ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická
Katedra výkonové elektroniky a strojů

DIPLOMOVÁ PRÁCE

Řízení laboratorního prototypu střídače

Autor práce: **Bc. Ondřej Hazuka**
Vedoucí práce: **Ing. Štěpán Janouš, PhD.**

2023/24

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Ondřej HAZUKA**
Osobní číslo:	**E22N0056P**
Studijní program:	**N0714A060013 Elektronika a informační technologie**
Specializace:	**Výkonová elektronika**
Téma práce:	**Řízení laboratorního prototypu střídače**
Zadávající katedra:	**Katedra výkonové elektroniky a strojů**

Zásady pro vypracování

1. Proveďte simulaci využívající 3-fázový střídač pracující do RL zátěže využívající PWM.
2. Implementujte řízení do DSP, pro implementaci kódu bude využito programovacího jazyka C.
3. Otestujte implementované řízení na laboratorním prototypu 3-fázového střídače pracujícího do RL zátěže.
4. Navrhněte a implementujte Grafické uživatelské rozhraní (GUI).
5. Navrhněte a laboratorně otestujte řízení proudu na výše zmíněném laboratorním prototypu.
6. Popište navržené řízení a shrňte dosažené výsledky.

Rozsah diplomové práce: **40-60**
Rozsah grafických prací: **2**
Forma zpracování diplomové práce: **elektronická**

Seznam doporučené literatury:

1. ZEMAN, K. PEROUTKA, Z. JANDA, M. *Automatická regulace pohonů s asynchronními motory*. Plzeň Západočeská univerzita, 2004, 200 s. ISBN: 80-7043-350-7.
2. VONDRÁŠEK, F. *Výkonová elektronika*. Plzeň : Západočeská univerzita , 2003, 267 s. ISBN: 80--7082-980-X.

Vedoucí diplomové práce: **Ing. Štěpán Janouš, Ph.D.**
Research and Innovation Centre for Electrical Engineering

Datum zadání diplomové práce: **6. října 2023**
Termín odevzdání diplomové práce: **24. května 2024**

Západočeská univerzita v Plzni
Fakulta elektrotechnická
L.S.

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

Prof. Ing. Václav Kůs, CSc.
vedoucí katedry

V Plzni dne 6. října 2023

Abstrakt

Cílem této práce je simulace vektorového řízení a vytvoření sestavy realizující toto řízení s rozšířením o skalární řízení. Zároveň také vytvořit uživatelské rozhraní pro ovládání pohonu, popsat funkčnost vytvořeného kódu a provést jeho testování. Účelem práce je vytvoření funkční sestavy pro účely výuky.

Práce popisuje komponenty moderního pohonu a jejich vlastnosti, algoritmy řízení a potřebné prostředky pro realizaci tohoto řízení společně s odvozováním a definováním potřebných rovnic, především pak matematického modelu asynchronního motoru.

Součástí praktické práce je, kromě popisu funkčnosti vlastního kódu, také popis postupu implementace jednotlivých funkcí a využití periferií použitého kontroléru včetně procesu kalibrace regulátorů a otestování funkčnosti včetně zkoumání přesnosti a rychlosti regulace. Je zde také sepsán přístup s jakým se sestavou pracovat a jak jí ovládat pro dosažení požadovaných stavů.

Klíčová slova

Pohon, měření, řízení, mikrokontroler, simulace, napěťový střídač, asynchronní motor, regulátor, skalární řízení, vektorové řízení, matematický model, PWM.

Abstract

The aim of this work is to simulate vector control and to create a report implementing this control with an extension to scalar control. At the same time, I also created a user interface to control the drive, described the functionality of the created code and performed its testing. The purpose of the work is to create a functional assembly for educational purposes.

The thesis describes the components of a modern drive and their characteristics, the control algorithms, and the necessary means to implement that control, together with the derivation and definition of the necessary equations, especially the mathematical model of an asynchronous motor.

In addition to the description of the functionality of the code itself, the practical work also includes a description of the process of implementation of the individual functions and the use of the peripherals of the used controller, including the process of calibration of the regulators and testing the functionality, including the testing of the accuracy and speed of control. The approach to operating the assembly and how to control it to achieve the desired states is also described.

Key Words

Drive, measurement, control, microcontroller, simulation, voltage inverter, asynchronous motor, controller, scalar control, vector control, mathematical model, PWM.

Obsah

Úvod ... 1
1 Pohon s 3-fázovým střídačem napětí ... 2
1.1 Výkonové spínací prvky ... 3
1.2 Napěťový střídač ... 7
1.3 Pulze-šířková modulace a její varianty ... 12
1.3.1 Subharmonická (referenční) PWM ... 12
1.3.2 Modulace prostorového vektoru (SVM) ... 13
1.4 Elektrický stroj ... 15
1.4.1 Náhradní schéma a matematický model asynchronního motoru ... 16
1.4.2 Matematický model AM vhodný pro vektorové řízení ... 17
1.5 Algoritmy řízení pohonů ... 21
1.5.1 Skalární řízení ... 21
1.5.2 Vektorové řízení ... 23
1.6 Mrtvé časy ... 26
2 Praktická část ... 27
2.1 Simulace vektorového řízení ... 27
2.2 Praktická realizace řízení ... 36
2.2.1 Popis kódu a funkčnosti programu ... 37
2.2.2 Výsledky testování praktické realizace ... 50
Zhodnocení a závěr ... 55
Zdroje ... 57
Přílohy ... 59

Seznam symbolů a zkratek

Značka	Popisek	Jednotka
DC	Dirrect current – stejnosměrný proud	-
AC	Alternating current – střídavý proud	-
U	Elektrické napětí	[V]
I	Elektrický proud	[A]
P	Výkon	[W]
f	Frekvence	[Hz]
ψ	Magnetický tok	[Wb]
t	Čas	[s]
CAN	Controlled Area Network	-
DMA	Dirrect Memory Access	-
AM	Asynchronní motor	-
PWM	Pulzně-šířková modulace	-
3f	3-fázový	-

Úvod

S aktuálním trendem čím dál častějšího nasazování elektrických pohonů vyvstává i potřeba co nejefektivnější a nejpřesnější regulace těchto pohonů ať z hlediska rychlosti otáčení či dodávaného momentu, potažmo výkonu. Příkladem může být rozvíjející se automatizace výroby pomocí průmyslových robotů nebo rozšiřování elektromobility. V současnosti jsou moderní elektricky regulované pohony nejčastěji realizovány s asynchronními motory nakrátko společně s 3-fázovými napěťovými střídači.

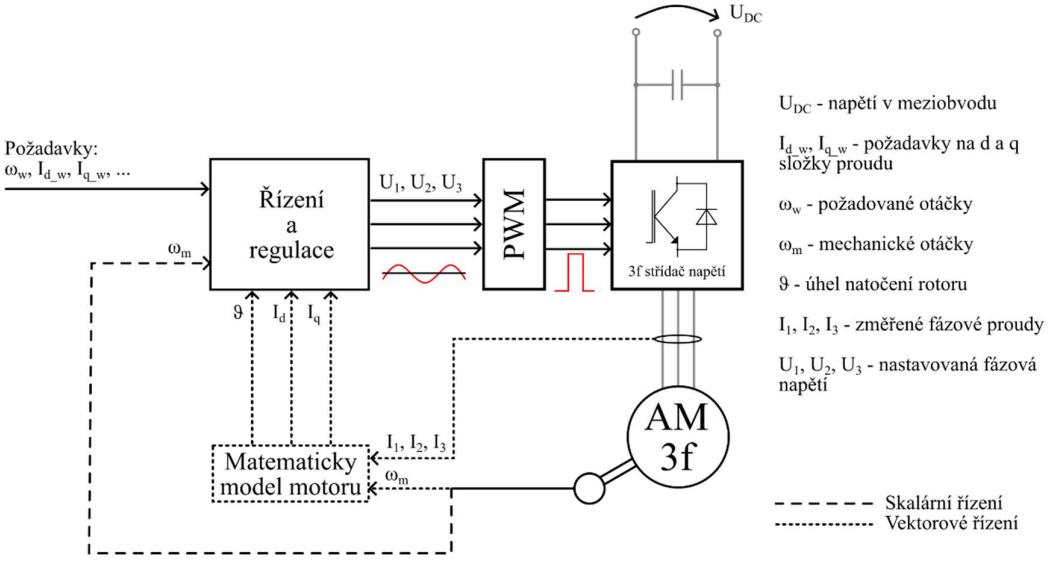
Tato skutečnost inspirovala tuto práci jejíž cílem je vytvoření sestavy pro řízení motoru pomocí 3-fázového napěťového střídače pro účely výuky realizující skalární a vektorové řízení. Program kontroléru řídící soustavu je psaný v jazyce C. Jakožto kontrolér je zde použit MLC Interface, v kombinaci s patřičnou knihovnou vytvořenou panem Ing. Tomášem Košanem, PhD., umožňující snadné využívání periférií tohoto kontroléru, především pak ePWM modulů a A/D převodníků důležitých pro tuto práci. Použitým střídačem je zde laboratorní prototyp ve standardní konfiguraci s 3f výstupem a čidly proudu, externě pak inkrementálním čidlem otáček společně s měřením napájecího napětí, které se zprostředkováno laboratorním zdrojem. Řízení této sestavy je zprostředkováno za pomocí CAN sběrnice umožňující komunikaci mezi kontrolérem a uživatelským rozhraním.

Z pohledu uživatele se jedná o jednoduchou aplikaci, kde má možnost výběru mezi skalárním a vektorovým řízením pomocí změny hodnoty proměnné vázané na daný typ řízení. Součástí práce je též popsat jak danou sestavu obsluhovat, respektive jak které proměnné pohon ovlivňují a jak dosáhnout požadovaných stavů.

Úkolem je také zmíněné vektorové řízení simulovat pomocí programu Simulink s nadstavbou Plecs a dosažené výsledky porovnat s fyzickým pohonem. Tyto výsledky jsou zhodnocené v závěru práce.

1 Pohon s 3-fázovým střídačem napětí

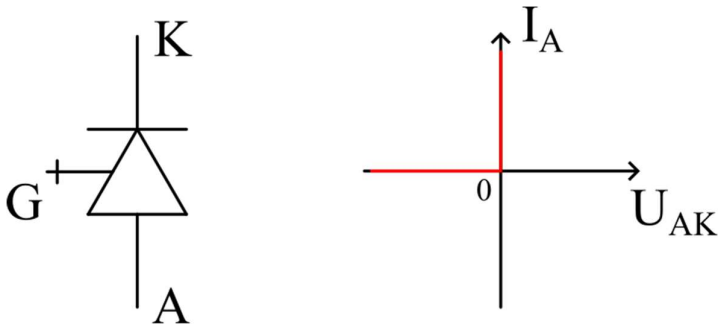
Nejčastěji používanou koncepcí je motor, uvažujme asynchronní, s 3-fázovým střídačem napětí. Úkolem střídače, jakožto výkonového polovodičového měniče, je vytvořit střídavé napětí ze stejnosměrného a tím přenášet výkon na střídavou stranu přičemž tento přenášený výkon reguluje. K tomu je zapotřebí příslušného řízení od kterého se odvíjí potřebná charakter chování pohonu. V případě použití skalárního řízení je postačující znalost napětí meziobvodu U_{DC} a parametry provozovaného motoru jako maximální a jmenovitý proud s napětím, počet pól párů, a jmenovité otáčky, případně frekvence, potažmo mechanické rychlosti motoru ω_m, ale ta však není nutná a je možné provozovat pohon i bez její znalosti. Dále je vhodné, nikoliv nutné, znát odpor statorového vinutí. Jako požadavek je zadávaná pouze hodnota požadovaných otáček ω_w. Při použití vektorového řízení navíc potřebujeme znát proudy na střídavé straně. V případě konfigurace s 3f střídačem napětí je jejich měření výhodné, jelikož se jejich průběh blíží sinusovému. Navíc měření proudů je důležité pro implementaci ochran před přehřátím, nadproudy a případnému zkratu. Je zde také zapotřebí matematického modelu, který ze změřených proudů a mechanických otáček estimuje úhel natočení rotorového toku ϑ pomocí něhož následně vypočítává velikost proudů I_d a I_q. Jako požadavek se zde mohou zadávat otáčky nebo d a q složka proudů, případně jejich různé kombinace v závislosti na požadavcích na pohon. Z bloku Řízení a regulace vystupují zadávaná fázová napětí, které se pulzně-šířkově modulují (PWM) z čehož vychází spínací pulzy pro měnič. [1]



Obr. 1: Blokové schéma elektrického pohonu s 3-fázovým střídačem napětí

1.1 Výkonové spínací prvky

Základy měničů tvoří spínatelné součástky, které dokážou přenášet výkony při jednotkách až desítkách kV a proudech až kA. I za zmíněných podmínek od nich stále vyžadujeme rychlé spínání a rozpínání, s čímž se pojí co nejnižší spínací ztráty, nízký odpor a snadné spínání. Obecně spínatelnou součástkou rozumíme prvek, který v propustném stavu umožňuje volný průchod proudu při ideálně nulovém úbytku napětí. Naopak v závěrném směru na sebe přebírá plné napětí bez jakéhokoliv protékajícího proudu. Přechod mezi propustným a závěrným stavem je ideálně nekonečně krátký, ovládaný řídící elektrodou, která neodebírá žádný výkon. Zároveň tento přechod může proběhnout kdykoliv a bez dalších nežádoucích efektů jako jsou překmity a další.

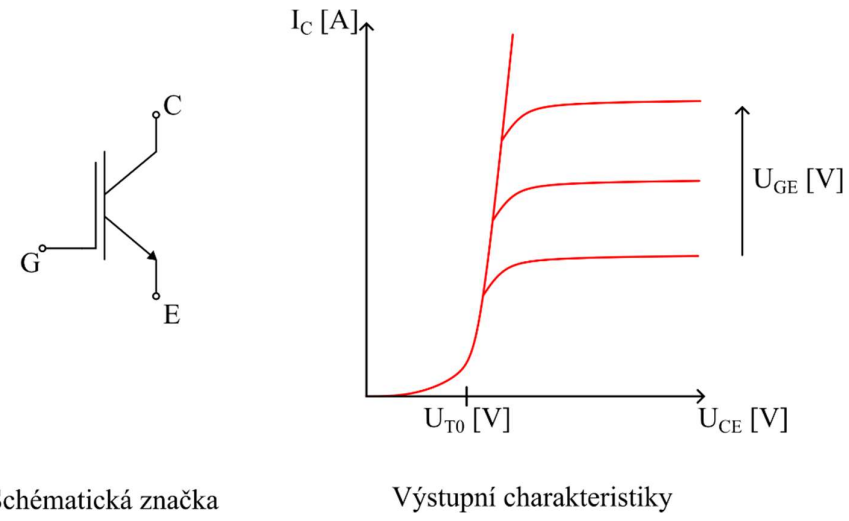


Obr. 2: Schématická značka a ideální VA-charakteristika obecně spínatelné součástky

Historicky používanými byly tyristory, avšak kvůli obtížnému vypínání jsou dnes použity pouze ve speciálních případech v aplikacích velmi vysokých výkonů, obvykle nad 1 MW. Současně jsou běžně používané tranzistory, především IGBT a MOSFET tranzistory.

IGBT tranzistor

IGBT tranzistor je bipolární tranzistor s izolovaným hradlem, jak vychází z anglického insulated-gate bipolar transistor, čímž se jedná v podstatě o kombinaci unipolárního a bipolárního tranzistoru spojující jejich výhody, především pak buzení tranzistoru. Hradlo IGBT tranzistoru má vlastnosti analogické FET tranzistoru. To odstraňuje nevýhodu výkonového bipolárního tranzistoru, který má malý proudový zesilovací činitel (zesiluje přibližně 10x až 20x), čímž komplikoval budiče tohoto tranzistoru především z hlediska potřebného dodávaného proudu do báze a tím i výkonu. Pro sepnutí FET tranzistoru, stejně tak IGBT, postačuje pouze napěťový signál o malém výkonu, jelikož řídící elektroda má vlastnosti velmi podobné kondenzátoru, vlivem izolace hradla – proud k řídící elektrodě slouží pouze k nabíjení a vybíjení kapacity řídící elektrody. Naopak výstupní strana je analogicky stejná s bipolárním tranzistorem, což zajišťuje dostatečnou napěťovou a proudovou zatížitelnost. [2]



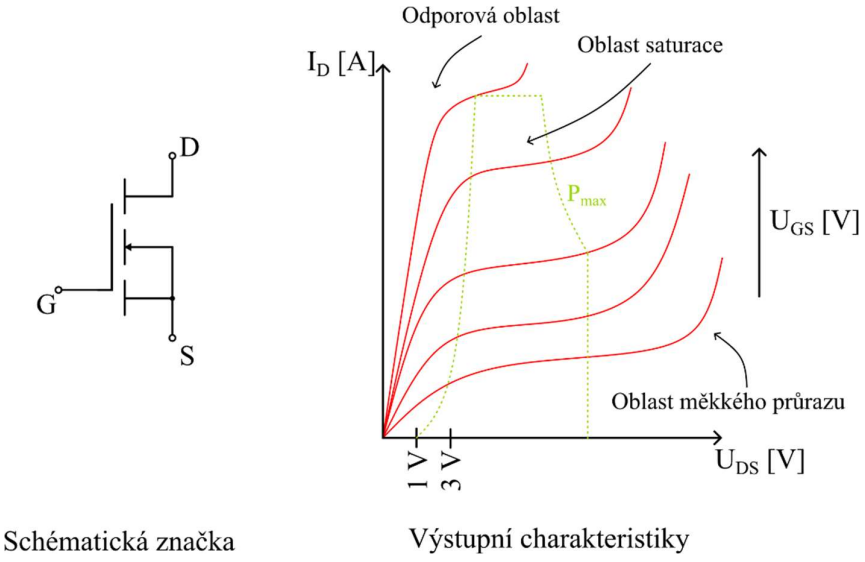
Obr. 3: Schématická značka a výstupní charakteristiky IGBT tranzistoru

Z výstupních charakteristik je vidět, že tranzistor má, obdobně jako dioda, prahové napětí U_{T0}, které se při zvyšování proudu již dále příliš nezvyšuje, a zároveň, stejně jako dioda, vede proud pouze v jednom směru. Z toho vyplývá, že vodivostní ztráty se počítají jako $P_{vod.} = U \cdot I$. Toto platí až do momentu desaturace. Desaturace nastává ve chvíli, kdy rychle narůstá ubytek napětí U_{CE}, přičemž proud I_C již roste minimálně. V tuto chvíli se již výrazně uplatňuje vnitřní odpor, který je vlastností bipolárního tranzistoru, stejně tak kladná zpětná vazba jejíž vlivem se odpor sníží až do kritické hodnoty proudu I_C, kdy se tranzistor začne chovat obdobně jako tyristor a již není možné ho vypnout zásahem do hradla, ale pouze zánikem zmíněného proudu. Kromě tohoto rizika také výrazně narůstají vodivostní ztráty tranzistorem. Avšak toto riziko může působit odpudivě, je možné, což je dnes běžnou výbavou, implementovat do budiče desaturační ochranu, která tomuto stavu zabraňuje. Chování diody v oblasti saturace se s výhodou využívá v aplikacích s velkými proudy a výkony kW až MW, jelikož, jak bylo již řečeno, napěťový úbytek na tranzistoru je stále téměř stejný. [2]

Spínací frekvence IGBT tranzistorů, respektive doba sepnutí, se pohybuje v rozmezí 0,1 až 1 µs. Doba potřebná pro rozepnutí je až 3x delší než pro sepnutí. S tím se pojí velké strmosti spínaných proudů $^{di}/_{dt}$, které se pohybují v rozmezí 1000 až 5000 A/µs. I přes to, že tato čísla zní působivě, jsou IGBT tranzistory vnímány spíše jako pomalejší součástky. Navíc se zvyšujícími se výkony je potřeba spínací frekvence snižovat z důvodu rychle narůstajících spínacích ztrát. Příkladem můžeme uvést aplikace s napájením 3x400 V se používají spínací frekvence 16 až 20 kHz, u měničů pro trakční aplikace typicky do 3 kHz. [2]

MOSFET tranzistor

Jak již bylo zmíněno u IGBT tranzistorů, MOSFET tranzistory, vycházející z anglického Metal Oxide Semiconductor Field Effect Transistor, se skládají z izolovaného hradla (Gate), kolektoru (Drain) a emitoru (Source). Kolektor s emitorem je spojován substrátem k němuž je připojeno ono hradlo izolované oxidu křemičitého (SiO_2). Napětím mezi hradlem a emitorem regulujeme otevření tranzistoru, který se při malých napětích U_{DS} (mezi kolektorem a emitorem), pod cca 1 V, chová jako řízený odpor a je v takzvané odporové (triodové) oblasti. Při zvýšení napětí U_{DS} nad 3 V přechází do oblasti saturace. V této části se již proud součástkou téměř nenavyšuje a zároveň roste U_{DS}. Při překročení maximální provozní oblasti tranzistoru, definované maximálním výkonem, vlivem nadměrného zatížení dochází k měkkému průrazu spojovacího kanálu a tranzistor zůstává otevřený. [3] [4]



Obr. 4: Schématická značka a výstupní charakteristiky MOSFET tranzistoru

Výhodou MOSFETu je velmi nízká teplotní závislost parametrů. Tudíž je tranzistor velmi konzistentní napříč celým svým rozsahem pracovních teplot. Stejně jako IGBT je jeho výhodou snadnost spínání, kdy stačí k jeho sepnutí nízký výkon, který pouze nabíjí a vybíjí kapacitu hradla, čímž snižuje náročnost na budič tranzistoru. Dále je díky své krátké době zapnutí a vypnutí vhodný pro aplikace s vysokou spínací frekvencí, kde se poměrně běžně používá spínací frekvence 20kHz i více. Jeho chování podobné rezistoru (hodnota odpovídá odporu kanálu v sepnutém stavu R_{DSon}) také umožňuje vést proud v obou směrech. To je zároveň i nevýhodou v případě aplikací vyžadující vysoké proudy, kde vodivostní ztráty odpovídají $P_{vod.} = R_{DS(on)} \cdot I_D^2$ a tak není vhodný pro aplikace, jelikož, jak rovnice popisuje,

ztráty vzrůstají s kvadrátem proudu. Jeho vlastnosti lze zlepšit paralelním řazením tranzistorů a tím snížit odpor. [3] [4]

Perspektivní technologie MOSFET tranzistorů

SiC MOSFET

Mezi perspektivní, a částečně již nasazované, technologie patří MOSFET tranzistory na bázi karbidu křemíku (SiC). Jejich přednostmi, oproti MOSFETům na bázi oxidu křemičitého, je přibližně desetkrát vyšší průrazné napětí při zachování stejné velikosti, nižší klidový proud (v uzavřeném stavu) a to i při vysokých teplotách. Zároveň mají lepší tepelnou vodivost, respektive teplo lépe odvádí, čímž umožňují zatížení vyšší proudovou hustotou, tedy vyšším proudem. K tomu navíc přispívá vyšší dovolená pracovní teplota, která dále napomáhá ke snížení nároků na chlazení. Neméně důležité jsou snížené spínací ztráty, což nadále umožňuje provoz při vyšší spínací frekvenci. Díky tomu je možné zmenšení následných filtračních členů, transformátorů a dalších. Zmíněná vyšší teplotní odolnost může být uváděna také jako nevýhoda, jelikož se pojí i se zkratovou odolností. Vzniklý zkrat je třeba vypnout v čase přibližně 1 až 2 µs, což je výrazně kratší čas než u IGBT tranzistorů a tato skutečnost zvyšuje nároky na rychlost detekce a vypnutí zkratu budičem. Obecně lze říci, že nabízí zlepšení všech vlastností MOSFET tranzistorů. Výraznou praktickou nevýhodou zde zůstává cena, která je, v porovnání s IGBT tranzistory stejné napěťové a proudové hladiny, stále vyšší. [5] [6]

