VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Tomáš Nepivoda

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

DRIVER KROKOVÉHO MOTORU S PODPOROU PRŮMYSLOVÉ KOMUNIKACE
STEPPER MOTOR DRIVER EQUIPPED WITH INDUSTRIAL COMMUNICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Tomáš Nepivoda

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Jakub Arm, Ph.D.

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Tomáš Nepivoda ***ID:*** 211165

Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Driver krokového motoru s podporou průmyslové komunikace

POKYNY PRO VYPRACOVÁNÍ:

Úkolem práce je navrhnout a vytvořit driver ke krokovému motoru, který bude ovládán pomocí jedné či více z průmyslových komunikací (CANOpen, EtherCAT, Profinet, Powerlink). K tomu je potřeba vybrat vhodné komponenty a definovat vhodný koncept.

1) Proveďte literární rešerši v oblasti driverů motoru a průmyslových komunikačních sítí v embedded oblasti.
2) Definujte požadavky na vytvářené zařízení.
3) Navrhněte vhodnou koncepci.
4) Realizujte a oživte hardwarovou část.
5) Implementujte softwarové vybavení.
6) Odzkoušejte a vyhodnoťte funkčnost pomocí ovládacího zařízení.

DOPORUČENÁ LITERATURA:

Zurawski, R. Embedded Systems Handbook: Networked Embedded Systems. CRC Press, 2017. ISBN: 9781315218298.

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Jakub Arm, Ph.D.

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

Abstrakt

Cílem této bakalářské práce je navrhnout a zrealizovat driver pro krokové motory, který bude možné ovládat nadřazeným systémem pomocí vybrané moderní průmyslové komunikační sítě. Základem práce je teoretická rešerše na téma drivery krokových motorů a průmyslové komunikační sítě v oblasti embedded systémů. Na základě teoretických poznatků je vybrána, a podrobněji popsána, vhodná komunikační síť, a vytvořen koncept navrhovaného driveru. Dále následuje popis návrhu a realizace hardwarového vybavení a následně implementace řídícího firmwaru. Poslední část práce je věnována vytvoření jednoduchého master zařízení, pomocí kterého je zrealizovaný driver vhodně otestován.

Klíčová slova

Krokový motor, driver krokového motoru, průmyslová komunikační síť, EtherCat, mikrokontrolér, vestavěné zařízení

Abstract

The main goal of this bachelor thesis is to design and realize driver for stepper motor, which could be controlled by supervising system using modern industrial communication network. The basis od this thesis is theoretical study on the subject of stepper motor drivers and industrial communication in the field of embedded systems. This theoretical study is then used for selecting the proper communication network and creating concept od the stepper motor. Thesis further describes design procedure and realization of hardware components and subsequent implementation of frimware. Last part of the thesis is dedicated to the creation of simple master device, which would be used for driver testing.

Keywords

Stepper motor, stepper motor driver, industrial communication network, EtherCat, microcontroller, embedded system

Bibliografická citace

NEPIVODA, Tomáš. Driver krokového motoru s podporou průmyslové komunikace. Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/134771. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce Jakub Arm.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	*Tomáš Nepivoda*
VUT ID studenta:	*211165*
Typ práce:	*Bakalářská práce*
Akademický rok:	*2020/21*
Téma závěrečné práce:	*Driver krokového motoru s podporou průmyslové komunikace*

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: 24. května 2021 ----------

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Jakubovi Armovi Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: 24. května 2021 ----------

