez ztráty kroku i v případě, že rychlost změny řídícího kmitočtu není omezena. Značí se f_{am} (Hz).

Mezní rozběhový moment je největší zátěžný moment, který krokový motor překoná při daném rozběhovém kmitočtu a určitém momentu setrvačnosti připojeným na hřídeli. Značí se M_{am} (N.m).

Nejvyšší provozní kmitočet je nejvyšší řídící kmitočet, při kterém krokový motor bez zátěže je schopen se otáčet v jenom smyslu. Motor se na tento kmitočet může rozběhnout nebo z něj zastavit bez ztráty kroku jen do určité hodnoty rychlosti změny řídícího kmitočtu. Značí se f_{bom} (Hz).

Mezní provozní kmitočet je nejvyšší řídící, při kterém krokový motor s určitou zátěží je schopen otáčet se v jenom smyslu. Motor se na tento kmitočet může rozběhnout, nebo z něj zastavit bez ztráty kroku jen do určité hodnoty rychlosti změny řídícího kmitočtu. Značí se f_{bm} (Hz).

Mezní provozní moment je nevětší zátěžný moment, který krokový motor překoná při určitém mezním provozním kmitočtu. Značí se M_{bm} (N.m).

Nejvyšší provozní moment je největší hodnota mezního provozního momentu. Značí se M_{bmax} (N.m).

Dynamický úhel zátěže je úhel, o který se liší okamžitá poloha otáčejícího se rotoru od magnetické klidové polohy odpovídající poslednímu zpracovanému impulsu řídícího signálu. Značí se δ (°).

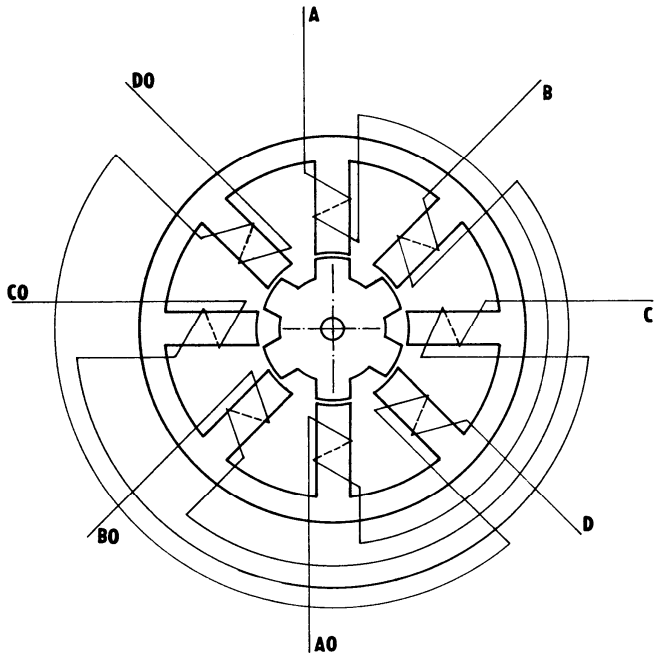
Maximální překmit je nevětší úhlová odchylka rotoru ve směru jeho pohybu od magnetické klidové polohy, kterou rotor zaujme po zpracování jednoho impulsu řídícího signálu. Značí se δ_m (°).

Budící proud je proud totožný s výstupním proudem ovladače. Značí se I (A).

Jmenovité napětí Je napětí totožné s výstupním napětím ovladače, tj. s napětím zdroje, kterým jsou napájeny výkonové spínací zesilovače ovladače krokového motoru. Značí se U (V).

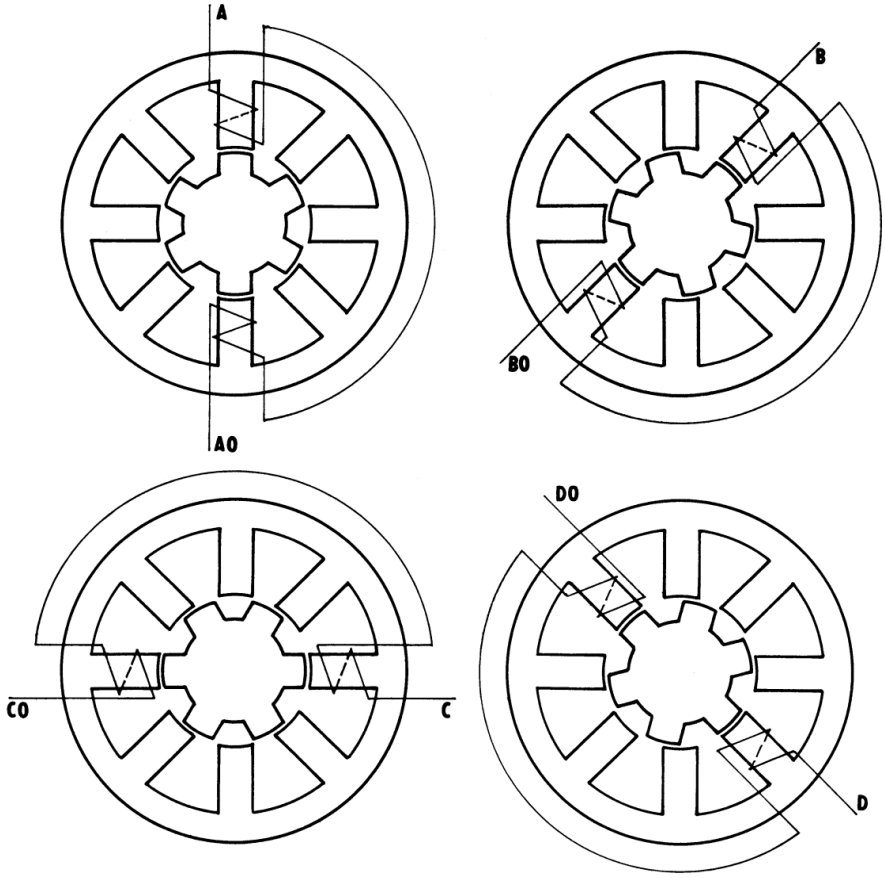
2.3 Princip krokového motoru

Nejjednodušeji se princip krokového motoru popisuje na krokovém motoru s pasivním rotorem. Pro účely vysvětlení byl použit čtyřfázový krokový motor s pasivním rotorem. Rotor tohoto typu motoru je tvořen pouze plechy nalisovanými na hřídel. V tomto případě má rotor směrem do vzduchové mezery 6 zubů bez vinutí. Na statoru jsou jednotlivé dvojice cívek navinuty na osmi zubech (pólech). Dvě protilehlé cívky tvoří jednu fázi. Jednotlivé fáze jsou označeny A, B, C, D a jsou připojeny k výstupům výkonového zesilovače. Ten spolu s elektronickým komutátorem zajišťuje jejich spínání v určitém pořadí podle použitého způsobu řízení. V tomto případě je použito unipolární buzení fází. Proud fází tedy protéká jenom jedním směrem. Řez magnetickým obvodem krokového motoru s pasivním rotorem je zobrazen na následujícím obrázku. [1]



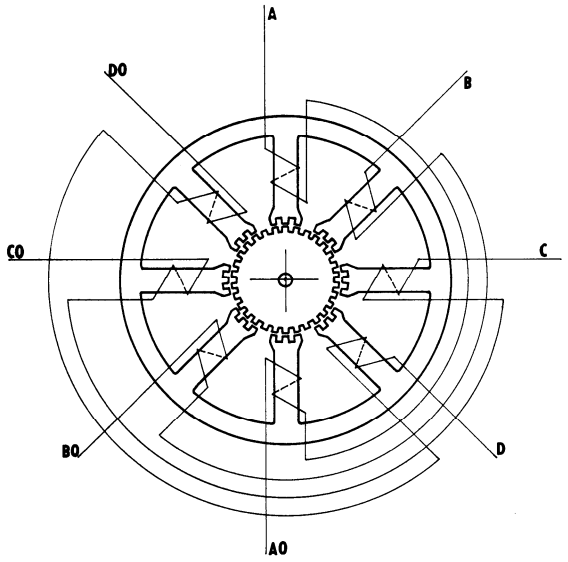
Obrázek 1 - Magnetický obvod krokového motoru s pasivním rotorem [1]

Po připojení motoru k ovladači se rotor nachází v náhodné poloze. Po zapnutí napájení ovladače se nejméně jedna fáze motoru nabudí a vytvoří magnetické pole. U krokových motorů s pasivním rotorem se rotor natočí tak, aby výsledný magnetický odpor (reluktance) byl co nejmenší. V tomto případě tak, aby se nejbližší rotorové zuby srovnaly pod póly právě buzené fáze. Na následujícím obrázku jsou zobrazeny situace, kdy jsou sepnuté jednotlivé fáze. Je vidět, že jak se přesouváme po pólech dál od aktuálně vybuzeného, rotorové zuby se nekryjí se statorovými čím dál více. Zároveň je patrné, že fáze musíme spínat postupně. Nemůžeme například přeskočit z fáze A rovnou na fázi C, jelikož by se mohl rotor otočit jakýmkoli směrem. [1]



Obrázek 2 - Znázornění pohybu rotoru při přepínání fází [1]

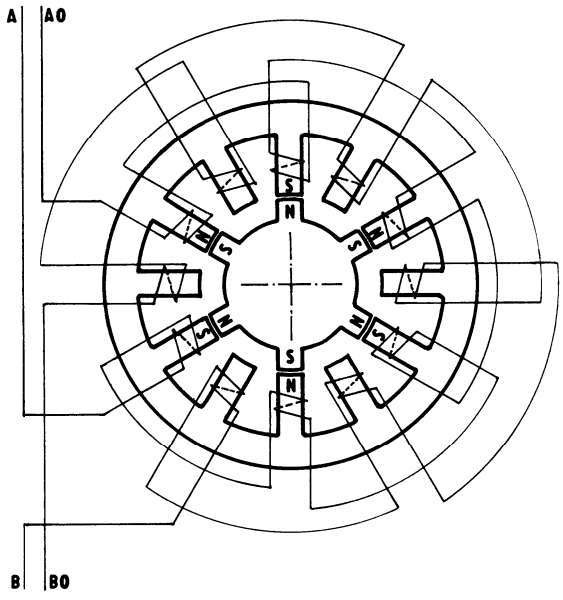
Motor použitý k vysvětlení základního principu má však relativně velkou velikost kroku. V daném případě 15° pro čtyřtaktní řízení a 7,5° pro osmitaktní řízení. V některých aplikacích je ovšem třeba dosáhnout vyššího rozlišení kroku. Toho lze dosáhnout drážkováním hlav pólů směrem do vzduchové mezery. Stejné drážkování je realizováno i na rotoru. Tímto způsobem nelze dosáhnout libovolné velikosti kroku ale pouze určitých hodnot. Na následujícím obrázku je zobrazen magnetický obvod čtyřfázového krokového motoru s pasivním rotorem, uspořádaný pro velikost kroku 3°.



Obrázek 3 - Magnetický obvod motoru s menší velikostí kroku [1]

2.4 Krokový motor s radiálně polarizovaným permanentním magnetem

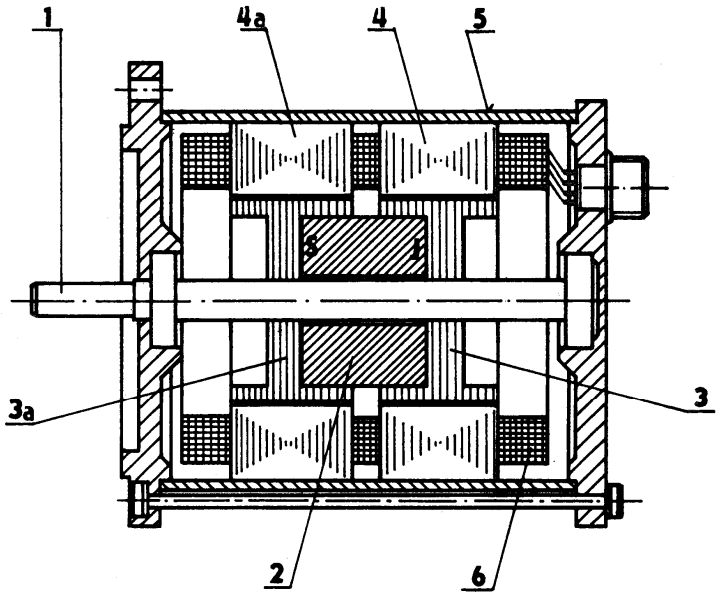
U tohoto typu motoru jsou na rotoru pólové nástavce zmagnetovány, přičemž stator má dvojnásobný počet pólů oproti rotoru. U tohoto typu motoru Je zapotřebí měnit magnetickou polaritu pólů statoru, je tedy zapotřebí použít bipolární způsob buzení nebo bifilární vinutí v cívkách (bude popsáno dále). Při správném buzení pólů statoru dojde k pootočení magnetického pole a následnému sesouhlasení polohy rotoru s tímto nově vzniklým polem. Následující obrázek ukazuje průřez magnetickým obvodem dvoufázového krokového motoru s radiálně polarizovaným permanentním magnetem. [1]



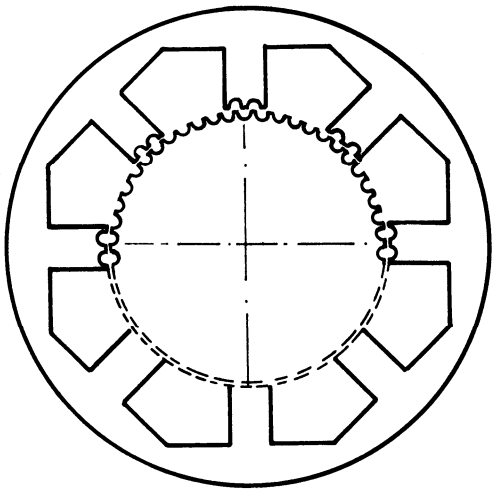
Obrázek 4 - Magnetický obvod dvoufázového krokového motoru s radiálně polarizovaným magnetem [1]

