Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Vektorové řízení synchronních motorů

Bakalářská práce

Jan Zágiba

Vedoucí práce: Ing. Václav Novák CSc.

České Budějovice 2022

Bibliografické údaje

Zágiba, J., 2022: Vektorové řízení synchronních motorů. [Field oriented control of synchronous motors. Bc.. Thesis, in Czech.] – 39 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Abstract

This bachelors thesis explaines field oriented control and develops software camable of modulating by it. The next goal is to make a custom hardware for this exact thesis. And making an example vehicle for testing software and hardware together on it.

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

V Českých Budějovicích dne 13. 4. 2022

Jan Zágiba

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Václavu Novákovi za vedení této práce a odborné rady během vývoje. Dále bych rád poděkoval své rodině, která mě během studia vždy podporovala.

Obsah

1 Úvod

Jelikož v dnešní době se čím dál tím víc snažíme ochraňovat planetu a co nejméně přispívat ke globálnímu oteplování je důležité dbát na efektivitu při návrhu a výrobě zařízení. A zároveň v posledních letech se rozvinul trend jízdy na elektrokoloběžkách a elektrokolech.

Všechny tyto aplikace používají baterie jako hlavní zdroj energie a velké množství baterií zvyšuje váhu a cenu těchto dopravních prostředků.

Prakticky všechny motory v této kategorii jsou stejnosměrné motory, i když se stále vyskytují motory komutátorové, tak jejich počty se snižují, protože motory bezkartáčové BLDC nabízí podstatné výhody, jako nízká potřeba údržby z důvodu absence kartáčů potřebných pro komutaci a vyšší poměr váhy oproti výkonu motoru. Jenže stejnosměrné motory vždy potřebují nějaký způsob komutace, aby se mohly otáčet a u komutátorových byla tato vlastnost vykonávána mechanicky, ale BLDC motory neobsahují žádnou mechanickou komponentu vykonávající tuto vlastnost, proto se musí vykonávat komutace ještě před motorem, a to zvyšuje komplexnost spojenou s používáním těchto motorů.

Na pohánění motorů lze použít několik různých způsobů a tato práce se zaměří na implementaci jednoho z nich. Následně pro tuto práci bude vyroben vlastní hardware zkonstruovaným přesně pro potřeby zvoleného způsobu pohánění, i když do jisté míry budou použita i běžně koupitelná řešení jako je například Raspberry. Jenže při realizaci hardwaru se objevil problém s globální nedostatkem čipů a tím pádem značně omezený možnosti nákupu jednotlivých součástek. A to přináší problém s případným selháním nějaké části, protože fronty na další čip jsou dlouhé i přes půl roku.

1.1 Cíle práce

Cílem práce je vytvořit funkční implementaci vektorového řízení s minimální používáním knihoven. A naprogramovat to na hardwaru s plným operačním systémem, aby byl samotný kód minimálně zatížen platformě specifickými částmi kódu. A tenhle software použít na hardwaru specificky vyrobeným pro tuto činnost. A kombinaci těchto věcí použít na nějakém prostředku osobní dopravy s tím, že dopravní prostředek musí být vhodný pro použití s hardwarem.

2 Druhy řízení BLDC motoru

Na řízení BLDC motorů lze použít několik způsobů řízení. Ale všechny metody, které tato práce bude zmiňovat používají pulzně šířkovou modulaci (PWM). Jediný rozdíl je v tom, jak spínají jednotlivé fáze motoru.

Všechny metody se snaží docílit, aby byl rotor 90° od pólu opačné polarity.

2.1 Lichoběžníkové (trapezoidní) řízení

Lichoběžníkové řízení motoru je nejjednodušší ze všech probíraných v této práci. Princip spočívá v tom, že jsou sepnutý 2 fáze a třetí je ponechána v nesepnutém stavu. Tento způsob řízení má i své nevýhody například zvlnění momentu motoru, nebo nižší účinnost, která se projevuje zahříváním motoru a zvýšenou hlučností motoru.

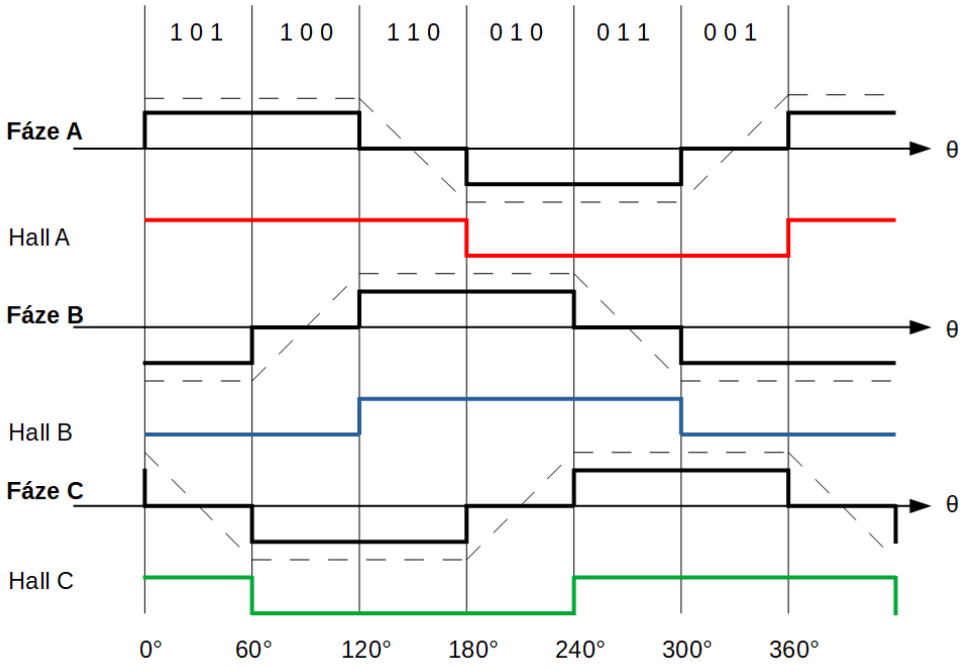

Figure 2.1: Spínání jednotlivých fází podle stavu Hallových senzorů

Na obrázku je vidět, jak se mění spínané fáze v závislosti na poloze rotoru. Poloha rotoru lze zjistit dvěma způsoby: přímým měřením a nepřímým měřením.

2.2 Přímá metoda měření

Přímé měření spočívá v tom, že používáme externí zařízení na měření polohy rotoru vůči statoru. Díky téhle skutečnosti můžeme použít velice jednoduchý hardware. Konkrétně jsme schopni navrhnout řídící obvod bez použití mikroprocesoru, jen s použitím komparátorů a obdobně složitých aktivních a pasivních prvků.

2.2.1 Nepřímá metoda měření

Nepřímé měření závisí na měření indukovaného napětí z odpojené fáze motoru. Na odpojené fázi jsme schopni změřit kdy přes pól motoru, ke kterému je připojená fáze, prochází magnet rotoru. Jenže tento způsob trpí nedostatečnou přesností při nízkých otáčkách, kdy se indukuje příliš nízké napětí, aby bylo spolehlivě odděleného od šumu. Proto se u nízkých otáček musí motor řídit o otevřené smyčce, aby dosáhl měřitelných hodnot.

2.3 Sinusové řízení

Sinusové řízení spočívá v tom, že používáme sinusovou křivku pro modulaci napětí na motoru. Tento způsob vyžaduje znát přesnou polohu rotoru. Pro výstupní modulaci PWM používá stejný způsob reprezentace magnetického pole jako vektorové řízení a to tím že dělí otáčky do šesti sektorů. A podle sektoru a úhlu v daném sektoru se vyhledá potřebná amplituda z tabulky hodnot. A tyto hodnoty se zkombinují jako u vektorového řízení a pošlou se na budič. Přes tohle všechno je tato metoda výpočetně nenáročná, protože všechny náročné výpočty jsou už vypočtené. [1]

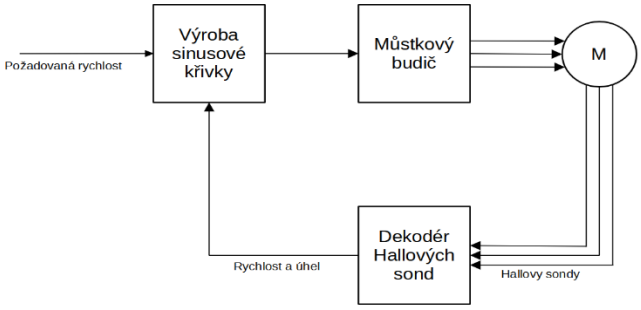

Figure 2.2: Schéma systému pro sinusové řízení

2.3.1 Vektorové řízení (FOC)

Vektorové řízení je nejkomplikovanější ze všech způsobů řízení, které tato práce probírá. Vektorové řízení dostalo své jméno podle vektoru napětí na motoru, jelikož PMSM motory jsou třífázové tak se jejich výsledný vektor skládá ze tří o 120° posunutých amplitud. Tento vektor je důležité zachovat v 90° odklonu oproti směru magnetického pole, protože vektor ve směru magnetického pole nepřispívá žádným kroutivým momentem, jen spotřebovává energii. A úkol vektorového řízení je zachovat tento vektor vždy v pravém úhlu.