podpis autora

Obsah

SEZNAM OBRÁZKŮ **9**

SEZNAM TABULEK **11**

ÚVOD **12**

1 TEORETICKÁ ČÁST **13**

1.1 KROKOVÉ MOTORY 13
1.1.1 Typy a jejich konstrukční uspořádání *14*
1.1.2 Porovnání jednotlivých typů *16*
1.1.3 Typické parametry *16*
1.1.4 Typická zapojení *17*
1.1.5 Základní principy řízení *18*
1.2 DRIVERY PRO ŘÍZENÍ KROKOVÝCH MOTORŮ 20
1.2.1 Popis základní funkce *20*
1.2.2 Rozdělení driverů do specifických skupin *21*
1.2.3 Typické parametry *22*
1.2.4 Řízení v otevřené a uzavřené smyčce *24*
1.2.5 Rychlostní rozběhové profily *26*
1.2.6 Driver R272-42(80)-ETH *28*
1.2.7 Driver N5 s podporou průmyslových sítí *30*
1.2.8 Mikroprogramovatelná jednotka R356 ..... *31*
1.2.9 Driver Pololu 36V4 ..... *33*
1.3 PRŮMYSLOVÉ KOMUNIKAČNÍ SÍTĚ 34
1.3.1 Rozdělení průmyslových sběrnic a jejich komunikačních protokolů *34*
1.3.2 EtherCat *35*
1.3.3 Powerlink *36*
1.3.4 Profibus *37*
1.3.5 Průmyslový ethernet pomocí Netx52 ..... *38*

2 PRAKTICKÁ ČÁST **39**

2.1 POPIS A POŽADAVKY NA VÝSLEDNÉ ZAŘÍZENÍ 39
2.1.1 Hlavní požadavky *39*
2.1.2 Požadované vnější vstupy a výstupy *40*
2.1.3 Požadované parametry *40*
2.2 BLOKOVÉ SCHÉMA 41
2.3 ETHERCAT KOMUNIKACE PODROBNĚJI 42
2.3.1 Možnosti topologie *42*
2.3.2 Požadavky na vybavení pro fyzickou vrstvu *42*
2.3.3 Požadavky na HW vybavení master zařízení *42*
2.3.4 Požadavky na HW vybavení slave zařízení *43*
2.3.5 Struktura EtherCat rámce *44*
2.3.6 Možnosti adresování *45*
2.3.7 Možnosti synchronizace *45*
2.3.8 EtherCat stavový automat *46*
2.3.9 Možnosti rozšířených profilů *47*

2.4 NÁVRH HARDWAROVÉ REALIZACE ... 48
2.4.1 Podrobné blokové schéma ... 48
2.4.2 Řídící mikrokontrolér STM32F446RET6 ... 49
2.4.3 Řídící servo-kontrolér TMC4361 ... 50
2.4.4 Budič krokového motoru TMC262 s externím mosfet H-můstkem ... 52
2.4.5 Komunikační obvod LAN9252 ... 54
2.4.6 Kompletní elektrické zapojení ... 55
2.4.7 Indikační panel s diodami ... 56
2.4.8 Konstrukční uspořádání prototypu ... 57
2.5 NÁVRH ŘÍDÍCÍHO FIRMWARU ... 59
2.5.1 Prostředí PlatformIO a framewrok Mbed ... 59
2.5.2 Popis hlavní smyčky ... 60
2.5.3 UML diagram rozvržení tříd ... 61
2.5.4 Struktura z pohledu jednotlivých vrstev a datového toku ... 62
2.5.5 Class – EasyCat ... 63
2.5.6 Class – EtherCatDatagram ... 64
2.5.7 Class– TMC4361_SPI_Adapter ... 65
2.5.8 Class – TMC4361 ... 67
2.5.9 Class - StepperMotor ... 73
2.5.10 Namespace – StepperMotorControl ... 74
2.6 OVLÁDACÍ ETHERCAT MASTER ... 77
2.6.1 Dostupný software a jeho možnosti ... 77
2.6.2 Řídící master aplikace pomocí prostředí Codesys ... 77
2.6.3 Vizualizace - Hlavní okno ... 80
2.6.4 Vizualizace - Testovací režim ... 81
2.6.5 Vizualizace - Základní režim řízení ... 82
2.6.6 Vizualizace - Pokročilý režim řízení ... 83
2.6.7 Vizualizace - Režim řízení na rychlost nebo polohu v uzavřené smyčce ... 84
2.6.8 Vizualizace – Zobrazování provozních hodnot ... 85
2.7 TESTOVÁNÍ VÝSLEDNÉHO ZAŘÍZENÍ ... 86
2.7.1 Požadavky na testování driveru ... 86
2.7.2 Krokový motor 17HS2408S a ABN enkodér S3806 ... 86
2.7.3 Doba propagace EtherCat rámce ... 87
2.7.4 Analýza EtherCat rámce ... 88
2.7.5 Měření skutečných otáček ... 89
3 ZÁVĚR ... 91
LITERATURA ... 93
SEZNAM SYMBOLŮ A ZKRATEK ... 97
SEZNAM PŘÍLOH ... 98