2.5 Krokový motor s axiálně polarizovaným permanentním magnetem

Tento typ krokového motoru, který je také označován jako hybridní, je v současné době nejpoužívanější. Na hřídeli rotoru tohoto typu motoru jsou nalisované dva pólové nástavce z plechů, mezi nimiž je vložen permanentní magnet tvaru mezikruží. Magnet má magnetické póly vytvořené na podstavách mezikruží, takže magnetický tok z něj vychází směrem do pólových nástavců. Pólové nástavce mají po obvodu směrem do vzduchové mezery zuby, jejichž počet ovlivňuje počet kroků na otáčku. Jednotlivé pólové nástavce rotoru jsou natočeny tak, aby se zuby jednoho kryly s drážkami druhého, přičemž šířka statorových i rotorových zubů je stejná. Na následujících obrázcích je řez krokovým motorem s axiálně polarizovaným magnetem a řez jeho magnetickým obvodem. 1 – hřídel, 2 – magnet, 3 a 3a – rotorové pólové nástavce, 4 a 4a – statorové svazky, 5 – kostra, 6 – vinutí. [1]



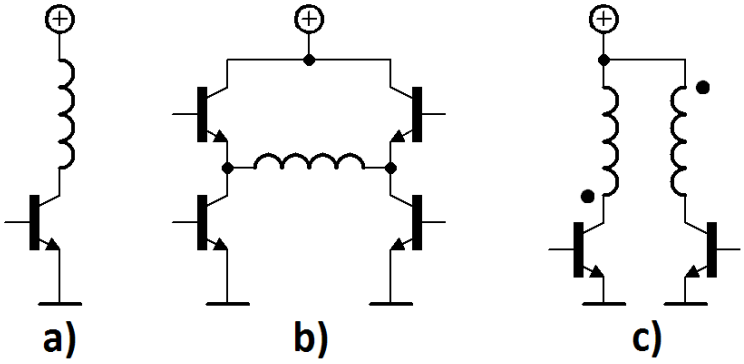
Obrázek 5 - Řez krokovým motorem s axiálně polarizovaným magnetem [1]



Obrázek 6 - Magnetický obvod krokového motoru s axiálně polarizovaným magnetem [1]

2.6 Typy vinutí krokových motorů

V různých typech krokových motorů se používají různé typy vinutí. Základní rozdělení vinutí je na monofilární a bifilární. Monofilární vinutí se skládá pouze z jedné cívky, jejíž magnetickou polaritu lze měnit změnou smyslu proudu ve vinutí. Bifilární vinutí se skládá ze dvou cívek, které magneticky působí proti sobě. Pro změnu polarity je potřeba sepnout proud druhou částí vinutí. Pokud má motor na rotoru aktivní části, musí se měnit polarita jednotlivých statorových pólů, aby docházelo k otáčení rotoru. U motorů s pasivním rotorem se polarita pólu měnit nemusí. Následující obrázek ukazuje zapojení jednotlivých typů vinutí. Na obrázku a) je monofilární vinutí buzené unipolárně. V takovémto zapojení nelze měnit polaritu pólů, používá se tedy jen u motorů s pasivním rotorem. Na obrázku b) je monofilární vinutí zapojeno bipolárně pomocí H-můstku. Toto zapojení umožňuje změnu polarity pólů, lze jím tedy řídit krokové motory s aktivním rotorem, avšak na každou fázi jsou potřeba 4 spínací prvky. Na obrázku c) je bifilární vinutí buzené unipolárně. V tomto zapojení jedna cívka budí severní polaritu pólu a druhá cívka jižní polaritu. Lze tak také řídit krokové motory s aktivním rotorem. [2]



Obrázek 7 - Různé typy buzení fází krokových motorů [2]

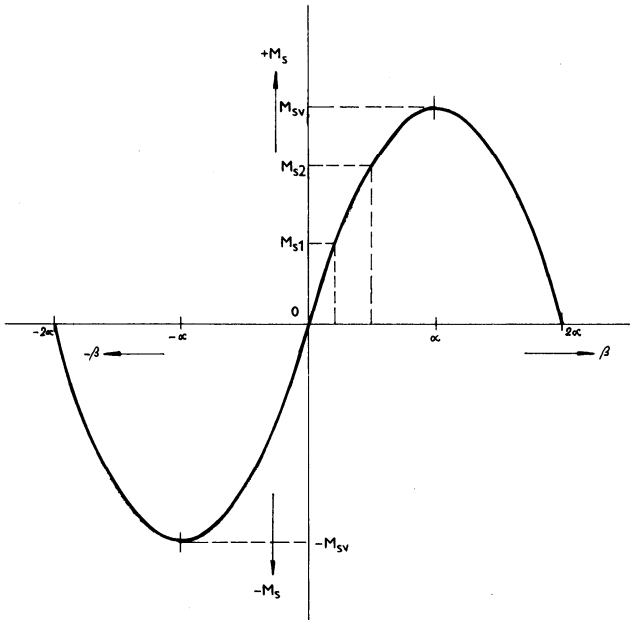
2.7 Základní charakteristiky krokových motorů

Zde jsou stručně popsány charakteristika statická a charakteristika dynamická.

2.7.1 Statická charakteristika

Tato charakteristika vyjadřuje závislost momentu krokového motoru na statickém úhlu zátěže. Statická charakteristika krokového motoru je charakteristikou synchronního stroje. Z průběhu charakteristiky je patrné, že moment krokového motoru roste se stoupajícím zátěžným úhlem β a v případě nulového zátěžného úhlu je moment nulový. Motor se v takové situaci nachází v magnetické klidové poloze. Důležitým bodem na statické charakteristice je statický vazební moment M_{SV}. V případě překročení tohoto momentu dojde k roztržení magnetické vazby a motor

proběhne. Tomuto momentu odpovídá zátěžný úhel o velikosti jmenovitého úhlu kroku α. Následující obrázek ukazuje příklad statické charakteristiky krokového motoru. [1]

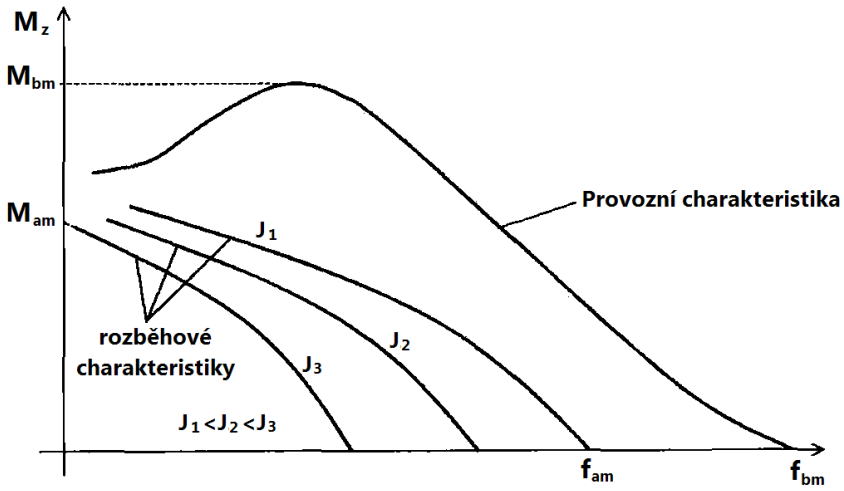


Obrázek 8 - Statická charakteristika krokového motoru [1]

2.7.2 Dynamická charakteristika

Dynamické charakteristiky krokového motoru udávají závislost momentu krokového motoru na spínací frekvenci. Dynamické charakteristiky se dále rozdělují na rozběhové a provozní. Rozběhová charakteristika vymezuje oblast možných zátěží krokového motoru a spínacích frekvencí, na které se motor musí rozběhnout a z nich zastavit bez ztráty kroku. Velikost této oblasti závisí na momentu setrvačnosti pohonu. Z rozběhové charakteristiky lze získat mezní rozběhovou frekvenci a mezní rozběhový moment. [2]

Provozní charakteristika udává provozní oblast krokového motoru. Tato oblast se skládá z rozběhové oblasti a oblasti omezené řiditelnosti. Oblast omezené řiditelnosti vyznačuje možné zátěže a spínací frekvence, na které se krokový motor dokáže dostat jen postupnou změnou řídící frekvence. Následující obrázek ukazuje příklady rozběhových charakteristik pro různé momenty setrvačnosti zátěže a pracovní charakteristiku.[2]



Obrázek 9 - Dynamické charakteristiky [2]

2.8 Použitý motor

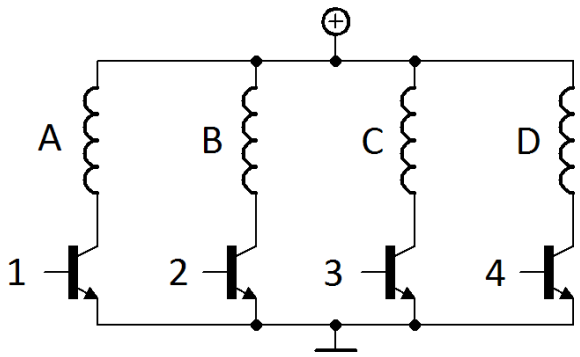
Pro účely této práce byl použit rotační hybridní krokový motor od firmy Microcon s aktivním rotorem. Použit byl typ SX34-2740. Motor má univerzálně vyvedené konce cívek, lze tedy vinutí fází uspořádat pro bipolární i unipolární způsob řízení. Tento motor má při bipolárním zapojení a jmenovitém proudu v obou fázích statický moment 4Nm. Jmenovitý proud fázemi je 2,75A při sériovém zapojení vinutí a 5,5 A při paralelním zapojení. Indukčnost je při sériovém zapojení 14 mH a při paralelním zapojení 3,5 mH. Odpor vinutí při sériovém zapojení je 1,68 Ω, při paralelním zapojení je 0,42 Ω. Jmenovitá délka kroku daného motoru je 1,8° s tolerancí ±0,1°. [5]

3 Způsoby řízení krokových motorů

Základní princip řízení krokového motoru spočívá ve spínání fázových proudů v předem určených sekvencích. Tyto sekvence se liší podle počtu fází motoru a jejich zapojení. Základní rozdělení typů zapojení krokových motorů je na unipolární a bipolární. Unipolární zapojení znamená, že jednotlivými fázemi protéká proud pouze jedním směrem. Toto zapojení má výhodu jednodušší řídící elektroniky, stačí k němu 4 tranzistory, které spínají proudy jednotlivými fázemi. Naproti tomu při bipolárním řízení může proud cívkami protékat oběma směry. To obnáší složitější řídící elektroniku. Každou fázi je třeba ovládat H-můstkem, který se každý skládá ze čtyř tranzistorů. Na dvoufázový motor s bipolárním zapojením fází je tedy třeba 8 tranzistorů. Informace pro tuto kapitolu byly čerpány z [1].

3.1 Unipolární řízení

Při unipolárním řízení protéká proud fázemi pouze jedním směrem. To přináší výhodu jednoduššího řízení, ke kterému stačí pouze jeden tranzistor na každou fázi. Na následujícím obrázku je zjednodušené zapojení čtyřfázového unipolárního krokového motoru. Dále budou popsány jednotlivé metody řízení při tomto zapojení. Znaménko (+) značí, že danou fází teče proud. Označení / znamená, že danou fází proud neteče. Tranzistory jsou zde označeny číslicemi 1 až 4 pro popis sekvencí v tabulkách. Fáze jsou označeny A, B, C a D.



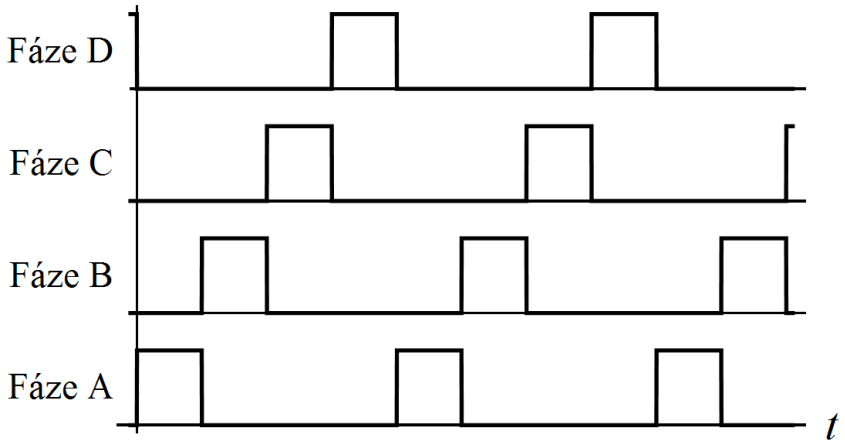
Obrázek 10 - Schematické unipolární zapojení čtyřfázového krokového motoru

3.1.1 Unipolární řízení čtyřtaktní s jednou aktivní fází

Při tomto způsobu řízení jsou spínány jednotlivé fáze postupně. Při přepnutí z jedné fáze na následující dojde k otočení rotoru o jmenovitou velikost kroku. Tento způsob řízení je principielně nejjednodušší způsob, jak krokový motor řídit. Následující tabulka a obrázek ukazuje průběhy proudů jednotlivými fázemi a pořadí spínání tranzistorů.