Aktuálně jsou běžně dostupné tranzistory na napěťové hladiny 600 V až 3,3 kV při zachování stále nízkého odporu sepnutého kanálu $R_{DS(on)}$. V posledních letech se také objevují studie o vývoji a laboratorním testování prvků na hladiny 6,5 a 10 kV. Obecně se předpokládá, že s budoucím vývojem se rozsah použitelných hladin navýší a poklesnou ceny, což umožní značné rozšíření technologie. [5] [6]

GaN MOSFET

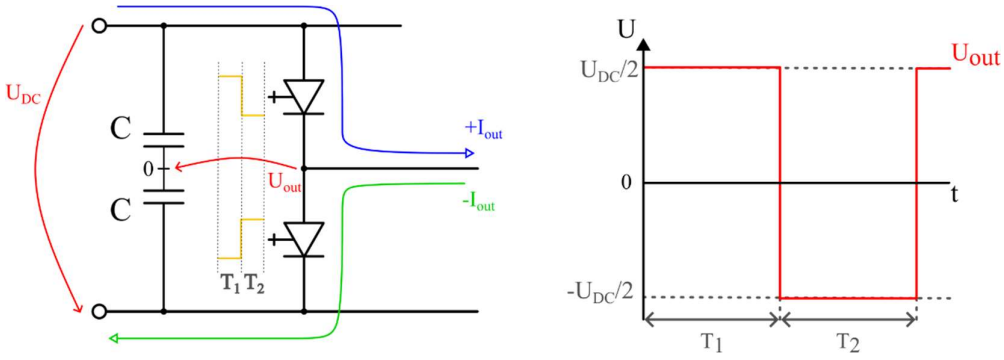
Druhou perspektivní technologií MOSFET tranzistorů jsou tranzistory na bázi Galium-nitridu. V porovnání s klasickými MOSFETy jsou přibližně 3x menší při zachování stejného výkonu. Zároveň mají výrazně nižší spínací ztráty s čímž se pojí i vyšší rychlost spínání, větší strmosti proudů, společně s menším potřebným nábojem pro sepnutí, a tím i použitelné spínací frekvence. Zároveň opět při stejné velikosti snáší vyšší napětí v uzavřeném stavu než dojde k průrazu. Také jejich teplotní odolnost je vyšší. To dovoluje vyšší pracovní teploty, ale zároveň kvůli zvýšené křehkosti a vysokému bodu tavení ztěžuje výrobu a tím zvyšuje cenu těchto tranzistorů. Největší nepříjemností je zhoršená tepelná vodivost, kvůli které se

z čipu špatně odvádí teplo a zvyšuje tak nároky na chlazení. Vlivem, již zmíněného, nízkého potřebného náboje, a tím i napětí (až pouhé 3 V), k sepnutí může být problematické nechtěné spínání prvku vlivem parazitních kapacit. [7]

Komerčně nabízené GaN MOSFETy v současnosti téměř není možné sehnat, jelikož jsou stále ve fázi výzkumu a vývoje. Z uvedených vlastností jsou vhodné pro použití na malých výkonech, především kvůli omezenému odvodu tepla, při vysokých spínacích frekvencích dosahujících až 1 MHz. Předpokládanými aplikacemi jsou velmi přesné regulační odvody s minimálními zvlněními a požadavky na filtrační členy, potažmo pro aplikace, kde se tohoto může využívat pro posuny výkonů na vyšší frekvence a tím zmenšovaní transformátorů na deskách plošných spojů. [7]

1.2 Napěťový střídač

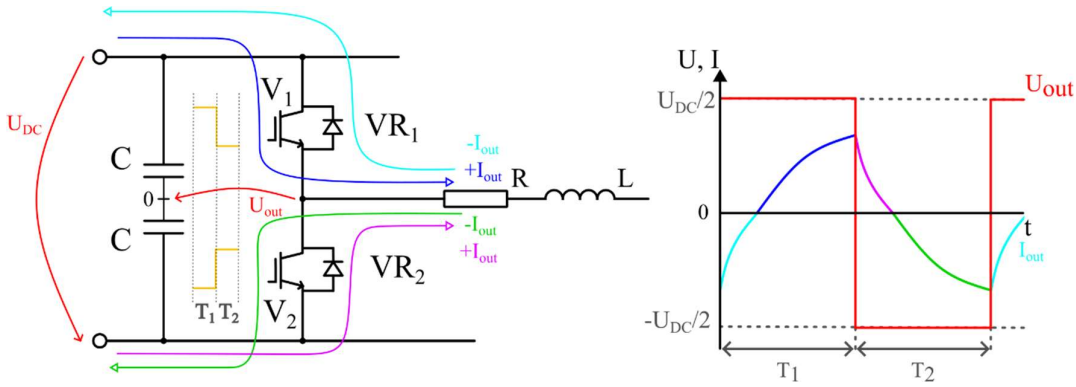
Napěťové střídače si lze zjednodušeně představit jako soubor spínačů, které střídavě připojují k zátěži kladný nebo záporný pól stejnosměrného meziobvodu čímž vytváří na výstupu střídavé obdélníkové napětí u kterého rychlost, respektive četnost, spínání určuje výstupní frekvenci *f*. Takový střídač se skládá z jednotlivých větví, ve kterých jsou 2 spínače, a společného napájení pro všechny větvě a zásobníkem energie, typicky kondenzátorem. Napětí výstupu U_{out} pak vztahujeme k virtuální nule, kterou představuje na polovinu rozdělený DC meziobvod – tedy $U_{DC}/2$. Tím se ono výstupní napětí jeví jako střídavé kolem nulové hodnoty s amplitudou $U_{DC}/2$. [1]



Obr. 5: Větev střídače s obecně vypínatelnými součástkami

Obecně spínatelné součástky můžeme nahradit buď MOSFET nebo IGBT tranzistory. Uvažujme zde také nejběžnější zátěž, tedy RL – odporově induktivní. Je nutné tedy počítat s proudem do zátěže jakožto se stavovou veličinou, kdy se tento proud nemůže měnit skokově a postupně se v podobě energie akumuluje do cívky. Změnou polarity připojeného napětí k větvi střídače tudíž nemůže proud ihned otočit svou polaritu společně s napětím,

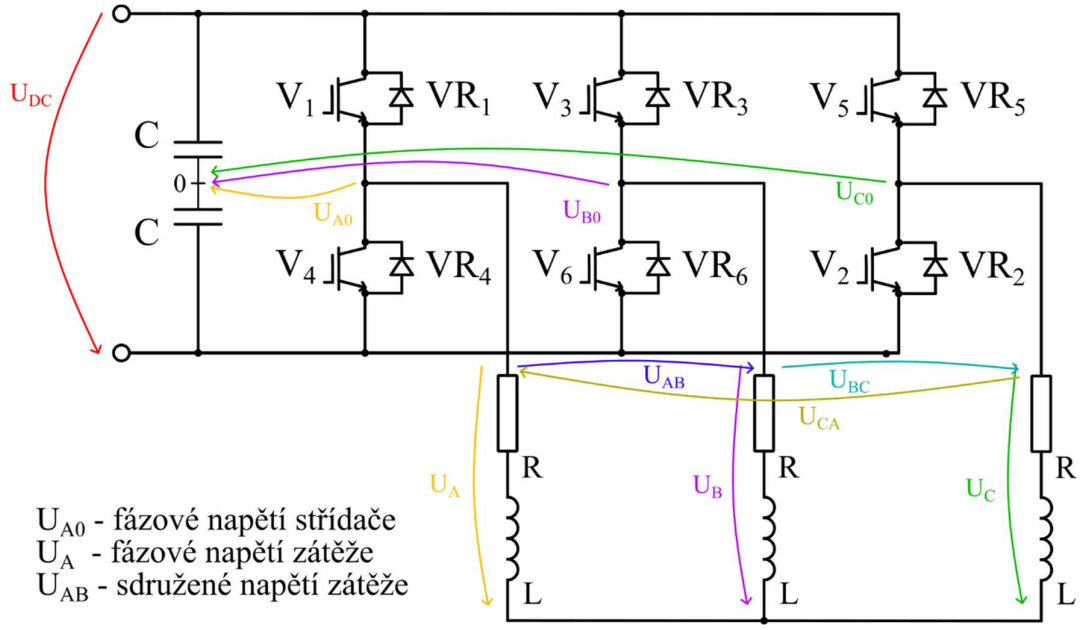
jako kdybychom měli čistě odporovou zátěž, ale jeho směr setrvává stejný a postupně se snižuje až do nuly. Teprve v tento moment může změnit svou polaritu. Abychom předešli nebezpečným přepětím, a zničení spínačů tímto přepětím, je nutné vytvořit pro proud vhodnou cestu, kterou se může uzavírat. Toho dosáhneme zařazením antiparalelních diod. Tuto situaci znázorňuje ilustrativní obrázek níže. [1]



Obr. 6: Větev střídače s reálnými součástkami a RL zátěží

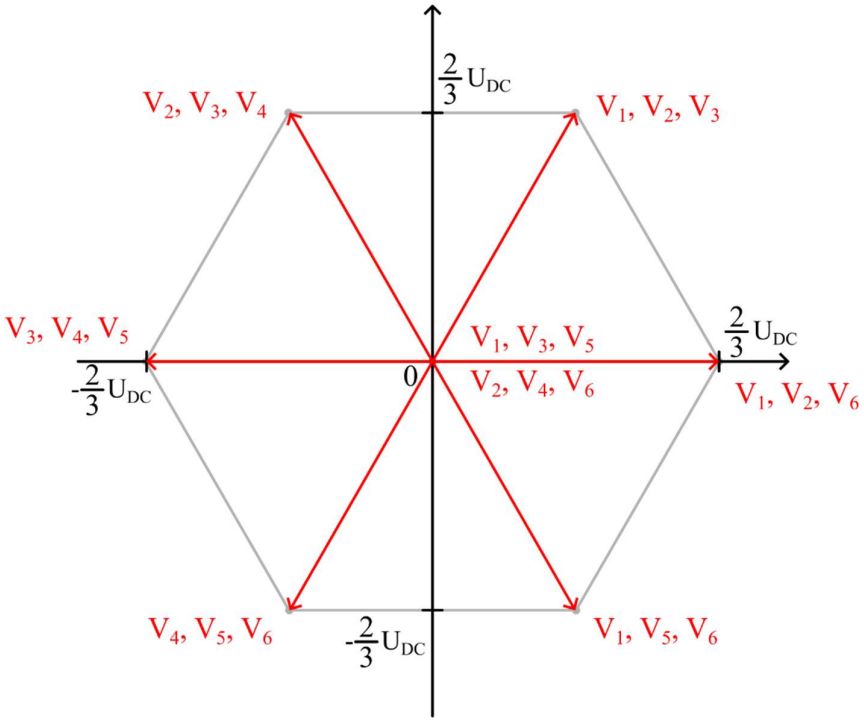
Při sepnutí horního prvku V_1 teče proud přes tento tranzistor do zátěže (modrá). Po vypnutí prvku V_1 a sepnutí prvku V_2 nemůže tento proud zaniknout a tak se uzavírá přes diodu VR_2 (fialová). Po zaniknutí tohoto proudu se jeho polarita obrací a protéká sepnutím tranzistorem V_2 (zelená). Po rozepnutí V_2 a sepnutí V_1 opět proud po exponenciále klesá až k nule (tyrkysová) a cyklus se opakuje. [1]

Spojením třech těchto větví získáme 3-fázový střídač. Přidáním čtvrté 4-fázový střídač a tak dále. Pro účely ilustrace tohoto zapojení, a zapojení použité v této práci, se budeme držet 3-fázové verze společně s RL zátěží představující motor zapojený do hvězdy.



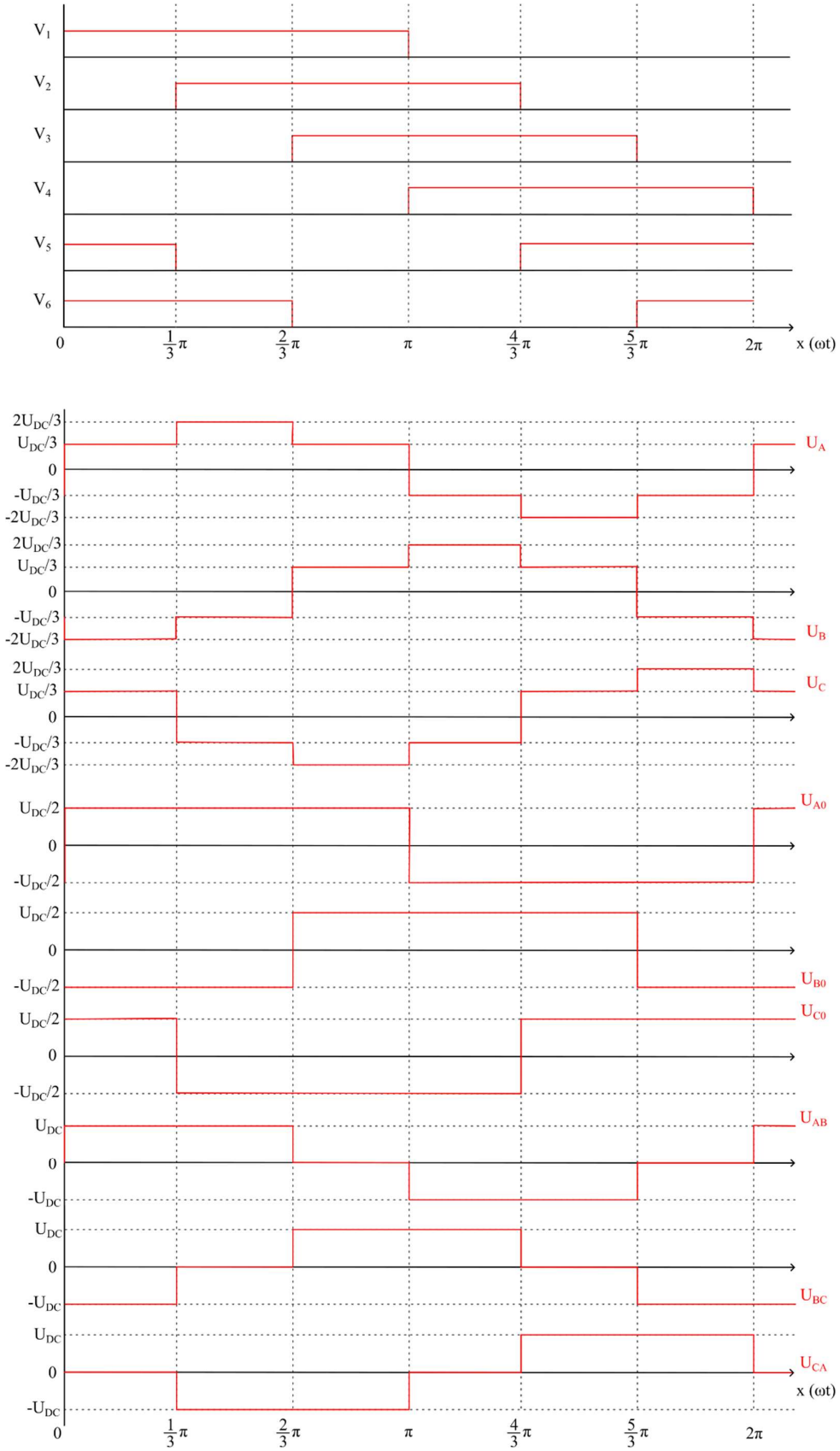
Obr. 7: 3-fázový střídač s RL zátěží a vyznačenými napětími

Ze zapojení je zřetelné, že zde vzniká 8 spínacích kombinací z toho 2 nulové, které představuje sepnutí všech horní nebo všech dolních prvků. Zakázanými kombinacemi je spínání prvků pod sebou, jelikož by tím došlo k vyzkratování DC meziobvodu. Příkladem funkce může být kombinace sepnutých prvků V_1, V_2 a V_6. Uvažujme symetrickou zátěž. Proud poteče z DC meziobvodu do zátěže přes zmíněný tranzistor V_1 na fázi A, na které vyvolá napětí U_A, a vrací se skrze fáze B a C, kde opět vyvolá úbytky U_B a U_C, přes tranzistory V_2 a V_6 na zápornou svorku DC meziobvodu. Zvolíme-li jako referenční bod středový bod zátěže spojené do hvězdy, pak se napětí U_A jeví jako kladné a U_B s U_C jako záporné. Vycházíme z již zmíněné symetrie zátěže z čehož vyplývá, že proud protékající větví A se rozdělí na dvě poloviny tekoucí větví B a C, což má za důsledek poloviční úbytky napětí na těchto větvích oproti větvi A. Pokud je plné napětí U_{DC}, pak napětí na větvi A je $\frac{2}{3}\cdot U_{DC}$ a větví B a C je napětí $\frac{1}{3}\cdot U_{DC}$, respektive $-\frac{1}{3}\cdot U_{DC}$, když napětí vztáhneme k zvolenému referenčnímu bodu. Z toho vyplývá, že maximálním napětím zátěže jsou zmíněné $\frac{2}{3}\cdot U_{DC}$. Obměnou spínacích kombinací získáme další kombinace, pomocí kterých dostáváme maximální možné napětí, ať kladné nebo záporné, na jednotlivé fáze zátěže. Tyto kombinace zobrazuje obrázek níže. [1]



Obr. 8: Vektory spínacích kombinací

Střídáním těchto kombinací v pořadí, jak jdou po obvodu za sebou, dosáhneme obdélníkového řízení střídače a tím tří fázového výstupního napětí, kde se proud přizpůsobuje zátěži. Průběhy napětí v závislosti na posloupnosti spínacích kombinací zobrazuje spínací diagram.



Obr. 9: Spínací diagram s průběhy všech napětí

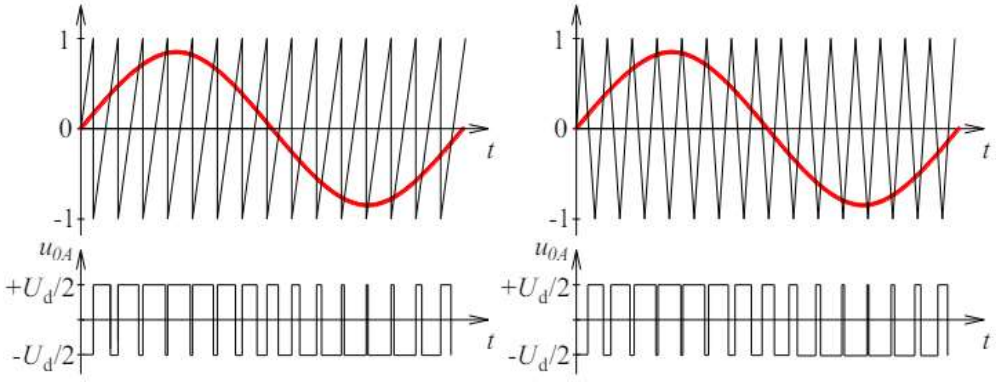
Průběhu napětí, a tím i proudu, více se blížícímu sinusovce lze dosáhnout improvizováním vektorů mezi definovanými stavy, případně v kombinaci s nulovými vektory. Tím lze ve střední hodnotě dosáhnout i jiných velikostí fázového napětí zátěže než $\frac{1}{3}\cdot U_{DC}$ a $\frac{2}{3}\cdot U_{DC}$. Nejběžnějším způsobem, kterým získáváme kombinaci spínacích pulzů, a tím tedy i hodnoty výstupního napětí, je pomocí pulzně-šířkové modulace (PWM). [1]

1.3 Pulze-šířková modulace a její varianty

Jak již bylo avizováno, PWM je důležitou součástí moderních pohonů kvůli svým vlastnostem jako je dosažení požadovaného napětí ve střední hodnotě, především pak první harmonické, a přiblížení se sinusovému průběhu proudu zátěží, mluvíme-li o napěťových střídačích. Jedná se asynchronní modulaci vhodnou pro výkonové součástky, jako IGBT a MOSFET tranzistory, jelikož jejím výstupem je prakticky stav zapnuto/vypnuto. [9]

1.3.1 Subharmonická (referenční) PWM

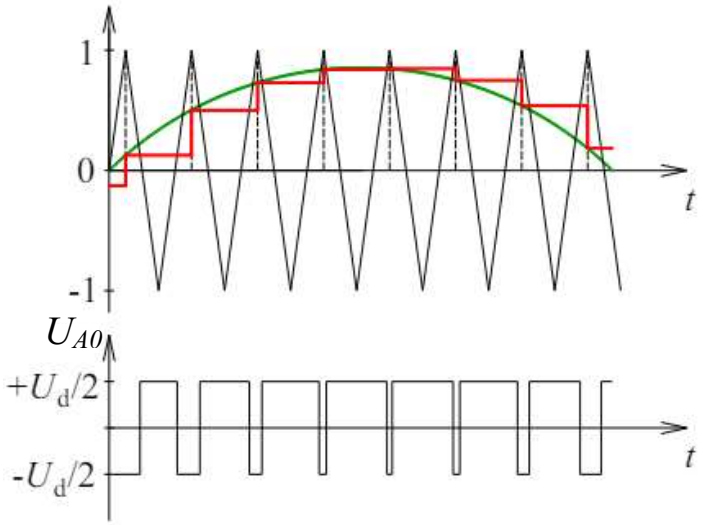
Metoda spočívá v porovnávání referenčního a nosného signálu. Jako nosný signál se typicky používá pilový nebo trojúhelníkový průběh. Jako referenční signál je použitý signál o požadovaném průběhu na výstupu měniče, tedy zpravidla sinus s patřičnou frekvencí. Při porovnávání těchto dvou signálů dostáváme dva stavy – okamžitá hodnota referenčního signálu je vyšší než okamžitá hodnota nosného signálu a výstup nabývá kladné hodnoty napětí, v opačném případě je výstupní napětí záporné. [9]



Obr. 10: Princip subharmonické PWM s pilovou (vlevo) a trojúhelníkovou (vpravo) nosnou (převzato z [9])

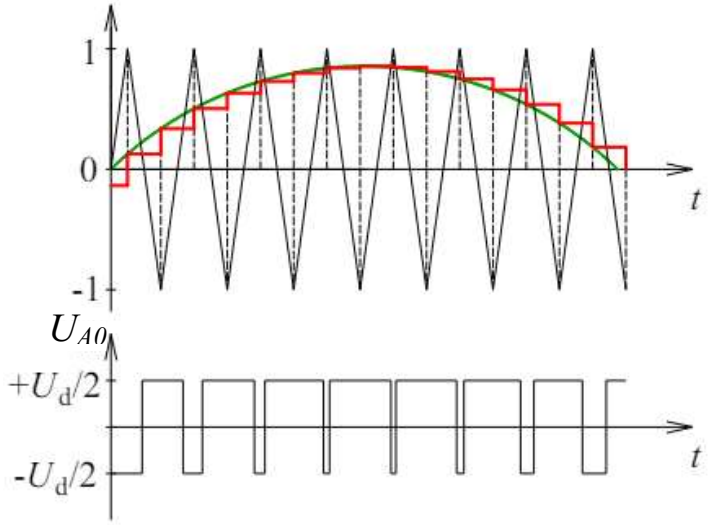
V případě analogové PWM, jako je na obrázku výše, dochází k překlopení stavu okamžitě při protnutí nosného signálu referenčním. V případě využití digitální PWM, tedy implementovanou v mikrokontroleru, k tomuto překlopení stavů dochází s mírným zpožděním, čímž vzniká nesprávná generace spínacích pulzů a tím zvýšení spínacích ztrát a obsahu vyšších harmonických ve výstupním napětím. Řešením je vzorkování referenčního

signálu pouze v definovaných bodech spíše než kontinuálně. To znamená, že hodnotu pro porovnání s nosným signálem vyčítáme pouze jednou za periodu nosného signálu, většinou v jednom z jeho maxim. [9]



Obr. 11: PWM vzorkovaná jednou za periodu nosného signálu (převzato z [9])

Možným vylepšením tohoto řešení je častější vzorkováním nosného signálu, konkrétně jednou za půl periodu nosného signálu, tedy v obou jeho maximech. [9]



Obr. 12: PWM vzorkovaná dvakrát za periodu nosného signálu (převzato z [9])

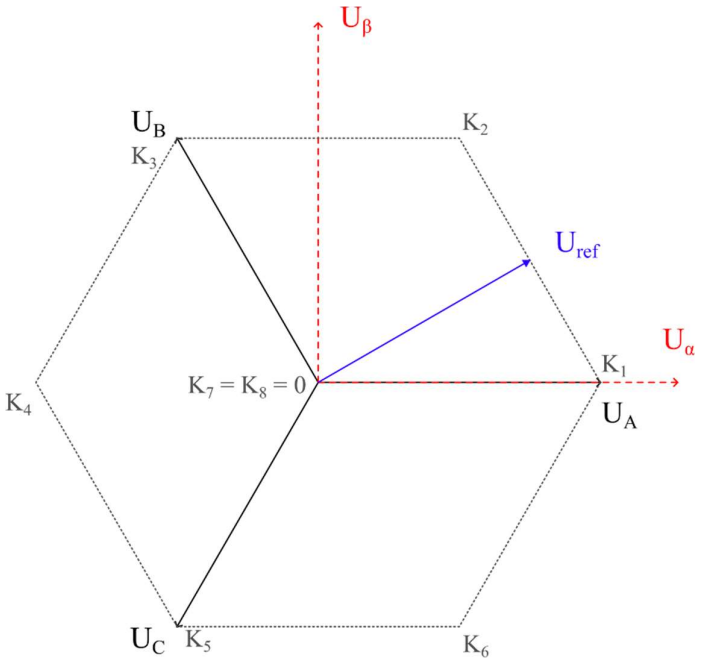
1.3.2 Modulace prostorového vektoru (SVM)

Oproti předchozí metodě je zde umístění spínacích pulzů předem dáno. Základem je zde obdélníkové řízení (viz. *Obr. 9*) využívající šesti aktivních a dvou nulových spínacích kombinací (viz. *Obr. 8*), značených níže jako K_1 až K_8, společně s prostorovým vektorem.