SEZNAM OBRÁZKŮ

1.1 Princip krokového motoru 13
1.2 Rozdělení krokových motorů....14
1.3 Krokový motor s pasivním rotorem [1] 14
1.4 Krokový motor s aktivním rotorem [1]....15
1.5 Hybridní krokový motor [1]....15
1.6 Bipolární zapojení krokového motoru [3]....17
1.7 Unipolární zapojení krokového motoru [3]....17
1.8 Průběh proudu při režimu mikrokrokování 1/4 [5] 19
1.9 Základní princi řízení motoru pomocí driveru [3] 20
1.10 Rozdělení driverů podle specifických vlastností....21
1.11 Vliv napětí při velkých rychlostech na momentovou charakteristiku [5] 23
1.12 Vliv napětí při malých rychlostech na momentovou charakteristiku [5] 23
1.13 Princip řízení v otevřené smyčce 24
1.14 Princip řízení v uzavřené smyčce....25
1.15 Porovnání vlastností při řízení v otevřené a uzavřené smyčce [6] 25
1.16 Porovnání průběhu dynamického momentu v závislosti na rychlosti motoru [6] 26
1.17 Rozběhový profil typu obdélník [7]....27
1.18 Rozběhový profil typu lichoběžník [7] 27
1.19 Rozběhový profil typu s-křivka [7]....27
1.20 Ukázka driveru R272-42-ETH [9] 28
1.21 Princip softwaru SMC-Program [9]....29
1.22 Rozdělení N5 driverů [10] 30
1.23 Ukázka N5 driveru s podporou průmyslových sítí [10] 30
1.24 Blokové zapojení pro konfiguraci a programování R356 [11]....32
1.25 Ukázka mikroprogramovatelné jednotky R356 [11]....32
1.26 Ukázka driveru Pololu 36V4 [12] 33
1.27 Rozdělení průmyslových sítí dle zařízení a aplikace [15]....34
1.28 Komunikační model EtherCat [15] 35
1.29 Průběh EtherCat rámce sítí [16]....35
1.30 Komunikační model Powerlink [15]....36
1.31 Základní architektura sítě Ethernet Powerlink [13] 36
1.32 Komunikační model Profibus [15]....37
1.33 Venšjí a vnitřní popis obvodu NetX 52 [19]....38
2.1 Vnější popis driveru....39
2.2 Základní blokové schéma 41
2.3 Obecná struktura ESC [22] 43
2.4 Struktura EtherCat rámce [24] 44
2.5 EtherCat - typy adresování a struktura adres [22]....45
2.6 EtherCat stavový automat [23] 47
2.7 Podrobné blokové schéma 48
2.8 MCU STM32F446RET6 a příslušný vývojový shield NUCLEO-F446RE [25] 50
2.9 Bloková struktura obvodu TMC4361 [27]....51
2.10 Servo kontrolér TMC4361 a příslušný vývojový shield TMC4361-BOB [27][26]....52
2.11 Budič TMC-262 a příslušný vývojový shield TMC262-BOB40 [28][29]....53
2.12 Komunikační obvod LAN9252 a příslušný vývojový shield EASYCAT [30][31] 54
2.13 Elektrické propojení řídící jednotky a budiče 55