Fixování úhlu by technicky šlo udělat pomocí třech PID regulátorů, ale vektorové řízení nabízí zjednodušení na výpočet tím, že sníží počet regulátorů. Po aplikaci prvního kroku též známého jako přímá Clarkova transformace, byly převedeny 3 napěťové hodnoty do alfa beta souřadnicového systému. V tomto systému máme stejný vektor, který se ale skládá jen už z dvou částí, toto je pokrok z 3 regulátorů na 2, ale stále to neřeší problém nutnosti navrhnout regulátor na střídavé veličiny. Jako druhý krok je odpoutání magnetického pole od statoru a spřáhnout ho s rotací rotoru, touto úpravou lze dosáhnout dvou veličin, kde jedna určuje sílu magnetického pole směřujícího k pólům permanentního magnetu a druhou veličinu, která svírá s první pravý úhel, z toho vyplývá, že druhá veličina je přímo úměrná kroutivému momentu. Když jsou známé přímo hodnoty ovládající magnetické pole v okolí rotoru, tak je regulace pomocí regulátorů značně zjednodušená, protože hodnoty těchto dvou veličin se chovají jako stejnosměrné hodnoty a jsou prakticky neměnné. Tato úprava se nazývá Parkova transformace. [2]

Přímá regulace kroutivého momentu motoru je veliké zjednodušení pro ovládání regulované hodnoty rychlosti. Na regulaci rychlosti je potřeba znát otáčky motoru v čase z těch se vypočte aktuální rychlost motoru a přes zápornou zpětnou vazbu se přes PID regulátor reguluje. Po provedené regulaci rychlosti se provedou inverzní transformace, aby souřadná soustava měla zase podobu tří fází, tento výsledek se posléze použije při modulaci pulzně šířkové modulace (PWM).

Pro tento způsob řízení motoru je absolutně kritické znát přesné proudy protékající skrz vinutí statoru motoru, současně naměřené hodnoty nemohou být změřené s velký časovým

rozestupem, ideálně by měly být změřené ve stejný okamžik, aby se zamezilo „rozmazání proudu" [1].

Náročnost algoritmu vyplývá z provádění 4 transformací a používání 3 PID regulátorů. Oproti předešle zmíněným metodám řízení je to značný skok ve výpočtu oproti hledání z tabulky se 6 stavy lichoběžníkového řízení, nebo hledání úhlů z předvypočítané tabulky hodnot sinusového řízení. Ale sdílí i pár společných věcí jako je potřeba znát přesnou polohu úhlu stejně jako sinusové řízení.

Figure 2.3: Ukázka 3 sinusových křivek oddělených od sebe 120°

Na obrázku je vidět, jak by správně měl vypadat výstup z vektorového řízení, tři sinusové křivky 120° posuté od sebe.

[1] Je posunutí vektoru proudu v průběhu měření a následné zkreslení

2.3.2 Clarkova transformace

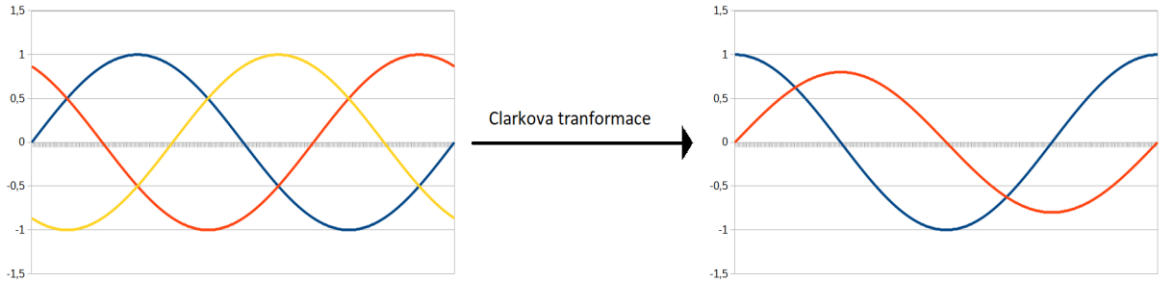

Figure 2.4: Znázornění přepočtu z tříosého do kartézského

Přímá Clarkova transformace

$$\begin{bmatrix} I_\alpha \\ I_\beta \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} I_U \\ I_V \\ I_W \end{bmatrix} \tag{1}$$

Přímá Clarkova transformace transformuje 3 fáze rotoru na 2fázovou reprezentaci rotoru. [3]

Inverzní Clarkova transformace

$$\begin{bmatrix} I_U \\ I_V \\ I_W \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} I_\alpha \\ I_\beta \end{bmatrix} \tag{2}$$

Inverzní Clarkovou transformací získáváme zpátky 3 o 60° posunuté fáze. [3]

2.3.3 Parkova transformace

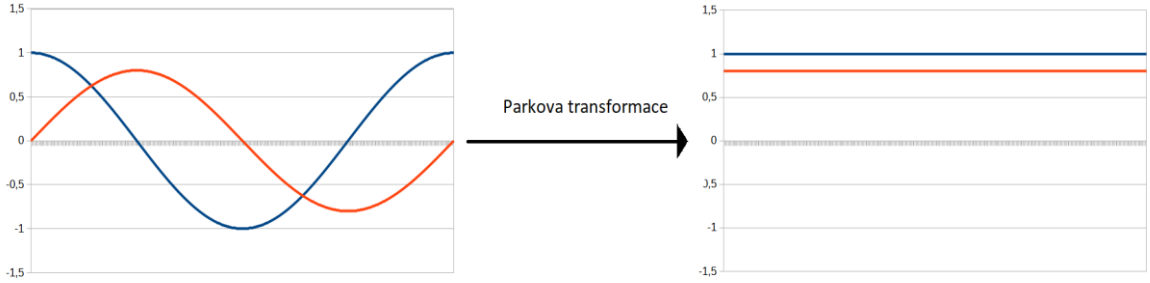

Figure 2.5: Ukázka přepočtu ze statického reprezentace do rotační

Přímá Parkova transformace.

$$\begin{bmatrix} I_d \\ I_q \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} I_\alpha \\ I_\beta \end{bmatrix} \tag{3}$$

Parkova transformace převádí 2fázovou reprezentaci vázanou k statoru na reprezentaci vázanou k rotoru [3]

Nepřímá Parkova transformace.

$$\begin{bmatrix} I_\alpha \\ I_\beta \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} I_d \\ I_q \end{bmatrix} \tag{4}$$

2.3.4 Rozdělení na sektory

Ve vektorovém řízení se musí kružnice úhlů rotoru rozdělit na sektory, a jelikož je pro PSMS motory použitý třífázový můstek s 6 spínači. A pro každou kombinaci je potřeba vytvořit sektor, ale jelikož v případě všech horních spínačů sepnutých najednou vzniklý vektor nikam nevede, a proto se jmenuje nulový, a totéž platí i pro stejný stav na dolní straně. A pro kombinaci 6 spínačů, kde každý má dva stavy vyjde celkový počet 8. [4]

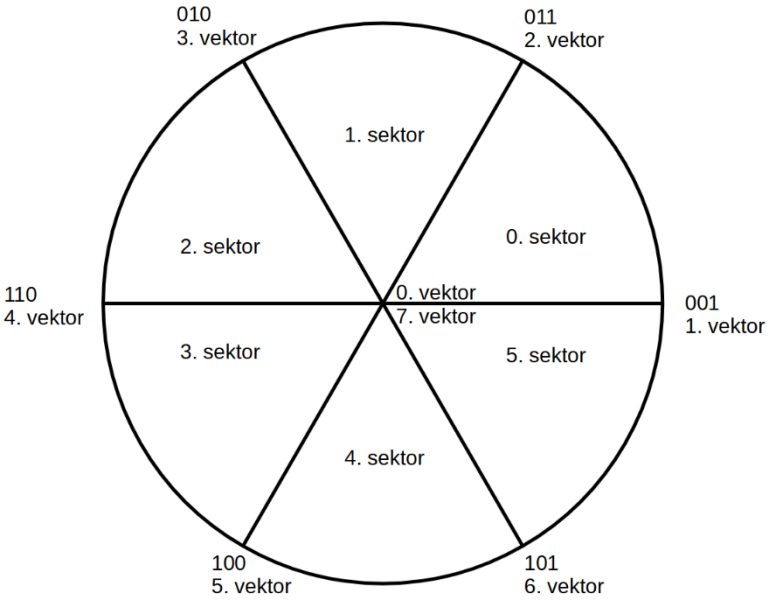

Figure 2.6: Zobrazení rozdělení sektorů a vrcholů se polaritou spínačů

Nulové sektory slouží k snížení napětí na motoru, a jsou potřeba oba dva nulové vektory jinak by vznikla deformovaná sinusová křivka.

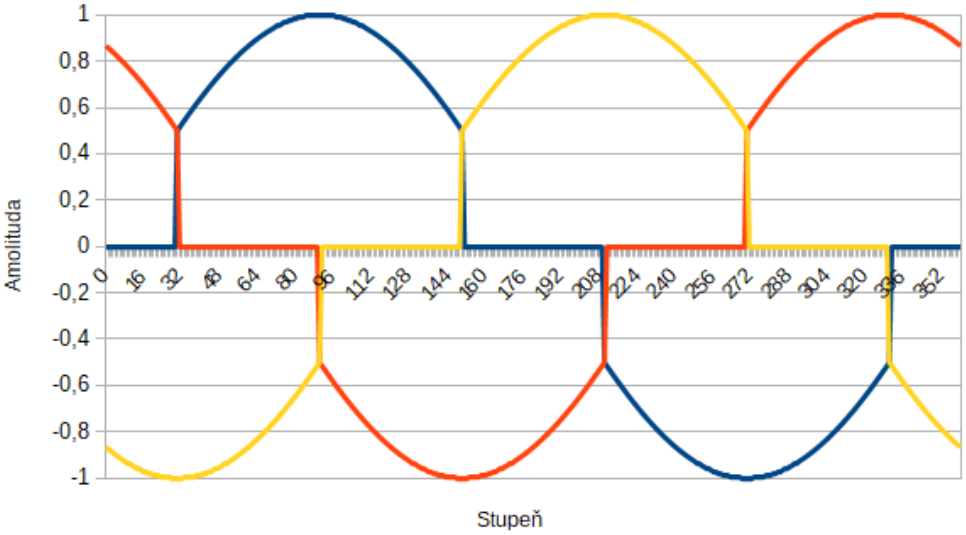

Figure 2.7: Znázornění důležitosti nulových vektorů (znázorněno při 50 % střídě)

2.3.5 Získávání pozice pomocí Hallových sond

Získávání pozice pomocí Hallových senzorů je jednoduchý způsob, jak získat pozici rotoru z důvodu, že rotor obsahuje magnety se střídající se orientací pólů.