Tabulka 1 - Unipolární řízení čtyřtaktní s jednou aktivní fází

Sekvence	A	B	C	D	tranzistory
1	+	/	/	/	1
2	/	+	/	/	2
3	/	/	+	/	3
4	/	/	/	+	4



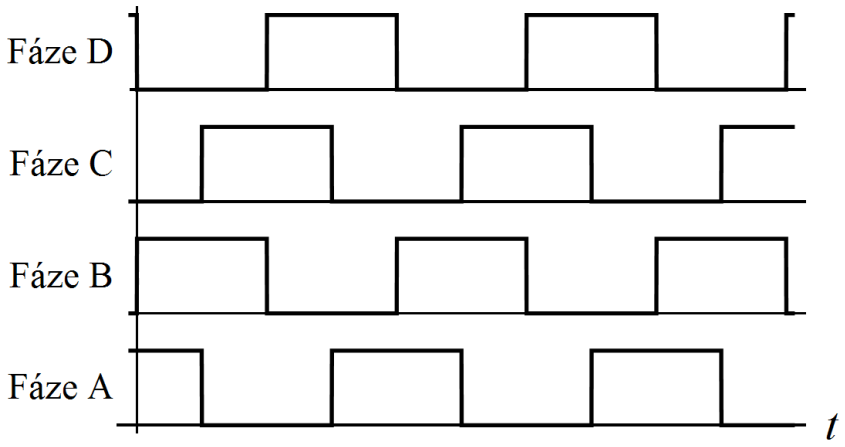
Obrázek 11 - Průběhy fázových proudů při unipolárním čtyřtaktním řízení s jednou aktivní fází [1]

3.1.2 Unipolární řízení čtyřtaktní se dvěma aktivními fázemi

Při tomto typu řízení jsou aktivní vždy dvě fáze najednou. To má za následek vyšší statický moment ale i vyšší spotřebu motoru. Zároveň dojde k úhlovému vychýlení rotoru o polovinu kroku, protože se rotor srovná do nové klidové polohy, která leží mezi klidovými polohami obou sepnutých fází. Následující tabulka a obrázek ukazují průběhy proudů fázemi a pořadí spínání tranzistorů.

Tabulka 2 - Unipolární řízení čtyřtaktní se dvěma aktivními fázemi

Sekvence	A	B	C	D	tranzistory
1	+	+	/	/	1, 2
2	/	+	+	/	2, 3
3	/	/	+	+	3, 4
4	+	/	/	+	4, 1



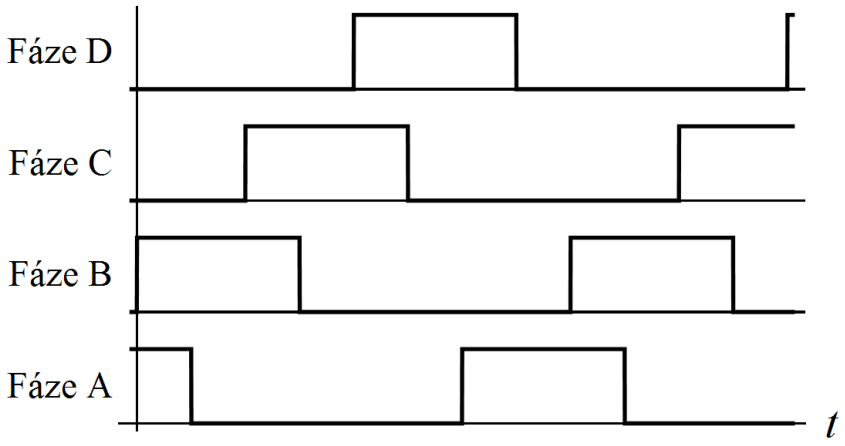
Obrázek 12 - Průběhy fázových proudů při unipolárním čtyřtaktním řízení se dvěma aktivními fázemi [1]

3.1.3 Unipolární řízení osmitaktní

Tento způsob řízení vznikl kombinací dvou předchozích způsobů řízení. Jsou zde střídavě buzeny jedna a dvě fáze. Jelikož řízení se dvěma aktivními fázemi vychýlí rotor o polovinu kroku, dojde při dalším přepnutí na řízení s jednou aktivní fází k posunu pouze o polovinu kroku. To ve výsledku zdvojnásobí rozlišení motoru, avšak dochází ke kolísání momentu. Následující tabulka a obrázek ukazují průběhy proudů fázemi a pořadí spínání tranzistorů.

Tabulka 3 - Unipolární řízení osmitaktní

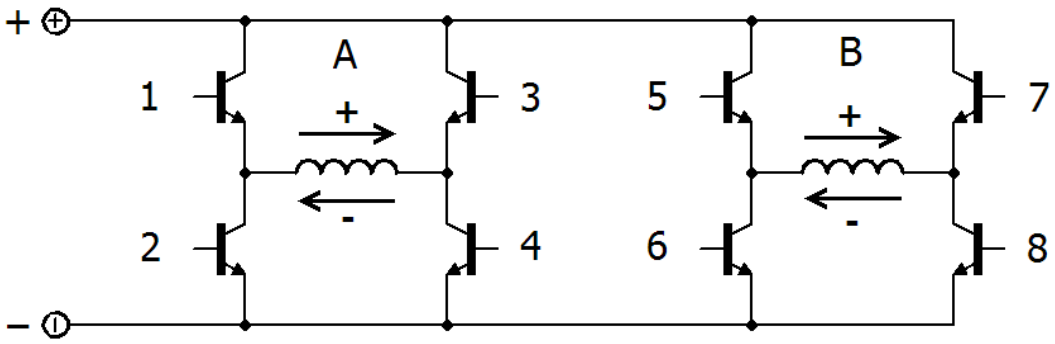
Sekvence	A	B	C	D	Tranzistory
1	+	+	/	/	1, 2
2	/	+	/	/	2
3	/	+	+	/	2, 3
4	/	/	+	/	3
5	/	/	+	+	3, 4
6	/	/	/	+	4
7	+	/	/	+	4, 1
8	+	/	/	/	1



Obrázek 13 - Průběhy fázových proudů při unipolárním osmitaktním řízení [1]

3.2 Bipolární řízení

Při bipolárním řízení může protékat proud fázemi v obou směrech. Aby proud mohl protékat fází v obou směrech, musí být buzena z H-můstků. Na každou fázi tak připadají čtyři tranzistory. Bipolární řízení je použito v této práci. Následující obrázek ukazuje určení směru proudu jednotlivými fázemi. Pokud teče proud fází v kladném směru, je označen (+), pokud teče proud fází v záporném směru, je označen (-). Pokud fází proud neteče, je označena /. Zapojení H-můstků je schematicky znázorněno na následujícím obrázku. Tranzistory jsou označeny číslicemi od 1 do 8 pro popis spínacích sekvencí v tabulkách a fáze jsou označeny A a B.



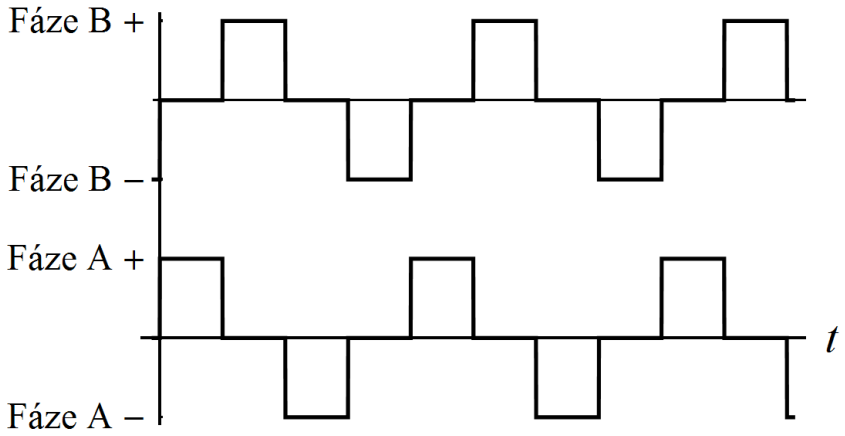
Obrázek 14 - Označení proudů v H-můstcích

3.2.1 Bipolární řízení čtyřtaktní s jednou aktivní fází

Při tomto druhu řízení je aktivní vždy jen jedna fáze. Jsou přepínány ve čtyřech krocích, z nichž každý vyvolá otočení rotoru o jmenovitý úhel kroku. Následující tabulka a obrázek ukazují řídící sekvenci jednotlivých kroků. A čísla tranzistorů, které jsou sepnuty současně.

Tabulka 4 - Bipolární řízení čtyřtaktní s jednou aktivní fází

Sekvence	A	B	Tranzistory
1	+	/	1, 4
2	/	+	5, 8
3	-	/	2, 3
4	/	-	6, 7



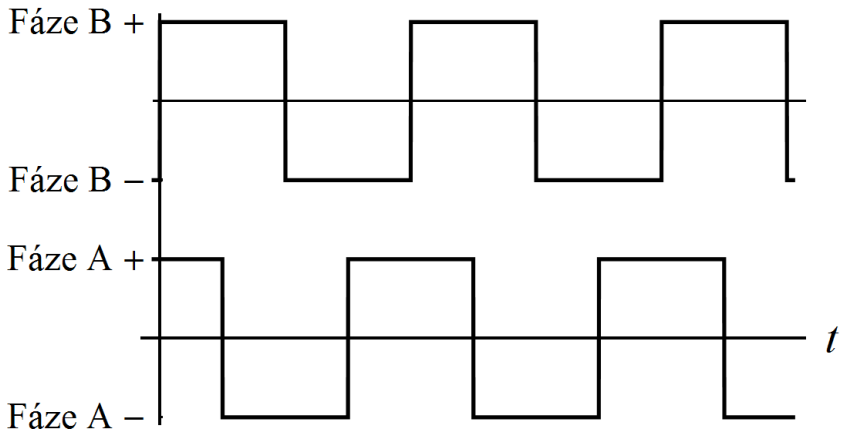
Obrázek 15 - Průběhy fázových proudů při bipolárním čtyřtaktním řízení s jednou aktivní fází [1]

3.2.2 Bipolární řízení čtyřtaktní se dvěma aktivními fázemi

Při tomto druhu řízení jsou vždy aktivní obě fáze motoru. To má za následek větší statický moment motoru a lepší tlumení oscilací motoru. To umožní dosáhnout vyšších krokovacích frekvencí. Současná magnetizace dvou fází najednou ovšem zvyšuje spotřebu motoru. Zároveň je rotor vychýlen o polovinu velikosti jednoho kroku, přičemž samotná velikost kroku zůstane nezměněna. Následující tabulka ukazuje sekvenci jednotlivých kroků pro řízení se dvěma aktivními fázemi.

Tabulka 5 - Bipolární řízení čtyřtaktní se dvěma aktivními fázemi

Sekvence	A	B	Tranzistory
1	+	-	1, 4, 6, 7
2	+	+	1, 4, 5, 8
3	-	+	2, 3, 5, 8
4	-	-	2, 3, 6, 7



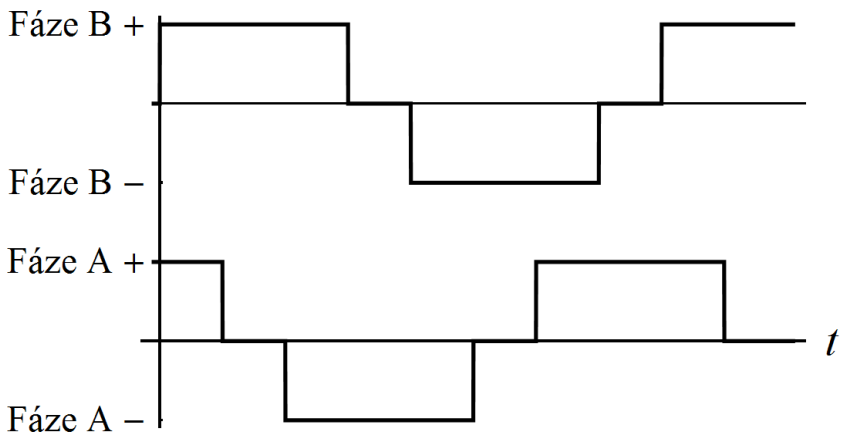
Obrázek 16 - Průběhy fázových proudů při čtyřtaktním bipolárním řízení se dvěma aktivními fázemi [1]

3.2.3 Bipolární řízení osmitaktní

Tento druh řízení vznikl kombinací dvou předchozích. Jsou při něm střídavě buzeny dvě a jedna fáze. Jeho hlavní výhodou je, že zdvojnásobí rozlišení motoru. Velikost kroku je tedy poloviční. V našem případě 0,9° Nevýhodou je kolísání momentu při přepínání fází. Následující tabulka ukazuje sekvenci pro osmitaktní řízení (s poloviční krokem).

Tabulka 6 - Bipolární řízení osmitaktní

Sekvence	A	B	Tranzistory
1	+	+	1, 4, 5, 8
2	+	/	1, 4
3	+	-	1, 4, 6, 7
4	/	-	6, 7
5	-	-	2, 3, 6, 7
6	-	/	2, 3
7	-	+	2, 3, 5, 8
8	/	+	5, 8



Obrázek 17 - Průběhy fázových proudů při osmitaktním bipolárním řízení [1]

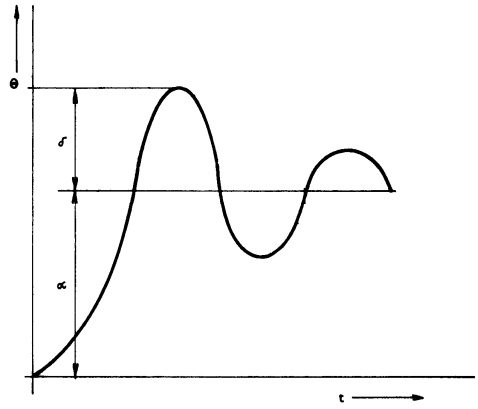
3.3 Řízení úhlu natočení hřídele krokového motoru

Hřídel krokového motoru dokáže zaujímat konečný počet přesně definovaných poloh. Tyto polohy jsou nazývány magnetické klidové polohy. Počet těchto poloh je definován tvarem magnetických obvodů rotoru a statoru krokového motoru. Úhel, o který se rotor otočí, při provedení jednoho kroku, se nazývá velikost kroku a značí se α. Velikost kroku je zpravidla udávána ve stupních. Motor použitý pro tuto práci má konstrukcí dáno 200 kroků na otáčku, což odpovídá velikosti kroku 1,8°. Při osmitaktním řízení lze dosáhnout poloviční velikosti kroku 0,9°. Hřídel rotoru se tedy dokáže otáčet pouze o násobky těchto velikostí kroku. Při čtyřtaktním řízení tak může vznikat odchylka od požadovaného úhlu až 0,9° a 0,45° při osmitaktním řízení. Tyto hodnoty jsou však pouze teoretické a odchylka od požadovaného úhlu může být ještě ovlivněna tolerancí kroku daného motoru, což je v případě použitého motoru ±0,1°, a hlavně statickým úhlem zátěže, který vychyluje rotor z magnetické klidové polohy. Při chodu motoru má vliv také dynamický úhel zátěže. Pro kontrolu úhlu natočení rotoru lze použít zpětnou vazbu ve formě enkodérů umístěných na hřídel motoru. Řízení se zpětnou vazbou se také nazývá řízení v uzavřené smyčce. Toto řízení není tolik náchylné na proběhnutí motoru při rozjezdech a při brzdění.