Prostorové vektory v komplexní rovině lze získat Clarkové transformací fázových napětí trojfázového systému A, B, C do dvouosého systému α, β, přičemž osa α je totožná s osou A. Tato transformace je dána rovnicí:

$$\begin{pmatrix} U_\alpha \\ U_\beta \end{pmatrix} = \frac{2}{3} \cdot \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{pmatrix} \cdot \begin{pmatrix} u_A \\ u_B \\ u_C \end{pmatrix}, \tag{1}$$

kterou získáváme následující rozložení vektorů. Referenční vektor U_{ref} představuje námi požadovaný stav a lze ho dosáhnout zkombinováním jednotlivých spínacích kombinací, v tomto konkrétním případě pomocí K_1, K_2 a nulového vektoru – K_7 nebo K_8.



Obr. 13: Vektory před a po Clarkové transformaci se zavedeným referenčním vektorem

Pro realizaci takového stavu je nutné nejen znát potřebné spínací kombinace, ale také dobu, po jakou mají být realizované. Řekněme, že Tc je spínací perioda a T_0, T_1 a T_2 jsou aktivní doby sepnutí jednotlivých vektorů, tedy doby K_7/K_8, K_1 a K_2. Potom pro výpočet jednotlivých dob platí rovnice:

$$U_{ref} \cdot T_c = z_0 \cdot K_0 + z_1 \cdot K_1 + z_2 \cdot K_2 = \frac{T_0}{T_c} \cdot K_0 + \frac{T_1}{T_c} \cdot K_1 + \frac{T_2}{T_c} \cdot K_2, \tag{2}$$

kde z_0, z_1 a z_2 reprezentují poměrné sepnutí odpovídající poměru doby sepnutí ku celkové době spínací periody. Zároveň tedy musí platit, že

$$T_c = T_0 + T_1 + T_2 \tag{3}$$

Finální pozice vektorů odvíjejících se od kombinací K_7/K_8, K_1 a K_2 je dána požadovanou amplitudou a úhlem výstupního napětí. [9]

1.4 Elektrický stroj

Současně používanými motory v kombinaci s měniči jsou asynchronní motory a to především s klecí nakrátko. To znamená, že rotor tohoto motoru se skládá z rotorových plechů a vodičů v drážkách v podobě měděných nebo hliníkový tyček spojených nakrátko na čelech rotoru zkratovacími kroužky. Důsledkem je uzavírání proudu čistě v obvodu rotoru díky čemuž není třeba vyvádět žádné vývody ven na svorky z čehož vyplývají dvě velké výhody tohoto motoru – rotor může zabírat většinovou část vnitřního prostoru motoru, čímž dosahuje vyšší hustoty výkonu na objem, a není na něm třeba provádět téměř žádnou údržbu. Stator se skládá z nosného tělesa (krytu), statorových plechů a vinutí jehož konce jsou vyvedeny na svorkovnicích. Druhou variantou je motor s kroužkovou kotvou. Ten má konstrukci rotoru stejnou jako motor s klecí nakrátko s tím rozdílem, že zde vinutí není zkratované na obou čelech rotoru, ale je vyvedeno na sběrací kroužky odkud je pomocí uhlíkových kartáčů vyvedeno na svorkovnici. To umožňuje řízený rozběh a regulaci motoru pomocí odporů, případně rekuperaci energie k dalšímu použití nebo navrácení do rozvodné sítě. Vzhledem k současnému použití frekvenčních měničů je tato regulace nevýhodná. Navíc je zde třeba udržovat zmíněné uhlíkové kartáče, které, kromě jiného, zabírají část vnitřního prostoru, čímž se musí celý motor zvětšit pro dosažení požadovaného výkonu, případně při stejném rozměru dosahují menších výkonů oproti motorům s klecí nakrátko. Z těchto důvodů se již tento druh asynchronních motorů nepoužívá nebo pouze ve speciálních případech. [8] [10]

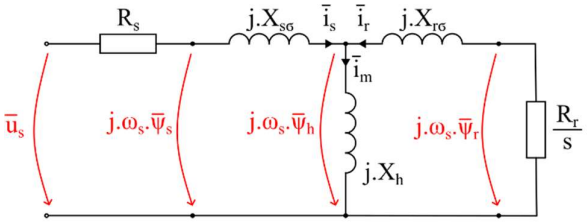
Dalším typem střídavých točivých strojů jsou synchronní motory. Jejich charakteristickou vlastností je, že dokáží pracovat pouze při přesně synchronní statorové a rotorové rychlosti jinak motor vypadne ze synchronismu a zastavuje se, avšak tato nevýhoda je díky moderním měničům a algoritmům řízení odstraněna. Konstrukcí statoru shodné s asynchronními. Hlavním rozdílem je konstrukce rotoru, který můžeme rozdělit na rotor hladký a rotor s vyniklými póly. Jestliže má motor hladký rotor, pak je jeho indukčnost ve všech směrech stejná a tím je indukčnost při jeho otáčení stále stejná. Rotor s vyniklými póly naopak stejnou indukčnost ve všech směrech (respektive podélnou a příčnou indukčnost) stejnou nemá a tak se při otáčení indukčnost mění. Dále je možné rozdělit synchronní motory dle buzení na stejnosměrné, buzení permanentními magnety a bez buzení, takzvané reluktanční stroje. Stejnosměrně buzené motory vykazují vysoké hodnoty momentu, avšak

musí být řešeno jejich napájení například přes kluzné kontakty nebo rotačním transformátorem. Toto řeší motory s permanentními magnety, které vytváří stacionární magnetické pole v rotoru. Nevýhodou je zde složité odbuzování záporným momentotvorným proudem, vysoká cena permanentních magnetů a nutnost hlídání kritické, Curieovy, teploty, kdy dochází ke ztrátě feromagnetických vlastností magnetů. V případě reluktančních strojů není třeba řešit napájeni točícího se rotoru ani teploty, jelikož rotor je tvořen pouze z rotorových plechů. Nevýhodou je zde nižší výkonová hustota oproti předchozím variantám. Kvůli již zmíněné ceně permanentních magnetů vzniklo více variant synchronních motorů využívající tyto magnety: SPMSM (s povrchovými magnety na rotoru), IPMSM (s vnitřními magnety na rotoru), PMA-RSM (magnety asistovaný reluktanční motor). [11]

1.4.1 Náhradní schéma a matematický model asynchronního motoru

Náhradní schéma asynchronního motoru pro ustálené stavy se stává z následujících částí:

- rezistory realizující odpor statoru a rotoru
- rozptylové indukčnosti, respektive jejich reaktance, reprezentující ztráty v magnetickém obvodu odpovídající vztahu $X_\sigma = \omega \cdot L_\sigma$
- reaktance realizující hlavní indukčnost přenášející výkon na stranu rotoru



u_s - napětí statoru
i_s - proud statoru
i_r - proud rotoru
i_h - proud hlavní indučností
R_s - odpor statorového vinutí
R_r - odpor rotoru
$X_{s\sigma}$ - reaktance rozptylové indukčnosti statoru
$X_{r\sigma}$ - reaktance rozptylové indučnosti rotoru
X_h - reaktance hlavní indukčnosti
s - skluz

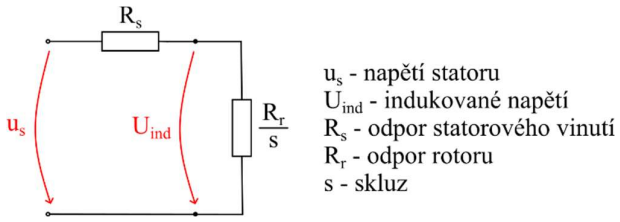
Obr. 14: Náhradní schéma asynchronního motoru pro ustálené stavy

Podle II. Kirchhoffova zákonu můžeme sestavit napěťové rovnice popisující toto náhradní schéma. [12]

$$|\overline{u_s}| = R_s \cdot \bar{\imath}_s + j \cdot X_{s\sigma} \cdot \bar{\imath}_s + j \cdot X_h \cdot \bar{\imath}_m \tag{4}$$

$$0 = \frac{R_r}{s} \cdot \bar{\imath}_r + j \cdot X_{r\sigma} \cdot \bar{\imath}_r + j \cdot X_h \cdot \bar{\imath}_m \tag{5}$$

Toto schéma lze dále zjednodušit na pouhý statorový odpor a odpor rotoru přepočítaný přes skluz na stranu statoru. [12]



Obr. 15: Zjednodušené náhradní schéma asynchronního motoru

Tyto náhradní schémata jsou vhodná pro skalární řízení (viz kapitola 1.5.1 Skalární řízení) a pro principiální pochopení přenosu výkonu ze statoru na hřídel, avšak nejsou vhodné pro sestavení matematického modelu použitelného pro vektorové řízení.

1.4.2 Matematický model AM vhodný pro vektorové řízení

Úkolem matematického modelu ve vektorovém řízení je především estimace úhlu natočení rotoru ϑ společně s výpočty proudů Id a Iq. Tento výpočet sice předešlý model umožňuje za použití funkce *arctg()*, ale není vhodný. Proto je nutné sestavit takové rovnice, které vychází, v případě proudového modelu při použití napěťového střídače, z měřených proudů a rychlosti otáčení.

Pro odvození je důležité si nejdříve zavést uvažované souřadné systémy:

- Souřadný systém statoru – index I, stojící souřadný systém
- Souřadný systém rotoru – index II, systém se otáčí rychlostí $\omega_{II} = p_p \cdot \omega_{mech}$
- Souřadný systém rotoru – index III, systém s obecně proměnnou rychlostí ω_{III} svázaný s magnetickým tokem statoru, hlavním tokem nebo tokem rotoru

Výchozím bodem jsou fázová napětí statoru.

$$\begin{aligned} u_{sa} &= R_s \cdot i_{sa} + u_{ind_a} = R_s \cdot i_{sa} + \frac{d\psi_{sa}}{dt} \\ u_{sb} &= R_s \cdot i_{sb} + u_{ind_b} = R_s \cdot i_{sb} + \frac{d\psi_{sb}}{dt} \\ u_{sc} &= R_s \cdot i_{sc} + u_{ind_c} = R_s \cdot i_{sc} + \frac{d\psi_{sc}}{dt} \end{aligned} \tag{6}$$

Za pomoci transformace Clarkeové převedeme tyto napětí na prostorový vektor v souřadném systému I.

$$\begin{aligned} \bar{u}_{s_I} &= k \cdot (u_{sa} + \bar{a} \cdot u_{sb} + \bar{a}^2 \cdot u_{sc}) = R_s \cdot \bar{\imath}_{s_I} + \frac{d\bar{\psi}_{s_I}}{dt} \\ u_{s0} &= k_0 \cdot (u_{sa} + u_{sb} + u_{sc}) = R_s \cdot i_{s0} + \frac{d\psi_{s0}}{dt} \end{aligned} \tag{7}$$

, kde $\bar{a} = e^{j \cdot 120°}$, čímž představuje posuv o 120° mezi jednotlivými fázemi. [12]

Obdobný postup aplikujeme i v případě odvozování rovnice platné pro rotor. Opět vycházíme z fázových napětí, tentokrát rotoru.

$$\begin{aligned} u_{ra} &= R_r \cdot i_{ra} + \frac{d\psi_{ra}}{dt} \\ u_{rb} &= R_r \cdot i_{rb} + \frac{d\psi_{rb}}{dt} \\ u_{rc} &= R_r \cdot i_{rc} + \frac{d\psi_{rc}}{dt} \end{aligned} \tag{8}$$

Opět zde využijeme transformace Clarkeové a převádíme rotorová fázová napětí na prostorový vektor, tentokrát v souřadném systému II, tedy otáčející se systém.

$$\begin{aligned} \bar{u}_{r_{II}} &= k \cdot (u_{ra} + \bar{a} \cdot u_{rb} + \bar{a}^2 \cdot u_{rc}) = R_r \cdot \bar{\imath}_{r_{II}} + \frac{d\bar{\psi}_{r_{II}}}{dt} \\ u_{r0} &= k_0 \cdot (u_{ra} + u_{rb} + u_{rc}) = R_r \cdot i_{r0} + \frac{d\psi_{r0}}{dt} \end{aligned} \tag{9}$$

Pro přechod do souřadného systému III je potřeba přepočet za pomocí úhlu ϑ popisující okamžité natočení souřadného systému III vůči systému I. Rychlost systému III pak odpovídá $\omega_{III} = \frac{d\vartheta}{dt}$. Přepočet veličin pak lze provést jako $\bar{x}_I = \bar{x}_{III} \cdot e^{j \cdot \vartheta}$. [12]

Obdobným postupem lze definovat přechod ze souřadného systému II do systému III. Okamžité otočení systému III vůči systému II tentokrát označíme jako ε. Relativní, jelikož oba systémy jsou rotující, rychlost systému III vůči systému II pak odpovídá $\Delta\omega_{III,II} = \frac{d\varepsilon}{dt}$, přepočet veličin tedy odpovídá $\bar{x}_{II} = \bar{x}_{III} \cdot e^{j \cdot \varepsilon}$. [12]

Pomocí těchto uvedených vztahů můžeme statorovou rovnici (7) převést ze souřadného systému I do systému III, kde má rovnice tvar

$$\begin{aligned} \bar{u}_{s_{III}} \cdot e^{j \cdot \vartheta} &= R_s \cdot \bar{\imath}_{s_{III}} \cdot e^{j \cdot \vartheta} + \frac{d}{dt} \cdot \left(\bar{\psi}_{s_{III}} \cdot e^{j \cdot \vartheta}\right) \\ \bar{u}_{s_{III}} \cdot e^{j \cdot \vartheta} &= R_s \cdot \bar{\imath}_{s_{III}} \cdot e^{j \cdot \vartheta} + \frac{d\bar{\psi}_{s_{III}}}{dt} \cdot e^{j \cdot \vartheta} + j \cdot \frac{d\vartheta}{dt} \cdot \bar{\psi}_{s_{III}} \cdot e^{j \cdot \vartheta} \\ \bar{u}_{s_{III}} &= R_s \cdot \bar{\imath}_{s_{III}} + \frac{d\bar{\psi}_{s_{III}}}{dt} + j \cdot \omega_{III} \cdot \bar{\psi}_{s_{III}} \end{aligned} \tag{10}$$

Obdobný postup opět aplikujeme i pro rovnici (9), tedy rotor.

$$\begin{aligned} \bar{u}_{r_{III}} \cdot e^{j \cdot \varepsilon} &= R_r \cdot \bar{\imath}_{r_{III}} \cdot e^{j \cdot \varepsilon} + \frac{d}{dt} \cdot \left(\bar{\psi}_{r_{III}} \cdot e^{j \cdot \varepsilon}\right) \\ \bar{u}_{r_{III}} \cdot e^{j \cdot \varepsilon} &= R_r \cdot \bar{\imath}_{r_{III}} \cdot e^{j \cdot \varepsilon} + \frac{d\bar{\psi}_{r_{III}}}{dt} \cdot e^{j \cdot \varepsilon} + j \cdot \frac{d\varepsilon}{dt} \cdot \bar{\psi}_{r_{III}} \cdot e^{j \cdot \varepsilon} \\ \bar{u}_{r_{III}} &= R_r \cdot \bar{\imath}_{r_{III}} + \frac{d\bar{\psi}_{r_{III}}}{dt} + j \cdot \Delta\omega_{III,II} \cdot \bar{\psi}_{r_{III}} \end{aligned} \tag{11}$$

Jelikož uvažujeme rotor s klecí nakrátko, pak platí, že $\bar{u}_{r_{III}} = 0$. Zároveň pro účely vektorového řízení zavedeme použitý rotující systém se souřadnicemi d, q s novým značením, kde ve statorové rovnici nahrazujeme $\omega_{III} = \omega_{s\psi}$, což odpovídá rychlosti rotace onoho systému, a v rotorové rovnici nahrazujeme $\Delta\omega_{III,II} = \omega_{r\psi}$, kde zároveň platí vztah mezi oběma rychlostmi dle vztahu $\omega_{r\psi} = \omega_{s\psi} - p_p \cdot \omega_{mech}$. Upravené rovnice (10) a (11) podle tohoto popisu pak vypadají následovně:

$$\begin{aligned} \bar{u}_s &= R_s \cdot \bar{\imath}_s + \frac{d\bar{\psi}_s}{dt} + j \cdot \omega_{s\psi} \cdot \bar{\psi}_s \\ 0 &= R_r \cdot \bar{\imath}_r + \frac{d\bar{\psi}_r}{dt} + j \cdot \omega_{r\psi} \cdot \bar{\psi}_r \end{aligned} \tag{12}$$

Pro další postup je třeba si definovat rovnice popisující proudy a magnetické toky ve stroji. Výchozím bodem je zde náhradní schéma asynchronního motoru (viz. Obr. 14). Proudy potom můžeme popsat následujícími rovnicemi:

$$\begin{aligned} \bar{\imath}_s + \bar{\imath}_r &= \bar{\imath}_m = \frac{\bar{\psi}_h}{L_h} \\ \bar{\imath}_r &= -\bar{\imath}_s + \frac{\psi_h}{L_h} = -\bar{\imath}_s + \frac{\bar{\psi}_h}{L_h} - \frac{L_{r\sigma}}{L_h} \cdot \bar{\imath}_r \\ \bar{\imath}_r &= -\frac{L_h}{L_r} \cdot \bar{\imath}_s + \frac{1}{L_r} \cdot \bar{\psi}_r \end{aligned} \tag{13}$$

Pro magnetické toky platí následující:

$$\begin{aligned} \bar{\psi}_s &= \bar{\psi}_r + L_{s\sigma} \cdot \bar{\imath}_s - L_{r\sigma} \cdot \bar{\imath}_r = \bar{\psi}_r + L_{s\sigma} \cdot \bar{\imath}_s + \frac{L_{r\sigma} \cdot L_h}{L_r} \cdot \bar{\imath}_s - \frac{L_{r\sigma}}{L_r} \cdot \bar{\psi}_r \\ \bar{\psi}_s &= \frac{L_h}{L_r} \cdot \bar{\psi}_r + \left(L_{s\sigma} + \frac{L_h}{L_r} \cdot L_{r\sigma} \right) \cdot \bar{\imath}_s \end{aligned} \tag{14}$$

Se zadefinovanými rovnicemi popisující proudové poměry a poměry magnetických toků můžeme za pomocí rovnic (13) a (14) upravit rovnici (12). Výsledkem této úpravy dostáváme:

$$\begin{aligned} u_{sd} &= R_s \cdot i_{sd} + \frac{L_h}{L_r} \cdot \frac{d\psi_{rd}}{dt} + \left(L_{s\sigma} + \frac{L_h}{L_r} \cdot L_{r\sigma} \right) \cdot \frac{di_{sd}}{dt} - \omega_{s\psi} \cdot \left(L_{s\sigma} + \frac{L_h}{L_r} \cdot L_{r\sigma} \right) \cdot i_{sq} \\ u_{sq} &= R_s \cdot i_{sq} + \left(L_{s\sigma} + \frac{L_h}{L_r} \cdot L_{r\sigma} \right) \cdot \frac{di_{sq}}{dt} + \omega_{s\psi} \cdot \frac{L_h}{L_r} \cdot \psi_{rd} + \omega_{s\psi} \cdot \left(L_{s\sigma} + \frac{L_h}{L_r} \cdot L_{r\sigma} \right) \cdot i_{sd} \\ 0 &= -R_r \cdot \frac{L_h}{L_r} \cdot i_{sd} + \frac{R_r}{L_r} \cdot \psi_{rd} + \frac{d\psi_{rd}}{dt} \\ 0 &= -R_r \cdot \frac{L_h}{L_r} \cdot i_{sq} + \omega_{r\psi} \cdot \psi_{rd} \end{aligned} \tag{15}$$

Následujícím krokem je dosazení derivace toku z třetí řádky rovnice (15) do první řádky stejné rovnice, čímž dosáhneme úpravy napěťové rovnice do statorového tvaru.

$$\begin{aligned}\frac{di_{sd}}{dt} &= -\alpha \cdot i_{sd} + \omega_{s\psi} \cdot i_{sq} + \beta \cdot \psi_{rd} + \delta \cdot u_{sd} \\ \frac{di_{sq}}{dt} &= -\omega_{s\psi} \cdot i_{sd} - \mu \cdot i_{sq} - \eta \cdot \omega_{s\psi} \cdot \psi_{rd} + \delta \cdot u_{sq} \\ \frac{d\psi_{rd}}{dt} &= R_r \cdot \frac{L_h}{L_r} \cdot i_{sd} - \frac{R_r}{L_r} \cdot \psi_{rd}\end{aligned} \tag{16}$$

Pro proudový model asynchronního motoru se dále vychází z třetí řádky rovnice (16), která popisuje derivaci toku ψ_{rd}, jehož velikost můžeme vyjádřit jako součet aktuální hodnoty s touto derivací, tedy jako $\psi_{rd} = \psi_{rd} + \frac{d\psi_{rd}}{dt}$. Toto vyjádření je důležité z hlediska následujících výpočtů rychlosti rotorového pole $\omega_{r\psi}$, jehož vyjádření vychází z druhé řádky rovnice (12), kterou dále rozepisujeme na složky. [12]

$$\begin{aligned}0 &= R_r \cdot I_{rd} + \frac{d\bar{\psi}_r}{dt} \\ 0 &= R_r \cdot i_{rq} + \omega_{r\psi} \cdot \bar{\psi}_r \;\rightarrow \omega_{r\Psi} = -i_{rq} \cdot \frac{R_r}{\bar{\psi}_r}\end{aligned} \tag{17}$$

Pro rotorový tok zároveň platí:

$$\psi_r = L_h \cdot i_s + L_r \cdot i_r \;\rightarrow \psi_{rq} = 0 = L_h \cdot i_{sq} + L_r \cdot i_{rq} \;\rightarrow i_{rq} = -i_{sq} \cdot \frac{L_h}{L_r} \tag{18}$$

Dosazením i_{rq} odvozeného v rovnici (18) do druhého řádku rovnice (17) dostáváme finální podobu vztahu popisující rychlost pole rotoru:

$$\omega_{r\psi} = i_{sq} \cdot \frac{L_h}{L_r} \cdot \frac{R_r}{\psi_r} \tag{19}$$

Výpočet rychlosti statorového pole $\omega_{s\psi}$ dosáhneme již prostým součtem rychlostí rotorového pole s mechanickou rychlostí získanou pomocí, například, čidla otáček.

$$\omega_{s\psi} = \omega_{r\psi} + \omega_{mech} \cdot p_p \tag{20}$$

Finální výpočet úhlu natočení souřadného systému ϑ, a tedy i natočení rotorového toku, dosáhneme integrací této spočtené rychlosti statorového pole.

$$\vartheta = \int \omega_{s\psi} \cdot dt \tag{21}$$

Výše popsané vztahy jsou postačující k sestavení matematického modelu motoru vyhovující pro vektorové řízení. [12]

1.5 Algoritmy řízení pohonů

Vlastnosti pohonů zásadně ovlivňuje u druh jeho řízení, respektive řízení spínání měniče. Mezi tři nejrozšířenější metody patří skalární řízení, vektorové řízení a přímé řízení momentu, které ovšem není náplní této práce.