Ale v běžném provedení se používají 3 sondy 120° od sebe. Jenže v praxi mají nevýhody jako nepřesnost při výrobě motoru se sondami, nebo odchylka při montáži externího měřícího zařízení na motor. Další problém je rozlišení, které závisí na počtů pól párů nacházející se v motoru, kde vyšší množství pól párů zvyšuje rozlišení. A v komerčním prostředí zvyšují cenu výroby, protože jsou to díly navíc.

Životnost sond může taky může hrát roli při provozu, protože při běhu zařízení, když sonda selže a začne dávat chybné hodnoty, tak je třeba mít záložní způsob měření, který kontroluje chyby z hlavního snímače (Hallovy sondy) a je schopen převzat úlohu, což zvedá komplexnost a tím i cenu. Ale zvýšení ceny lze omezit tím, že se použije bočník (shunt) na odhad pozice rotoru, který vždy musí být součástí hardwaru, který podporuje FOC.

2.3.6 Získávání pozice pomocí rotačních snímačů

Rotační snímače mají většinu výhod a nevýhod shodných s Hallovými sondami, až na cenu a přesnost senzoru. A právě díky vysoké přesnosti jsou velice užitečné při aplikacích s náročností na přesnou polohu, například v aplikacích, kde za žádnou cenu se motor nesmí otočit v opačném směru, než je požadováno. Existuje několik druhů rotačních snímačů jako například absolutní a inkrementální rotační snímače.

I když by byl tento způsob nejlepší pro tuto práci, tak se nemůže použít, protože vybraný motor nemá integrovaný snímač a kvůli nedostatku místa na hřídeli motoru díky způsobu uchycení nelze použít.

2.3.7 Získávání pozice snímáním proudů motoru

Tato metoda je nejspolehlivější z pohledu životnosti komponentů, protože odpory mají vysokou životnost, při používání v limitech od výrobce můžou fungovat do nekonečna. Získávání dat spočívá v měření proudu v moment, když jsou spínače sepnutý a proudí skrz něj proud.

Jenže při snímání proudu na malé střídě může docházet ke zkreslení signálu z důvodu nízkého SNR (signal to noise ratio). Proto tenhle druh snímání není optimální od nízkých rychlostí až po zastavený motor. To samé platí při rozjezdu motoru, proto se musí používat speciální postupy pro rozběh motoru s tímto způsobem snímáním polohy.

2.3.7.1 Snímání proudu na třech a dvou odporech

Snímání odporu na dvou, nebo třech bočnících má výhodu oproti jednomu bočníku, v tom že potřebují na měření proudu skrz motor pouze jeden cyklus.

2.3.7.2 Snímání proudu na jednom odporu

Používání pouze jednoho bočníku je lepší v tom, že stačí koupit pouze jeden odpor a tím pádem je výsledný produkt levnější. Ale nevýhodou je, že se musí snímat proud ve dvou cyklech.

Problém s tímto způsobem měření je při nízkých, nebo vysokých střídách, kdy je čas mezi spínáním jednotlivých tranzistorů nedostatečný pro AD převodník na provedení měření. Tento problém jde částečné odvrátit posunováním spínáním v, že se posune část sepnutí z jedné strany symetrického PWM cyklu do druhého. [5]

3 Funkční diagram FOC

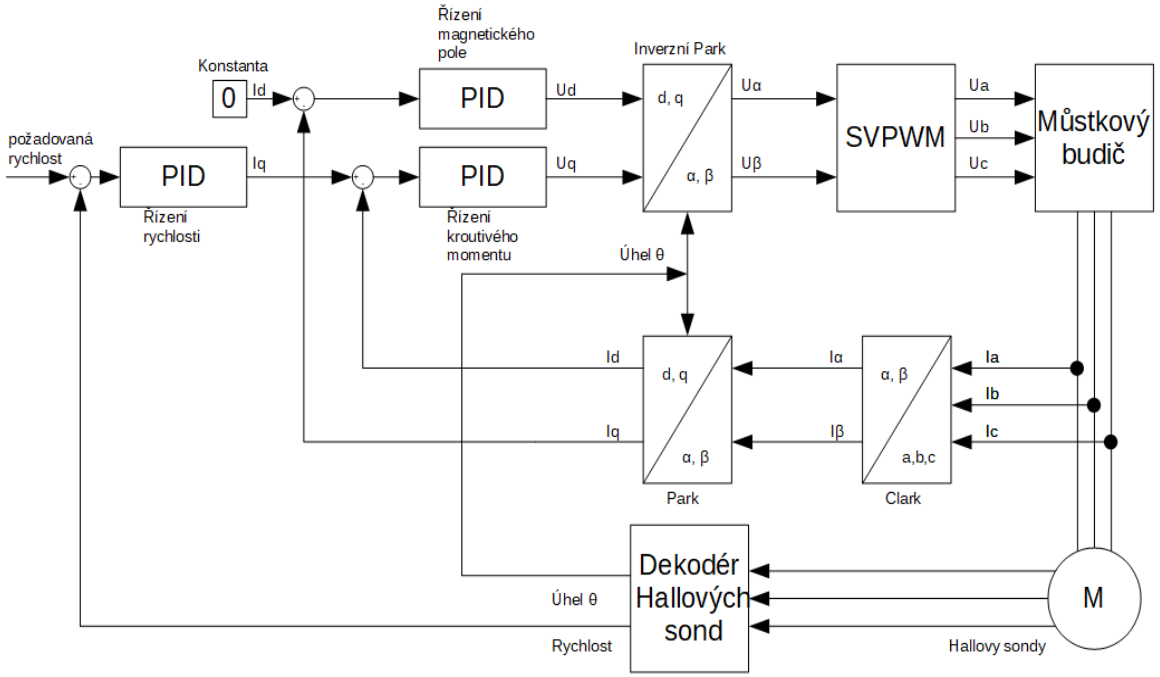

Figure 3.1: Schématické zobrazení částí potřebných pro vektorové řízení

Vektorové řízení je uzavřená smyčka a jedinou hodnotu, kterou ovládané je rychlost otáčení. Rychlost otáčení se přičte k záporné rychlosti motoru a tato hodnota jde do PI regulátoru pro rychlost. Hodnota z regulátoru se přičte k záporné kvadraturní složce proudu a výsledek se předá do PI regulátoru kroutivého momentu. PI regulátoru přímé složky jako regulační hodnotu proudu používá buď 0 nebo metodu oslabování magnetického pole [2]. Nula se používá, když nepotřebujeme, aby motor přesáhl svoje maximální otáčky a oslabování pole, když to potřebujeme.

Id a Iq se s úhlem motoru předá Parkově transformaci a poté Clarkově, a signál se pomocí PWM (Pulzně šířková modulace) namoduluje pro budič tranzistorů. Při spínání tranzistoru se měří proud protékající skrz motor. Provede se Clarkova transformace, ale Parkova transformace potřebuje aktuální úhel motoru. Bez použití Hallových sond by se použít jiný

[2] Oslabování pole je, když se použije Id vektor jde do záporných čísel a tím snižuje magnetický tok motoru

způsob na odhadování polohy rotoru. Poloha rotoru se přepočítá na rychlost a pošle se na začátek smyčky.

4 Popis hardware

4.1 Baterie

Výběr baterií je jedna z nejdůležitějších ne-li ta nejdůležitější součást jakéhokoliv mobilního zařízení poháněné elektřinou. Existuje mnoho typů baterií od olověných článků až po lithium-polymerové články. Náročnost výběru vyplývá z rovnováhy mezi cenou za watt, objemem na watt a váhou na watt.

Tradiční olověné baterie jsou nejlevnější na watt, ale nabízejí nejhorší váhu a objem na watt. I když mají schopnost poskytovat vysoké proudy na krátkou dobu.

NiMH baterie jsou lepší než olověné, co se týče objemu a váhy na watt. Jenže tenhle druh baterií má problém s částečným nabíjením, protože nabíjení baterie bez kompletního vybití předem snižuje značně kapacitu kvůli tzv. paměťovému efektu.

Li-ion a li-po baterie mají podobné vlastnosti mezi sebou co se týče váhy na watt i objemu na watt. Důležitý rozdíl mezi nimi je maximální proud skrz článek, li-ion baterie mívají nižší maximální proud skrz článek. Tato hodnota se často jako C[3] a li-po mívají vyšší C. A navíc mají nejvyšší váhu/kapacitu i objem/kapacitu ze všech zmíněných.

Tato práce používá baterii NANO-Tech s kapacitou 2200 mAh a vybíjením 60 C stálým a 120 pulzním. Baterie používá nejiskřící konektor XT60.

[3] C číslo se používá pro zjednodušený popis, kolik baterie zvládne dodávat vůči její kapacitě tzn. 20 C je 20krát kapacita baterie jako maximální stálí proud.

Figure 4.1 Použitá baterie

4.1.1 Vyvážené nabíjení

Vyvážené nabíjení se používá, aby se zabránilo předčasnému selhání článků ve více článkových bateriích. Princip spočívá v tom, že se při nabíjení článků kontroluje napětí na každém článku a přemosťují se jednotlivé články, aby měly všechny články stejné napětí.

4.2 Řídící jednotka

Procesorem řídící jednotky je Raspberry Pi Zero W. Tento procesor byl vybrán, protože lze při nedostatku polovodičů koupit a výpočetní výkon procesoru je víc jak dostatečný. A velkého množství knihoven a přídavných modulů dělá Raspberry velmi lákavou možnost. Jediná nevýhoda proti klasickému řešení je, že nemá integrované AD převodníky a počet PWM kanálů je o jeden menší, než je potřeba, ale tyto problémy jdou vyřešit pomocí přídavných čipů. Zato přináší možnost mít pouze jeden procesor na vykonávání všech úkonů.