3.3.1 Oscilace rotoru

Při přechodu z jedné magnetické polohy do druhé nastává vlivem momentu setrvačnosti rotoru k překmitnutí nové rovnovážné polohy. Tento jev se nazývá oscilace. Rotor harmonicky kmitá kolem nové rovnovážné polohy s malým tlumením, daným převážně třením. Amplituda prvních překmitů je blízká velikosti kroku motoru. Tyto oscilace mohou způsobit rezonanci celého motoru, pokud je řídící kmitočet blízký rezonančnímu kmitočtu motoru. Celý motor se tak rozvibruje, což může vést až ke ztrátě kroku. Zároveň je motor také značně hlučný. Jelikož je motor při určitých řídících kmitočtech vlivem rezonance nestabilní, snažíme se těmto řídícím kmitočtům vyhnout. Problém oscilací se dá také zmírnit osmitaktním řízením, jelikož při něm není jednorázová změna úhlu

tak velká. Téměř úplně tento problém vymizí při použití mikrokrokování, kde jsou jednorázové změny úhlu pouze zlomkem velikosti kroku. Na následujícím obrázku je zobrazen průběh úhlu natočení rotoru θ. Úhel α je jmenovitá velikost kroku a úhel δ značí maximální překmit.



Obrázek 18 - Znázornění oscilací [1]

3.4 Řízení otáček krokového motoru

Pro změnu otáček krokového motoru je potřeba změnit frekvenci kroků. Otáčky krokového motoru lze získat z následujícího vztahu: $n = \frac{60.f_Z.\alpha}{360}$, kde f_Z je kmitočet kroků a α je velikost kroku [1]. Pro přesné řízení rychlosti je tedy nutné přesně definovat kmitočet kroků. Při pomalých rychlostech však není pohyb rotoru plynulý. Po příchodu řídícího impulsu se motor skokově přesune do nové magnetické klidové polohy a potom se již nehýbe. Rychlost tedy není konstantní a také dochází k již zmíněnému jevu oscilací. Pro vyšší řídící kmitočty však již rotor nestíhá oscilovat kolem rovnovážné polohy, jelikož ta rychle přechází na další. Okamžitá úhlová rychlost rotoru je tedy pro vyšší řídící kmitočty konstantní. Problém nespojité rychlosti při nízkých řídících kmitočtech lze do jisté míry vylepšit použitím mikrokrokování. Rotor při něm zaujímá mnohem více magnetických klidových poloh než při standardním čtyřtaktním řízení a jeho pohyb je plynulejší.

3.4.1 Řízení otáček mikrokontrolérem

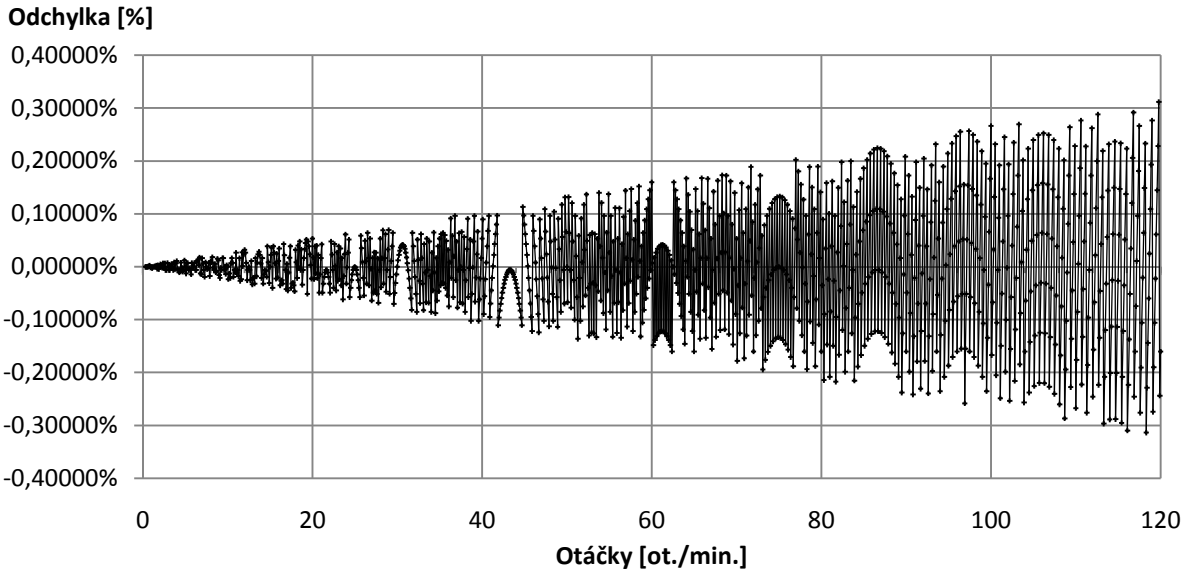
V moderních aplikacích je pohon s krokovým motorem řízen mikrokontrolérem. Ten musí zajistit správné posloupnosti spínání proudů cívkami a řídící kmitočet, který udržuje požadovanou rychlost motoru. Mikrokontrolér tedy zastává funkci elektronického komutátoru. Pro udržení konstantního řídícího kmitočtu se v mikrokontroléru využívá systému přerušení. Čítač mikrokontroléru, který čítá do určité hodnoty, což trvá přesně definovanou dobu, vyvolá po dosažení komparační hodnoty přerušení, ve kterém se provede krok. To typicky znamená změnu proudů cívkami a otočení rotoru motoru do nové klidové magnetické polohy.

Při tomto typu řízení je tedy důležité nastavení čítače. Je nutné, aby doba jednoho tiku čítače byla dostatečně krátká. To zajistí vyšší rozlišení rychlosti při vyšších rychlostech. Ale zároveň nesmí být doba tiku čítače moc krátká. To by způsobilo omezení minimální rychlosti. V následující tabulce jsou konkrétní možnosti použitého mikrokontroléru na desce Arduino ATmega2560, který pracuje na frekvenci 16MHz. Jsou zde uvedeny doby jednoho tiku, maximální periody a minimální rychlosti motoru pro všechny možné nastavení předděličky čítače.

Tabulka 7 - Možné nastavení čítače

Předdělička	Doba tiku [µs]	Maximální perioda [ms]	Minimální otáčky [ot./min]
1	0,0625	4,096	73,24
8	0,5	32,768	9,16
64	4	262,14	1,14
256	16	1048,56	0,29
1024	64	4194,24	0,07

Z tabulky je patrné, že pro předděličku 1 a 8 by motorem nešlo otáčet pomaleji, než 9,16 a 73,24 otáček za minutu, což by mohlo být v některých aplikacích nežádoucí. Předdělička 1024 by zase měla malou přesnost nastavení rychlosti při vyšších otáčkách. Jako optimální se tedy jeví předděličky 64 a 256. S předděličkou 64 je minimální rychlost 1,14 ot./min. a odchylky v nastavené rychlosti při rychlostech okolo 100 ot./min jsou zhruba do 0,06%. Kdežto při použití předděličky 256 je minimální rychlost 0,29 otáčky za minutu, ale odchylky v okolí 100 otáček za minutu jsou až 0,27%. Tyto odchylky od požadované rychlosti jsou způsobeny neschopností mikrokontroléru nastavit libovolnou velikost komparační hodnoty pro čítač. Komparační hodnota musí být v každém případě celé číslo a chyba tak vzniká zaokrouhlováním. Následující graf ukazuje průběh odchylky skutečné rychlosti od žádané rychlosti s použitou předděličkou 256. Je vidět, že maximální odchylka stoupá lineárně s požadovanou rychlostí.



Obrázek 19 - Závislost odchylky skutečné rychlosti od požadované rychlosti

4 Arduino

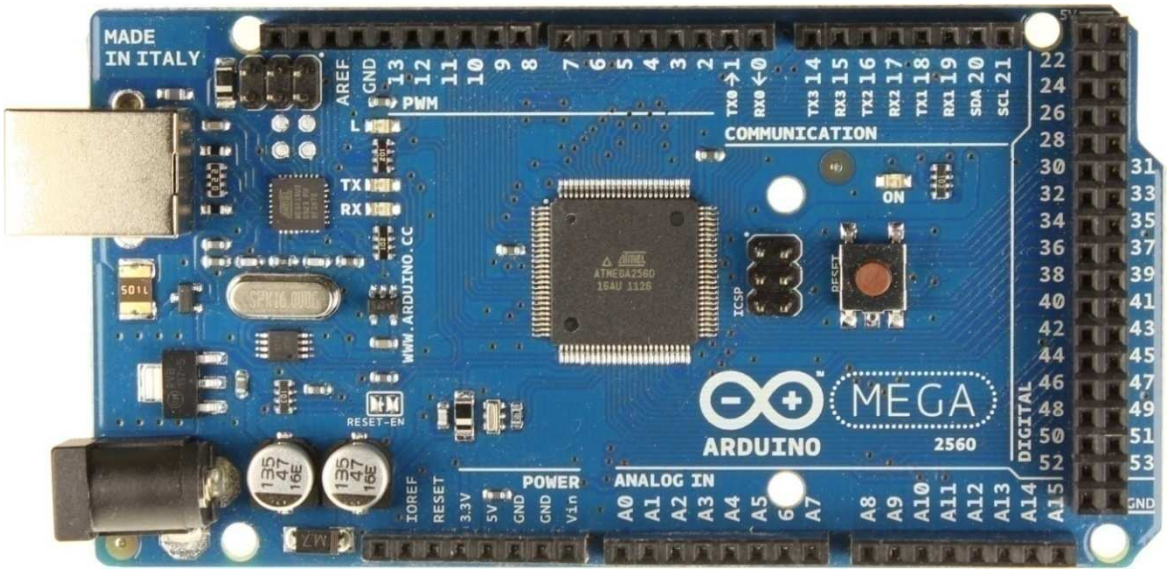
Jedná se o open source vývojovou platformu, která je založená na mikrokontrolérech Atmel AVR. Součástí projektu je i vývojové prostředí, které vychází z projektu Wiring a projektu Processing. Cílem těchto projektů bylo co nejvíce zjednodušit programování i pro neodbornou veřejnost. [3]

4.1 Hardware

Vývojové desky Arduino obsahují osmibitové mikrokontroléry od firmy Atmel. Existuje celá řada vývojových desek s různými mikrokontroléry. Například ATMega8, ATMega168, ATMega328, ATMega1280 a ATMega2560. Dále jsou vývojové desky vybaveny různými podpůrnými obvody, jako například obvody pro řízení sériové komunikace. Na starších deskách byly použity obvody od FTDI, na novějších jsou ATmega16U2. Tyto komunikační obvody zajišťují komunikaci s počítačem emulací virtuálního COM portu přes USB. Všechny vývojové desky jsou vybaveny různým počtem vstupně výstupních digitálních portů, analogově-digitálními převodníky, PWM výstupy a komunikačními porty. K vývojovým deskám je k dispozici velké množství periferního hardwaru, jako např. displeje, bluetooth moduly, ultrazvukové senzory, výkonové spínací prvky a spousta dalších. Tato široká variabilita umožňuje použít Arduino pro širokou škálu aplikací. Ke většině periferních zařízení jsou navíc k dispozici volně dostupné ovládací knihovny. [3]

4.2 Hardware použitý v projektu

Pro účely tohoto projektu byla použita vývojová deska Arduino Mega 2560. Je osazena osmibitovým procesorem Atmel AVR ATMega2560. Pro řízení sériové komunikace je použit procesor Atmel ATMega16U2. Vývojová deska má 54 digitálních vstupně výstupních pinů, z nichž 15 lze použít jako výstupy PWM. Dále má 16 analogových vstupů, 4 UART porty, USB konektor, napájecí konektor, konektor pro programování ICSP a tlačítko resetu. Deska je také vybavena krystalovým oscilátorem pracujícím na frekvenci 16MHz. Vývojová deska může být napájena buď externím napájením 7-12V, nebo z USB, přičemž výběr zdroje probíhá automaticky. Samotný mikrokontrolér je napájen 5V. Tento typ mikrokontroléru disponuje 256KB flash paměti pro uložení kódu, z nichž 8KB je vyhrazeno pro bootloader. Dále mikrokontrolér disponuje 8KB paměti SRAM a 4KB paměti EEPROM. Jednotlivými digitálními vstupně-výstupními piny může protékat proud 40mA. Piny jsou vybaveny pull-up rezistory o hodnotě 20-50 KΩ. Pull-up rezistory jsou v základním nastavení odpojeny. USB port na vývojové desce je vybaven opakovatelnou proudovou pojistkou, která při překročení proudu 500mA odpojí vývojovou desku od počítače, aby nedošlo k poškození počítače. Na následujícím obrázku je vidět Vývojová deska použitá v této práci. [4]



Obrázek 20 - Arduino Mega2560 [4]

4.3 Propojení s výkonovým spínacím zesilovačem

Jelikož logické obvody Arduina i výkonového spínacího zesilovače pracují na úrovni TTL 5V, není zde třeba používat úrovňový převodník. Výstupy z Arduino jsou rovnou připojeny na logické vstupy řízení výkonových tranzistorů. K propojení bylo potřeba vyrobit propojovací kabel. Na straně vstupů obvodů pro řízení tranzistorů je 15-ti pinový konektor cannon DA-15 female. Tranzistory jsou spínány signály 1 až 8 a zbytek pinů tvoří zem. Jako výstupy z vývojové desky Arduina byly vybrány digitální piny 6 až 13, které je pro další práci rovněž možno použít jako výstupy PWM. Jako propojovací kabel byl použit UTP kabel.