2.14 Elektrické propojení řídící jednotky a komunikační jednotky 56
2.15 Indikační panel....56
2.16 Univerzální krabička S-BOX516P a vsazené vstupy/výstupy 57
2.17 Propojení vývojových shieldů Nucleo F446RE a EasyCat 57
2.18 Plošný spoj pro moduly TMC4261-BOB a TMC262-BOB....58
2.19 Konstrukční uspořádání vytvořené krabičky prototypu driveru....58
2.20 Ukázka projektu pomocí PlatformIO 59
2.21 Vývojový diagram hlavní smyčky 60
2.22 UML diagram rozvržení tříd 61
2.23 Digram jednotlivých vrstev a datového toku firmwaru 62
2.24 UML diagram pro třídu EasyCat 63
2.25 Datový rámec TMC4361 65
2.26 UML diagram pro třídu TMC4361_SPI_Adapter....66
2.27 UML diagram pro třídu TMC4361 69
2.28 UML diagram pro třídu StepperMotor....73
2.29 UML diagram pro jmenný prostor StepperMotorControl 74
2.30 Přechodový diagram hlavního stavového automatu....74
2.31 Struktura Codesys projektu 78
2.32 Konfigurace EtherCat master zařízení pomocí Codesys....79
2.33 Diagnostické zprávy k úspěšném startu EtherCat sítě a připojení slave zařízení....79
2.34 Codesys EtherCat master - hlavní okno vizualizace 80
2.35 Codesys EtherCat master - okno vizualizace pro testování registrů TMC4361 81
2.36 Codesys EtherCat master - okno vizualizace pro řízení motoru v základním režimu 82
2.37 Codesys EtherCat master - okno vizualizace pro řízení motoru v pokročilém režimu 83
2.38 Codesys EtherCat master - okno vizualizace pro řízení motoru v pokročilém režimu 84
2.39 Codesys EtherCat master - okno vizualizace pro vyčtené provozní hodnoty 85
2.40 Topologie EtherCat sítě při měření....87
2.41 Codesys EtherCat master status 87
2.42 Analyzovaný EtherCat datagram pomocí WireShark 88
2.43 Mechanické spojení hřídelí krokového motoru a enkodéru 89
2.44 Měření otáček pomocí osciloskopu....89
2.45 Záznam obrazovky osciloskopu při měření otáček pomocí enkodéru 90

SEZNAM TABULEK

Tabulka 1.1 Výhody a nevýhody jednotlivých typů krokových motorů [3] 16
Tabulka 1.2 Stěžejní parametry krokového motoru 16
Tabulka 1.3 Princip jednofázového řízení s plným krokem 18
Tabulka 1.4 Princip dvoufázového řízení s plným krokem 18
Tabulka 1.5 Princip dvoufázového řízení s poloviční krokem 19
Tabulka 1.6 Technické parametry driveru R272-80-ETH [9] 29
Tabulka 1.7 Tabulka technických parametrů pro driver N5 2-1 2-2 (CanOpen) [10] 31
Tabulka 1.8 Technické parametry jednotky R356 [11] 32
Tabulka 1.9 Technické parametry driveru Pololu 36V4 [12] 33
Tabulka 2.1 Požadované vstupní a výstupní rozhraní driveru 40
Tabulka 2.2 Základní požadované parametry driveru 40
Tabulka 2.3 b Hlavní použité komponenty pro hardwarovou realizaci 48
Tabulka 2.4 Důležité parametry a periferie mikrokontroléru STM32F446RET6 [25] 49
Tabulka 2.5 Seznam implementovaných částí a jejich základní popis 61
Tabulka 2.6 Přehled datových typů 64
Tabulka 2.7 Hlavní využité registry TMC4361 [27] 68
Tabulka 2.8 Přehled řídících režimů 75
Tabulka 2.9 Přehled provedených testů 86
Tabulka 2.10 Parametry motoru 17HS2408S a enkodéru S3806 [37] [38] 86

ÚVOD

Krokové motory jsou v dnešním průmyslu a mnoha dalších odvětvích velmi používaným pohonem. Tento typ pohonu vyžaduje pro kvalitní a přesné řízení specifické zařízení zvané driver či kontroler krokového motoru. V oboru průmyslových aplikací, jako jsou automatizační výrobní linky či jednoúčelové stroje, je systém skládán z komponent různého charakteru a jejich součinnost je řízena pomocí centrální řídící jednotky, která s nimi komunikuje pomocí určitého typu průmyslové komunikační sítě.