Rozšiřitelnost je například: přidání kamery na záznam jízdy, přidání GPS modulu na přesný záznam pozice, sonar nebo radar na měření vzdálenosti od objektů před vozidlem. A to všechno pouze s přidáním modulu a stáhnutí pluginu do Raspberry.

Jak je psáno v úvodu této práce, návrh trpěl nedostatkem a nedostupností polovodičů, a proto se místo původního DRV8305NPHPR můstkového budiče, musel být použit STSPIN32F0251 víceúčelový kontroler určený na spínání motorů používajících 230 V usměrněných. Ale tato práce ho bude používat pouze jako můstkový budič, i když výpočetní výkon ARM Cortexu M0 by mohl být dostatečný. Ale Cortex M0 nemá FPU (aritmetickou

jednotku pro počítání čísel s pohyblivou desetinnou čárkou), proto by musely bát všechny výpočty ve formátu s pevnou řádovou čárkou.

Jsou mezi sebou připojený pomocí SPI, kde Raspberry provádí všechny výpočty a zadává povely a STM provádí samotnou komutaci a zaznamenává všechna napětí potřebná pro výpočet vektorového řízení. A snímání Hallových senzorů a měření napětí baterie je jaké zpracováno pomocí STM.

Výzvou je propojit komunikaci SPI mezi oběma zařízeními, protože Raspberry používá SPI pouze v Master režimu a software přímo nepodporuje poloduplexovou komunikaci. A jelikož vždy bude zpoždění mezi naměřením hodnot STM a výpočtem Raspberry a SPI je synchronní komunikace kdy vždy přenosový rámec začíná master. Běžné řešení tohoto problému je mít GPIO pin, který změní hodnotu, když je slave připraven poslat zprávu zpět.

Raspberry používá RaspbiOS jako svůj operační systém.

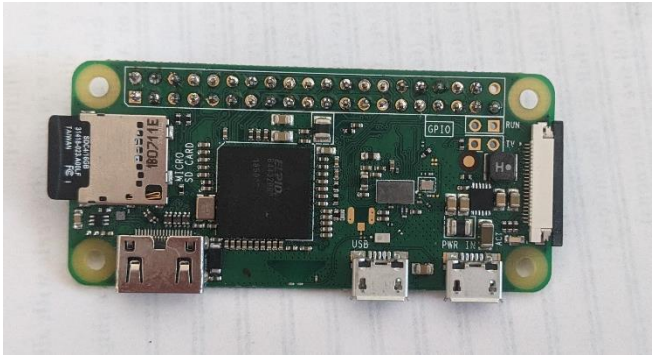

Figure 4.2 Fotka Raspberry Pi Zero W

4.3 Výkonová část

Výkonová část se měla skládat z 6 MOSFETů IRF7749L1TRPBF, které vytváří můstek pro spínání třífázového motoru. Jenže polovodičová krize zhoršila dostupnost i MOSFETů takže bylo potřeba nahradit jiným. Jako náhrada byl vybrán FDMT80060DC, který má srovnatelný parametry a byl dostupný ke koupi.

Oba dva tranzistory mají stejnou hodnotu RDS(on)[4] okolo 1,1mΩ. Kde maximální hodnoty proudu jsou víc jak 100 A [6] a vysokou spínací rychlost.

Celá výkonová část musela být odrušena pomocí velkého elektrolytického kondenzátoru a několika menších keramických, zemnící plocha výkonové části je oddělená od zemnící plochy mikrokontroleru.

4.4 Motor

Motor použitý pro demonstraci je D6368 model s 190KV a nominálním výkonem 864 W. Tento motor sice až moc výkonný pro tuto práci, ale to se dá vyřešit pomocí softwarového limitu.

Figure 4.3 Fotka motoru už přidělaná k trucku

Tento motor je osazen Hallovými sondami, které zaznamenávají polohu, tyto sondy obvykle jsou integrovány společně s výkonným zesilovačem a Schmittovým tvarovačem. A stejně podle informacích dodaných k motoru měl tohle obsahovat i motor použitý v této práci, ale tomu tak není. Motor sice obsahuje Hallovy sondy, ale podle měření se zdá, že žádný zesilovač neobsahuje, ale Schmittův tvarovač ano. Po následném testování bylo zjištěno, že se jedná o Hallův senzor s otevřeným drainem.

Jelikož jsou senzory navržené na 5 V a STM v momentální konfiguraci může měřit pouze k 3,3 V je třeba mít před AD převodník napěťový dělič.

[4] RDS(on) je odpor tranzistoru při sepnutém stavu

4.5 Snímače proudu a napětí

Přesná hodnota proudu je potřeba pro FOC algoritmus. Ale měření proudu má svoje obtížnosti, jako odpor na rezistoru musí být dostatečně malý, aby na něm byl co nejmenší úbytek napětí a aby byl schopen ustát vysoký ztrátový výkon.

Proud přímou metodou měření se zjišťuje tak, že na rezistoru malého odporu změříme úbytek napětí a jelikož známe přesnou hodnotu odporu, tak můžeme spočítat jaký proud protéká skrz rezistor.

Kdyby byl úbytek napětí až moc vysoký, tak by motor nedostával maximální možné napětí a podle rovnice:

$$P = \frac{U^2}{R}$$

Například, odpor motoru je cca 0,55Ω a když by měl bočník hodnotu 1Ω, tak by na něm bylo napětí 23 V a spotřebovával by 23 W energie. Motor by měl jen 12,77 V a výkon 296,66 W z původních 2340 W. Ale kdyby se použil odpor s malou hodnotou jako je 1 mΩ, tak ztrátový výkon by byl nízký. Celkový odpor v obvodu by se zvýšil z cca 0,55 Ω na 0,551 Ω a úbytek výkonu by byl ≈ 4,2 W.

Při snižování napětí na motoru výkon jde dolů kvadraticky za každý volt.

Proto je potřeba mít co nejnižší hodnotu odporu. Ale zesilovače mají jen určité zesílení, než jejich výstup se stane nepoužitelným, proto nemůžeme mít rezistory příliš nízké, aby zesilovač mohl správně pracovat.

Tištěný obvod je osazen 3 MAX9919FASA+T zesilovači na zesilování napětí na rezistoru mezi zátěží (motorem) a zemí. Tento zesilovač lze používat ve více funkcích. Ale nejoptimálnější funkce pro tuto práci je zesilování jen v kladném napětí a jelikož se jedná o zesilovač s fixním zesílením (45) tak není potřeba přidávat další pasivní součástky kromě kondenzátorů na filtrování napájecího napětí.

Mezi zesilovačem a rezistorem jsou dvě dolní propusti které snižují množství šumu který jde do zesilovače. Snížení šumu v tomto kroku je velice důležité, protože každá odchylka bude 45krát zesílená. I malá odchylka z důvodu šumu může znatelně výsledek řízení.

4.6 Bezpečnostní prvky

Jednou z bezpečnostních pojistek je, když Raspberry neodpoví do určité doby (dva celé cykly modulace PWM) tak STM uvede všechny tranzistory do nevodivého stavu a vyčkává, dokud se Raspberry znovu neozve.

4.7 Dálkové ovládání

Dálkové ovládání je prakticky nezbytné pro zařízení, kde by jiný uživatelský vstup by byl velmi nepraktický. A další limitace se týká ergonomiky při používání, jelikož při jízdě je třeba používat obě ruce na vyrovnávání, tak není možné používat více než jednu ruku na zadávání rychlosti.

Na přijímací straně se nachází NRF24L01 modul pro plně duplexovou komunikaci, tento modul byl vybrán díky nízké ceně, velké podpoře knihoven a možnosti propojit s různými ovladači volně dostupných ke koupi.

Jako základ pro ovladač byl použit náhradní ovladač pro autíčko na dálkové ovládání. I když ovládání mělo stejnou frekvenci, modulaci a způsob přenosů packetů, nepodařilo se spárovat ho s NRF24L01 modulem, z důvodu nemožnosti zjistit kanál a adresu trubky pro přenos rychlosti.

Po neúspěchu spárování se vnitřní ovládací logika změnila na kombinaci Arduina nano a druhého modulu NRF24L01, kde všechny mechanické části zůstanou stejné (potenciometry a baterie).

Figure 4.4 Fotka použitého ovladače

4.7.1 NRF24L01

NRF24L01 modul je určený pro bezdrátovou komunikaci mezi stejnými moduly, nebo moduly kompatibilními. Jeho velkou předností je velká flexibilnost a cena. Samotný modul používá komunikační rozhraní SPI s dvěma dalšími kontrolními piny CE a IRQ. CE je určený na přepínání mezi konfiguračním módem a RX/TX módem a IRQ po nastavení potřebných registrů signalizuje, že se v RX bufferu nachází nějaká data.

V konfiguraci lze nastavit na jakém kanálu a adrese bude probíhat komunikace mezi moduly. Z testů při hledání adresy a kanálu komerčního ovladače vyplynulo, že přibližně prvních 40 kanálů není příliš vhodných pro komunikaci, protože tyhle frekvence využívá 2,4 GHz WI-FI a může dojít k degradaci signálu na větší vzdálenosti.

I když při použití v této práci není nutná obousměrná komunikace, tak možnost požadování potvrzení o přijetí spojená s CRC dává velice nízkou ztrátovost při řízení.

4.8 Tištěný spoj

Tištěný spoj má rozměry 200 mm šířky na 100 mm výšky a 1,6 mm tlustý. Měděný povrch se nachází na obou stranách, ale deska je osazena pouze z jedné strany. Tloušťka mědí 1oz (0,089 mm) přináší problémy pro přenos velkého množství proudu, protože by šířka spoje musela být nepřijatelně široká např. pro proud 25 A a zvýšení teploty o 20 °C na vnější vrstvě by musela být šířka 43,490 mm [5]. A kvůli změně designu v průběhu nebylo dost času na správné přidělání vnějších měděných spojů na zvýšení přenosu proudu, proto výsledný maximální proud je jen 15 A. Ale některé součástky jsou schopné snášet až původně zamýšlených 50 A, jako například tranzistory, bočníky a zesilovače.