1.5.1 Skalární řízení

Skalární řízení, známé také jako *U/f*, patří mezi nejjednodušší, ale prakticky použitelné pouze pro asynchronní motory, metody. Jednoduchost je zde také zaplacena horší dynamikou a tak je použitelné pouze pro nenáročné aplikace jako kompresory, ventilátory a další, kde se zátěžný moment nemění skokem. Jeho princip je založen na udržování konstantního magnetického toku pomocí udržování konstantního poměru napětí U a frekvence f přiváděných na svorky statoru, případně snižování tohoto poměru při odbuzovaní, aby nemohlo dojít k přesycení stroje. Jelikož zadávané napětí vychází přímo ze statorové frekvence, pak při rozběhu, kdy se ona statorová frekvence f_s blíží nule, napětí také blíží nule. Z toho vyplývá, že rozběhový moment je velmi malý a je třeba ho kompenzovat. Pro to vycházíme z rovnice:

$$|\bar{U}_{ind}| = |\bar{\psi}_s| \cdot \omega_s = K_U \cdot f_s; K_U = \frac{U}{f_s}, \tag{22}$$

kde K_U je konstantou, U_{ind} je indukované napětí, ψ_s je statorový magnetický tok a f_s je statorová frekvence. [1] [12]

Napětí na statoru U_s pak odpovídá rovnici

$$|\bar{U}_s| = |\bar{U}_{ind}| + \Delta U, \tag{23}$$

kde ΔU je skalár. Jeho výpočet lze získat třemi metodami – jako funkci statorového proudu ($\Delta U = f(|\bar{I}_s|)$), jako funkci statorové frekvence ($\Delta U = f(f_s)$) nebo jako funkci rotorové frekvence ($\Delta U = f(f_r)$). Poslední zmíněný případ je nejpoužívanější a vychází ze zjednodušeného náhradního schématu asynchronního motoru (viz. Obr. 15). Vycházejíc z tohoto schématu pak ΔU odpovídá následující rovnici. [12]

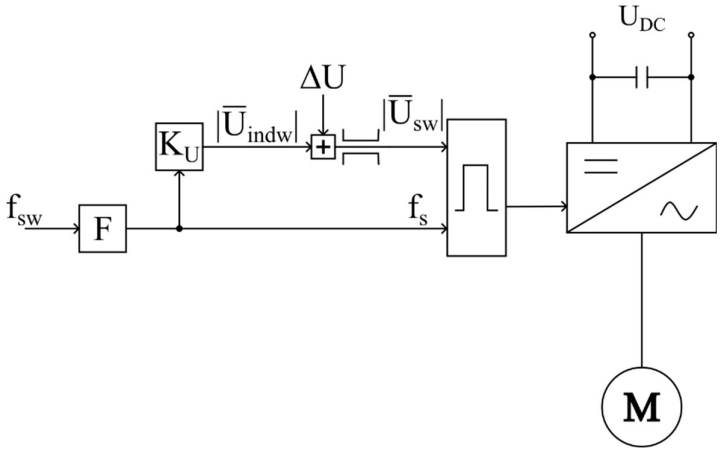
$$\begin{gathered}\Delta U = R_s \cdot \frac{U_{ind}}{\frac{R_r}{s}} = R_s \cdot \frac{U_{ind}}{R_r \cdot f_s} \cdot f_r \cong \frac{(U_{sN})_{ef} \cdot \sqrt{2} \cdot R_s}{R_r \cdot f_{sN}} \cdot f_r \\ \rightarrow \Delta U \cong K_{fr} \cdot f_r \;\; ; K_{fr} = \frac{(U_{sN})_{ef} \cdot \sqrt{2} \cdot R_s}{R_r \cdot f_{sN}},\end{gathered} \tag{24}$$

kde U_{sN} odpovídá jmenovitému napětí stroje. [12]

Další úbytky představují napěťové úbytky na polovodičových součástkách a vlivem mrtvých časů, avšak tato závislost je nelineární funkce a kompenzuje se velmi obtížně. [12]

Skalární řízení bez čidla otáček

Je zde použit výše popsaný princip zadávání statorového napětí a frekvence s tím, že při nižší statorové frekvence než je jmenovitá je udržován konstantní magnetický tok a při frekvencích vyšších než jmenovitých se napětí udržuje konstantní a motor se odbuzuje. Blok F funguje jako filtr, respektive jako rampa, zajišťující dostatečně pomalý nárust statorové frekvence tak, aby motor pracoval v lineární části momentové charakteristiky. Implementace tohoto filtru není pro funkčnost nutná, avšak je doporučené filtr použít. Statorová frekvence je pak zadávána do generátoru spínacích pulzů a zároveň přenásobena konstantou K_U podle vztahu (22) a následně přičten člen ΔU, který ovšem není možné vypočítat dle vztahu (24), jelikož neznáme rotorovou frekvenci. Proto je nutné ΔU vhodně odhadnou ze statorové frekvence, kde vycházíme z toho, že velikost tohoto napětí je nejvyšší když $f_s \rightarrow 0$ a při jmenovité frekvenci je vliv tohoto napětí téměř nulový. Následně je statorové napětí omezeno na vhodnou hodnotu U_{sw} a zadávané do generátoru spínacích pulzů, tedy PWM. Strukturální schéma je znázorněno obrázkem níže. [12]

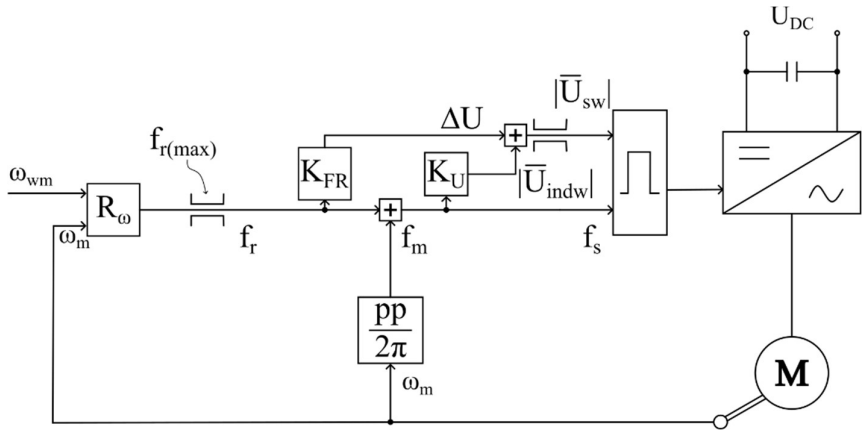


Obr. 16: Strukturální schéma skalárního řízení bez čidla otáček

Skalární řízení s čidlem otáček

V případě pohonu s čidlem otáček odpadá blok filtru (F), který je nahrazen regulátorem zadávající rotorový kmitočet dle rozdílu požadované a aktuální rychlosti získané právě čidlem otáček, jehož výstup je vhodné filtrovat. Tuto hodnotu rotorové frekvence je nutné vhodně omezit tak, aby, stejně jako v předchozím případě, motor pracoval v lineární část momentové charakteristiky. K takto získané rotorové frekvenci je nutné následně přičíst mechanickou frekvenci získanou přenásobení mechanické rychlosti konstantou $\frac{p_p}{2\pi}$. Výstupem je požadovaná statorová frekvence f_s zadávaná do PWM modulu jakožto část nutná pro výpočet referenčního signálu. Z této f_s se stejně jako v případě pohonu bez čidla

pomocí konstanty K_U vypočítává požadavek na napětí. Opět je zde potřeba přičíst člen ΔU, který je tentokrát již možné vypočítat dle vztahu (24), jelikož v tomto případě již je rotorová frekvence f_r známá. Výstupní součet těchto napětí je opět třeba omezit na vhodnou hodnotu vycházející z jmenovitého napětí použitého stroje, potažmo z velikosti napětí stejnosměrné meziobvodu U_{DC}. Schéma tohoto pohonu je znázorněno níže. [12]



Obr. 17: Strukturální schéma skalárního řízení s čidlem otáček

1.5.2 Vektorové řízení

Značně dynamičtější metodou je vektorové řízení zakládající se na rozdělení napájecího proudu na d a q složku řídící separátně moment a budící tok. K fungování je zapotřebí matematického modelu řízeného stroje díky kterému získáváme informaci o úhlu natočení magnetického toku a z něj dále pomocí Clarkeovy a Parkovy transformace dostáváme informaci, které tranzistory v měniči sepnout. Velkou výhodou tohoto řízení, kromě již zmíněné dynamičnosti, je také možnost řízení účiníku. Navíc při správném nastavení lze takto řídit i synchronní stroje bez rizika vypadnutí ze synchronismu. [1]

Principem vektorového řízení je navázání proudů na rotující souřadný systém tak, aby se regulované veličiny, konkrétně v tomto případě proudy, jevili jako dvě stejnosměrné složky a bylo tak možné k jejich regulaci použít klasické proporčně-integrační regulátory. Do každého tohoto regulátoru se zadávají požadavky na tento proud. Proud I_d ovlivňuje moment na hřídeli, jedná se tedy o takzvanou momentotvornou složku. Naopak proud I_q ovlivňuje nabuzení motoru a tím i jeho otáčky a proto se nazývá tokotvornou složkou. Každá tato složka disponuje vlastním regulátorem. Výstupem je napětí ΔU_{sdw}, respektive ΔU_{sqw}. Tyto napětí se sčítají s výstupními předpočítanými napětími U_{sd0} a U_{sq} z dopředného modelu ulehčující regulátorům proudu. Zároveň dopředný model realizuje takzvanou křižnou vazbu mezi d a q složkou, tedy jak se vzájemně tyto složky ovlivňují. Vstupem do dopředného

modelu požadavky na proudy I_{sdw} a I_{sqw} společně s mechanickou rychlostí ω_{mech} a požadavkem na rotorový tok ψ_{rw}, který je nejprve potřeba vypočítat z požadavku na proud I_{sdw}. Pro tento výpočet vycházíme z třetí řádky rovnice (16), kterou upravíme na následující tvar. [12]

$$\frac{d|\bar{\psi}|_r}{dt} = R_r \cdot \left(\frac{L_h}{L_r} \cdot i_{sd} - \frac{|\bar{\psi}|_r}{L_r}\right) \tag{25}$$

Z této rovnice pak můžeme vytknout $|\bar{\psi}|_r$ odpovídající našemu hledanému ψ_{rw}.

$$\psi_{rw} = L_h \cdot i_{sdw} - \frac{L_r}{R_r} \cdot \frac{d\Psi_{rw}}{dt}, \tag{26}$$

kde derivace v případě ručního zadávání požadavku má minimální vliv a je možné jí tak zanedbat. Toto je navíc podpořeno tím, že výpočet derivace toku je při realizaci obtížný a tak je výsledná použitá rovnice zjednodušena na tvar popsaný níže. [12]

$$\psi_{rw} = L_h \cdot i_{sdw} \tag{27}$$

Rovnice samotného dopředného modelu vychází z prvního a druhého řádku rovnice (15). Předpokladem je, že v ustáleném stavu platí přibližná rovnost indukčností $L_h \cong L_r$. Pak mají rovnice tvar následující.

$$\begin{aligned} u_{sd} &\cong R_s \cdot i_{sd} - \omega_{s\psi} \cdot (L_{s\sigma} + L_{r\sigma}) \cdot i_{sq} \cong U_{sd0} \\ u_{sq} &\cong R_s \cdot i_{sq} - \omega_{s\psi} \cdot (L_{s\sigma} + L_{r\sigma}) \cdot i_{sd} + \omega_{s\psi} \cdot \left|\bar{\psi}_r\right| \cong U_{sq0}, \end{aligned} \tag{28}$$

kde platí, že $\omega_{s\psi} = \omega_{r\psi} + p_p \cdot \omega_{mech}$. Zároveň je zde vidět zmíněná křižná vazba, kdy do napětí U_{sd0} vstupuje prostřednictvím rozptylových indukčností proud i_{sq} a stejným způsobem do napětí U_{sq0} vstupuje proud i_{sd}. [12]

Dříve zmíněným součtem složek napětí finálně dostáváme požadavky na napětí U_{sdw} a U_{sqw}, které nejprve pomocí inverzní Parkovy transformace a následně inverzní transformace Clarkeové převedeme na jednotlivá fázová napětí U_{saw}, U_{sbw} a U_{scw} vstupující do PWM modulu za účelem generování spínacích pulzů. [12]

Jak bylo řečeno na začátku, pro fungování je zapotřebí matematického modelu jehož výstupy jsou složky proudu I_d a I_q společně s úhel natočení ϑ důležitý pro Parkovu transformaci. Jeho vstupy jsou mechanické otáčky ω_{mech} společně se změřenými fázovými proudy. Pro realizaci měření všech 3 fází proudu je možné volit úsporné řešení, kdy nám postačují pouze 2 čidla proudu a třetí proud lze dopočítat za pomocí vztahu

$$i_a + i_b + i_c = 0, \tag{29}$$

který vychází z prvního Kirchhoffova zákona. Pro výpočet samotného matematického modelu následuje převedení těchto změřených proudů na proudy I_d a I_q za pomocí transformace Clarkeové a Parkovy transformace. Ze znalosti těchto proudů lze již vypočítat

z třetího řádku rovnice (16) vytknutím ψ_{rd} a integrací tak, aby byl výpočet realizovatelný v mikrokontroleru zmíněný magnetický tok, který je popsán následující rovnicí:

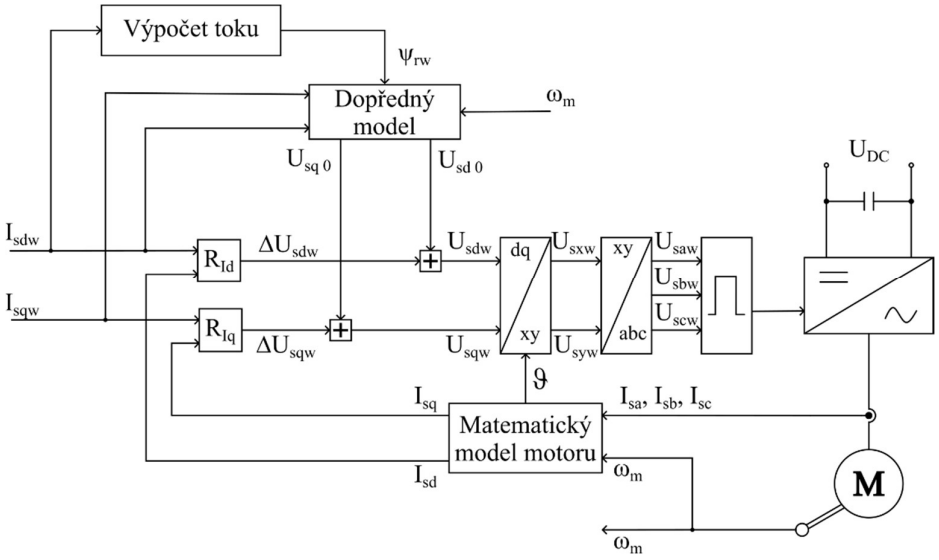
$$\psi_{rd} = \psi_{rd} - \frac{R_r}{L_r} \cdot (\psi_{rd} - L_h \cdot i_{sd}) \cdot dt, \tag{29}$$

kde dt značí délku výpočetního kroku. Následuje výpočet dle vztahu (19) a (20), ze kterých získáváme rychlost pole rotoru a statoru. Posledním krokem je výpočet samotného úhlu natočení ϑ vycházejícího ze vztahu (21), tedy integrace v čase, kterou můžeme opět zdiskretizovat na tvar

$$\vartheta = \vartheta + \omega_{s\psi} \cdot dt, \tag{30}$$

který je opět důležitý pro realizaci v mikrokontroleru. [12]

Strukturální schéma popsaného pohonu je znázorněno na obrázku níže.



Obr. 18: Strukturální schéma vektorového řízení

1.6 Mrtvé časy

Jak bylo již zmíněno v kapitole 1.1 Výkonové spínací prvky, ideální spínatelná součástka dokáže spínat a rozpínat okamžitě, kdežto u reálných součástek sepnutí a rozepnutí trvá v řádu jednotek ns až desítek μs. Kapitola 1.2 Střídač napětí také hovořila o zakázaných spínacích kombinacích, které by vedly ke zkratu DC meziobvodu. Těmito kombinacemi jsou veškeré případy, kdy by došlo k sepnutí prvků přímo nad sebou a tím propojení kladné a záporné svorky DC meziobvodu, tím tvorbě zkratu, který by byl omezen pouze odporem kanálů výkonových součástek. K tomu může dojít i po velmi krátký čas, kdy jednu součástku vypínáme a druhou zapínáme a to ve stejný čas. Dojde tedy k tomu, že jedna z dvojice je už přiotevřená a druhá se ještě nestihla zavřít. [13]

Řešením je vkládání takzvaných mrtvých časů. Cílem je dám vypínající se výkonové součástce dostatek času na její uzavření než začneme otevírat její protějšek. Prakticky je toto realizované zpoždění náběžné hrany spínacího signálu. To může být realizováno buď softwarově, případně lze zajistit i pomocí hardwaru. Po dobu komutace, kdy jsou oba prvky vypnuté, proud přebírají zpětné diody. Nevýhodou této praktiky je deformace výstupního proudu, která se zvyšuje s délkou mrtvého času. Proto je nutné mrtvé časy volit vhodně tak, aby zamezily, i částečným, zkratům meziobvodu a zároveň nebyli příliš dlouhé k minimalizaci vlivu na tvar proudu zátěže. [13]

2 Praktická část

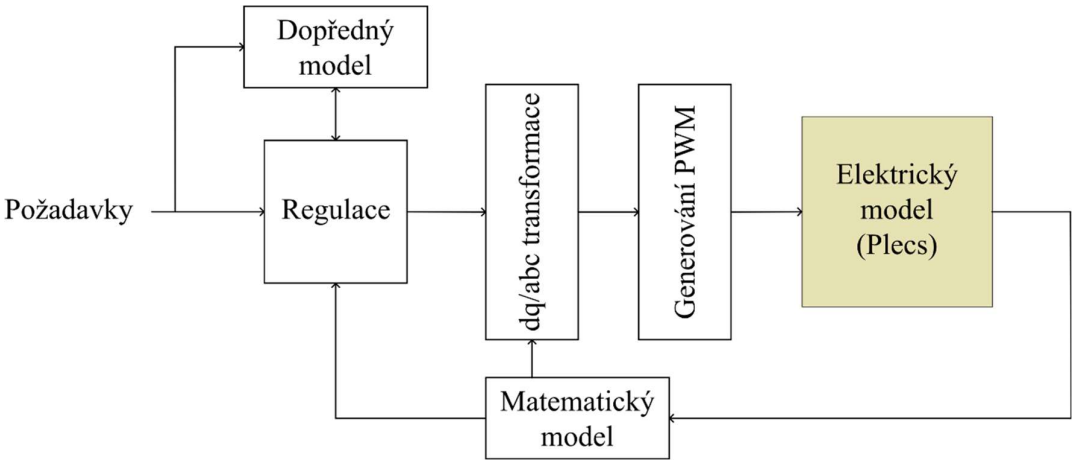
Jak již bylo řečeno v úvodu, cílem praktické části této práce je nejprve vytvořit simulaci vektorového řízení a následně realizovat toto řízení na skutečné zařízení s možností ovládání uživatelem. Důvodem provedení simulace je otestování řídícího algoritmu a správnosti matematických rovnic vedoucích na matematický model použitého asynchronního motoru, zároveň je tak možné i přibližný odhad nastavení regulátorů. Pro tento matematický model a také simulaci je třeba znát parametry motoru, které mi byli poskytnuty vedoucím práce a shrnuje je tabulka níže. Spínací frekvence f_{spin} je stanovena na 10 kHz.

Tab. 1.: Tabulka shrnující parametry použitého motoru

Parametr	Značení	Hodnota	Parametr	Značení	Hodnota
Odpor statoru	R_s	1,85 Ω	Hlavní indukčnost	L_m	33 mH
Odpor rotoru	R_r	1,53 Ω	Počet pól párů	pp	2
Rozptylová indukčnost statoru	L_{ss}	5,3 mH	Setrvačnost	J	0,01 kg·m^2
Rozptylová indukčnost rotoru	L_{rs}	4,3 mH			

2.1 Simulace vektorového řízení

K simulaci jsem použil MatLab s nadstavbou Simulink a rozšířením Plecs. Tento simulační model lze rozdělit na několik částí: Regulace, dopředný model, transformace dq/abc, matematický model, generátor PWM pulzů a samotný elektrický obvod.