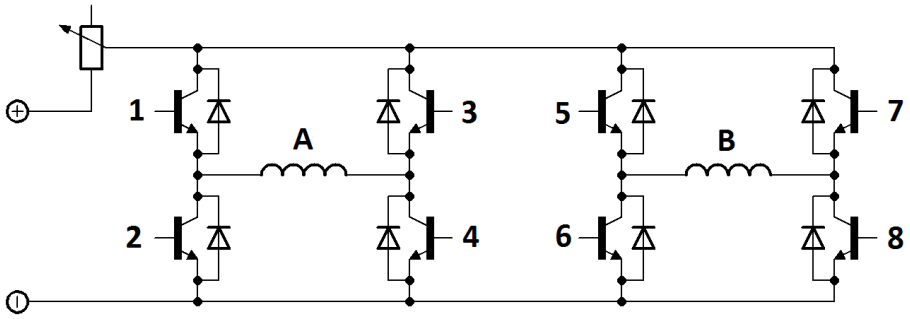
4.4 Programování

Programování probíhá přímo z vývojového prostředí Arduino software. Je potřeba pouze nastavit na jaký COM port je arduino připojeno k počítači. Mikrokontroloréry na vývojových deskách jsou z výroby vybaveny bootloaderem. To znamená, že k programování není potřeba externí programátor, ale zkompilovaný program je do mikrokontroléru nahrán po sběrnici USB, která může být mimo programování využita ke komunikaci s počítačem. Vývojová deska je také vybavena ICSP (In Circuit Serial Programming) konektorem. Lze tedy pro naprogramování mikrokontroléru použít i externí programátor. Při programování je využíván protokol STK500.

5 Výkonová část

Jako výkonový spínací zesilovač krokového motoru byl použit měnič s IGBT tranzistory LOSER III (červené akvárko). Toto zapojení disponuje osmi samostatně spínatelnými IGBT tranzistory, což umožňuje zapojit je jako dva samostatné H-můstky. Tyto dva H-můstky dokážou bez problému řídit dvoufázový krokový motor v bipolárním zapojení. Na jeden H-můstek jsou tedy použity 4 IGBT tranzistory. Jednotlivé tranzistory jsou spínány podle vstupních signálů na vstupním konektoru výkonového modulu. Piny 1 až 8 na konektoru odpovídají tranzistorům 1 až 8.

5.1 Schematické zapojení H-můstků



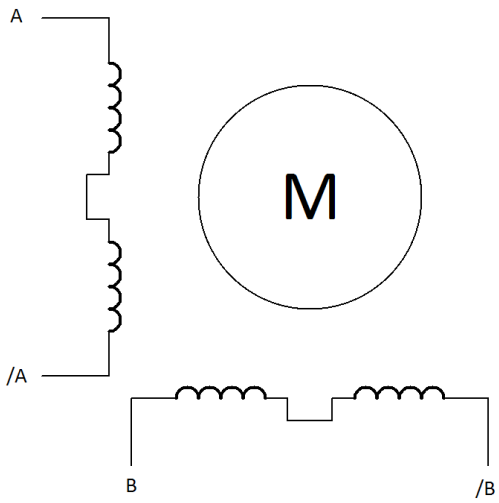
Obrázek 21 - Schematické zapojení H-můstků

Ze schématu je patrné, že tranzistory musíme spínat po dvojicích, abychom dosáhli požadovaných směrů proudů v cívkách krokového motoru. Pro další úvahy byly určeny proudy fázemi následovně: Při sepnutých tranzistorech 1 a 4 protéká cívkou A proud v kladném směru (+). Při sepnutých tranzistorech 2 a 3 protéká cívkou A proud v záporném směru (-). Při sepnutých tranzistorech 5 a 8 protéká cívkou B proud v kladném směru (+). Při sepnutých tranzistorech 6 a 7 protéká cívkou B proud v záporném směru (-). Dále je třeba dávat pozor, aby nedošlo k současnému sepnutí tranzistorů zapojených pod sebou. Nikdy tedy nesmí být sepnuté tranzistory 1 a 2, 3 a 4, 5 a 6, 7 a 8. Pokud by došlo k jejich současnému sepnutí, znamenalo by to zkrat na zdroji. Samotný modul s tranzistory má sice proti tomuto stavu integrovanou ochranu, avšak její sepnutí znamená zablokování tranzistorů. Proto je třeba zajistit, aby tyto nechtěné stavy nenastaly již na úrovni řídícího programu ve vývojové desce Arduino. Vzhledem k indukčnímu charakteru zátěže jsou všechny tranzistory rovněž vybaveny zpětnou diodou.

5.2 Zapojení krokového motoru

Tento motor má univerzálně vyvedené konce cívek. Je možno jej zapojit několika způsoby. Pro účely této bakalářské práce byly vinutí fází zapojeny bipolárně a jednotlivé cívky byly spojeny sériově. Pro řízení proudu byl před vinutí motoru zapojen reostat, kterým byl nastaven jmenovitý

budicí proud motoru. Toto řešení není sice ideální vzhledem k větším ztrátám takového pohonu, avšak pro základní řízení krokového motoru je dostačující. Konkrétní zapojení vodičů tohoto motoru: Středy vinutí jsou spojeny. Žlutý s modrým a oranžový s hnědým. Vývody první fáze jsou A – červený, /A – černý. Vývody druhé fáze jsou B – bílý a /B – zelený. [6]



Obrázek 22 - Zapojení fází motoru

6 Program pro Arduino

Program pro mikrokontroléry ATMega2560 je psán ve vývojovém softwaru Arduino software. Tento software je open source. Vývojové prostředí obsahuje knihovny s mapováním pinů pro jednotlivé vývojové desky Arduino. To umožňuje využívat v kódu příkazy, které jsou univerzální napříč všemi platformami Arduino. Není potřeba uživatelem definovat nastavení vnitřních registrů procesoru, vše provádí vývojové prostředí automaticky, ovšem je umožněno do tohoto nastavení v případě potřeby zasahovat. V následujících částech této práce jsou popsány jednotlivé části kódu a jejich funkce. Celý program je také v příloze č. 2.

6.1 Nastavení výstupů

Nejprve pomocí příkazu #define přiřadím výrazům T1 až T8 hodnoty 6 až 13. Výrazy T1 až T8 odpovídají tranzistorům 1 až 8 z obrázku 21. Hodnoty 6 až 13 jsou čísla použitých pinů na desce Arduino (Obrázek 20). Tyto piny byly použity, protože je lze také použít jako výstupy PWM. Tato definice byla provedena, aby při dalším psaní programu bylo hned jasné, který tranzistor se kterým výstupem řídí.

```
#define T1 6
#define T2 7
#define T3 8
#define T4 9
#define T5 10
#define T6 11
#define T7 12
#define T8 13
```

Dále je třeba tyto piny nadefinovat jako výstupní. To se provede ve funkci void setup(). Tato funkce je zavolána pouze jednou po startu procesoru. Typ portu se definuje pomocí funkce pinMode(). Tato funkce má dva parametry, pin (číslo pinu který se nastavuje) a mode. Digitální piny vývojových desek Arduino mohou pracovat ve třech módech. OUTPUT, INPUT a INPUT_PULLUP. OUTPUT nastaví příslušný pin jako výstupní, INPUT jako vstupní a INPUT_PULLUP jako vstup s pullup odporem. Dále se ve funkci setup() nastaví hodnoty výstupů po zapnutí. Hodnota se na výstup zapisuje pomocí funkce digitalWrite(). Tato funkce má dva argumenty, pin a value. Hodnota pin říká, který digitální pin se bude nastavovat. Do value se zapisuje buď HIGH nebo LOW, chceme-li na výstupu logickou 1 nebo 0. Tímto jsou všechny potřebné piny nastaveny jako výstupní a je na nich po restartu procesoru hodnota 0.

```
pinMode(T1, OUTPUT);      digitalWrite(T1, LOW);
pinMode(T2, OUTPUT);      digitalWrite(T2, LOW);
pinMode(T3, OUTPUT);      digitalWrite(T3, LOW);
pinMode(T4, OUTPUT);      digitalWrite(T4, LOW);
pinMode(T5, OUTPUT);      digitalWrite(T5, LOW);
pinMode(T6, OUTPUT);      digitalWrite(T6, LOW);
pinMode(T7, OUTPUT);      digitalWrite(T7, LOW);
pinMode(T8, OUTPUT);      digitalWrite(T8, LOW);
```

6.2 Definice proměnných

V další části programu je potřeba nadefinovat globální proměnné. První proměnná je typu boolean s názvem Run. Tato proměnná je určena k zapínání trvalého chodu motoru. Když je proměnná TRUE, motor se volně otáčí. Když je FALSE, motor stojí. Druhá proměnná je také typu boolean s názvem Dir (Direction). Hodnoty TRUE a FALSE nastavují otáčení doprava nebo doleva. Další proměnné jsou bajty využité při sériové komunikaci. Tyto proměnné jsou použity ke čtení příchozích bajtů při sériové komunikaci. Dále jsou použity dvě proměnné SekIndex a MaxIndex. Obě jsou typu integer. SekIndex slouží k uchování informace, v jaké části cyklu přepínání cívek motoru se program nachází. Proměnná MaxIndex nastavuje maximální hodnotu proměnné SekIndex, při které dojde k návratu na první hodnotu. MaxIndex může nabývat zatím dvou hodnot, 4 a 8, při čtyřtaktním nebo osmitaktím řízení. Následuje série proměnných typu long, které jsou využívány při nastavování úhlu rotoru. Čtyřbitový datový typ byl použit pro svůj velký rozsah hodnot od -2147483648 do 2147483647, které by měly na nastavování úhlu v rozumném rozsahu stačit. Jsou to PozadovanyUhel, AktualniUhel a OtocitOUhel. K těmto třem patří ještě proměnná ZmenaUhlu, která je použita pro přičítání změny úhlu po provedení kroku. Další proměnná typu long je ZadanaRychlost. Pomocí této proměnné probíhá zadávání rychlosti rotace motoru. Poslední proměnná s názvem Mode je typu byte a slouží k nastavování módu řízení. Pokud je její hodnota 1, řízení probíhá čtyřtaktně, tedy s plným krokem, a jednou aktivní cívkou. Pokud je rovna 2, řízení probíhá čtyřtaktně s plným krokem a dvěma aktivními cívkami. Pokud je hodnota 3, je motor řízen osmitaktně, tedy s polovičním krokem.

```
boolean Run = false;
boolean Dir = false;

byte iB1 = 0;
byte iB2 = 0;
byte iB3 = 0;
byte iB4 = 0;
byte iB5 = 0;

long PozadovanyUhel = 0L;
long AktualniUhel = 0L;
long OtocitOUhel = 0L;
long PocetKroku = 0L;
int  ZmenaUhlu = 18;

unsigned int SekIndex = 1;
unsigned int MaxIndex = 4;
byte Mode = 1;

long ZadanaRychlost = 60L;
```

6.3 Funkce spínající proudy cívek

V další části kódu jsou nadefinovány funkce, které zajišťují spínání jednotlivých tranzistorů výkonového zesilovače tak, aby proud protékal cívkami požadovaným směrem. Je zde použito šest funkcí. Pro každou fázi lze nastavit proud v kladném i záporném směru a lze danou fázi i vypnout. Funkce jsou pojmenovány Aplus(), Aminus() a Anic(). Pro fázi B jsou Bplus(), Bminus() a Bnic(). V každé funkcí jsou nejprve vypnuty tranzistory, které nebudou použity, a až poté jsou sepnuty požadované tranzistory. Například pro nastavení proudu do kladného směru fáze A jsou nejprve výstupy na tranzistory 2 a 3 nastaveny do logické 0 a až poté jsou výstupy pro tranzistory 1 a 4 nastaveny do logické 1. Nikdy tak nedojde k sepnutí dvou tranzistorů pod sebou a případnému zkratu. Systém spínání tranzistorů je odvozen z (Obrázek 14 - Označení proudů v H-můstcích). Ukázka funkcí pro nastavování proudů fází A. Sekvence pro nastavování proudů ve fázi B jsou analogické.

```
void Aplus()  {
      digitalWrite(T2, LOW);
      digitalWrite(T3, LOW);
      digitalWrite(T1, HIGH);
      digitalWrite(T4, HIGH);  }
void Aminus() {
      digitalWrite(T1, LOW);
      digitalWrite(T4, LOW);
      digitalWrite(T2, HIGH);
      digitalWrite(T3, HIGH);  }
void Anic()   {
      digitalWrite(T1, LOW);
      digitalWrite(T2, LOW);
      digitalWrite(T3, LOW);
      digitalWrite(T4, LOW);   }
```