Plocha je fyzicky rozdělená na dvě části výkonovou a procesorovou. Výkonová část má vlastní zem[6] a vlastní filtraci, kromě velkého kondenzátoru na vstupu, který filtruje parazitickou induktanci kabelů vedoucích z baterie.

Na desce jsou přítomná 4 napětí: Napětí baterie, +15 V pro spínaní tranzistorů, +5 V pro zesilovače a Raspberry a +3,3 V pro STM a nRF24. 15 V se zajišťuje tím, že je připojen externí spínaný zdroj, který snižuje napětí na 15 V a jeho výstup je přiveden zpátky na tištěný obvod. 5 V a 3,3V jsou zajištěné dvěma lineárními zdroji. Na 15 V byl použit spínaný zdroj kvůli vysoké účinnosti a jelikož je první v kaskádě a napájí i obvody pro spínání tranzistoru. 5 a 3,3 jsou lineární, protože jejich výstup není výkonově náročný a tím pádem nebude ztrátový výkon na stabilizátoru neúnosně veliký. A jelikož Raspberry na 5 V má maximální spotřebu 1,1W [7] a STSPIN má spotřebu 0,06237W [8].

Na desce jsou vývody určený pro programátor ST-Link, aby se dal STM programovat. Hallovy sondy mají též svůj vývod na desce, ale STM funguje na 3,3 voltové logice, takže je potřeba napěťový dělič, aby nedošlo k poškození STM.

Raspberry má na desce plnohodnotný vývod pro přímé připojení, ale ne všechny piny jsou připojené, připojené jsou pouze potřebné piny jako: 5 V, zem, a piny pro SPI.

[5] Tato hodnota byla spočítána pomocí online nástroje https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-pcb-trace-width
[6] Zem je na desce rozdělena do dvou celků výkonová a výpočetní

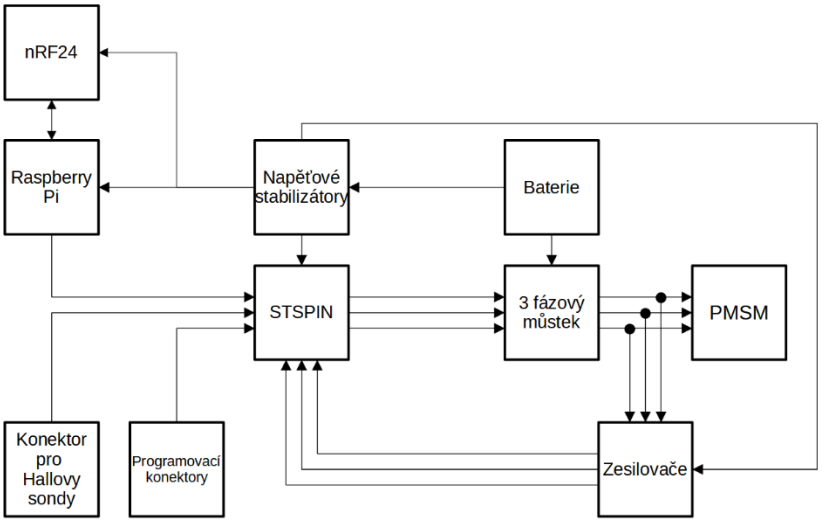

Figure 4.5 Blokové schéma prvků tištěného spoje

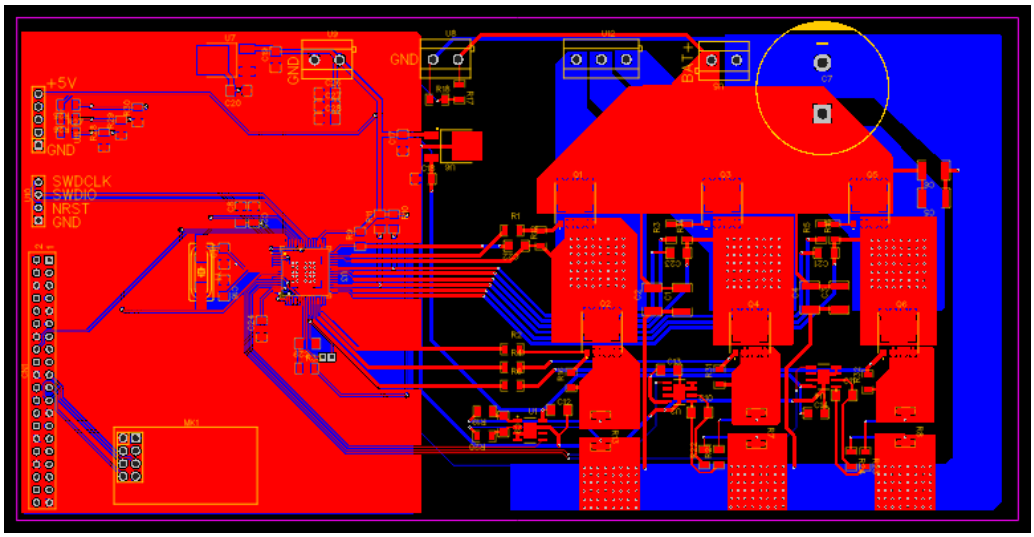

Figure 4.6 Skutečný design desky (červená je horní vrstva, modrá je spodní vrstva)

5 Popis programu

Jako hlavní řídící jednotka se používá Raspberry s RaspbiOS, díky tomu lze použít vysoký programovací jazyk jako je například Python, nebo Java. Jenže Java nemá takový množství knihoven k Raspberry jako má Python. Jenže Python má nevýhodu v oblasti rychlosti

vykonávání programu. A to v případě řízení FOC je vážný problém, protože výpočty musí být vykonány v určitém časovém rozmezí.

Jako most mezi Pythonem a C++ se používá PyBind11, který je velice jednoduchý na použití v prostředí C++. Ale velká nevýhoda je značná komplikovanost na platformě Windows, protože většina návodů a dokumentací nenabízí postup pro prostředí Windows, jen pro Unixové systémy.

5.1 Raspberry

5.1.1 C++

Třída PID je obal pro konstanty PI (D není potřeba v tomto kontextu) jejich limity a hodnoty z předchozích odchylek od zadané hodnoty. Třída obsahuje dva settery jeden pro limity a druhý pro konstantu. Jediná funkce kromě setterů je samotné provedení PI regulace.

Třída Response je kontejner pro uchovávání hodnot z přenosu a je vybavený funkcí na parsování vstupu. Tato třída je zpřístupněna přes API Pythonu jen jako třída na předávání dat mezi SPI a C++.

Soubor Transformace.h obsahuje všechny transformace potřebné pro provádění FOC a definuje strukturu polarKoord, která je určena na uchovávání polárních koordinátu vektoru řízení.

Soubor Komplex.h je pouze struktura na komplexní čísla.

Soubor main.cpp obsahuje oba dva hlavní body vstupu pro Python PWM start(Response&) a PWM process(Response&, int rychlost). Start je určený na připravení všech pomocných struktur a uvedení motoru do startovní pozice. Process je hlavní metoda FOC za tohle funkcí je schovaný celý algoritmus FOC.

5.1.2 Python

Funkce main spouští vlákno pro obsluhu STM a zajišťuje uživatelské rozhraní na zadávání rychlosti. Také nastavuje všechny potřebné věci jako jsou GPIO výstupy/výstupy a SPI konfiguraci.

Funkce regulovat(), když se spustí vlákno tak provede úvodní naklonění motoru a na 500 ms se vlákno uspí, aby měl motor čas se natočit. Po probuzení vlákna se provede přenos dat přes SPI a hodnoty se použijí do druhé start funkce. Po ukončení startu vstoupí vlákno do nekonečného cyklu čekání na data z STM, výpočtu a odesílání dat zpět.

Aby mohlo SPI fungovat musí se povolit v operačním systému.[9]

Pro komunikaci s nRF24 byla použita knihovna RF24, která zprostředkovává komunikaci mezi Raspberry a nRF24, tato knihovna je použitá i v programu Arduina. [10]

5.2 Program a konfigurace na STSPIN

Programování na platformě STM32 se skládá z dvou kroků, konfigurace zařízení a periferií a samotného programování.

5.2.1 Konfigurace

Pro konfiguraci STM32 mikrokontrolerů se používají dvě programovací prostředí STM32CUBEMX a STM32CUBE IDE, kde první se používá čistě na konfiguraci a generování předkonfigurovaných tříd pro další vývojová prostředí. STM32CUBE IDE je aplikace sdružující programovací prostředí se zmíněným konfigurátorem, programátorem a debuggerem. Pro jednoduchost instalace a následnou práci s mikrokontrolerem je doporučené používat IDE. Alternativy jako Arm Keil a podobné mají programátor s debuggerem a programovací prostředí, ale integrace s CUBEMX je prakticky nulová a uživatel musí si vyrobit vlastní způsob integrace.

Samotná konfigurace spočívala v nastavení pinů vnitřního procesoru a zdrojů pro časovou základnu. Jako zdroj frekvence se používá externí krystalický oscilátor s frekvencí 8 MHz. Uvnitř kontroléru se zvýší frekvence přes fázový závěs na 48 MHz a tahle frekvence se rozdělí do periferií. Tenhle krok je důležitý pro určení periody PWM a celkové rychlosti výpočtu.