Obr. 19: Blokové rozdělení modelu

Regulace

K realizaci regulátorů jsem použil blok MatLab function pro psaní standardního MatLab kódu. Regulátory jsou standardní PI s úpravou pro digitální implementaci. Základem pro realizaci integrační složky je proměnná typu *persistent*. Tento druh proměnné zaručuje, že

poslední hodnota zůstává uložena napříč jednotlivými kroky modelu a tak umožňuje postupné nasčítávání oné integrační složky. Jako příklad zde použiji regulátor rotorové frekvence. U něj je zavedena zpětná vazba od měření frekvence otáček rotoru a požadavek uživatele. Z těchto parametrů se počítá odchylka od požadované hodnoty *e*. Z odchylky, a nasčítané sumy integrační složky, se pomocí zesílení regulátoru vypočítává požadovaná rotorová frekvence. Pokud by tento požadavek přesáhl stanovené meze (jelikož se jedná o malý a měkký motor, byla zvolena maximální rotorová frekvence 5 Hz), omezí se výstup regulátoru na tuto hodnotu a vynuluje se derivace suma integrační složky. V případě, že požadavek je v mezích se pak vypočítá derivace sumy integrační složky podle odchylky *e*, zesílení a časové konstanty regulátoru, která se následně přenásobená délkou kroku *dt* přičte k celé sumě. Postup je obdobný i u regulátorů proudu.

```
function f_rot = PI_reg(f_w, f_mech, dt)

persistent i_sum;
if isempty(i_sum)
    i_sum = 0;
end

Kp = 2;
Ti = 0.15;
f_max = 5;

e = f_w - f_mech;

f_rot = i_sum + Kp * e;

if f_rot > f_max
    f_rot = f_max;
    d_sum = 0;
elseif f_rot < -f_max
    f_rot = -f_max;
    d_sum = 0;
else
    d_sum = e * Kp/Ti;
end

i_sum = i_sum + d_sum * dt;
```

Obr. 20: Kód regulátoru rotorové frekvence jako MatLab funkce

Dopředný model

Dopředný model, anglicky Feed Forward, předpočítává napětí d a q složky z požadavků na proudy a aktuální rychlosti, čímž usnadňuje práci regulátorům, které pak mohou být nastaveny na přesnější regulaci s menšími překmity. Navíc ještě zprostředkovává takzvanou křižnou vazbu, tedy jak se jednotlivé složky (d a q) vzájemně ovlivňují. Opět jsem k realizaci využil blok MatLab function. Dopředný model vychází z rovnic uvedených v kapitole 1.5.2 Vektorové řízení. Nejdříve je však potřeba vypočítat požadovaný rotorový tok, v simulaci značný jako Psi_r_w, vycházející z požadavku na proud v ose d a derivace tohoto požadavku. Derivaci je možné zanedbat, jelikož, co se velikosti týče, nemá značný vliv a navíc je požadavek na tento proud zadáván ručně. Výsledkem je pouhé přenásobení požadavku na proud hodnotou hlavní indukčnosti. Se znalostí tohoto požadavku je již možné aplikovat zmíněné rovnice pro dopředný model.

```
function [Uq_0, Ud_0] = FF(w_mech, Id_w, Iq_w, Rs, pp, Lss, Lrs, Lm)

Psi_r_w = Lm * Id_w;

Ud_0 = Rs * Id_w - pp * w_mech * (Lss + Lrs) * Iq_w;
Uq_0 = Rs * Iq_w + pp * w_mech * (Lss + Lrs) * Id_w + pp * w_mech * Psi_r_w;
```

Obr. 21: Kód dopředného modelu jako MatLab funkce

Matematický model

Matematický model, opět realizován jako blok MatLab function, vychází z rovnic uvedených v kapitole 1.4.1 Náhradní schéma a matematický model asynchronního motoru upravených tak, aby bylo možné výpočet realizovat i v prostředí MatLabu. To znamená aplikace dvou proměnných typu *persistent* a to konkrétně pro proměnou *thehtas*, sloužící jako paměť pro uložení předchozího úhlu natočení, a *Psirs*, také sloužící jako paměť pro uložení předchozí hodnoty magnetického toku rotoru. Pro funkčnost modelu je třeba zavést měření proudů jednotlivými fázemi a zpětnou vazbu v podobě informace z čidla otáček společně s parametry motoru, které jsou uložené v podobě MatLab Script se jménem IMparam. Prvním krokem ve výpočtech je pomocí Clarkeové transformace převést proudy z abc souřadnic do αβ a následně pomocí Parkovy transformace do dq souřadnicového systému. Z d-složky proudu, délky časového kroku, indukčnosti s odporem rotoru a předchozí hodnotou rotorového toku můžeme jeho hodnotu aktualizovat. Jednoduchou if podmínkou je zamezeno, aby, například hned v prvním kroku simulace, došlo k dělení nulou. Takže převrácená hodnota rotorového toku je z vlastní hodnoty rotorového toku počítána pouze v případě, že rotorový tok nabývá hodnoty vyšší než 0.01 Wb, jinak je stanovena na konstantu. Dále se počítá rychlost rotorového a statorového pole, ze kterého se dále počítá úhel natočení ϑ. Na konec se pouze omezí rozsah úhlu natočení na $\pm\pi$.

```
12  Ialpha = sqrt(2/3) * (I_s(1) - 1/2 * I_s(2) - 1/2 * I_s(3));
13  Ibeta = sqrt(2/3) * (sqrt(3)/2 * I_s(2) - sqrt(3)/2 * I_s(3));
14
15  I_AlBe = [Ialpha; Ibeta];
16
17  Idq = [cos(thetas) sin(thetas); -sin(thetas) cos(thetas) ] * I_AlBe(1:2, 1);
18
19  Psirs = Psirs - Rr * dt/Lr *(Psirs - Lm * Idq(1));
20
21  if Psirs > 0.01
22      invPsirs = 1/Psirs;
23  else
24      invPsirs = 1/0.01;
25  end
26
27  w_rot = Rr * Lm/Lr * invPsirs * Idq(2);
28  w_s = w_rot + w_mech * pp;
29
30  thetas = thetas + w_s * dt;
```

Obr. 22: Útržek kódu z matematického modelu

Transformace dq/abc

Transformace z dq souřadného systému je nutná pro vytvoření PWM pulzů pro jednotlivé fáze. Zde je třeba znát úhel natočení ϑ a požadavky z nadřazených regulátorů. Zde se jedná

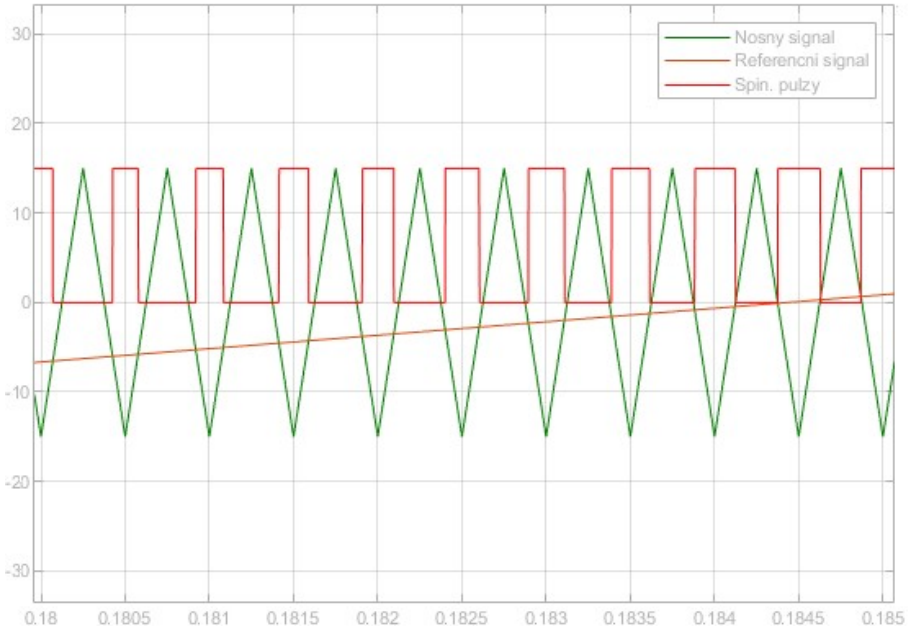
o standardní inverzní Parkovu transformaci a transformaci Clarkeové sloučených do jedné matice, respektive tří rovnic. Signály jsou na výstupu spojeny a vedeny jako jeden z důvodu úspory spojů a tím větší přehlednosti.

```
function U_s = dq_abc(Ud, Uq, theta)

U_u = sqrt(2/3) * (Ud * cos(theta) - Uq * sin(theta));
U_v = sqrt(2/3) * (Ud * cos(theta - 2*pi/3) - Uq * sin(theta - 2*pi/3));
U_w = sqrt(2/3) * (Ud * cos(theta + 2*pi/3) - Uq * sin(theta + 2*pi/3));

U_s = [U_u, U_v, U_w];
```

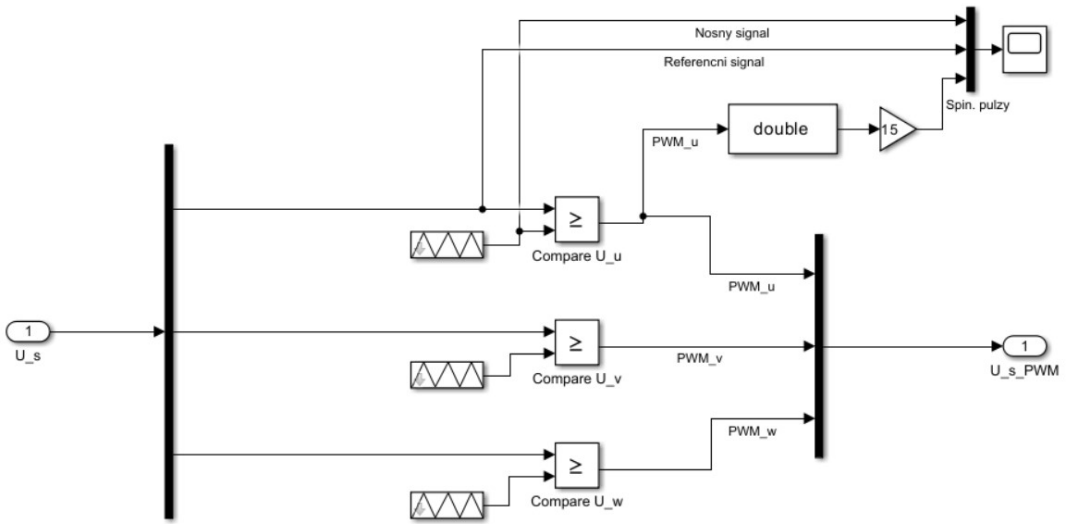
Obr. 23: Transformace dq/abc s použitím úhlu natočení theta (9)

Generování PWM pulzů

Generování PWM pulzů jsem udělal pro přehlednost jako samostatný subsystém, kde se porovnávají jednotlivé požadavky na napětí s trojúhelníkovým signálem s frekvencí 10 kHz odpovídající zadání a amplitudou $\pm U_{DC}/2$. Na obrázku níže byla velikost spínacích pulzů přenásobená hodnotou $U_{DC}/2$ pro lepší viditelnost v grafu. Prakticky spínací pulzy nabývají hodnoty pouze 0 a 1.

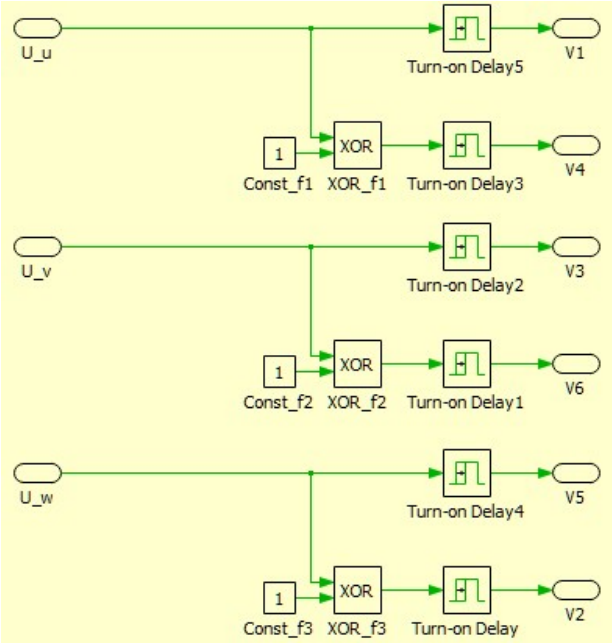


Obr. 24: Znázornění tvorby spínacích pulzů



Obr. 25: Vnitřek bloku subsystému PWM_gen

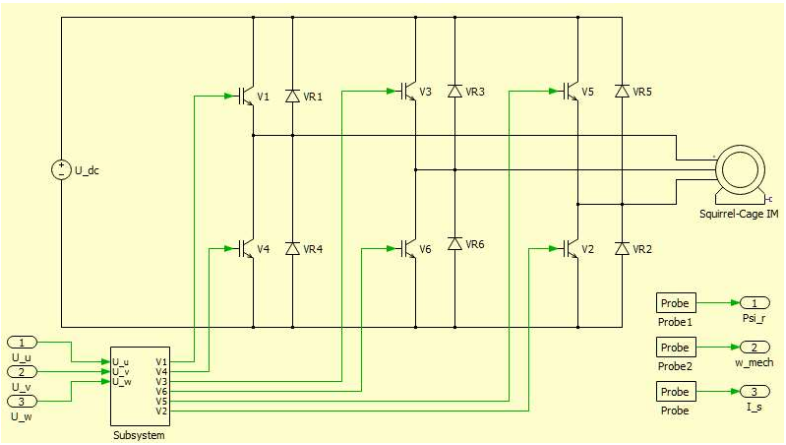
Ke generování spínacích pulzů patří i implementace mrtvých časů. Tu jsem se rozhodl implementovat až téměř analogově přímo v rozšíření Plecs pomocí bloků zpožďujících náběžnou hranu o definovaný čas. Tyto bloky jsem zahrnul do subsystému, který zároveň rozděluje spínací pulzu mezi jednotlivé tranzistory ve větvi a zajišťuje, že nedojde k větvovému zkratu. To jsem realizoval pomocí exkluzivního or (XOR) bloku do kterého jsem zavedl spínací pulzy z generátoru PWM pulzů a pak konstantu o hodnotě 1. Horní tranzistor je tedy přímo ovládán spínacími pulzy a druhý přes tento blok. V případě, že aktuální pulz nabývá hodnoty 1, tak dvě jedničky na vstupu XOR dají na výstupu logickou nulu. Naopak v případě, že spínací pulz je zrovna v hodnotě 0, tak nula a jednička na vstupu XORu způsobí logickou jedničku na výstupu. Díky této logice je vždy sepnut pouze jeden z dvojice tranzistorů.



Obr. 26: Realizace mrtvých časů a střídaní dvojic tranzistorů

Elektrický obvod

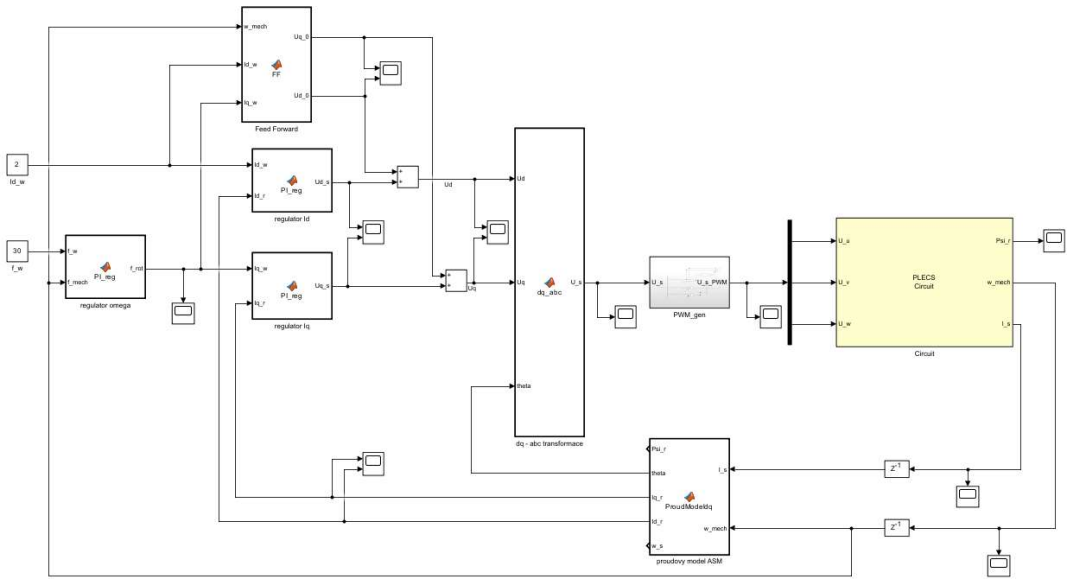
K tvorbě elektrického obvodu vhodně posloužilo rozšíření Plecs díky již implementovaných výkonovým prvkům a modelům točivých elektrických strojů. Elektrický obvod tvoří vestavěný model asynchronního motoru s kotvou nakrátko a tranzistory se svými zpětnými diodami. Nachází se zde také subsystém, který realizuje zmíněné mrtvé časy a střídaní tranzistorů, a také sondy (Probe), které slouží jako zpětná vazba. Konkrétně poskytují informaci o proudech jednotlivých fází, frekvenci otáček motoru a velikost magnetického toku, která je ovšem pouze informativní a pro fungování není potřebná. Napájení zde představuje klasický stejnosměrný zdroj napětí parametricky nastavený na 30 V.



Obr. 27: Elektrický obvod sestavený v Plecs

Celkový model

Celkový model vzniknul sestavením jednotlivých výše popisovaných částí. Posloupnost je taková, že na vstupu jsou zadávané požadavky. V případě požadavku na frekvenci otáček probíhá zpracovaní v regulátoru rychlosti jehož výstupem je požadavek na q složku proudu. Požadavek na d složku proudu je zadávána uživatelem. Oba tyto požadavky jsou vstupem do dopředného modelu a individuálních regulátorů proudů, jejichž výstupem je požadavek na napětí, stejně tak jako u dopředného modelu. Výstupy těchto bloků se podle složek patřičně sečtou a postupují do bloku transformace z dq souřadného systému do 3-fázového abc systému z jehož výsledku se utváří spínací pulzy vedené na výkonové prvky pohánějící asynchronní motor, který poskytuje pomocí sond informace matematickému modelu zprostředkovávající zpětnou vazbu regulátorům a estimovaný úhel natočení nutný pro transformaci. V modelu jsou za každým krokem vložené osciloskopy pro snadné sledování dějů probíhajících v jednotlivých blocích. Tento model je znázorněn na obrázku níže.

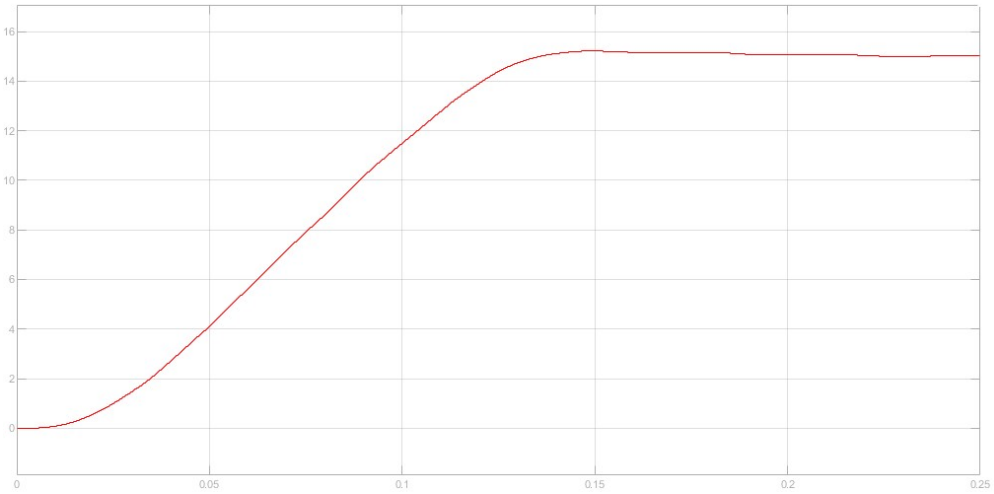


Obr. 28: Celý model vektorového řízení v Simulinku

Výsledky simulace

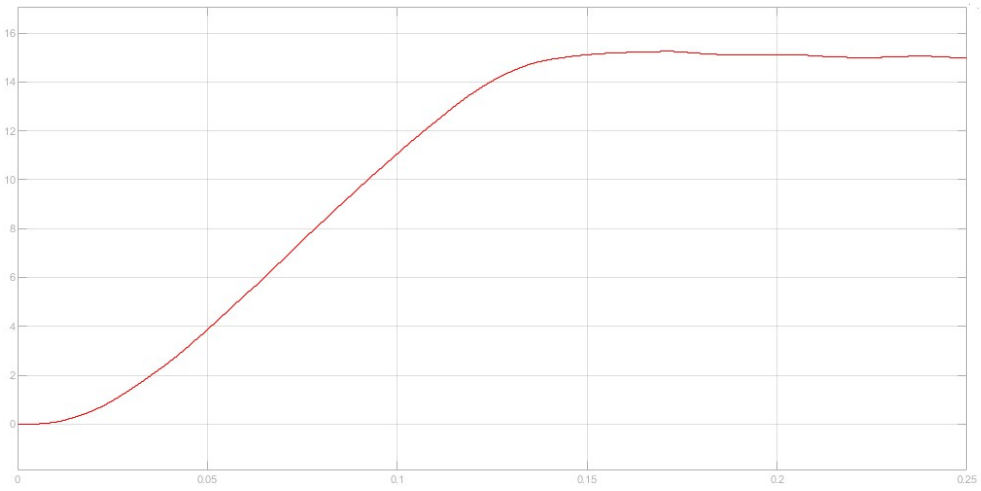
Simulaci jsem testoval ve dvou režimech. V prvním případě jsem výstup z regulátoru rotorové frekvence převedl na vstup obou regulátorů proudu. V tomto případě uživatel zadává pouze požadavek na rychlost a zbytek již regulátory vyřeší sami. V druhém případě je požadavek na d složku proudu zadáván ručně, tedy požadavek na moment. O zadávání požadavku na tokotvorný proud se, stejně jako v předchozím případě, stará regulátor rotorové rychlosti. Obě tyto simulace budou dále požity pro porovnání s reálným pohonem.

Pro zkoušku jsem zvolil frekvenci otáček odpovídající 15 Hz na statoru, odpovídajících 900 ot/min. Zde je vidět, že se otáčky ustálí do 0,17 s od rozběhu.



Obr. 29: Simulace rozběhu se zadáním rychlosti otáček

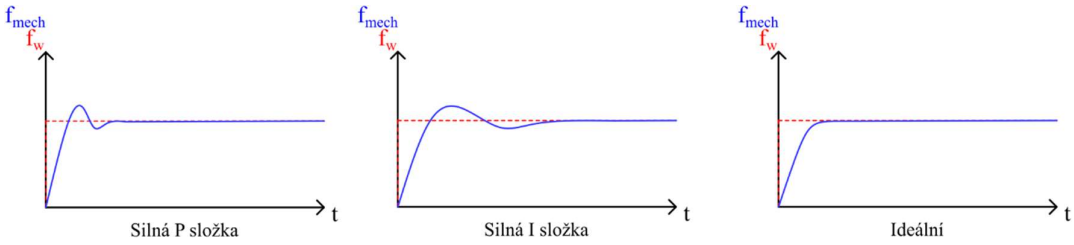
V druhém případě jsem ponechal stejný požadavek na frekvenci stejný na 15 Hz a nově zadával i požadavek na d složku proudu, který jsem zvolil 0,7 A. Z průběhů je vidět, že rozběh je velmi podobný avšak v prvním případě byl o trochu pomalejší a k ustálení došlo až v čase 0,2 s, tedy o přibližně 0,02 s déle. Přisoudit to můžeme tomu, že pro dynamické stavy není vhodné nechával požadavek na momentotvorný proud konstantní a přechodové děje jsou rychlejší, když jeho výpočet necháme na regulátorech.