Tato bakalářská práce klade důraz na shrnutí teoretických informací z oblasti krokových motorů, driverů pro řízení krokových motorů a možností implementace průmyslových sítí v oblasti embedded systémů. V oblasti krokových motorů jsou shrnuty základní principy, jednotlivé typy, důležité parametry a základní principy řízení. V oblasti driverů pro krokové motory je popsána základní funkce, typické parametry, rozdělení dle specifických vlastností a dále provedena rešerše dostupných driverů se zaměřením na drivery podporující ovládání pomocí průmyslových komunikačních sítí nebo jiných typů sběrnic. V oblasti průmyslových komunikačních sítí je popsáno jejich rozdělení a vybrané moderní komunikační standarty podrobněji popsány. Dále jsou zmíněna konkrétní dostupná řešení, kterými je možné docílit integrace těchto standartu do embedded systému, tak aby bylo zajištěno, že je možné jejich zařazení v rámci většího fungujícího celku, který je propojen právě jednou z průmyslových komunikačních sítí. Cílem tohoto teoretického rozboru je vytvořit jasný a faktický přehled, ze kterého je čerpáno v dalších částech práce.

Praktická stránka práce je zprvu zaměřena na vytvoření vhodného konceptu, definování jasně formulovaných požadavků na výslednou funkcionalitu, ke které práce postupnými kroky směřuje, zvolení vhodné komunikační sítě a její podrobnější popis. Dále je praktická část práce členěna na část návrhu hardwarového vybavení a na část návrhu řídícího firmwaru. V obou těchto částech je podrobně popsán postup návrhu, jak textovou, tak příslušnou grafickou formou. Poslední část práce je věnována oživení, otestování funkčnosti a zhodnocení zrealizovaného driveru. Nejdříve jsou zmíněny možnosti pro implementaci jednoduchého master zařízení, poté je pomocí dostupných prostředků takovýto master vytvořen a předpokládaná funkčnost driveru otestována.

1 TEORETICKÁ ČÁST

Následující kapitoly, spadající do teoretické části, obsahují rozbor hlavních částí, kterými jsou krokové motory, drivery pro řízení krokových motorů a průmyslové komunikační sítě. Tyto části jsou stěžejní vzhledem k následnému vlastnímu návrhu driveru.

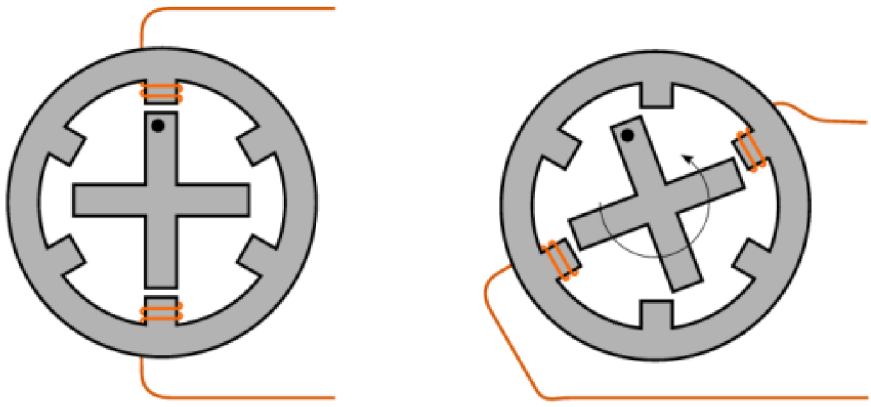
1.1 Krokové motory

Krokový motor je bezkomutátorový elektrický synchronní točivý stroj, který je specifický svým pohybem po tzv. krokách, čehož dosahuje tím, že vstupní stejnosměrné pulzy převádí na pohyb rotoru o přesně definovaný úhel natočení. Počet stabilních poloh krokového motoru je dán počtem pólových dvojic a dále je ovlivněn způsobem řízení. Klíčovou vlastností tohoto typu elektromotoru je tedy možnost řízení na určitou polohu bez znalosti aktuální polohy (není potřeba zpětná vazba).