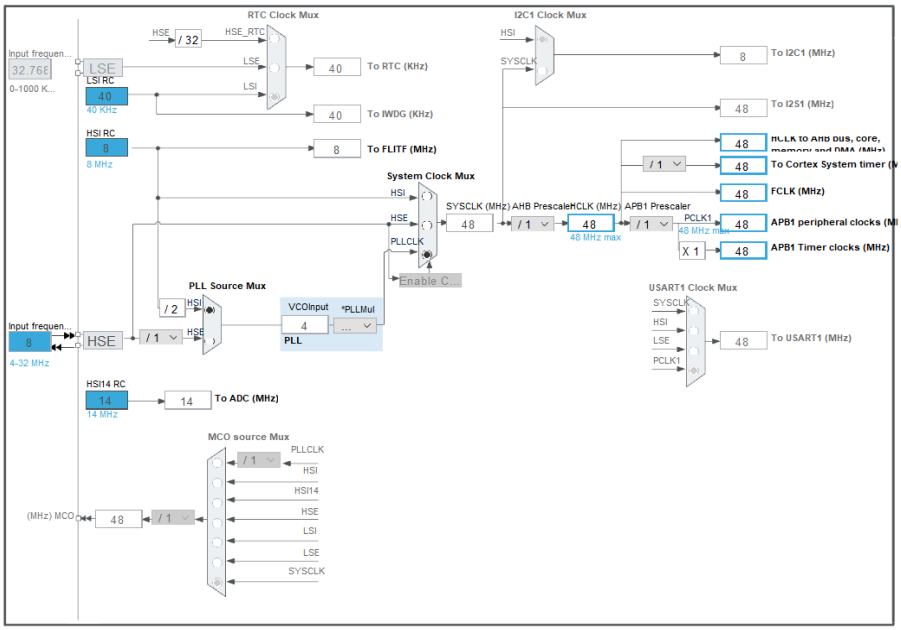

Figure 5.1 Zobrazení veškerého nastavení časových základen

Další krok je nastavit konektory, jenže konektory zobrazené v konfigurátoru neodpovídají vždy s konektory na fyzickém čipu.

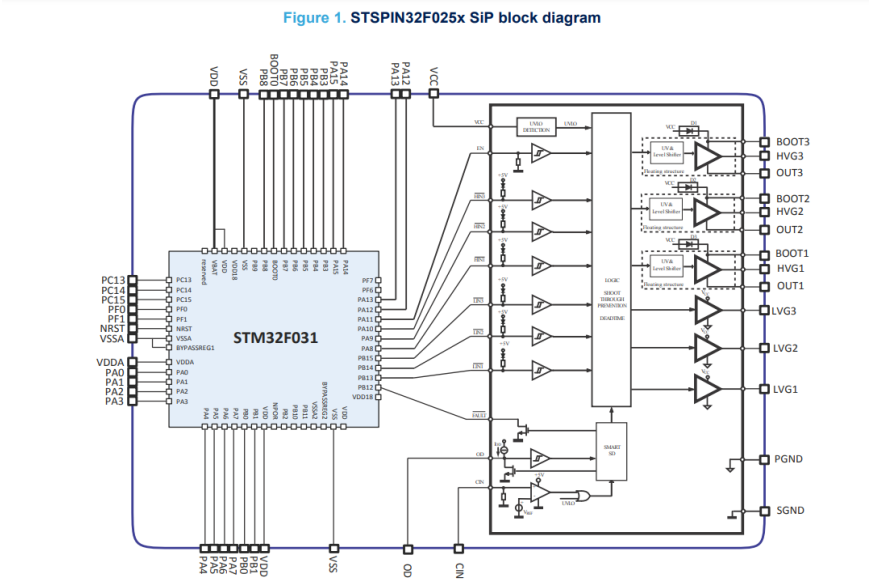

Figure 5.2 Schéma propojení mezi vnitřním mikrokontrolerem a budičem můstku [8]

Na obrázku je vidět propojení mezi vnitřním kontrolérem, budičem můstku a vnějšími piny. Například PA8 až PA18 a PB12 až PB15 slouží k předávání PWM signálu na můstek s tím,

že PA8 slouží jako spouštěč můstku a PB12 jako vstup pro případnou chybu jako například nadproud.

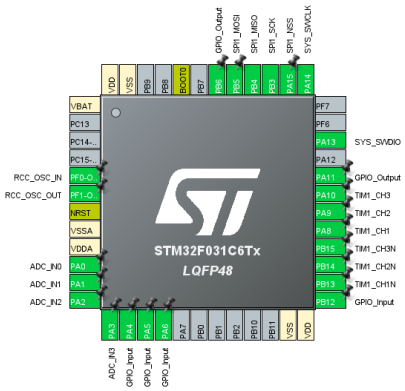

Figure 5.3 Zobrazení funkce jednotlivých výstupů z vnitřního mikrokontroleru

Piny PA0 až PA3 jsou nakonfigurovány jako vstupy do AD převodníku, piny PA0 až PA2 zaznamenávají napětí z bočníkových zesilovačů a pin PA3 zaznamenává napětí na baterii. Piny PA4 až PA6 jsou digitální vstupy pro Hallovy sondy. PF0 a PF1 jsou vstupy oscilátoru. PB3 až PB6 a PA15 jsou používány na komunikaci přes SPI, kde PA15 je hardwarový CS a PB6 je pin pro indikaci, že proběhl další cyklus. PA14 a PA13 jsou používány dohromady s NRS pinem k programování a debuggování.

S nakonfigurovanými výstupy se musí přejít ke konfiguraci jednotlivých periferií. AD převodník musí být nakonfigurovaný tak, aby měřil ve správný čas při dosažení periody cyklu. Další na konfiguraci je potřeba užití přímého přístupu do paměti (DMA) a jaký DMA kanál má používat.

Stejně tak TIM1 časovač je potřeba nakonfigurovat. Konfigurace se skládá z dvou částí obecné konfigurace celého časovače a konfigurace pro jednotlivé kanály. V obecné konfiguraci se nastavuje frekvence, konstanta dělení vstupní frekvence (prescaler), směr přičítání a dead time[7]. Frekvence se nastavuje přes periodu, že lze nastavit kolik cyklů je potřeba, aby uběhla 1 perioda. Každý kanál lze samostatně konfigurovat, ale v prvotní

[7] Dead time je čas který se vkládá do přepínání můstku, aby nedošlo ke zkratu důsledkem pomalého rozepnutí tranzistoru.

konfiguraci musí mít všechny kanály stejnou konfiguraci. Nastavení obsahují módy modulace, periodu pro střídu, polaritu výstupů při vysoké a nízké hodnotě.

5.2.2 Program na obsluhu

Program se nachází na dvou místech, buď v mainu, nebo v různých obsluhách přerušení. V mainu se nachází pouze inicializační části programu jako HAL_TIM_PWM_Start pro konfiguraci kanálu PWM, nebo nastavení správné hodnoty na výstupu pro Raspberry.

HAL_SPI_TxRxCpltCallback je obsluha vyvolaná po přenosu všech dat a následnému přesunu do paměti. Při vykonání této funkce se nastavují střídy jednotlivých kanálů a zároveň tato funkce zajišťuje vypnutí modulace v případě zamrznutí Raspberry.

HAL_ADC_ConvCpltCallback funkce se spouští po dosažení periody čítače TIM1 a konverzi hodnot. Následně funkce přečte všechny analogové hodnoty bočníků a baterie, a následně přečte digitální hodnoty Hallových sond a převede tyto hodnoty do bufferů SPI ve formátu specifikovaným v kapitole 5.6. PWM přenos signálu.

5.3 Arduino

Program na arduinu má za úkol číst z potenciometru napětí a převést ho do procentní formy. Následně pokud se nová hodnota liší od té staré tak se pošle na Raspberry. Tento cyklus se opakuje 10krát za vteřinu.

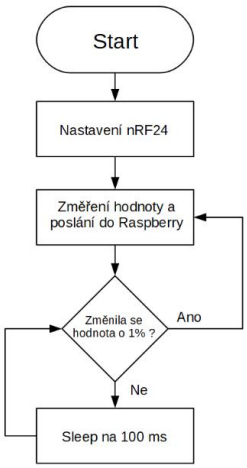

Figure 5.4 Vývojový diagram programu na Arduinu

5.4 PID v softwaru

PID regulátor je implementován bez použití externích knihoven. V případě této práce jsou implementovány pouze proporcionální a integrační části regulátoru, protože derivační část není potřeba ve vektorovém řízení. Implementace má možnost omezit maximální a minimální výstupní akční veličinu a omezit wind-up integračního členu. Kp a Ki konstanty byly určeny experimentálně.

5.5 Přepočet hodnot z transformací na PWM signál

Pro tento krok lze použít vícero způsobů, jak docílit modulace. Nejpřímější způsob je použít inverzní Clarkovu transformaci a s modulací na trojúhelníkový signál. Ale lze použít i způsob, který přímo vychází z výstupu inverzní Parkovi transformace. Tento způsob se odlišuje tím, že nejdřív zjistí, v jakém sektoru se rotor momentálně nachází a až po zjištění se začnout přepočítávat hodnoty a přesné časy obou nulových vektorů se dopočtou až nakonec. Tento způsob je jednodušší tím, že nepotřebuje modulovat trojúhelníkový signál.

Celé rozlišování sektoru je napsané pomocí struktury switch-case, která poskytuje vysokou rychlost, protože umožňuje kompilátoru vyrobit jednoduchý jump table. Kdyby struktury switch-case nahrazena rozvětveným if-else, tak by se výkon snížil, protože jednoduchý jump table je pro cache procesoru jednodušší načíst dopředu oproti rozvětvenému programu který vzniká při použití if-else.

5.6 PWM přenos signálu

V původním plánu bylo použít Waveshare Servo Driver HAT na pohánění budiče můstku. V jádru tohohle modulu je PCA9685 určený pro pohánění LED pásků. Což by nebyl problém i když šel adresovat jen po jednom jeden výstupu, protože pokud bude odchylka proti předchozímu stavu dostatečně malá tak se vektor momentu motoru posune nepatrně od ideálu.

Ale po předělání je PWM signál přenášen přes SPI na STM. Přenos funguje tak že, když STM pošle data o napětí, tak po krátké prodlevě způsobené výpočtem pošle Raspberry data zpět.