Obr. 30: Simulace rozběhu se zadáním rychlosti otáček a d složky proudu

Ladění regulátorů

Ladění regulátorů jsem dělal experimentálně. Pro ladění regulátorů jsem vycházel z naměřených hodnot a jejich průběhů. Z tohoto průběhu pak je možné určit jak upravit jednotlivé složky regulátoru. Jako příklad zde můžeme použít rozběh motoru, respektive skokovou změnu požadavku na otáčky f_w a jak na tento požadavek reaguje mechanická frekvence f_{mech}. Podle doby ustálení lze určit, zda je třeba například potřeba zeslabit P složku, I složku, případně zesílit a podobně. Ukázky jak vypadají teoretické výstupní hodnoty vůči požadavku při silných nastaveních regulátoru a při ideálním nastavení popisuje obrázek níže.

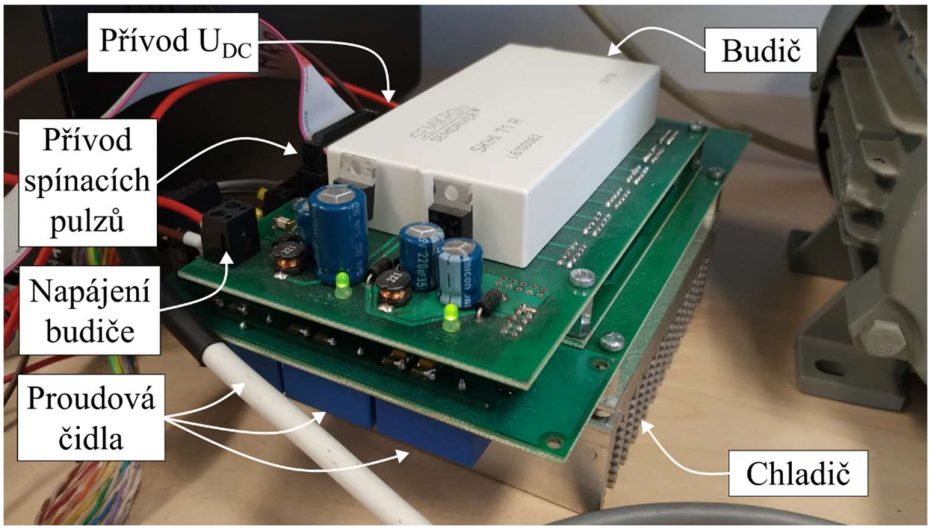


Obr. 31: Teoretické průběhy výstupních hodnot při různých nastaveních regulátoru

Teorie ladění regulátorů je velmi rozsáhlou problematikou a cílem této práce není její zkoumání a popis, pouze je zde v krátkosti popsán postup vyhodnocování výstupních dat a jejich zpracování při experimentálním ladění použitých regulátorů.

2.2 Praktická realizace řízení

Pro praktickou realizaci jsem využil onoho laboratorního prototypu střídače využívající součástky výrobce SEMIKRON osazeného budičem stejného výrobce společně se třemi proudovými čidly. Budič měniče je napájen 24 V DC s pomocí laboratorního zdroje DIAMENTRAL L240R51D. Stejně tak samotné napětí pro stejnosměrný meziobvod je napájeno z tohoto zdroje, který pro účely vývoje a testování algoritmů z důvodu bezpečnosti poskytuje pouze 30 V. Samotný měnič je chlazen pasivním vzduchovým chladičem.

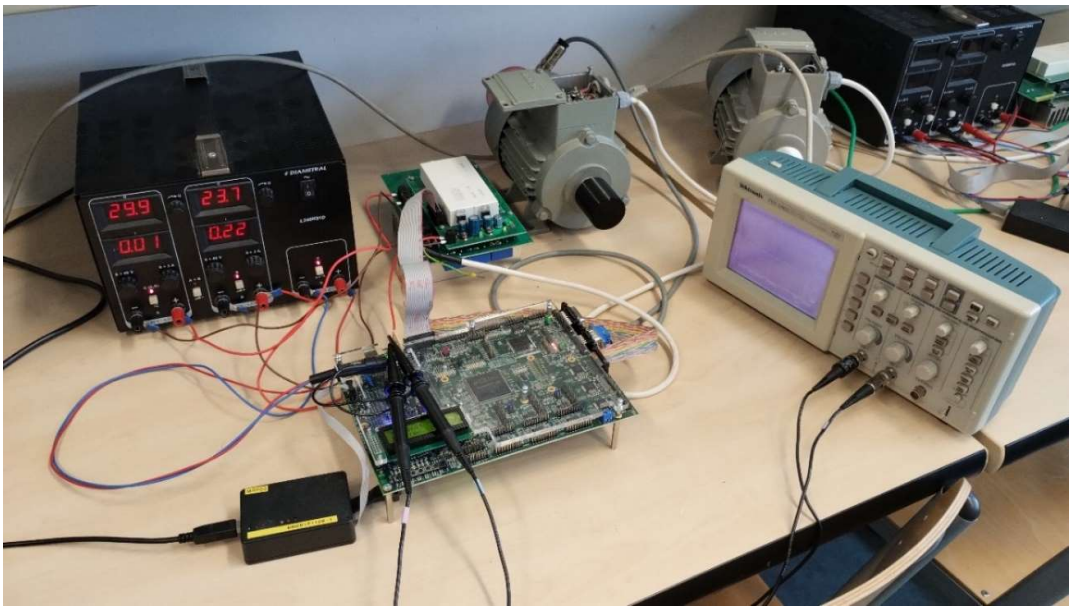


Obr. 32: Použitý měnič

Měnič napájí malý asynchronní motor o jmenovitém výkonu 0,25 kW s jmenovitým napětím 83 V při zapojení do hvězdy a frekvencí 50 Hz. Jmenovité otáčky jsou stanoveny na 1350 ot/min, což při přepočtu na frekvenci odpovídá 22,5 Hz. Z toho vyplývá, že se jedná o motor s 2 pól-páry. Motor je navíc osazen inkrementálním čidlem otáček (QEP).

Pro řízení tohoto měniče jsem použil kontrolér MLC vyvinutý, společně s patřičnou knihovnou pro jeho používání, panem Ing. Tomášem Košanem, PhD. pro výzkumné a vývojové účely na Západočeské univerzitě v Plzni. Nejdůležitějšími periferiemi pro tuto aplikaci pro mne jsou ePWM moduly zprostředkovávající generování PWM signálu pro spínaní výkonových prvků. Vzhledem k tomu, že střídač je třífázový, musel jsem využít 3 tyto moduly. Další velmi důležitou periferií jsou A/D převodníky pomocí kterých jsem realizoval měření napětí ve stejnosměrném meziobvodu a fázových proudů. Následně jsem využil CAN periferie pro účely komunikace s uživatelským rozhraním běžícím na počítači, který je ke zmíněné CAN sběrnici připojen pomocí adaptéru na USB. Pro diagnostické účely a sledování jednotlivých veličin v průběhu testování jsem také využil D/A převodníků tak, aby bylo možné data zobrazit a sledovat jejich průběh a chování na osciloskopu. Výhodou

tohoto kontroléru je také přítomnost floating-point jednotky, která umožňuje programovaní s formátem dat s plovoucí řádovou čárkou (*float*) bez významné ztráty rychlosti výpočtu. Pro napájení je opět použit výše zmíněný laboratorní zdroj, konkrétně jeho 5 V výstup. Pro nahrávání programu do kontroléru je používáno JTAG rozhraní.



Obr. 33: Fotografie použité sestavy pro realizaci řízení

Detailnější fotografie jednotlivých komponent této sestavy jsou vloženy v příloze A včetně štítku použitého asynchronního motoru.

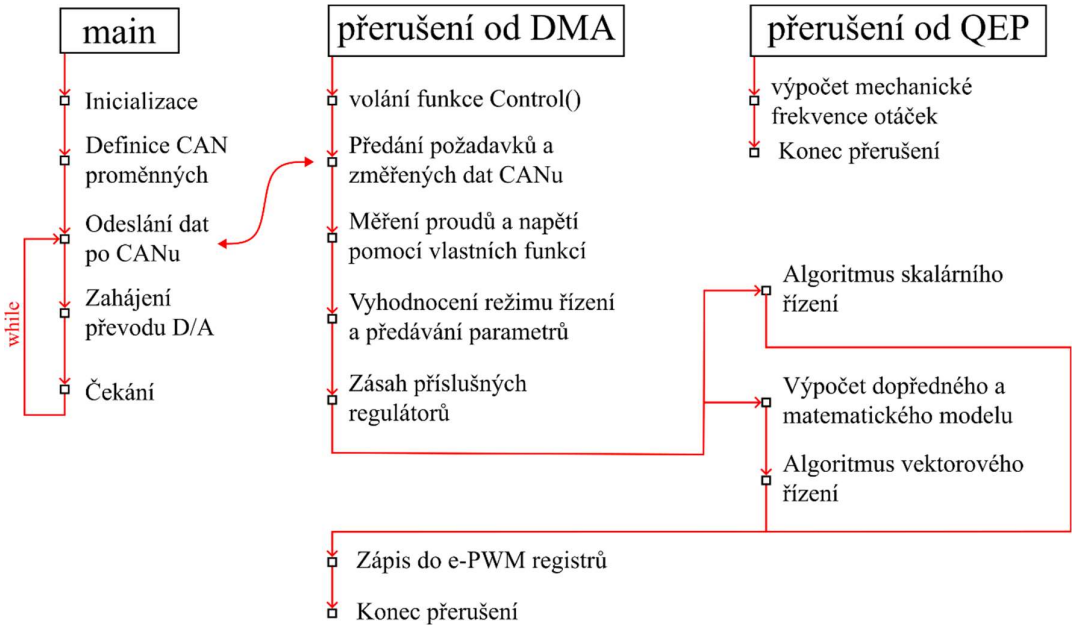
2.2.1 Popis kódu a funkčnosti programu

K psaní kódu jsem využil programovacího jazyku C psaného v Code Composer Studio s rozšířením o obecnou knihovnu “math.h“ kvůli zahrnutí některých matematických funkcí jako je sinus, odmocnina a další společně s knihovnou “MLC_drv.h“. Tento kód lze rozdělit na několik sekcí:

- Vyhodnocení napětí, proudů a otáček
- Vyhodnocení režimu řízení
- Algoritmy řízení
- Zadávání parametrů motoru
- Odesílání dat pomocí CAN

Vykonávání instrukcí je řešené v rámci přerušení od vhodných periferií systémem rozdělení dílčích částí kódu do volaných funkcí pro dobrou čitelnost kódu. Celý kód, včetně komentářů vysvětlující jednotlivé řádky, je k dispozici k nahlédnutí jako příloha B.

Struktura kódu a jeho posloupnost lze popsat obrázkem 34.

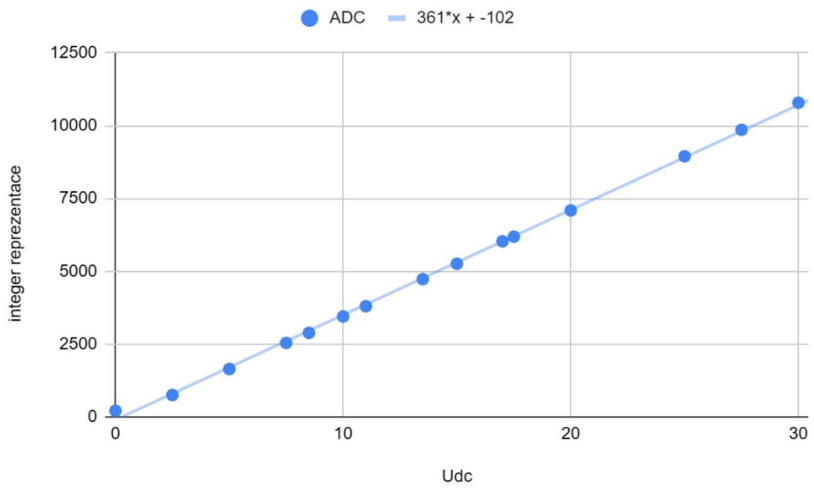


Obr. 34: Schéma posloupnosti kódu

Vyhodnocení napětí, proudů a otáček

K vyhodnocení napětí a proudů jsem použil třech A/D převodníků, které jsou navázané na přenos pomocí DMA (direct memory access – funkce umožňující přenos dat bez účasti procesoru). Tento proces je inicializován přerušením od A/D převodníku signalizující dokončení převodu, kdy DMA započne přenos těchto dat, který následně také iniciuje přerušení signalizující dokončení přenosu. V rutině tohoto přerušení je pak následně volána funkce *Control()*, jejíž úkolem je, mimo jiné, volání funkcí *Get_Udc()* získávající napětí stejnosměrného meziobvodu a funkce *Get_Iout()* obstarávající měření a dopočítávání proudů. Všechny tyto hodnoty jsou pak ukládány do struktury MEAS s definovaným jménem vals (jakožto measured values – změřené hodnoty), což usnadňuje přístup k těmto datům a ulehčuje práci s těmito daty.

Jelikož výsledky převodu A/D převodníků jsou uloženy v jejich registru, je třeba k tomuto registru přistupovat pomocí ukazatele na tuto strukturu. Z nahlédnutí do tohoto registru s pomocí debugovacího režimu jsem zjistil, že měření DC-linku se nachází na kanále 4, avšak toto číslo přímo neodpovídá napětí, nýbrž ho popisuje pomocí integerového čísla, které je teprve potřeba převést na reálné. Toto jsem provedl pomocí jednoduché lineární aproximace s použitím běžné Excel tabulky, kdy jsem postupně zvyšoval napětí na zdroji a toto napětí si zapisoval společně s jeho integerovou reprezentací. Graf znázorňující naměřená data a proložení aproximační křivkou znázorňuje obrázek 35.



Obr. 35: Aproximace napětí DC-linku

Se znalostí těchto konstant je pak možné je zavést jako definované hodnoty (*#define*) a použít pro výpočet vlastního napětí, viz. obrázek níže.

```
235 // funkce pro ziskani a vypocet napeti DC-linku
236 void Get_Udc(MEAS* vals)
237 {
238
239     // prenos vysledku prevodu AD prevodniku pomoci DMA
240     volatile int16_t* DMA_result = MLC_ADC1_get_res_ptr();
241
242     // mereni DC linku je na 4 kanalu -> pointer na 4. pozici
243     // konstanty ziskany linearni aproximaci
244     vals->U_dc = ((*(DMA_result + 4) - Udc_offset) / Udc_aprox);
245
246 }
```

Obr. 36: Kód funkce pro měření napětí stejnosměrného meziobvodu

Postup pro měření proudů je stejný s tím rozdílem, že jsou měřené pouze dva proudy ze tří, a to na kanálech 2 a 0. Proud třetí se musí pak dopočítat podle rovnice (29). Navíc jsou v rámci této funkce rovnou vypočítávány proudy I_d a I_q kvůli zpětné vazbě uživateli. Tyto proudy jsou získány standardně pomocí transformace Clarkeové a Parkově transformaci.

```
250 // funkce pro ziskani a vypocet proudu fazi
251 void Get_Iout(MEAS *vals)
252 {
253     float I_alpha, I_beta;
254 
255     // prenos vysledku prevodu AD prevodniku pomoci DMA
256     volatile int16_t* DMA_result = MLC_ADC1_get_res_ptr();
257 
258     // mereni proudu 1. faze je na 2 kanalu -> pointer na 2. pozici
259     // konstanty ziskany linearni aproximaci
260     vals->iL1 = ((*(DMA_result + 2) - Iout_offset) / Iout_aprox);
261 
262     // mereni proudu 2. faze je na 0 kanalu -> pointer na 0. pozici
263     vals->iL2 = ((*(DMA_result) - Iout_offset) / Iout_aprox);
264 
265     // vypocet proudu treti faze
266     vals->iL3 = -(vals->iL1 + vals->iL2);
267 
268     // vypocet proudu I_aplha
269     // I_alpha = sqrt(2/3) * (Ia - 1/2 * Ib - 1/2 * Ic)
270     I_alpha = odmoc2_3 * (vals->iL1 - 0.5 * vals->iL2 - 0.5 * vals->iL3);
271 
272     // vypocet proudu I_beta
273     // I_beta = sqrt(2/3) * (sqrt(3)/2 * Ib - sqrt(3)/2 * Ic)
274     I_beta = odmoc2_3 *(odmoc3/2 * vals->iL2 - odmoc3/2 * vals->iL3);
275 
276     // vypocet proudu I_d
277     vals->Id = I_alpha * cos(Theta) + I_beta * sin(Theta);
278 
279     // vypocet proudu I_q
280     vals->Iq = -I_alpha * sin(Theta) + I_beta * cos(Theta);
281 }
```

Obr. 37: Kód funkce pro měření a výpočet fázových proudů

Vyhodnocování otáček disponuje svojí vlastní periferií využívanou pro čtení použitého inkrementálního čidla otáček, která generuje přerušení jednou za definovaný čas, který je určen během inicializace. Toto generování přerušení je zároveň spojené se zápisem počtu nasčítaných pulzů za dobu mezi přerušeními do příslušného registru. Opět je potřeba počet těchto pulzů převést na samotné otáčky. K tomu je potřeba znát počet pól-párů a konstanty čidla vypočítanou jako

$$K_{QEP} = \frac{1}{4 \cdot IRC \cdot T_{vz}}, \tag{31}$$

kde *IRC* odpovídá počtu pulzů za jednu otáčku a T_{vz} odpovídá vzorkovací periodě. Poté je již možné provést výpočet vlastních otáček jak to ukazuje útržek programu níže.

```
222 void Calc_Wmech()
223 {
224 
225     // vycteni poctu pulzu z registru
226     int32_t DCNT = (int32_t)(EQep1Regs.QPOSLAT);
227 
228     // vypocitani mechanicke frekvence pomoci napocitanych pulzu a konstanty cidla
229     PIr_speed.val_actual = -DCNT * K_qep * pp;
230 
231 }
```

Obr. 38: Výpočet vlastní rychlosti

Znaménko mínus je zde čistě pro účely, aby se zadávaný směr otáčení shodoval s měřených a regulace tak mohla fungovat správně. Na rozdíl od předchozím je hodnota otáček ukládána do struktury RGL se jménem PIr_speed (odpovídající Regulate – PI regulator of speed) pod veličinu val_actual značící aktuální hodnotu sledované veličiny. Důvodem je dosáhnutí univerzálnosti funkce PI regulátoru, která bude vysvětlena dále v textu.

Vyhodnocení režimu řízení

K volbě režimu řízení zde slouží dvě proměnné a to jest *MODE* a *CTRL*. První z dvojice značí výběr módu, respektive druhu, řízení a *CTRL* určuje rozsah v jaké uživatel řídí daný pohon, tedy jaké požadavky zadává.

Tab. 2.: Tabulka popisující režimy a druhy řízení pohonu

MODE	*CTRL*	Druh řízení	Zadávané veličiny
0	-	Zkrat zátěže	-
1	-	Skalární řízení	f_w
2	0	Vektorové řízení	f_w
2	1	Vektorové řízení	f_w, I_d
2	2	Vektorové řízení	I_d, I_q

Jak tabulka popisuje, je zřejmé, že pohon je možné řídit ve vícero režimech. Nultý mód značí stav, je zátěž připojena k zápornému pólu stejnosměrného meziobvodu, tudíž je vyzkratována a není na ní připojené žádné napětí. V tomto případě jsem si tento postup mohl dovolit, jelikož pracují s asynchronním motorem. Pokud by se jednalo o synchronní motor s permanentními magnety, pak by tento postup nebylo kvůli indukovanému napětí z magnetů provést a musel by se tento stav řešit jinak, například použitím tzv. Trip zones, které úplně zablokují spínací pulzy na výkonové prvky, případné úplné odpojení od periferie. V módu 1 se jedná o běžné skalární řízení s čidlem otáček ve zpětné vazbě, kde se zadává pouze požadavek na otáčky. Mód dva značí přechod na vektorové řízení, konkrétně s režimem kontroly 0 se z pohledu uživatele jeví obdobně jako předchozí stav a opět se zadává pouze požadovaná frekvence, respektive otáčky, kde požadavky na proudy řeší regulátor otáček. V režimu kontroly 1 je stále regulátor otáček aktivní a stará se již pouze o požadavek na q-složku proudu, proud I_d je zadáván uživatelem. V posledním režimu kontroly, režimu 2, je již regulátor otáček zcela vyřazen a obě složky proudu jsou přímo zadávány uživatelem.

Z pohledu kódu se jedná o běžný stavový automat vyhodnocující zmíněné dvě proměnné a na základě toho přiřazuje regulátorům příslušné požadavky a volá samotné funkce regulátoru společně s funkcí příslušného řízení. Navíc zde vstupuje faktor, že pro každý režim řízení je jiné vhodné nastavení regulátorů a tak se zadávané zesílení a časové konstanty násobí příslušnými korekčními faktory tak, aby se z pohledu uživatele zesílení při přechodu mezi jednotlivými režimy zdáli stále stejné.

```
115     if (MODE == 1)
116     {
117         // predani hodnot zesileni regulatoru s prisnusnym koeficientem
118         PIr_speed.Kr = Kp_f;
119         PIr_speed.Ti = Ti_f;
120 
121         // volani funkce PI regulatoru rychlosti
122         PI_regul(&PIr_speed);
123 
124         // volani funkce pro vypocet a zapis do registru pro U/f rizeni
125         Scalar(&vals, &PIr_speed);
126 
127     } else if (MODE == 2)
128     {
129         // volani proudoveho modelu pro vypocet slozek dq proudu a Thety
130         Proudovy_model(&PIr_speed, &vals);
131 
132         if (CTRL == 0)
133         {
134             // predani hodnot zesileni regulatoru s prisnusnym koeficientem
135             PIr_speed.Kr = Kp_f * k_kpf_0;
136             PIr_speed.Ti = Ti_f * k_tif_0;
137 
138             // volani funkce PI regulatoru rychlosti
139             PI_regul(&PIr_speed);
140 
141             // navazani I-reg. na w-reg.
142             PIr_Iq.val_wanted = PIr_speed.val_out;
143             PIr_Id.val_wanted = PIr_speed.val_out;
144         }
145         else if(CTRL == 1)
```

Obr. 39: Útržek z kódu ukazující stavový automat, který přiřazuje zesílení, požadavky a volá příslušné funkce

Pozn.: Volání funkce regulátorů proudů není vynechané, ale je až dále v kódu.