Z konstrukčního hlediska jsou krokové motory, stejně jako všechny ostatní elektromotory, rozděleny na stacionární část (stator) a nestacionární část (rotor).

Základní fyzikální princip

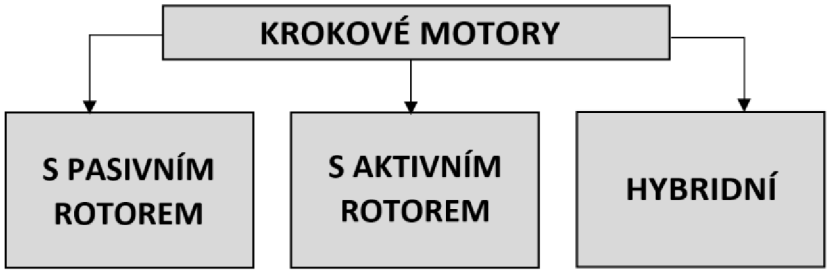
Princip pohybu o jeden krok je vysvětlen na třífázovém krokovém motoru s pasivním rotorem (konstrukčně popsaném v další kapitole) – připojení fáze s označením A na stejnosměrné napětí vyvolá magnetické pole mezi protilehlými pólovými nástavci fáze – je vyvolán tok magnetického pole, který hledá cestu nejmenšího magnetického odporu a vyvolá tím pootočení rotoru do pozice, kde je tento odpor nejmenší. Po odpojení fáze A a připojení fáze B následuje tento postup znovu, nyní však pro fázi B. Tento jednoduchý princip popisuje, jak pohybovat rotorem krokového motoru. [1][2]



1.1 Princip krokového motoru

1.1.1 Typy a jejich konstrukční uspořádání

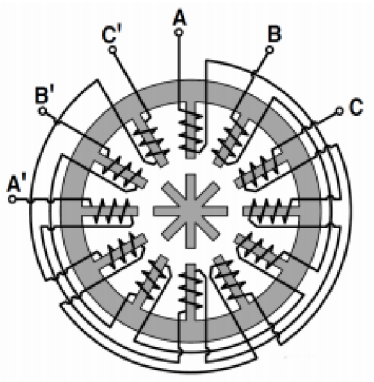
Krokové motory jsou děleny do tří základních skupin. Jednotlivé skupiny mají podobný princip funkce, liší se však především konstrukčním uspořádáním a vlastnostmi, které jednotlivé typy předurčují pro konkrétnější použití.



1.2 Rozdělení krokových motorů

Krokový motor s pasivním rotorem

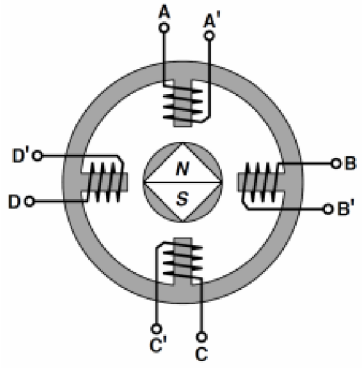
Krokové motory s pasivním rotorem, také nazývané jako reluktanční, jsou z hlediska konstrukčního uspořádání rozděleny na rotor, tvořený svazkem nalisovaných plechů na hřídeli, ze které vystupují pólové nástavce bez vinutí. Další částí je stator, který tvořen také z konstrukce nalisovaných plechů s vyniklými póly, na kterých jsou navinuta budící vinutí. Dvě protilehlá budící vinutí zapojená do série tvoří jednu budící fázi krokového motoru. Jak již z názvu vyplývá, takČ rotor neobsahuje žádný zdroj magnetického pole. Toto pole je vytvářeno postupným spínáním jednotlivých fází buzení a pohyb motoru nastává v důsledku reluktance. Protilehlé póly představují při vybuzení severní a jižní pól, a mezi nimi vzniká magnetický tok. Stator má v tuto chvíli tendenci zaujmout pozici, ve které bude představovat pro průchod magnetického toku nejmenší reluktanci neboli magnetický odpor. [1][2]