Datové rámce pro komunikaci musí rozdělit 12bitové hodnoty po 8bitových celcích. A pro ušetření dat na přenesení se 4bitové zbytky seskupí dohromady do jednoho 8bitového celku. Jako poslední blok se přenáší stav 3 Hallových sond, tyto data jsou už v digitálním formátu proto potřebují jen 3 bity. Při 4 AD převodnících a jednom bajtu pro Hallovy sondy je výsledná délka 7 bajtů.

Seskupení přebytkových 4bitových hodnot je vždy stejné, horní 4 bity jsou rezervované pro předchozí bajt a spodní 4 bity jsou pro následující bajt.

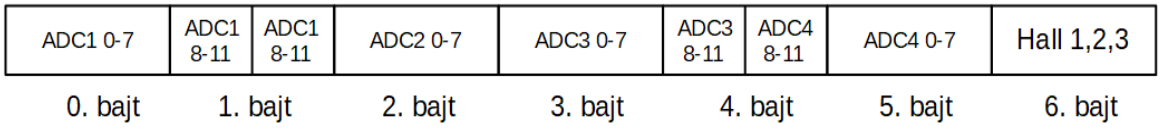

Figure 5.5 Rozložení dat ve přenosu ze STSPINu

Pro komunikaci od Raspberry do STM se používá jiný datový rámec, protože všechny 3 hodnoty PWM co se posílají zpět musí být 16bitové, musí se zase dělit do 8bitových bloků. Inspirace pro tento rámec vychází z formátu Big Endian kde bajty s vyšším pořadím se dávají mají přednost před nižšími.

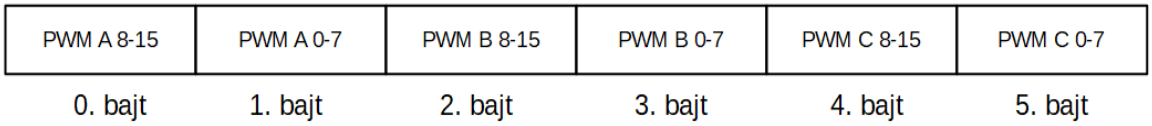

Figure 5.6 Rozložení dat ve přenosu z Raspberry

Největší problém přenosu je dostatečná šířka pásma na přenos všech dat, protože za každý cyklus se přenese 13 bajtů a to při 20 kHz je 260 kB (254 KiB). Tohle by mohlo být problém při zvyšování frekvence pro přesnější a jemnější řízení motoru.

5.7 Řídící smyčka na STSPIN

Řídící smyčka na mikrokontroleru STM32F031C6Tx obsažený v STSPIN32F0251 je vázaná na frekvenci modulace PWM signálu. Běžná frekvence pro modulaci je 20kHz,

protože je za hranicí slyšitelnosti pro většinu lidí. Perioda 20kHz je 50 mikrosekund to znamená, že program vykonává veškeré úkony v tomto časovém rámci.

V každý rámec začíná, když čítač PWM se dostane na maximální hodnotu před počítáním směrem dolů, v tomto momentu lze měřit všechny proudy, a proto rámec začíná zde. První věc, co se provede je změření všech napěťových hodnot (3 proudy z bočníků, napětí baterie a stav Hallových senzorů), poté se čeká na převod analogových hodnot na digitální, kromě Hallových senzorů, které používají digitální logiku. Následný krok je připravit přenosový rámec SPI a oznámit Raspberry připravenost na odeslání hodnot. Přenosový rámec je popsaný v předchozí kapitole.

Potom se nastaví vnitřní příznak pro čekání na hodnotu a ukončí se první přerušení zavolané dosažením periody PWM. STM čeká na odezvu od Raspberry a když žádná odezva nepřijde ani v dalším přerušení odpojuje můstkový budič od energie a motor začne zpomalovat. Ale v moment, když odpoví Raspberry včas, tak se příznak vynuluje a hodnoty PWM se aplikují na PWM pro další cyklus.

5.7.1 Přepočty hodnot

Velké množství hodnot vyprodukované STM se musí přepočítat do použitelných hodnot. AD převodníky měřící proud skrz motor dávají hodnotu naměřenou na předzesilovačích od bočníků. Proud se vypočte jako:

$$I = X \cdot \frac{U_{ADC}}{2^{12}} \div 45 \div R \tag{5}$$

Kde X je hodnota z AD převodníku, R je hodnota bočníku, 2 na 12 je rozlišení AD převodníku, 45 je zesílení předzesilovače a U_{ADC} je napětí pro AD převodník. Známé hodnoty jsou: napětí pro AD převodník je 3,3 V a odpor bočníku 1 mΩ.

Po dosazení známých hodnot rovnice vypadá takto:

$$I = X \cdot \frac{3{,}3}{2^{12}} \div 45 \div 0{,}001 \tag{6}$$

A druhá analogová hodnota napětí na bateriích se přepočítává pomocí rovnice 5, ale kvůli rozsahu AD převodníku se musí napětí snižovat přes napěťový dělič, který má hodnoty 820kΩ a 68kΩ.

$$U = X \cdot \frac{U_{ADC}}{2^{12}} \cdot \frac{820 + 68}{68} \tag{7}$$

Kde X je hodnota z AD převodníku a U_{ADC} je 3,3 V.

PWM modulace je zadávána 16bitovou hodnotou, která má hodnotu periody 1200. Prescaler je nastavený na 1, aby při 48 MHz 1200 cyklů dohromady snížilo frekvenci na 20 kHz. Ale hodnota 1200 je dosažená tím, že PWM je středově symetrická a jedna perioda jsou 2 periody při normální pilové modulaci.

6 Fyzická konstrukce

Jako dopravní prostředek pro demonstraci byl zvolen longboard, protože má dostatečnou velikost na všechny potřebné součásti a na rozdíl od kola nebo koloběžky je konstrukčně jednodušší na výrobu.

Longboard je zkonstruovaný z 20 mm prkna uříznutého na délku 1 metr a šířkou 18,5 cm. Prkno má zakulacené konce a přebroušený povrch. Jako trucky jsou použity Puente trucky zakoupené na Aliexpressu, k nim byly dokoupený úchyt motoru, který se dá přichytit na truck. Na přenos výkonu na kolečka se je pomocí pastorku HTD5M 15T se šířkou 15 mm a 10 mm vnitřním obvodem a 16 mm vnějším obvodem. Dále řemenice HTD 5M o šířce 15 mm a délce 570 mm. Na kolečku je přichycené ozubené kolečko s 36 zuby.

Některé z dílů bylo potřeba přizpůsobit pro tuto konkrétní aplikaci. U ozubeného kolečka byla potřeba srazit hrana, aby se vešlo do kolečka. Kolečko bylo potřeba provrtat, aby se tam vešly šrouby na upevnění kolečka, na opačné straně kolečka se z epoxidu odlila podložka pro matky. Samotný držák nebyl schopný pevně upevnit držák, a proto bylo potřeba vyříznou nové díry a drážky.

Jako obal pro baterie a řídící desku byla použita krabice od klávesnice, kde jsou komponenty jsou upevněny suchým zipem. Celý obal je upevněn k prknu pomocí suchých zipů.

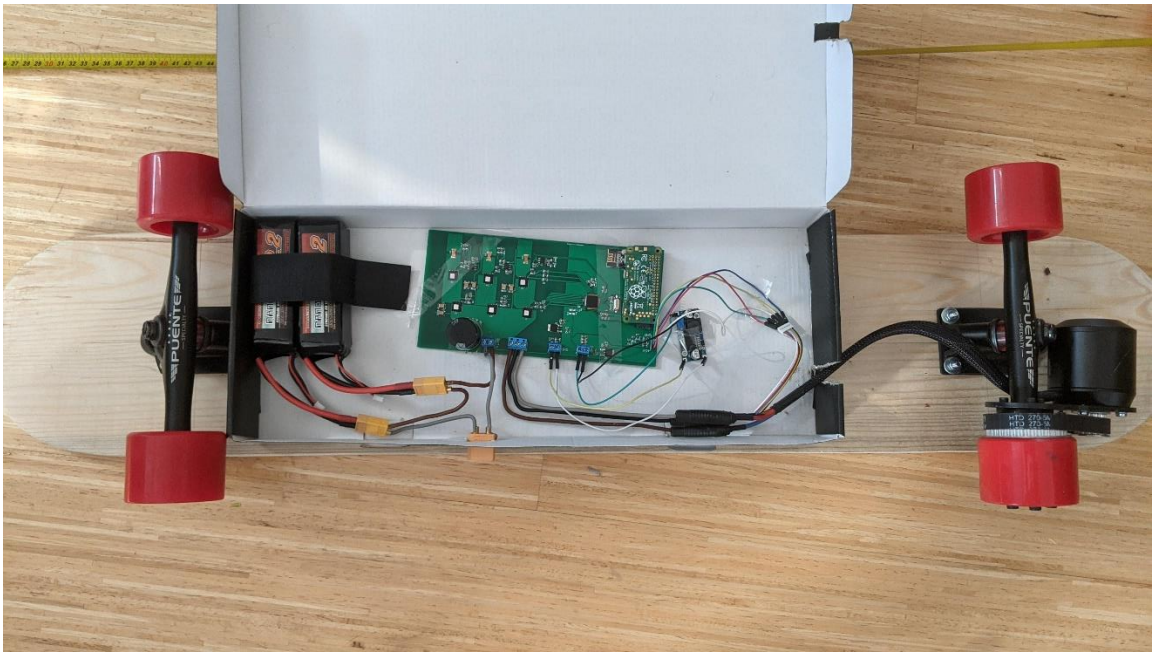

Figure 6.1: Pohled na kompletní výrobek

Figure 6.1: Pohled na longboard zboku

Figure 6.3: Pohled seshora

7 Testování

Testování celého sestavení je velmi důležité z bezpečnosti pasažéra, protože porucha může být od ztráty výkonu ve špatný moment až po nemožnost zastavení a následné nehodě. Dále jako důležité kritérium je odezva na povely uživatele.