6.4 Tabulky sekvencí kroků

Řídící program umí řídit krokový motor třemi způsoby. Čtyřtaktně s jednou aktivní fází, čtyřtaktně se dvěma aktivními fázemi a osmitaktně. Pro každý způsob je v programu uložena jedna tabulka nastavování proudů fázemi. Program danou tabulkou prochází nahoru nebo dolů, podle nastaveného směru, zvyšováním nebo snižováním hodnoty SekIndex. Podle toho, na jakém řádku tabulky se nachází, volá příslušné funkce pro nastavování proudů jednotlivými cívkami. Při dosažení konce tabulky je nutno přeskočit na opačnou stranu jednorázovou změnou hodnoty SekIndex. Při změně způsobu řízení ze čtyřtaktního na osmitaktní je potřeba změnit maximální hodnotu proměnné SekIndex. Maximální hodnota se kontroluje pomocí proměnné MaxIndex, která nabývá hodnoty 4 pro čtyřtaktní řízení a hodnoty 8 pro osmitaktní řízení. Následuje tabulka pro řízení motoru čtyřtaktně (plným krokem), s jednou aktivní fází.

```
void FullStep1C (){
  switch (SekIndex) {
    case 1:    Aplus();    Bnic();      break;
    case 2:    Anic();     Bplus();     break;
    case 3:    Aminus();   Bnic();      break;
    case 4:    Anic();     Bminus();    break;
    default:   Anic();     Bnic();      break;
    }
  }
```

Tabulky využívají funkci switch k rozhodování, na jakém řádku se zrovna nacházejí. Pro případ jiné hodnoty SekIndex, než tabulka obsahuje, se provede řádek default. Je to ochrana proti chybám programu a neměla by nastat. V tomto případě se při nedefinovaných podmínkách cívky motoru z bezpečnostních důvodů odpojí. Další tabulka ukazuje čtyřtaktní řízení se dvěma aktivními fázemi.

```
void FullStep2C (){
  switch (SekIndex) {
    case 1:    Aplus();    Bminus();    break;
    case 2:    Aplus();    Bplus();     break;
    case 3:    Aminus();   Bplus();     break;
    case 4:    Aminus();   Bminus();    break;
    default:   Anic();     Bnic();      break;
    }
  }
```

Poslední tabulka ukazuje osmitaktní způsob řízení. Jedná se o kombinaci dvou předchozích způsobů a je jím dosaženo dvojnásobného rozlišení motoru. Pro použití této tabulky je ovšem třeba zvětšit maximální hodnotu proměnné SekIndex na 8.

```
void HalfStep (){
  switch (SekIndex) {
    case 8:    Aplus();    Bplus();     break;
    case 7:    Aplus();    Bnic();      break;
    case 6:    Aplus();    Bminus();    break;
    case 5:    Anic();     Bminus();    break;
    case 4:    Aminus();   Bminus();    break;
    case 3:    Aminus();   Bnic();      break;
    case 2:    Aminus();   Bplus();     break;
    case 1:    Anic();     Bplus();     break;
    default:   Anic();     Bnic();      break;
    }
  }
```

6.5 Funkce na provedení kroku

Funkce na provedení kroku nejprve otestuje, zda je proměnná Dir nastavena na točení motorem doleva nebo doprava. Podle nastaveného směru buď zvýší, nebo sníží hodnotu proměnné SekIndex o 1. Jestliže by mělo dojít k překročení maximální hodnoty indexu, přeskočí index na začátek tabulky. V případě, že by byl index menší než 1, se nastaví na hodnotu maximálního indexu, udanou v proměnné MaxIndex. Nakonec je potřeba zjistit, v jakém módu řízení motoru pracuje. Podle toho se program odkáže na příslušnou tabulku a provede nastavení proudů cívkami podle příslušného řádku tabulky. Ukázka funkce pro provedení kroku:

```
void Krok() {
  if(Dir == false){
    ++SekIndex;
    if(SekIndex > MaxIndex) SekIndex = 1;
    }
  if(Dir == true){
    --SekIndex;
    if(SekIndex < 1) SekIndex = MaxIndex;
    }
  switch (Mode) {
    case 1:    FullStep1C();    break;
    case 2:    FullStep2C();    break;
    case 3:    HalfStep();      break;
    default:   break;
    }
  }
```

6.6 Sériová komunikace

Pro sériovou komunikaci využívá platforma Arduino několik základních funkcí, jako jsou funkce na přijetí bajtu, odeslání bajtu a nastavení parametrů sériové komunikace. Vytvořený komunikační protokol mezi ovládacím programem na počítači a Arduinem posílá najednou vždy 5 bajtů. První bajt nese základní informaci o tom, co má Arduino vykonat. Například udělat krok, změnit směr, nastavit požadovaný úhel, atd. Zbylé 4 bajty mohou u některých instrukcí nést doplňující informace. Například požadovaný úhel natočení, který je zapsán pomocí proměnné long využívající 4 bajty. Program Arduina tedy přijme 5 bajtů a poté testuje první z nich pomocí funkce switch. Různým hodnotám jsou přiřazeny různé příkazy. U příkazů vyžadujících doplňující informaci se ještě z jednotlivých bajtů poskládá hodnota do jedné požadované proměnné. Některé příkazy zároveň odesílají zpět do počítače zprávy o stavu programu Arduina a o krokovém motoru. Například aktuální úhel natočení nebo pro kontrolu jaký bajt obdržel. Následující ukázka kódu ukazuje část programu pro obsluhu sériové komunikace.

```
if (Serial.available() >= 5) {

    iB1 = Serial.read();
    iB2 = Serial.read();
    iB3 = Serial.read();
    iB4 = Serial.read();
    iB5 = Serial.read();

    switch (iB1) {
      case 0:   Run = false;
                TCCR1B = 0b00001000;
                Serial.println(" ZASTAVENO");
                break;
      case 20:  PozadovanyUhel = 0L;
                PozadovanyUhel += (long)iB2 << 24;
                PozadovanyUhel += (long)iB3 << 16;
                PozadovanyUhel += (long)iB4 << 8;
                PozadovanyUhel += (long)iB5;
                SrovnejUhel();
                break;
```

6.6.1 Seznam příkazů použitých pro sériovou komunikaci

Následující seznam popisuje odezvy Arduina na jednotlivé příchozí bajty podle jejich hodnoty.

0 - Zastaví motor.

1 - Rozjede motor, v tomto režimu se nepočítá úhel.

2 -Změní smysl otáčení doprava. Po směru hodinových ručiček.

3 - Změní smysl otáčení doleva. Proti směru hodinových ručiček.

4 - Sníží zadanou rychlost o 1 ot./min.

5 - Zvýší zadanou rychlost o 1 ot./min.

6 - Změní typ řízení na čtyřtaktní s jednou aktivní cívkou.

7 - Změní typ řízení na čtyřtaktní se dvěma aktivními cívkami.

8 - Změní typ řízení na osmitaktní.

9 - Provede jeden krok v nastaveném směru a započítává změnu úhlu.

10 - Uvolní motor. Vinutími neprotéká proud.

11 - Provede test kabelu. Postupně sepne všechny výstupy.

20 - Změní požadovaný úhel a srovná hřídel motoru do požadovaného úhlu.

21 - Vynuluje aktuální a požadovaný úhel.

22 - Otočí hřídel motoru o požadovaný úhel.

23 - Vypíše aktuální a požadovaný úhel.

30 - Změní zadanou rychlost s přesností na 2 desetinná místa.

128 - Slouží pro kontrolu spojení. Vypíše: ,,připojeno“.

6.7 Nastavení čítače

Řízení otáček motoru probíhá z obsluhy přerušení vyvolaného komparátorem čítače. Čítač dokud nedosáhne hodnoty shodné s komparačním registrem a vyvolá přerušení. Je tedy Nutné čítač nastavit, aby vyvolával přerušení v požadovaných intervalech. Je nutné nastavit předděličku na 256 a povolit přerušení. Následující kód ukazuje nastavení pro čítač TIMER1. Oba řídící registry jsou nejprve nastaveny do 0, počáteční hodnota, od které čítač čítá je také nastavena do 0 a do komparačního registru se nahraje inicializační hodnota. Dále je čítač nastaven do komparátorového režimu a je povoleno přerušení vyvolané komparátorem. Zároveň jsou před nastavováním zakázána přerušení pomocí cli() a na konci opět povolena pomocí funkce sei(). Jelikož vývojové prostředí Arduino software nedisponuje knihovnami na nastavení čítačů do požadovaného módu, muselo být nastavení provedeno klasickou cestou pomocí zápisu hodnot do registrů. Při nastavování čítače byl použit datasheet použitého mikrokontroléru [7].

```
  cli();
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1  = 0;
  OCR1A = 1250;
  TCCR1B = 0b00001000;
  TIMSK1 |= (1 << OCIE1A);
  sei();
```

6.8 Obsluha přerušení

V obsluze přerušení se program nejprve rozhodne, jestli se má motor pouze volně točit, nebo otáčet o určitý úhel danou rychlostí. V případě volného točení je proměnná Run = true a čítač při každém přerušení udělá krok. V případě, že se má motor pouze otočit o zadaný úhel, musí přerušení odpočítávat počet kroků, které má udělat. V každém cyklu sníží hodnotu požadovaných kroků o 1. Když dosáhne 0, čítač se zastaví. Obsluhu přerušení ukazuje následující kód.

```
ISR(TIMER1_COMPA_vect){
  if(Run == true) {
    Krok();
  } else {
      Krok();
      --PocetKroku;
      if(PocetKroku == 0) TCCR1B = 0b00001000;
  } }
```

6.9 Výpočet komparační hodnoty

Pokud přijde požadavek na změnu rychlosti, musí být vypočtena nová hodnota pro komparátor čítače. Ta je závislá na nastavení předděličky čítače, počtu kroků motoru na jednu otáčku a na požadované rychlosti. Po úpravách dostaneme následující vzorec pro výpočet hodnoty komparátoru:

$$KomparacniHodnota = \frac{60}{ZadanaRychlost} * \frac{1}{PocetKroku * DobaTiku}$$

Pro nastavení předděličky čítače 256 je doba jednoho tiku čítače 16µs a daný motor má 200 kroků na jednu otáčku. V programu je zadaná rychlost 100x větší aby mohla být nastavována s přesností na 2 desetinná místa. Dále je nutné vzít v úvahu změnu počtu kroků na dvojnásobek při osmitaktním řízení. Vypočtená hodnota je poté nahrána do komparačního registru OCR1A. Následuje ukázka kódu.

```
void VypoctiRychlost() {
  int PocKrok = 200;
  if(Mode == 3) PocKrok = 400;
  float DobaTiku = 0.000016;
  int PocetTiku =
    round(((float)6000/ZadanaRychlost)*(1/(PocKrok*DobaTiku)));
  TCNT1 = 0;
  OCR1A = PocetTiku;
}
```

6.10 Srovnání úhlů

Po příchodu požadavku na změnu úhlu natočení rotoru zavolá program funkci na srovnání úhlu. Jelikož si program udržuje v paměti aktuální hodnotu natočení rotoru lze vypočítat rozdíl mezi aktuálním úhlem a požadovaným úhlem, o který se má rotor otočit. Pro kladný nebo záporný rozdíl úhlů je nastavena proměnná Dir na otáčení doprava nebo doleva. Podle nastaveného způsobu řízení se vypočte počet kroků, které je třeba vykonat a aktivuje se čítač, který otáčí rotorem nastavenou rychlostí, dokud ho neotočí o požadovaný počet kroků. V případě, že je požadovaná změna úhlu menší, než velikost kroku při daném způsobu řízení, se úhel nemění. Ke změně úhlu dojde až poté, co rozdíl úhlů vzroste nad velikost kroku. Proměnná PocetKrokuSign je použita pro správné započtení změny úhlu. Do změny aktuálního úhlu je zapotřebí přičítat počet provedených kroků i se záporným znaménkem, kdežto pro čítač stačí počet kroků v absolutní hodnotě. Následuje ukázka funkce na srovnávání úhlů.

```
void SrovnejUhel() {
  long PocetKrokuSign = 0L;
  OtocitOUhel = PozadovanyUhel - AktualniUhel;
  if(OtocitOUhel < 0) Dir = false;
  if(OtocitOUhel > 0) Dir = true;
  PocetKrokuSign = OtocitOUhel / ZmenaUhlu;
  PocetKroku = abs(PocetKrokuSign);
  if(PocetKroku == 0){
  }
  else {
    TCCR1B = 0b00001100;            //rozjede citac
    AktualniUhel = AktualniUhel + (PocetKrokuSign * ZmenaUhlu);
  }
}
```

6.11 Ostatní funkce

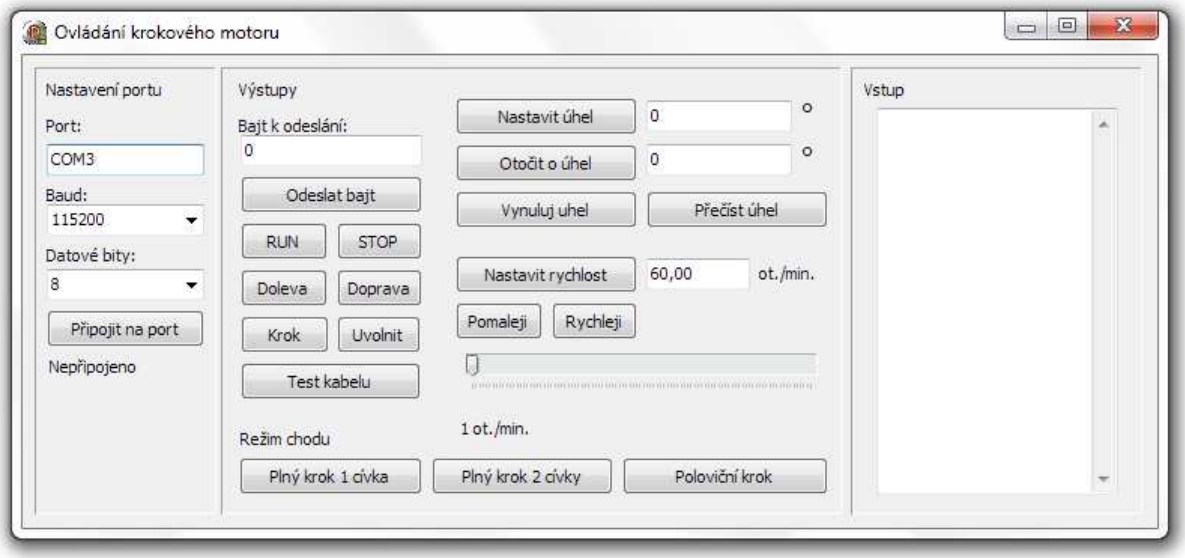
Program dále obsahuje například funkci na ověření spojení desky Arduino s výkonovým zesilovačem. Tato funkce postupně sepne všechny tranzistory, a pokud je spojení v pořádku, na výkonovém zesilovači se postupně rozsvítí indikační LED diody jednotlivých tranzistorů. Další pomocnou funkcí je funkce na uvolnění motoru. Tato funkce zastaví čítač a vypne proud oběma fázemi. Celý program je také připravený na přidávání dalších funkcí a rozšiřování.