Algoritmy řízení

Kapitolu algoritmů řízení je kvůli přehlednosti rozděleno na následující podkapitoly:

- PI regulátory a jejich zobecnění
- Implementace skalárního řízení
- Implementace vektorového řízení
- Dopředný a matematický model

PI regulátory a jejich zobecnění

Jak již bylo naznačeno kapitole pojednávající o vyhodnocení otáček, kód PI regulátorů je stavěn tak, aby se pouze volal s parametrem v podobě ukazatele na strukturu týkající se veličiny, kterou je třeba regulovat. To je vidět například na řádku 122 obrázku 39. Pro dosažení univerzálnosti kódu je nejprve třeba definovat zmíněnou strukturu.

```
23 typedef struct RGL
24 {
25     float A_max;
26     float A_min;
27     float Kr;
28     float Ti;
29     float d_sum;
30     float sum;
31     float val_actual;
32     float val_wanted;
33     float val_out;
34 } RGL;
```

Obr. 40: Definice struktury RGL vhodnou pro regulátory

Součástí této struktury, jmenováno od shoda, je horní omezení výstupní hodnoty, dolní omezení výstupní hodnoty, zesílení proporcionální složky, časová konstanta, derivace sumy a suma vycházející z integrační části regulátoru, aktuální hodnota regulované veličiny, požadavek a výstupní veličina regulátoru. Vzhledem k tomu, že tyto hodnoty včetně omezení a zesílení jsou pro jednotlivé regulátory různé, tak je nutné je mít definované separátně právě pomocí této struktury.

Samotná PI regulátor je standardní v podobě pro digitální zpracování. Nejprve proběhne vyhodnocení regulační odchylky, tedy rozdíl požadované a aktuální hodnoty regulované veličiny, a následně součet sečtení se sumou a přenásobení této odchylky zesílením. Následuje kontrola zda nebyl překročen limit výstupní hodnoty. Pokud ano, tak se výstup omezí na definované maximum, potažmo minimum. Pokud omezení překročeno nebylo, proběhne výpočet derivace sumy a výpočet nové sumy integrací z předchozího kroku, z této derivace sumy a délky kroku.

```
285 // funkce pro regulaci pomoci proporcionalne integracniho regulatoru
286 void PI_regul(RGL *PIr)
287 {
288     float e;
289
290     // odchylka pozadovane a namerene hodnoty
291     e = PIr->val_wanted - PIr->val_actual;
292
293     // vypocet pozadovaneho vystupu
294     PIr->val_out = PIr->sum + PIr->Kr * e;
295
296     // omezeni maximalniho vystupu na definovane meze, nulovani integrace
297     if(PIr->val_out > PIr->A_max) {PIr->val_out = PIr->A_max; PIr->d_sum = 0;} else
298     if(PIr->val_out < PIr->A_min) {PIr->val_out = PIr->A_min; PIr->d_sum = 0;} else
299                     {PIr->d_sum = e * PIr->Kr / PIr->Ti;}
300
301     // vypocet nove sumy z predchozi + jeji derivace/zmena
302     PIr->sum = PIr->sum + PIr->d_sum * dt;
303 }
```

Obr. 41: Kód PI regulátoru pracující se strukturami

Struktura pro regulaci rychlosti se nazývá PIr_speed, pro regulaci proudu I_d PIr_Id a pro regulaci proudu I_q je PIr_Iq. Požadavky a parametry, jako je například zesílení, je třeba jim průběžně předávat. Příklad předávání je opět vidět na obrázku 39, řádek 142 a 143.

Implementace skalárního řízení

Algoritmus skalárního řízení je proveden pomocí funkce *Scalar()*. Na začátku je proveden výpočet poloviny napětí stejnosměrného meziobvodu a poloviny TBPRD registru (amplituda nosného signálu pro PWM) za účelem ušetření výpočetního času, jelikož jsou tyto hodnoty v kódu využity vícekrát. Následuje výpočet zadávané statorové frekvence jakožto součtu rotorové (z regulátoru) a mechanické frekvence, ze které se vypočte derivace ϑ (Thety) opět za pomocí délky kroku. Běžnou integrací pak získáváme novou hodnotu Thety, která je následně omezena na rozsah $\pm\pi$. Podle rovnic (22), (23) a (24) proběhne výpočet zadávaného statorového napětí U_s, které je dále normováno na $U_{DC}/2$ kvůli snadnému omezení, kterým se zajišťuje přechod do odbuzování a zároveň zamezuje přetečení komparačního registru ePWM. Následuje výpočet okamžitých hodnot napětí jednotlivých fází za pomocí Thety z funkce sinus s patřičnými posuny o 120°, respektive $\pm\frac{2\cdot\pi}{3}$. Tyto úhly jsou předpočítané ručně a zavedené jako definované konstanty pro urychlení výpočtů. Při zápisu do komparačního registru je třeba brát v potaz, že 0 odpovídá hodnotě TBPRD/2. O tuto hodnotu je tedy třeba provést odsazení a nanormovat rozsah hodnoty okamžitého fázového napětí na $\pm TBPRD/2$. Je důležité připomenout, že zde použitá hodnota TBPRD je již vydělena 2 na začátku funkce a proto toto dělení již není v zapisování do komparačních registrů zahrnuto.

```
// funkce realizujici skalarni rizeni
void Scalar(MEAS* vals, RGL* PIr_speed)
{

    float U_dc, TBPRD, d_U, f_s, U_L1, U_L2, U_L3, U_s, d_Theta;

    // vypocteni poloviny napeti DC-linku pro dalsi vypocty
    U_dc = vals->U_dc / 2;

    // vycteni TBPRD z registru, respektive periody pily, deleni 2 pro dalsi vypocty
    TBPRD = (EPwm1Regs.TBPRD) / 2;

    // vypocet pozadovane statorove frekvence
    f_s = PIr_speed->val_out + PIr_speed->val_actual;

    // vypocet derivace Thety
    d_Theta = f_s * 2 * PI * dt;

    // vypocet nove Thety
    Theta = Theta + d_Theta;

    // omezeni rozsahu Thety
    if(Theta >  PI) {Theta = Theta - 2 * PI;}
    if(Theta < -PI) {Theta = Theta + 2 * PI;}

    // priblizny vypocet ubytku napeti
    d_U = Kf * PIr_speed->val_out;

    // vypocet statoroveho napeti a normovani na Udc/2
    U_s = (Ku * f_s + d_U) / U_dc;

    // omezeni vystupni amplitudy na maximalne Udc/2
    if(U_s >  1) {U_s =  1;}

    // vypocet napeti jednotlivych fazi
    U_L1 = U_s * sin(Theta);
    U_L2 = U_s * sin(Theta - pi2_3);
    U_L3 = U_s * sin(Theta + pi2_3);

    // zapis od compare registru e-PWM
    EPwm1Regs.CMPA.half.CMPA = (uint16_t)(TBPRD + (TBPRD * U_L1));
    EPwm2Regs.CMPA.half.CMPA = (uint16_t)(TBPRD + (TBPRD * U_L2));
    EPwm3Regs.CMPA.half.CMPA = (uint16_t)(TBPRD + (TBPRD * U_L3));

}
```

Obr. 42: Kód algoritmu skalárního řízení

Implementace vektorového řízení

Algoritmus je opět realizován stylem vlastní funkce *Vector()* do které, oproti předchozí variantě vstupuje navíc kromě regulátorů a změřených hodnot dopředný model prostřednictvím struktury FF (odpovídající výrazu Feed Forward). Opět ihned na začátku funkce proběhne výpočet $U_{dc}/2$ a $TBPRD/2$ ze stejných důvodů jako tomu bylo v případě skalárního řízení. Další výpočet realizuje součet výstupních napětí z regulátorů proudu a napětí dopředného modelu. Může opět nastat situace, že vektor napětí překročí hodnotu poloviny stejnosměrného napětí a došlo by k přemodulování. Tento vektor napětí se vypočte jako druhá odmocnina součtu mocnin obou složek napětí. Aby byl zachován poměr U_{ds} a U_{qs}, jsou obě složky přenásobeny stejnou konstantou tak, aby vektor napětí přesně odpovídal $U_{dc}/2$. Opět následuje nanormování na polovinu napětí stejnosměrného meziobvodu pro snadné zapsaní do komparačních registrů e-PWM modulu. Fázová napětí jsou zde vypočtena

podle inverzní Parkovy transformace a transformace Clarkeové. Theta potřebná pro tuto transformaci je získávána z matematického modelu, jehož funkce se volá pouze v případě vektorového řízení v rámci obsluhování rutiny výběru druhu řízení jak je vidět na obrázku 38. Výpočet uvedený výše je použit pouze v případě skalárního řízení.

Dopředný a matematický model

Dopředný model je vcelku snadnou funkcí respektující rovnice (26), (27) a (28), jehož součástí je i výpočet požadavku na magnetický tok. Na začátku kódu je nejdříve proveden přepočet z Hz na rad^{-1} kvůli zachování validnosti rovnic a respektování obecné rovnice pro výpočet reaktancí cívek. Následuje estimace požadovaného toku a samotných složek napětí.

```
438 // funkce vypocitavajici dopredny model pro vektorove rizeni
439 void FeedForward(RGL *PIr_Id, RGL *PIr_Iq, RGL *PIr_speed, FF *Forward)
440 {
441     float w_stat, Psi_r_w;
442 
443     // vypocet w z f pro dalsi vypocty
444     w_stat = 2 * PI * PIr_speed->val_actual;
445 
446     // estimace pozadovaneho magnetickeho toku
447     Psi_r_w = L_m * PIr_Id->val_out;
448 
449     // predvypocet potrebnych napeti
450     Forward->Ud_0 = R_s * PIr_Id->val_out - w_stat * (L_rs + L_ss) * PIr_Iq->val_out;
451     Forward->Uq_0 = R_s * PIr_Iq->val_out + w_stat * (L_rs + L_ss) * PIr_Id->val_out + w_stat * Psi_r_w;
452 
453 }
```

Obr. 43: Kód funkce dopředného modelu

Matematický, konkrétně proudový, model motoru zprostředkovává především zmíněný výpočet úhlu natočení Theta. Model vychází z třetího řádku rovnice (16) popisující výpočet magnetického toku ψ_{rd}, v kódu značení jako Psi_rs. Tento tok se poté kontroluje, zda není příliš malý nebo dokonce nulový, což by způsobovalo problémy při dělení takto nízkou hodnotou až nulou, a proto se omezuje na danou minimální hodnotu. Při té příležitosti se předpočítává jeho převrácená hodnota. Následují výpočty, které jsou prakticky přepsané rovnice (19), (20) a (21), tedy výpočet rotorové a statorové rychlosti z jejíž znalosti konečně získáváme za pomoci integrace s délkou kroku úhel natočení Theta, který opětovně omezen na rozsah ±π.

```
399 // funkce pro vypocet proudoveho modelu a estimaci uhlu natoceni
400 void Proudovy_model(RGL* PIr_speed, MEAS* vals)
401 {
402
403     float w_mech, invPsi_rs, w_rot, w_stat;
404
405     // posilam f_mech -> prepocet na w_mech
406     w_mech = PIr_speed->val_actual * 2 * PI;
407
408     // vypocet magnetickeho toku Psi_rs
409     Psi_rs = Psi_rs - R_r * (dt / L_r) * (Psi_rs - L_m * vals->Id);
410
411     // vypocet 1 / Psi_rs -> zamezeni deleni 0
412     if (Psi_rs > 0.01)
413     {
414         invPsi_rs = 1 / Psi_rs;
415     }
416     else
417     {
418         invPsi_rs = 1 / 0.01;
419     }
420
421     // vypocet rotorove rychlosti w_rot
422     w_rot = R_r * (L_m / L_r) * invPsi_rs * vals->Iq;
423
424     // vypocet statorove rychlosti w_stat
425     w_stat = w_rot + w_mech;
426
427     // integrace Thety
428     Theta = Theta + w_stat * dt;
429
430     // omezeni rozsahu Thety
431     if(Theta >  PI) {Theta = Theta - 2 * PI;}
432     if(Theta < -PI) {Theta = Theta + 2 * PI;}
433
434 }
```

Obr. 44: Kód výpočtu proudového modelu motoru

Zadávání parametrů motoru

Obstarání parametrů použitého motoru je zprostředkováno pomocí připnutého hlavičkového souboru s názvem Mot_params.h, který potřebné parametry popisuje. Výhodou tohoto přístupu je, že je možné si obdobný hlavičkový soubor vygenerovat, například z MatLab scriptu, s hodnotami požadovaného stroje. Pokud bude dodržené definované názvosloví, pak je kód programu použitelný i pro ostatní stroje, kde stačí pouze vyměnit onen hlavičkový soubor bez jakéhokoliv dalšího zásahu do programu.

```
// parametry motoru
#define pp 2                    // pocet polparu motoru
#define R_s 1.86                // odpor statoru
#define R_r 1.53                // odpor rotoru
#define L_m 0.033               // magnetizacni (hlavni) indukcnost
#define L_r 0.0373              // indukcnost rotoru
#define L_ss 0.0053             // rozptylova indukcnost statoru
#define L_rs 0.0043             // rozptylova indukcnost rotoru
#define I_max 3.65              // maximalni proud
#define f_n 50                  // jmenovita frekvence motoru
#define U_n 83                  // jmenovite napeti motoru
```

Obr. 45: Definice parametrů použitého stroje ve vlastním hlavičkovém souboru

Odesílání dat pomocí CAN

Jakožto komunikace s uživatelským rozhraní bylo zvoleno použití CAN sběrnice. Pro její implementaci jsem využil knihovnu "can_debug.h", která byla také vyvinuta panem Ing. Tomášem Košanem, PhD. umožňující snadné definování proměnných určených pro přenos po této sběrnici. Všechny tyto proměnné jsou definované v "main.c" speciálním datovým typem, respektive strukturou, *variable_t*. Do zdrojového kódu modulátoru jsou pak propsány jako externí (*extern*) proměnná pro snadné předávání dat mezi rozhraním a samotným kódem. Pro připravení samotné proměnné je třeba nejdříve definovat její jméno, které bude popisovat danou proměnnou v samotném rozhraní, společně s datovým typem a počáteční hodnotou. Nakonec je povolena změna proměnné pomocí CANu prostřednictvím nastavení *readonly* na 0 – tedy zakázání parametru umožňující pouze čtení proměnné u kterých je toto potřeba. Následuje už pouze přidání této funkce do obslužné rutiny CANu pomocí funkce *CanDebugAddVar()*. Takto zahrnutými proměnnými jsou: napětí meziobvodu, mechanická a požadovaná frekvence otáček, fázové proudy, změřené a požadované proudy I_d a I_q, zesílení jednotlivých regulátorů a proměnné *MODE* a *CTRL* pro volbu režimu řízení.

```
110     // incializace datoveho typu a popisku promennych pro CAN
111     ctrl_Udc.raw_data.float_var = 0;
112     ctrl_Udc.var_name = "Udc";
113     ctrl_Udc.var_type = is_float;
114     ctrl_Udc.readonly = 1;
115     CanDebugAddVar(&ctrl_Udc);
```

Obr. 46: Ukázka definice proměnné pro odesílání po CAN sběrnici

Ke všem použitým proměnným jsem ke jménu přidal na jeho začátek "ctrl_", aby bylo zamezeno konfliktům jmen a stále byla zachována jednoznačnost významu dané proměnné.

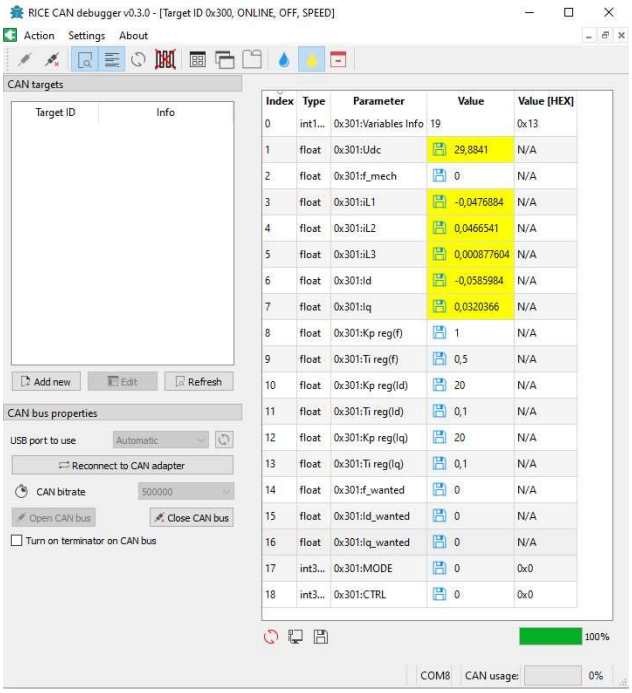
Propsání přijatých a odesílaných dat se provádí v rámci kroku modulátoru během přerušení od DMA přenosu. Na začátku tohoto přerušení jsou nejprve předány požadavky uživatele příslušným proměnným a, pro vizuální oddělení, se teprve propisují změřená data. Samotné odesílání probíhá v rámci "main.c", kde jsou za pomocí jednoduché čekací rutiny

data periodicky odesílána, jelikož není třeba kontrolér zatěžovat zbytečně častým odesíláním těchto dat. Použití tohoto čekání jsem si zde mohl dovolit použít, jelikož, jak bylo vysvětleno u předchozích částí kódu, jsou veškeré funkce realizující řízení pohonu obsluhovány v rámci přerušení a tak toto čekání nijak nezpožďuje vykonávání oněch funkcí.

```
81    // predani pozadavku uzivatele strukture
82    PIr_speed.val_wanted = ctrl_fw.raw_data.float_var;
83    MODE = ctrl_MODE.raw_data.int32_var;
84    CTRL = ctrl_CTRL.raw_data.int32_var;
85    Kp_f = ctrl_Kp_f.raw_data.float_var;
86    Ti_f = ctrl_Ti_f.raw_data.float_var;
87    Kp_Id = ctrl_Kp_Id.raw_data.float_var;
88    Ti_Id = ctrl_Ti_Id.raw_data.float_var;
89    Kp_Iq = ctrl_Kp_Iq.raw_data.float_var;
90    Ti_Iq = ctrl_Ti_Iq.raw_data.float_var;
91    Id_w = ctrl_Idw.raw_data.float_var;
92    Iq_w = ctrl_Iqw.raw_data.float_var;
93
94    // predani zmerenych hodnot uzivateli
95    ctrl_Udc.raw_data.float_var = vals.U_dc;
96    ctrl_Id.raw_data.float_var = vals.Id;
97    ctrl_Iq.raw_data.float_var = vals.Iq;
98    ctrl_I1.raw_data.float_var = vals.iL1;
99    ctrl_I2.raw_data.float_var = vals.iL2;
100   ctrl_I3.raw_data.float_var = vals.iL3;
101   ctrl_fm.raw_data.float_var = PIr_speed.val_actual;
```

Obr. 47: Předávání požadavků uživatele a změřených hodnot uživateli

Pro zobrazení na počítači je pak použit program RICE CAN debugger vyvinut, jak již z názvu vyplývá, v rámci výzkumného centra RICE pro účely vytvoření jednoduchého rozhraní umožňující řízení po CAN sběrnici.



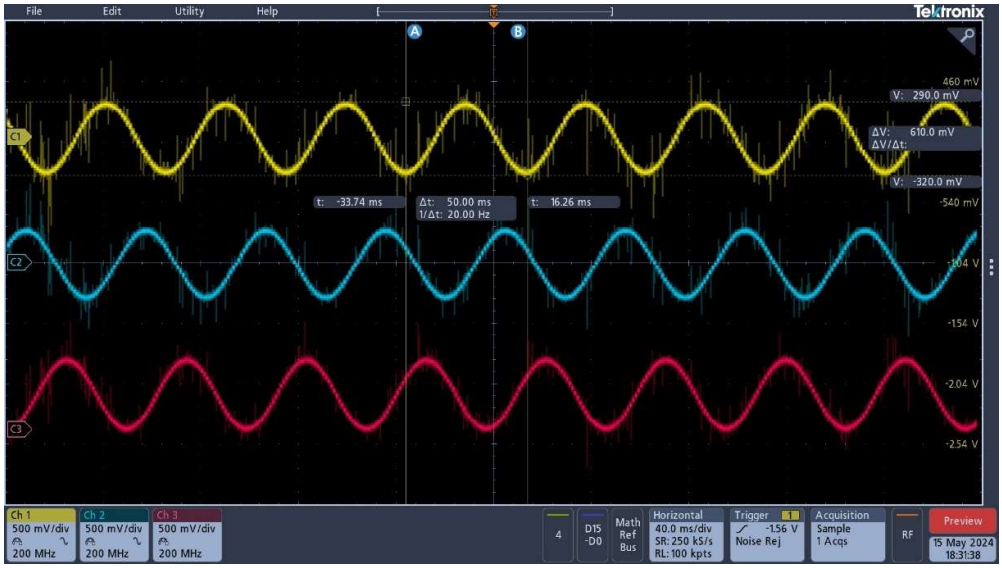
Obr. 48: Rozhraní RICE CAN debuggeru

V rámci “main.c“ se také provádí započetí převodu D/A převodníků spouštěné zároveň s odesíláním dat po CAN sběrnici. Toto má za důsledek, že se zobrazované průběhy na osciloskopu jeví jako více hranaté. Důvodem toho je opět to, že není třeba tento převod provádět tak často a aktuální nastavení je pro diagnostické účely dostačující.

2.2.2 Výsledky testování praktické realizace

Výsledky a jejich zhodnocení je možné provést z porovnávání dosažených hodnot vůči požadavku společně s průběhy získaných z osciloskopu.

Jako první kritérium uvažuji přesnost regulace. Jako prvotní kontrolu jsem zvolil měření frekvence referenčního signálu PWM. Tento signál jsem si pomocí D/A převodníků přivedl na osciloskop, aby bylo možné zkontrolovat, zda jsou jednotlivé fáze skutečně posunuté o 120° a za pomocí kurzorů ověřit, zda mají správnou frekvenci. Pro tuto kontrolu jsem zvolil skalární řízení, jelikož se zde přímo zadává požadovaná frekvence a tak je snadné porovnání požadavku, který jsem nastavil na 20 Hz, a skutečnosti. Na obrázku níže je vidět, že referenční signály jsou skutečně posunuty o 120° a frekvence odpovídá požadavku.



Obr. 49: Snímek z osciloskopu pro kontrolu fázového posuvu a frekvence

Přesnost regulace není pouze otázkou frekvence referenčního signálu, ale také skutečných změřených hodnot. Jako příklad zde můžeme uvést regulaci proudů při vektorovém řízení. Pro otestování jsem zvolil nejdříve hodnoty proudů $I_d = I_q = 1\text{A}$. Jako druhou variantu jsem zvolil $I_d = 0{,}8$ A a $I_q = 2$ A. Na obrázcích je vyznačen barevnými šipkami požadavek s příslušnou změřenou hodnotou. Jak je vidět, tak v obou případech skutečná hodnota velice blízko odpovídá žádané a regulaci lze nazvat vcelku přesnou.