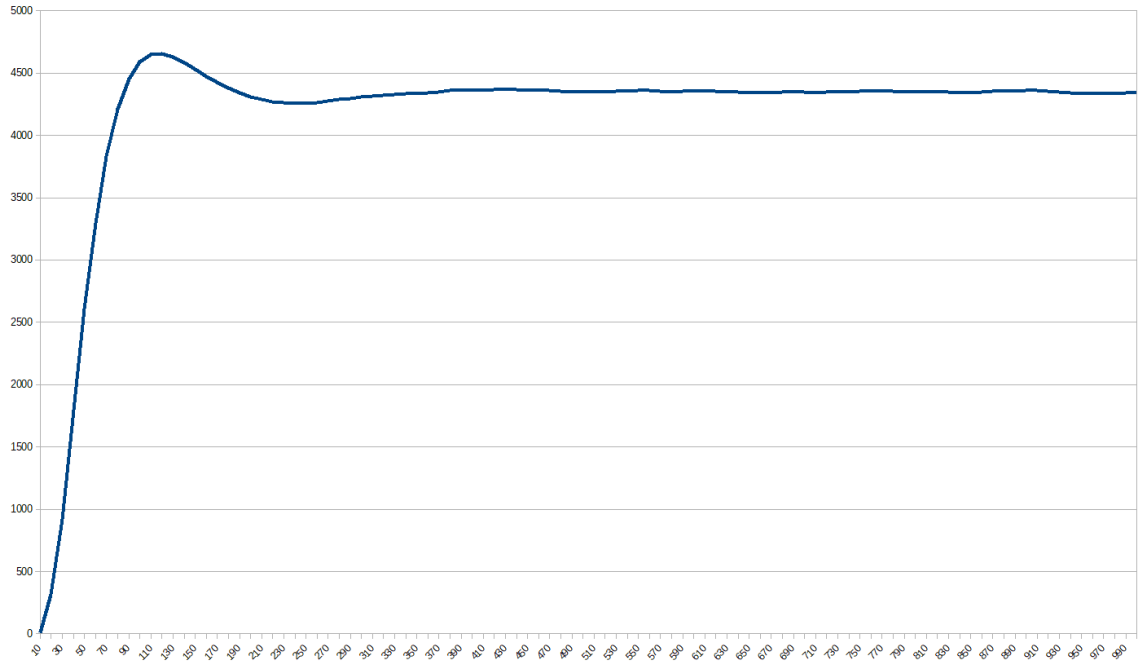

Figure 7.1: Graf rozběhu motoru bez zátěže, kde vodorovná osa je čas v ms a svislá osa jsou mechanické otáčky

Na Figure 7.1 lze vidět rozběh motoru naprázdno a taky je vidět překmit způsobený právě během naprázdno. PI regulátor byl experimentálně nastaven na běh motoru se zátěží.

Jako další byl proveden test při odpojení Raspberry od zbytku systému za běhu a ověření funkčnosti bezpečností pojistky. U tohoto testu je jednoduchý zjistit úspěšnost, ale díky odpojení Raspberry není možné změřit žádná data, protože všechny výpočty se odehrávají na Raspberry.

Jeden z problémů, co se objevil je nedostatečné větrání v krabici, kde je všechno upevněno, některé komponenty (stabilizátory a tranzistory) se pomalu přehřívají. Tady je zase problém s možností měření teploty, protože jediný způsob dostupný je použít bezkontaktní teploměr určený na běžné měření teploty u lidí Dr. Max ThermoMAX Comfort s maximální teplotou

cca 50 °C. Proto maximální doba provozu je omezena. A proto v další iteraci krabice budou větrací otvory pomáhající s udržením teploty.

8 Závěr

Tato práce měla za cíl vytvořit funkční prototyp vozidla osobní přepravy za použití vektorového řízení běžícího na hardwaru speciálně vyrobeného pro něj. Co se týká vektorového řízení tak tato část po úspěšném vyzkoušení fungovala. Hardwarová stránka nebyla plně úspěšná z důvodu nedostatku všech komponentů, co by byly potřeba na méně komplexní řešení. A mechanická stránka věci je pouze dostatečná z důvodu podcenění potřeby chlazení při dlouhodobějším používání.

Na základě výše zmíněných závěrů by se dalo říct, že cíle práce byly splněny.

Seznam zkratek

AD – Analogově digitální

API – Application programing interface (Programovací rozhraní aplikace)

BLDC – Brushless DC (Bezkartáčový stejnosměrný motor)

CS – Chip select (Výber čipu)

DMA – Direct memory access (Přímý přístup do paměti)

FOC – Field oriented control (Vektorové řízení)

FPU – Floating point unit (Jednotka na výpočet s plovoucí desetinnou čárkou)

GPIO – General purpose I/O (Víceúčelový vstup/výstup)

PMSM – Permanent magnet synchronous machine (Synchronní motor s permanentními magnety)

MOSFET – Metal Oxide Semiconductor Field Effect Transistor (Tranzistor řízený polem)

NiMH – Nickel Metal Hydride baterie

nRF24 – NRF24L01

PI – Proporcionálně integrační

PID – Proporcionálně integračně derivační

PWM – Pulse width modulation (Pulzně šířková modulace)

Raspberry – Raspberry Pi Zero W

RX – Receive (Příjem)

SNR – Signal to noise ration

SPI – Serial Peripheral interface

STM – STM32F031C6Tx

STSPIN – STSPIN32F0251

TX – Transmit (Odesílání)

Seznam použité literatury

[1] ABACAN, Aldrin. Sensored 3-Phase BLDC Motor Control Using Sinusoidal Drive [online]. Microchip Technology, 2020 [cit. 2022-04-11]. Dostupné z: http://ww1.microchip.com/downloads/en/Appnotes/00003453A.pdf

[2] SOLBAKKEN, Yngve. VECTOR CONTROL FOR DUMMIES [online]. Switchcraft, 2017 [cit. 2022-04-11]. Dostupné z: https://www.switchcraft.org/learning/2016/12/16/vector-control-for-dummies

[3] Park, Inverse Park and Clarke, Inverse Clarke Transformations MSS Software Implementations User Guide [online]. Microsemi Corporation, 2013 [cit. 2022-04-11]. Dostupné z: https://www.microsemi.com/document-portal/doc_view/132799-park-inverse-park-and-clarke-inverse-clarke-transformations-mss-software-implementation-user-guide

[4] SOLBAKKEN, Yngve. SPACE VECTOR PWM INTRO [online]. Switchcraft, 2017 [cit. 2022-04-11]. Dostupné z: https://www.switchcraft.org/learning/2017/3/15/space-vector-pwm-intro

[5] TORRES, Daniel a Jorge ZAMBADA. Single-Shunt Three-Phase Current Reconstruction Algorithm for Sensorless FOC of a PMSM [online]. Microchip Technology, 2009 [cit. 2022-04-11]. Dostupné z: https://ww1.microchip.com/downloads/en/Appnotes/01299A.pdf

[6] Datasheet – FDMT80060DC [online]. Fairchild semiconductor corporation, 2015 [cit. 2022-04-11]. Dostupné z: https://cz.mouser.com/datasheet/2/308/1/FDMT80060DC_D-2312771.pdf

[7] How much power does Pi Zero W use? [online]. RasPi.TV, 2017 [cit. 2022-04-11]. Dostupné z: http://raspi.tv/2017/how-much-power-does-pi-zero-w-use

[8] Datasheet – STSPIN32F0251 [online]. STMicroelectronics, 2021 [cit. 2022-04-11]. Dostupné z: https://www.st.com/resource/en/datasheet/stspin32f0251.pdf

[9] HYMEL, Shawn a Byron J. Raspberry Pi SPI and I2C Tutorial [online]. Sparkfun, 2018 [cit. 2022-04-11]. Dostupné z: https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial/all

[10] Optimized high speed nRF24L01+ driver class documentation [online]. 2021 [cit. 2022-04-11]. Dostupné z: https://nrf24.github.io/RF24/

[11] Datasheet – STM32F031x6 [online]. STMicroelectronics, 2021 [cit. 2022-04-11]. Dostupné z: https://www.st.com/resource/en/datasheet/stm32f031k6.pdf

[12] Datasheet – nRF24L01 [online]. Nordic, 2008 [cit. 2022-04-11]. Dostupné z: https://www.sparkfun.com/datasheets/Components/SMD/nRF24L01Pluss_Preliminary_Product_Specification_v1_0.pdf

[13] Best Practices for Board Layout of Motor Drivers [online]. Texas Instruments, 2021 [cit. 2022-04-11]. Dostupné z: https://www.ti.com/lit/an/slva959b/slva959b.pdf?ts=1649707056158

[14] Datasheet – MAX9919 [online]. Maxim Integrated Products, 2021 [cit. 2022-04-11]. Dostupné z: https://datasheets.maximintegrated.com/en/ds/MAX9918-MAX9920.pdf

[15] PID Controller Implementation in Software [online]. 2020 [cit. 2022-04-13]. Dostupné z: https://www.youtube.com/watch?v=zOByx3Izf5U

Seznam příloh

Příloha 1 – Soubory programu

Příloha obsahuje všechny programy roztříděných do 3 složek:

- Složka s programem na Raspberry
    - start.py zdrojový kód obsluhy
    - složka cpp, která obsahuje všechny soubory potřebné pro vyrobení sdílené knihovny
- Složka s konfigurací a programem STSPINu

- Složka s programem Arduina

Příloha 2 – Výkres tištěného spoje

Příloha obsahuje kompletní výkresy a schémata, které byly potřeba k výrobě tištěného spoje. Tyto výkresy jdou otevřít pouze v programu EasyEDA.

Příloha 3 – Výroba sdílené knihovny

Pro výrobu sdílené knihovny je potřeba naistalovat PyBind11 přes manažera balíčků v RaspbiOS a potom ve stejném adresáři jako je start.py spustit následující příkaz:

c++ -O3 -Wall -shared -std=c++11 -fPIC $(python3 -m pybind11 --includes) *.cpp *.h -o FOC$(python3-config --extension-suffix)