7 Obslužný program pro PC

Ovládací program pro PC byl vytvořen pomocí jazyka object pascal. Program má jednoduchou grafickou podobu umožňující přímé ovládání pomocí tlačítek a zadávání hodnot do textových polí. Program zajišťuje obsluhu komunikace po sériové lince pomocí volně dostupné knihovny synaser. Program odesílá do Arduina bajty jednotlivých příkazů a zároveň přijímá a vypisuje odpovědi. Čtení příchozí komunikace probíhá pomocí samostatného vlákna, aby program pracoval plynule.

7.1 Popis ovládacího prostředí

Okno programu je rozděleno na 3 základní části. Nastavení sériové komunikace, ovládací tlačítka a výstup komunikace. Po zapnutí programu je nejprve nutné se spojit s Arduinem. Je zapotřebí nejprve zvolit správný COM port, nastavit rychlost komunikace (typicky 115200 baud) a stisknout tlačítko připojit. Pokud vše proběhne v pořádku, pod tlačítkem se objeví nápis ,,Připojeno". Dále je možné odesílat příkazy do Arduina a řídit tak krokový motor. Tlačítka ,,RUN" a ,,STOP" roztočí, respektive zastaví motor v režimu volného otáčení. Tlačítky ,,doleva" a ,,doprava" se mění smysl otáčení motoru. Tlačítko ,,krok" provede motorem jeden krok v nastaveném směru, přičemž počítá úhel. Tlačítkem ,,uvolnit" se nastaví budicí proud vinutí na nulu. Tlačítko ,,test kabelu" slouží k ověření propojení mezi Arduinem a výkonovými prvky. Další série tlačítek pracuje s úhly. Tlačítko ,,nastavit úhel" nastaví absolutní hodnotu úhlu, na který se má rotor otočit. Zadávat úhel lze s přesností na 1 desetinné místo. Tlačítko ,,otočit o úhel" otočí rotor o zadaný úhel, přičemž kladné hodnoty otáčí po směru hodinových ručiček a záporné hodnoty proti směru. Tlačítko ,,vynuluj úhel" nastaví hodnotu aktuálního úhlu v Arduinu na 0. Tlačítko ,,přečíst úhel" vypíše aktuální úhly natočení a požadovaný úhel natočení. Další tlačítka a jezdec zajišťují nastavování rychlosti otáčení motoru. Tlačítko ,,nastavit rychlost" nastaví rychlost otáčení motoru s přesností na 2 desetinná místa. Tlačítka ,,pomaleji" a ,,rychleji" sníží nastavenou rychlost o 1 ot./min. respektive zvýší o 1 ot./min. Pohyblivý jezdec umožňuje nastavovat rychlost v rozsahu od 1 ot./min. do 120 ot./min. Nakonec jsou zde tři tlačítka na nastavení způsobu řízení krokového motoru. Lze nastavovat plný krok s jednou aktivní cívkou, plný krok se dvěma aktivními cívkami a poloviční krok. Nakonec je v pravé části ovládacího panelu výstup komunikace mezi počítačem a Arduinem.



Obrázek 23 - Ovládací panel na PC

8 Závěr

Cílem této bakalářské práce bylo seznámit se s konstrukcí a způsoby řízení krokových motorů. Získané poznatky měly být poté využity k vytvoření jednoduchého systému na řízení krokového motoru. Cílem bylo uvést krokový motor do pohybu a umožnit nastavování polohy hřídele, případně nastavit rychlost otáčení hřídele motoru.

Na realizaci systému byl jako motor použit dvoufázový hybridní krokový motor od firmy Microcon, jako výkonový spínací zesilovač byl využit modul s osmi IGBT tranzistory a jako elektronický komutátor byla využita vývojová deska Arduino Mega2560 s mikrokontrolérem Atmel. Ovládací panel umožňující zadávání požadovaných hodnot natočení rotoru a rychlosti otáčení byl realizován jako počítačový program. Tento program by propojen s vývojovou deskou Arduino pomocí USB a výstupy vývojové desky byly propojeny s tranzistory ve výkonovém modulu, na které byl již připojen krokový motor. Výkonový spínací zesilovač byl napájen ze stejnosměrného zdroje a jmenovitého proudu ve fázích motoru bylo dosaženo pomocí reostatu.

Hlavní náplní práce bylo programování mikrokontroléru na vývojové desce Arduino. Vytvořený program umožňuje nastavovat pozici rotoru krokového motoru a měnit rychlost otáčení rotoru. Navíc také umožňuje měnit způsoby řízení motoru ze čtyřtaktního s jednou nebo dvěma aktivními fázemi na osmitaktní. Program také obsahuje funkce jako krokování, reverzaci a uvolnění motoru.

Celý tento systém nabízí do budoucna možnosti rozšíření, jako jsou například řízení proudu fázemi, přidání možnosti mikrokrokování, zavedení zpětné vazby o poloze a přidání rozjezdových ramp. Program by také bylo možné upravit i pro řízení unipolárních motorů. Ovládací panel na počítači lze také dále upravovat a rozšiřovat podle potřeby. Cíle této bakalářské práce se tedy podařilo naplnit.

9 Seznam použité literatury

[1] RUIDER, Pavel, Jan POHORSKÝ a J. KYLAR. *Krokové motory*. 1985, 35 s.

[2] HRABOVCOVÁ, Valéria, Ladislav JANOUŠEK, Pavol RAFAJDUS a Miroslav LIČKO. *Moderné elektrické stroje*. Žilina: EDIS, 2001. ISBN 80-7100-809-5.

[3] *Domovská stránka projektu Arduino* [online]. © 2014 [cit. 2014-04-26]. Dostupné z: http://www.arduino.cc/

[4] Stránka vývojové desky Arduino Mega2560. *Arduino* [online]. ©2014 [cit. 2014-04-26]. Dostupné z: http://arduino.cc/en/Main/arduinoBoardMega2560

[5] Dokumentace motorů Microcon. *Microcon* [online]. 2014 [cit. 2014-04-27]. Dostupné z: http://www.microcon.cz/pdf2014/13-20.pdf

[6] Zapojení vinutí motorů SX34. *Microcon* [online]. 2012 [cit. 2014-04-29]. Dostupné z: http://www.microcon.cz/zapojenivinuti2012web/zapojenivinutipdf2012/SX34NP.pdf

[7] Datasheet mikrokontroléru ATmega2560. *Atmel* [online]. 2014 [cit. 2014-05-05]. Dostupné z: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

Při psaní této práce jsem také čerpal ze své předchozí práce *Řízení polohy krokového motoru pomocí vývojové platformy Arduino a modulu s H-můstky.* Práce byla výstupem individuálního projektu A1B14IND.

10 Seznam obrázků

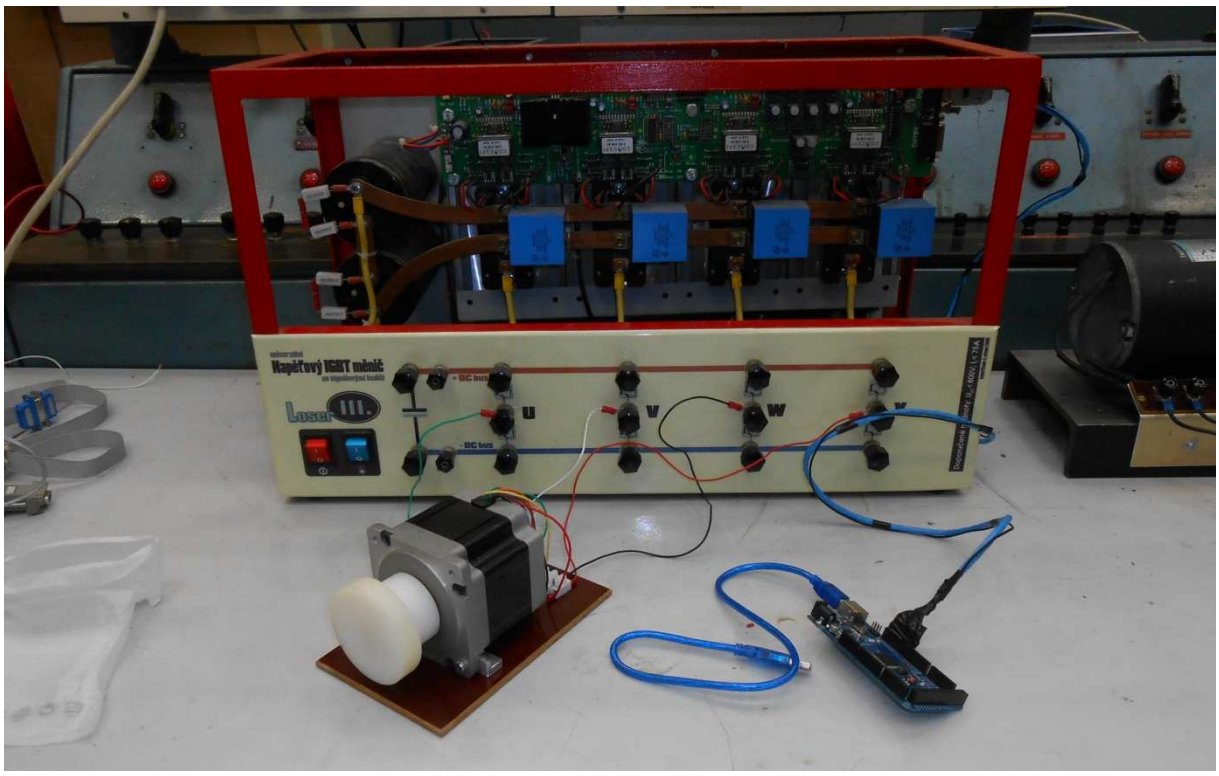
11 Seznam tabulek

12 Obsah přiloženého CD

Bakalářská práce.pdf	- Bakalářská práce ve formátu PDF
Ovládací panel.exe	- Program na ovládání krokového motoru
Složka Arduino	- Zdrojový kód pro vývojovou desku Arduino
Složka Panel zdrojový kód	- Zdrojové kódy ovládacího panelu