Expression	Type	Value	Address
vals	struct MEAS	{U_dc=29.3440208,iL1=-0.159...	0x0000E8D8@Data
U_dc	float	29.3496475	0x0000E8D8@Data
iL1	float	-0.187321752	0x0000E8DA@Data
iL2	float	1.09790003	0x0000E8DC@Data
iL3	float	-0.642438471	0x0000E8DE@Data
Id	float	0.920849085	0x0000E8E0@Data
Iq	float	1.01818264	0x0000E8E2@Data
Plr_speed	struct RGL	{A_max=5.0,A_min=-5.0,Kr=0...	0x0000E8E4@Data
Plr_Id	struct RGL	{A_max=58.7949333,A_min=-...	0x0000E900@Data
Plr_Iq	struct RGL	{A_max=58.8174362,A_min=-...	0x0000E912@Data
A_max	float	58.699295	0x0000E912@Data
A_min	float	-58.682415	0x0000E914@Data
Kr	float	3.5	0x0000E916@Data
Ti	float	0.140000001	0x0000E918@Data
d_sum	float	-0.375497371	0x0000E91A@Data
sum	float	13.1299257	0x0000E91C@Data
val_actual	float	0.994557679	0x0000E91E@Data
val_wanted	float	1.0	0x0000E920@Data
val_out	float	12.8435621	0x0000E922@Data
Theta	float	-0.250049591	0x0000E8C6@Data
MODE	int	2	0x0000E8C0@Data
CTRL	int	2	0x0000E8C1@Data
Id_w	float	1.0	0x0000E8CA@Data
Iq_w	float	1.0	0x0000E8CC@Data
f_mech_w	float	0.0	0x0000E8CE@Data

Obr. 50: Porovnání požadavků a skutečných hodnot při regulaci proudu, varianta 1

Expression	Type	Value	Address
vals	struct MEAS	{U_dc=29.2708855,iL1=-1.5756...	0x0000E61E@Data
U_dc	float	29.2962017	0x0000E61E@Data
iL1	float	1.61584401	0x0000E620@Data
iL2	float	-1.27805984	0x0000E622@Data
iL3	float	-0.43197	0x0000E624@Data
Id	float	0.826602817	0x0000E626@Data
Iq	float	1.94321907	0x0000E628@Data
Plr_speed	struct RGL	{A_max=5.0,A_min=-5.0,Kr=0....	0x0000E62A@Data
A_max	float	5.0	0x0000E62A@Data
A_min	float	-5.0	0x0000E62C@Data
Kr	float	0.25	0x0000E62E@Data
Ti	float	0.00249999994	0x0000E630@Data
d_sum	float	0.0	0x0000E632@Data
sum	float	-4.78515625	0x0000E634@Data
val_actual	float	7.71484375	0x0000E636@Data
val_wanted	float	0.0	0x0000E638@Data
val_out	float	-5.0	0x0000E63A@Data
Plr_Id	struct RGL	{A_max=58.7499275,A_min=-5...	0x0000E640@Data
Plr_Iq	struct RGL	{A_max=58.8793221,A_min=-5...	0x0000E652@Data
Theta	float	1.26883495	0x0000E608@Data
MODE	int	2	0x0000E600@Data
CTRL	int	2	0x0000E601@Data
Id_w	float	0.800000012	0x0000E60C@Data
Iq_w	float	2.0	0x0000E60E@Data
f_mech_w	float	0.0	0x0000E610@Data

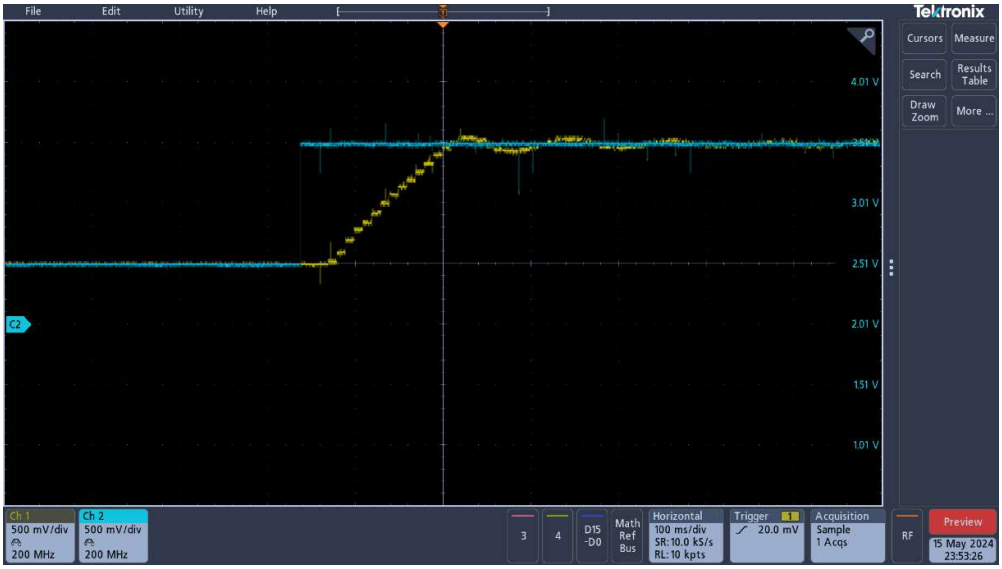
Obr. 51: Porovnání požadavků a skutečných hodnot při regulaci proudu, varianta 2

Stejný postup jsem aplikoval v případě nastavení režimu s vektorovým řízením a zadáváním proudu I_d společně s požadovanými otáčkami. Opět je zde vidět, že skutečné hodnoty jsou blízké těm požadovaným.

Expression	Type	Value	Address
vals	struct MEAS	{U_dc=29.40028,iL1=-0.745698...	0x0000E61E@Data
U_dc	float	29.4143448	0x0000E61E@Data
iL1	float	-1.83455575	0x0000E620@Data
iL2	float	1.24850345	0x0000E622@Data
iL3	float	0.734775066	0x0000E624@Data
Id	float	0.666781425	0x0000E626@Data
Iq	float	2.25085473	0x0000E628@Data
Plr_speed	struct RGL	{A_max=5.0,A_min=-5.0,Kr=0....	0x0000E62A@Data
A_max	float	5.0	0x0000E62A@Data
A_min	float	-5.0	0x0000E62C@Data
Kr	float	0.25	0x0000E62E@Data
Ti	float	0.00249999994	0x0000E630@Data
d_sum	float	3.7109375	0x0000E632@Data
sum	float	4.97424316	0x0000E634@Data
val_actual	float	11.9628906	0x0000E636@Data
val_wanted	float	12.0	0x0000E638@Data
val_out	float	4.98945618	0x0000E63A@Data
Plr_Id	struct RGL	{A_max=58.6486626,A_min=-5...	0x0000E640@Data
Plr_Iq	struct RGL	{A_max=58.5473976,A_min=-5...	0x0000E652@Data
Theta	float	2.68012524	0x0000E608@Data
MODE	int	2	0x0000E600@Data
CTRL	int	1	0x0000E601@Data
Id_w	float	0.639999986	0x0000E60C@Data
Iq_w	float	0.0	0x0000E60E@Data
f_mech_w	float	12.0	0x0000E610@Data

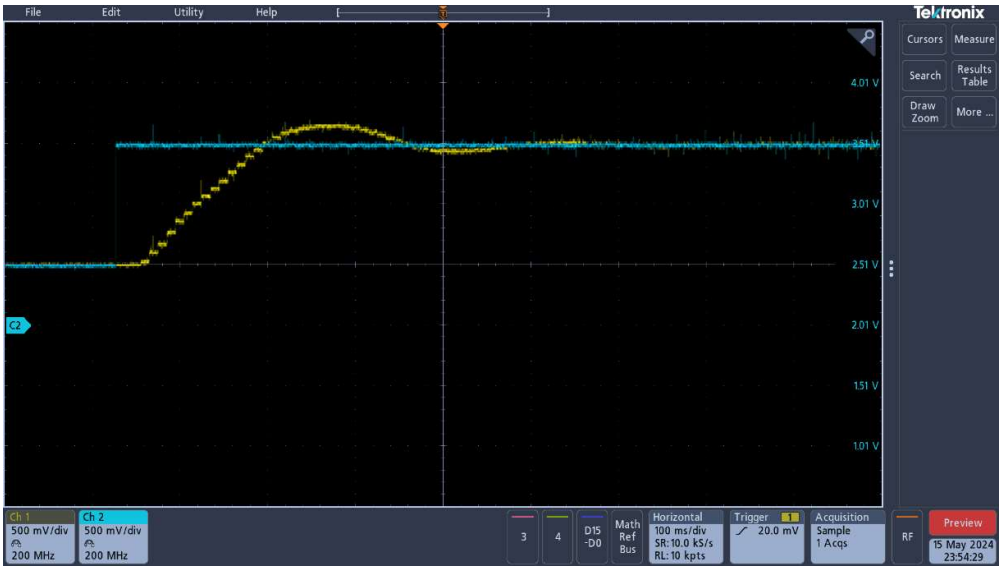
Obr. 52: Porovnání požadavku a skutečné hodnoty při zadávání požadavku na frekvenci a d-složku proudu

Kromě této přesnosti regulace je také potřeba zkontrolovat průběh veličin v přechodových stavech, tedy například při rozběhu. Jako příklad zde uvedu rozběh v režimu skalárního řízení, na kterém lze dobře demonstrovat důsledky nesprávně naladěného regulátoru, který popisuje podkapitola Ladění regulátoru v kapitole 2.1 Simulace vektorového řízení, s teoretickým doplněním obrázkem 31. Testovaný případ je rozběh z nulové frekvence na frekvenci 25 Hz.



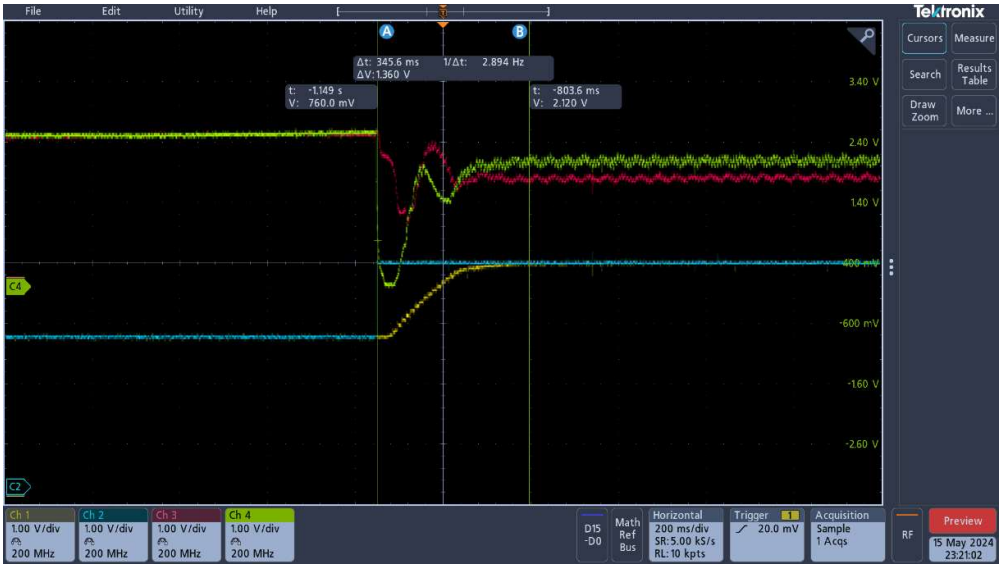
Obr. 53: Snímek z osciloskopu zobrazující požadovanou (modrá) a skutečnou (žlutá) frekvenci, vysoké Kp

Jak je vidět, tak skutečná frekvence před svým ustálením na požadované hodnotě vykazuje krátké kmity, což značí příliš vysoké zesílení regulátoru K_p.



Obr. 54: Snímek z osciloskopu zobrazující požadovanou (modrá) a skutečnou (žlutá) frekvenci, vysoké T_i

Zde jsou naopak kmity pomalé, což naopak vypovídá o příliš silné integrační složce regulátoru.

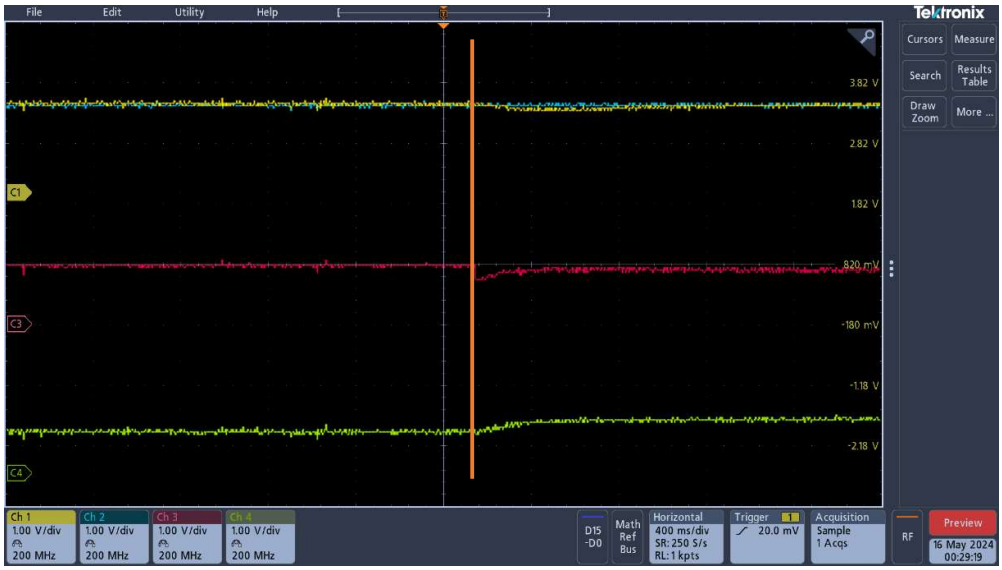


Obr. 55: Snímek z osciloskopu zobrazující požadovanou (modrá) a skutečnou (žlutá) frekvenci, ideální

Na tomto posledním obrázku je vidět, že zde již nejsou žádné kmity a nastavení regulátoru můžeme považovat za korektní. Navíc zde můžeme vidět, že samotný rozběh trval přibližně 340 ms. Také jsou zde zobrazené průběhy proudů I_d (červený) a I_q (zelený).

Další rozběhy a odezvy hodnot na požadavky jsou obsáhnuty v příloze C společně s textovým dokumentem popisující co se na jednotlivých obrázcích nachází.

Také jsem zaznamenal přechody mezi jednotlivými režimy. Jako ukázka je zde uveden přechod mezi režimem CTRL = 1 (zadávaná f_w a proud I_d) na režim $CTRL = 0$ (zadávaná pouze f_w) . Přechod nastal v momentě vyznačeném oranžovou čarou. Modrá křivka zde značí požadavek na frekvenci otáček, žlutá aktuální hodnotu. Červená křivka zobrazuje d-složku proudu, zelená q-složku. Je vidět, že dojde k mírnému poklesu otáček, na který ihned reaguje regulátor proudu I_q a stroj přibudí tak, aby se otáčky doregulovali na požadovanou hodnotu. Proud I_d jeví také pokles, jelikož byl vstupní požadavek jeho regulátoru změněn z konstanty zadávané uživatelem navázán výstup regulátoru otáček – tedy nastal skok, který byl v krátké době doregulován na vhodnou hodnotu a ustálil se.



Obr. 56: Snímek z osciloskopu zachytávající přechod z režimu CTRL = 1 na CTRL = 2

Další přechody mezi režimy jsou také uvedené v příloze C s příslušnými popisy, jelikož si myslím, že uvádět zde veškeré přechody mezi režimy není nutné a v případě zájmu jejich bližšího zkoumání je možné v uvedené příloze.

Zhodnocení a závěr

Práce popisuje obecné schéma a používané komponenty moderních elektrických pohonů, jako jsou nejběžněji používané výkonové spínací prvky společně s jejich vlastnostmi a parametry společně s jejich porovnáním a vhodností použití pro určité aplikace. Důležitou komponentou je samotný napěťový střídač, jeho zapojení a řízení. Mezi nejběžněji používané algoritmy se řadí skalární řízení a vektorové řízení. Pro tuto práci je nejzajímavější právě vektorové řízení, jehož stěžejní komponentou je matematický model motoru, který je odvozen a popsán rovnicemi v kapitole 1.4 Elektrický stroj.

Cílem práce bylo vektorové řízení a jeho algoritmus otestovat na simulačním modelu k čemuž jsem využil program Simulink s nadstavbou Plecs s již existujícími modely výkonových prvků, elektrických strojů a možnosti psaní kódu MatLab Scriptu, který jsem použil pro psaní kódu regulátorů, transformací a matematického modelu. Po tomto otestování bylo mým úkolem řízení implementovat do kontroléru a řídit RL zátěž. Ta byla zvolena v podobě malého asynchronního motoru napájeného z laboratorního prototypu napěťového střídače. Kód jsem, dle zadání, psal v programovacím jazyce C s využitím několika knihoven uzpůsobeným k práci s daným kontrolérem. Výsledkem je kód, který umožňuje řízení daného elektrického stroje v režimu skalárního řízení nebo vektorového řízení, které se dále dělí na 3 podrežimy umožňující výběr veličin, které budou uživatelem zadávány jako požadavky. Výsledky těchto testování jsou obsaženy v kapitole 2.2.2 Výsledky testování praktické realizace a příloze D. Samotný kód je v příloze B.

Pro komunikaci s uživatelským rozhraním je použita CAN sběrnice přivedená na CAN adaptér, který je možno pomocí standardního USB připojit k počítači. Za pomocí programu RICE CAN debugger je pak možno s řídícím kontrolérem komunikovat, předávat mu požadavky uživatele a číst aktuální měřené hodnoty. Návod k ovládání pohonu je součástí kapitoly 2.2.1 Popis funkčnosti kódu.

Možným vylepšením je zavedení diskrétního časového kroku simulace. Aktuálně jsou zásahy řídících bloků prováděny v každém kroku simulace, tedy téměř spojitě. Kdežto ve skutečnosti probíhá zásah těchto bloků pouze jednou za daný čas. K tomuto účelu je možné využít spínaný subsystém, který se mi ovšem nepovedlo naladit tak, aby simulace proběhla v celém svém rozsahu. Vzhledem k tomu, že účelem nebylo provést co nejpřesnější simulaci, ale takto pouze otestovat funkčnost algoritmu, tak jsem se tímto problémem dále nezabýval.

Při porovnávání výsledků simulace a skutečného pohonu jsou výsledky simulace optimističtější. Důvodem může být zmíněný vliv zpožděných reakcí reálného systému oproti regulovanému, ale také přesnost matematického modelu, jelikož parametry motoru pro jeho výpočet v simulaci jsou naprosto přesné, ale v reálné aplikaci tyto parametry s vysokou pravděpodobností přesně neodpovídají použitému motoru. Také se zde může propisovat ne zcela ideální nastavení regulátorů vzhledem k jejich různému potřebnému nastavení pro každý režim separátně a tak by bylo možné regulaci zkvalitnit jejich lepším naladěním.

Jakožto dalším možným krokem ke zlepšení je zavedení ochran realizovaných pomocí tzv. Trip zones, které řeší například nadproudy nebo třeba odpojení e-PWM periferie v režimu, kdy není pohon řízen, o čemž se zmiňuje kapitola 2.2.1. K tomuto řešení ochran pomocí Trip zones jsem se z časových důvodů nevěnoval.

Výsledkem práce je sestava připravená k použití za účely výuky, jak bylo zamýšleno, umožňující ovládání za pomocí zmíněné CAN sběrnice. Výhodou je možnost změny režimu i za běhu motoru. Navíc je kód psaný tak, aby byl použitelný i pro jiné asynchronní motory než se kterým jsem pracoval. Parametry použitého motoru jsou do kódu zavedeny vlastním hlavičkovým souborem, který, při dodržení daných názvů veličin, stačí nahradit stejnojmenným souborem s parametry použitého stroje.

Zdroje

[1] Elektrické pohony a výkonová elektronika | prof. Ing. Václav Kůs, CSc.; Západočeská univerzita v Plzni: Fakulta Elektrotechnická. Vydáno 2005. [Cit. 02.04.2024]. ISBN 978-80-7043-422-8

[2] Moderní výkonové polovodičové prvky a jejich aplikační možnosti | Ing. Jaroslav Novák, CSc.; ČVUT v Praze: Obor elektrotechniky spínacími ztrátami. [Cit. 01.05.2024]. Dostupné z: http://www.odbornecasopisy.cz/elektro/casopis/tema/moderni-vykonove-polovodicove-prvky-a-jejich-aplikacni-moznosti--14473

[3] Trakční střídač 2,5kW pro asynchronní motor elektrického zahradního traktoru | Petr Krist; Vysoké učení technické v Brně: Fakulta elektrotechniky a komunikačních technologií. Poslední změna 2012. [Cit. 24.04.2024]. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=53840

[4] MOSFET tranzistory – struktury, funkce, charakteristiky | Martin Olejár; ELWEB. [Cit. 02.05.2024]. Dostupné z: http://www.elweb.cz/clanky.php?clanek=94

[5] Budící obvody výkonového tranzistoru SiC MOSFET | Vojtěch Vítek; Vysoké učení technické v Brně: Fakulta elektrotechniky a komunikačních technologií. Poslední změna 2013. [Cit. 24.04.2024]. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=68991

[6] Inteligentní budič SiC tranzistorů | Jiří Hodný; ČVUT v Praze: Kybernetika a robotika. Poslední změna: 2021. [Cit. 02.05.2024]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/96675/F3-DP-2021-Hodny-Jiri-DP_HODNY.pdf?sequence=-1&isAllowed=y

[7] Analýza polovodičových prvků Si MOSFET a GaN se základní topologií budících obvodů | Alexandr Justin; Západočeská univerzita v Plzni: Fakulta Elektrotechnická. Poslední změna: 2019. [Cit. 02.05.2024]. Dostupné z: https://dspace5.zcu.cz/bitstream/11025/39434/1/BP_Alexandr_Justin.pdf

[8] Střídavé motory – Motor s kotvou na krátko | Ing. Radovan Hartmann; SPŠ a VOŠ technická Brno. Poslední změna 11.2012. [Cit. 23.04.2024]. Dostupné z: https://www.sokolska.cz/DUMy/ELE/VY_32_INOVACE_41-07.pdf

[9] Modulační techniky pro víceúrovňové střídače | Bc. Petr Stejskal; ČVUT v Praze: Elektrotechnika, energetika a managment. Poslední změna: 2015. [Cit. 08.05.2024]. Dostupné z: https://dspace.cvut.cz/bitstream/handle/10467/61749/F3-DP-2015-Stejskal-Petr-modulacni_techniky_pro_viceurovnove_stridace.pdf?sequence=3&isAllowed=y

[10] Střídavé motory – Motor s kroužkovou kotvou | Ing. Radovan Hartmann; SPŠ a VOŠ technická Brno. Poslední změna 11.2012. [Cit. 23.04.2024]. Dostupné z: https://www.sokolska.cz/DUMy/ELE/VY_32_INOVACE_41-08.pdf

[11] Synchronní stroj spouštěný ze sítě | Bc. Ruben Kaczmarczyk; Vysoké učení technické v Brně: Fakulta elektrotechniky a komunikačních technologií. Poslední změna 2018. [Cit. 23.04.2024]. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=176186

[12] Automatická regulace pohonů s asynchronními motory | doc. Ing. Karel Zeman, CSc. | doc. Ing. Zdeněk Peroutka, Ph.D. | Ing. Martin Janda; Západočeská univerzita v Plzni: Fakulta Elektrotechnická. Vydáno 2007. [Cit. 23.04.2024]. ISBN 978-80-7043-350-8

[13] Potlačení mrtvých časů v napěťovém střídači | Bc. Vojtěch Matys; Západočeská univerzita v Plzni: Fakulta elektrotechnická. Poslední změna 2012. [Cit. 09.05.2024]. Dostupné z: https://dspace5.zcu.cz/bitstream/11025/2331/1/Potlaceni%20mrtvych%20casu%20v%20napetovem%20stridaci.pdf

Přílohy

Příloha A: Soubor fotografií dokumentující použitý hardware k řešení této diplomové práce

Příloha B: Soubor s projektem vypracovaným v Code Composer Studiu obsahující kód vytvoření v rámci této práce včetně všech příslušných knihoven

Příloha C: Snímky osciloskopu zachytávající dynamické stavy v průběhu řízení pohonu

Příloha D: Simulace vektorového řízení v MatLab Simulinku společně s MatLab scriptem popisující parametry pohonu a simulace