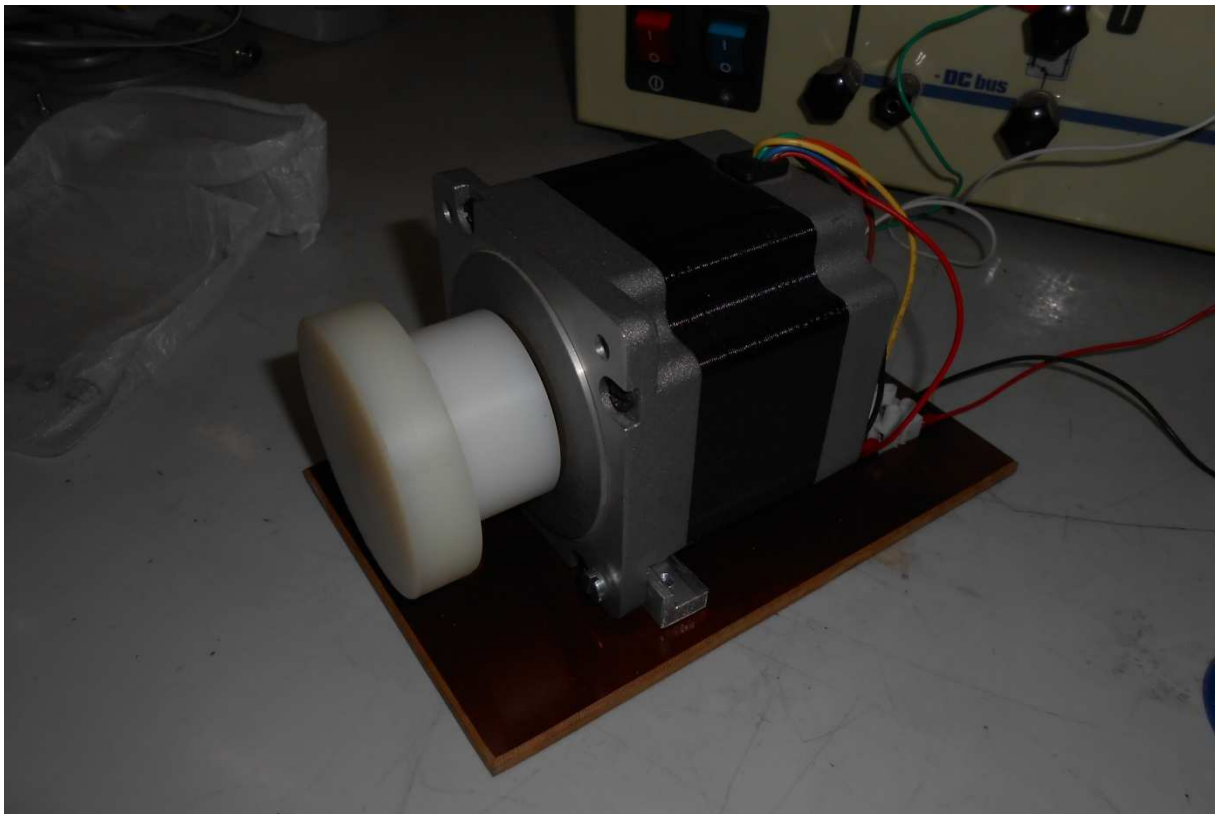
13 Přílohy

13.1 Obrazová příloha

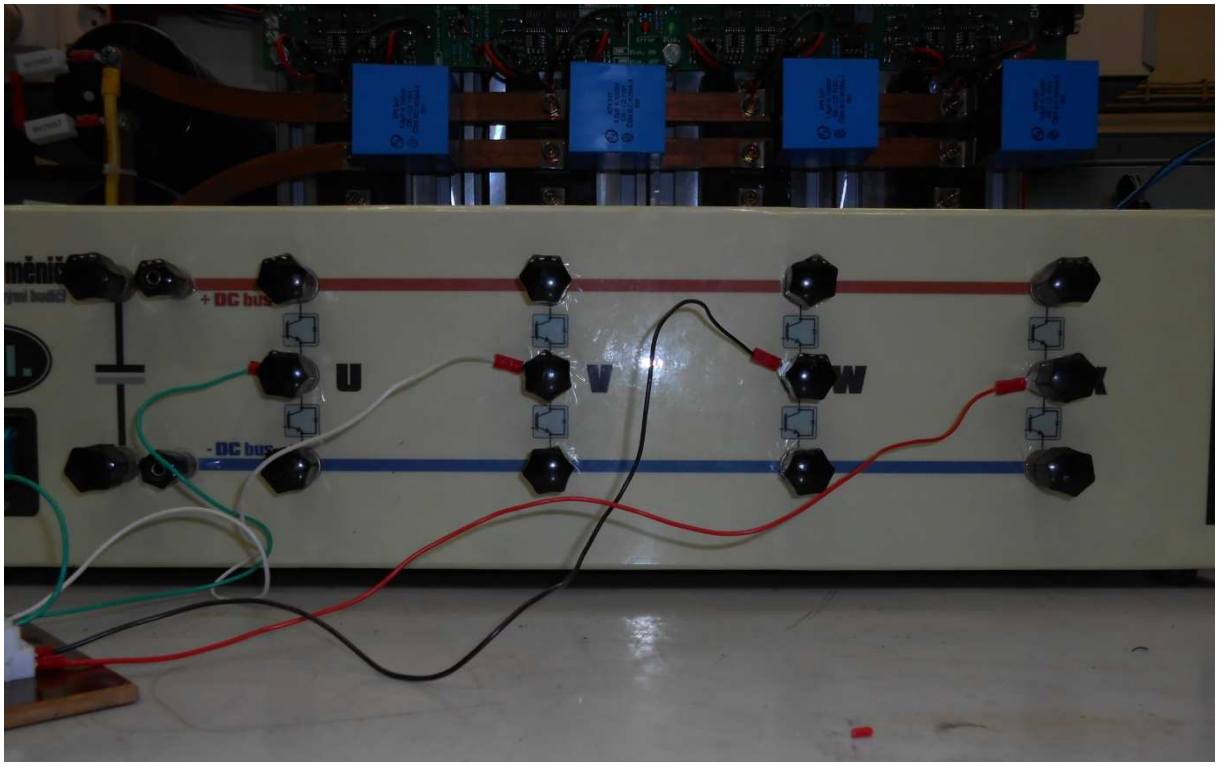


Obrázek 24 - Zapojení výkonových spínacích prvků, krokového motoru a vývojové desky Arduino

Obrázek 25 - Detail zapojení krokového motoru



Obrázek 26 - Použitý krokový motor



Obrázek 27 - Zapojení fází krokového motoru

13.2 Celý zdrojový kód pro vývojovou desku Arduino

```
  #define T1 6
  #define T2 7
  #define T3 8
  #define T4 9
  #define T5 10
  #define T6 11
  #define T7 12
  #define T8 13

  boolean Run = false;
  boolean Dir = false;
  byte iB1 = 0;
  byte iB2 = 0;
  byte iB3 = 0;
  byte iB4 = 0;
  byte iB5 = 0;
  long PozadovanyUhel = 0L;
  long AktualniUhel = 0L;
  long OtocitOUhel = 0L;
  long PocetKroku = 0L;
  int  ZmenaUhlu = 18;
  unsigned int SekIndex = 1;
  unsigned int MaxIndex = 4;
  unsigned int cekej = 50;
  byte Mode = 1
  long ZadanaRychlost = 60L;

void setup() {
  Serial.begin(115200);

  cli();
  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1  = 0;
  OCR1A = 1250;
  TCCR1B = 0b00001000;
  TIMSK1 |= (1 << OCIE1A);
  sei();

  pinMode(T1, OUTPUT);      digitalWrite(T1, LOW);
  pinMode(T2, OUTPUT);      digitalWrite(T2, LOW);
  pinMode(T3, OUTPUT);      digitalWrite(T3, LOW);
  pinMode(T4, OUTPUT);      digitalWrite(T4, LOW);
  pinMode(T5, OUTPUT);      digitalWrite(T5, LOW);
  pinMode(T6, OUTPUT);      digitalWrite(T6, LOW);
  pinMode(T7, OUTPUT);      digitalWrite(T7, LOW);
  pinMode(T8, OUTPUT);      digitalWrite(T8, LOW);
}

ISR(TIMER1_COMPA_vect){
  if(Run == true) {
    Krok();
  } else {
      Krok();
      --PocetKroku;
      if(PocetKroku == 0) TCCR1B = 0b00001000;
  }  }
void Aplus()  {  digitalWrite(T2, LOW);
                 digitalWrite(T3, LOW);
```

```
                digitalWrite(T1, HIGH);
                digitalWrite(T4, HIGH);  }
void Aminus() {  digitalWrite(T1, LOW);
                digitalWrite(T4, LOW);
                digitalWrite(T2, HIGH);
                digitalWrite(T3, HIGH);  }
void Anic()   {  digitalWrite(T1, LOW);
                digitalWrite(T2, LOW);
                digitalWrite(T3, LOW);
                digitalWrite(T4, LOW);   }

void Bplus()  {  digitalWrite(T6, LOW);
                digitalWrite(T7, LOW);
                digitalWrite(T5, HIGH);
                digitalWrite(T8, HIGH);  }
void Bminus() {  digitalWrite(T5, LOW);
                digitalWrite(T8, LOW);
                digitalWrite(T6, HIGH);
                digitalWrite(T7, HIGH);  }
void Bnic()   {  digitalWrite(T5, LOW);
                digitalWrite(T6, LOW);
                digitalWrite(T7, LOW);
                digitalWrite(T8, LOW);   }

void FullStep1C (){
  switch (SekIndex) {
    case 1:    Aplus();    Bnic();      break;
    case 2:    Anic();     Bplus();     break;
    case 3:    Aminus();   Bnic();      break;
    case 4:    Anic();     Bminus();    break;
    default:   Anic();     Bnic();      break;
    }}
void FullStep2C (){
  switch (SekIndex) {
    case 1:    Aplus();    Bminus();    break;
    case 2:    Aplus();    Bplus();     break;
    case 3:    Aminus();   Bplus();     break;
    case 4:    Aminus();   Bminus();    break;
    default:   Anic();     Bnic();      break;
    }}
void HalfStep (){
  switch (SekIndex) {
    case 8:    Aplus();    Bplus();     break;
    case 7:    Aplus();    Bnic();      break;
    case 6:    Aplus();    Bminus();    break;
    case 5:    Anic();     Bminus();    break;
    case 4:    Aminus();   Bminus();    break;
    case 3:    Aminus();   Bnic();      break;
    case 2:    Aminus();   Bplus();     break;
    case 1:    Anic();     Bplus();     break;
    default:   Anic();     Bnic();      break;
    }}

void UvolnitMotor() {
  Run = false;
  TCCR1B = 0b00001000;
  Anic();    Bnic();
  }

void VsechnyVystupyO() {
  digitalWrite(T1, LOW);
```

```
  digitalWrite(T2, LOW);
  digitalWrite(T3, LOW);
  digitalWrite(T4, LOW);
  digitalWrite(T5, LOW);
  digitalWrite(T6, LOW);
  digitalWrite(T7, LOW);
  digitalWrite(T8, LOW);
}

void TestKabelu() {
  unsigned int cekat = 500;
  Run = false;
  VsechnyVstupy0();
  digitalWrite(T1, HIGH);
  delay(cekat);
  digitalWrite(T1, LOW);
  digitalWrite(T2, HIGH);
  delay(cekat);
  digitalWrite(T2, LOW);
  digitalWrite(T3, HIGH);
  delay(cekat);
  digitalWrite(T3, LOW);
  digitalWrite(T4, HIGH);
  delay(cekat);
  digitalWrite(T4, LOW);
  digitalWrite(T5, HIGH);
  delay(cekat);
  digitalWrite(T6, LOW);
  digitalWrite(T7, HIGH);
  delay(cekat);
  digitalWrite(T7, LOW);
  digitalWrite(T8, HIGH);
  delay(cekat);
  digitalWrite(T8, LOW);
}

void Krok() {
  if(Dir == false){
    ++SekIndex;
    if(SekIndex > MaxIndex) SekIndex = 1;
    }
  if(Dir == true){
    --SekIndex;
    if(SekIndex < 1) SekIndex = MaxIndex;
    }
  switch (Mode) {
    case 1:    FullStep1C();    break;
    case 2:    FullStep2C();    break;
    case 3:    HalfStep();      break;
    default:   break;
    }
  }

void SrovnejUhel() {
  long PocetKrokuSign = 0L;
  OtocitOUhel = PozadovanyUhel - AktualniUhel;
    if(OtocitOUhel < 0) Dir = false;
    if(OtocitOUhel > 0) Dir = true;
  PocetKrokuSign = OtocitOUhel / ZmenaUhlu;
  PocetKroku = abs(PocetKrokuSign);
```

```
  if(PocetKroku == 0){   }
  else {
    TCCR1B = 0b00001100;
    AktualniUhel = AktualniUhel + (PocetKrokuSign * ZmenaUhlu);
  }}

void VypoctiRychlost() {
  int PocKrok = 200;
  if(Mode == 3) PocKrok = 400;
  float DobaTiku = 0.000016;
  int PocetTiku =
      round(((float)6000/ZadanaRychlost)*(1/(PocKrok*DobaTiku)));
  TCNT1 = 0;
  OCR1A = PocetTiku;
}

void loop() {
  long OtocitO = 0L;

  if (Serial.available() >= 5) {
    iB1 = Serial.read();
    iB2 = Serial.read();
    iB3 = Serial.read();
    iB4 = Serial.read();
    iB5 = Serial.read();

    switch (iB1) {
      case 0:   Run = false;
                TCCR1B = 0b00001000;
                Serial.println(" ZASTAVENO");
                break;
      case 1:   Run = true;
                TCCR1B = 0b00001100;
                Serial.println(" JEDU");
                break;
      case 2:   Dir = true;
                Serial.println(" Doprava (Dir = true)");
                break;
      case 3:   Dir = false;
                Serial.println(" Doleva (Dir = false)");
                break;
      case 4:   ZadanaRychlost = ZadanaRychlost - 100;
                if(ZadanaRychlost < 100) ZadanaRychlost = 100;
                VypoctiRychlost();
                Serial.print(ZadanaRychlost/100, DEC);
                Serial.println(" ot/min");
                break;
      case 5:   ZadanaRychlost = ZadanaRychlost + 100;
                VypoctiRychlost();
                Serial.print(ZadanaRychlost/100, DEC);
                Serial.println(" ot/min");
                break;
      case 6:   Mode = 1;
                MaxIndex = 4;
                ZmenaUhlu = 18;
                VypoctiRychlost();
                Serial.println("Mode Fullstep1C");
                break;
      case 7:   Mode = 2;
                MaxIndex = 4;
                ZmenaUhlu = 18;
```

```
                VypoctiRychlost();
                Serial.println("Mode Fullstep2C");
                break;
        case 8:   Mode = 3;
                MaxIndex = 8;
                ZmenaUhlu = 9;
                VypoctiRychlost();
                Serial.println("Mode HalfStep");
                break;
        case 9:   Krok();
                if(Dir == false){AktualniUhel -= ZmenaUhlu;}
                if(Dir == true) {AktualniUhel += ZmenaUhlu;}
                Serial.println("Proveden krok");
                break;
        case 10:  UvolnitMotor();
                Serial.println("Motor uvolnen");
                break;
        case 11:  TestKabelu();
                Serial.println("Probiha test kabelu");
                break;
        case 20:  PozadovanyUhel = 0L;
                PozadovanyUhel += (long)iB2 << 24;
                PozadovanyUhel += (long)iB3 << 16;
                PozadovanyUhel += (long)iB4 << 8;
                PozadovanyUhel += (long)iB5;
                SrovnejUhel();
                break;
        case 21:  AktualniUhel = 0L;
                PozadovanyUhel = 0;
                break;
        case 22:  OtocitO = 0L;
                OtocitO += (long)iB2 << 24;
                OtocitO += (long)iB3 << 16;
                OtocitO += (long)iB4 << 8;
                OtocitO += (long)iB5;
                PozadovanyUhel = PozadovanyUhel + OtocitO;
                SrovnejUhel();
                break;
        case 23:  Serial.print("A: ");
                Serial.print(AktualniUhel, DEC);
                Serial.print("  P: ");
                Serial.println(PozadovanyUhel, DEC);
                break;
        case 30:  ZadanaRychlost = 0L;
                ZadanaRychlost += (long)iB2 << 24;
                ZadanaRychlost += (long)iB3 << 16;
                ZadanaRychlost += (long)iB4 << 8;
                ZadanaRychlost += (long)iB5;
                VypoctiRychlost();
                break;
        case 128: Serial.println("Pripojeno");
                break;
        default:  break;
        }
  }
}
```