1.3 Krokový motor s pasivním rotorem [1]

Krokový motor s aktivní rotorem

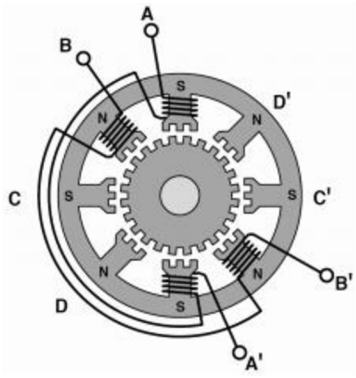
Krokové motory s aktivním motorem, také nazývané jako krokové motory s radiálně polarizovaným permanentním magnetem, jsou z hlediska konstrukčního uspořádání rozděleny na rotor tvořený póly s permanentními magnety, kde se střídá jižní a severní pól magnetického pole. Stator je tvořen dvojnásobným počtem pólových nástavců, který by měl být dělitelný čtyřmi, s vinutími, která jsou spojena do dvou fází. Liché pólové nástavce jsou sériově spojeny, a to samé sudé pólové nástavce, čímž je dosáhnuto vyvedení pouze dvou řídících fází. Princip tohoto typu krokového motoru je založen na působení sil magnetického pole. Abychom měnili orientaci magnetického pole, tak je při řízení nutné měnit polaritu napájecího proudu. [1][2]



1.4 Krokový motor s aktivním rotorem [1]

Hybridní krokový motor

Hybridní krokové motory, také nazývané jako krokové motory s axiálně polarizovaným permanentním magnetem, jsou speciálním druhem krokových motorů, který v dnešní době převyšuje přechozí dva typy. Důvodem je to, že hybridní krokový motor sdružuje výhodné vlastnosti krokového motoru s pasivním a aktivním rotorem. Rotor je tvořen permanentním magnetem a na pólech magnetu, které jsou orientovány jako sever nebo jih, jsou nástavce z magneticky měkkého materiálu. Velmi častým typem je hybridní krokový motor, který má na rotoru 50 nástavců, což odpovídá velikosti kroku na 1,8 stupně. [1][2]



1.5 Hybridní krokový motor [1]

1.1.2 Porovnání jednotlivých typů

Následující **Tabulka 1.1** znázorňuje porovnání jednotlivých typů krokových motorů v rámci jejich výhod a nevýhod.

Tabulka 1.1 Výhody a nevýhody jednotlivých typů krokových motorů [3]

	Typ krokového motoru		
Srovnání	S pasivním rotorem	S aktivním rotorem	Hybridní
Výhody a nevýhody	• Malá setrvačnost • Velký krok • Rychlý • Levný • Žádný přídržný moment	• Celkem velký krok • Velky přídržný moment	• Nejdražší provedení • Velmi malý krok • Velký moment • Vysoká rychlost

1.1.3 Typické parametry

Při výběru krokového motoru pro konkrétní aplikaci je důležité znát parametry, které krokové motory nejvíce odlišují. Následující **Tabulka 1.2** uvádí výběr a popis parametrů.

Tabulka 1.2 Stěžejní parametry krokového motoru

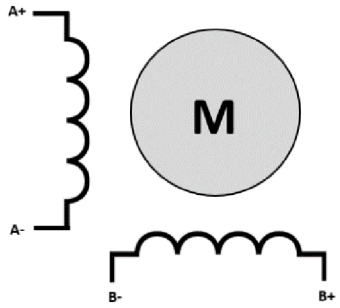
Název parametru	**Popis parametru**
Typ motoru	S pasivním/aktivním rotorem, Hybridní
Pracovní režim motoru	Bipolární/Unipolární
Velikost motoru	Např. standartní velikosti NEMA 14,17,23,24 nebo 34
Počet fází [n]	Počet fází pro ovládání motoru
Maximální napětí [V]	Maximální možné napětí pro napájení jedné fáze
Proud na fázi [A]	Maximální jmenovitý proud jednou fází motoru
Počet kroků [n]	Počet kroků pro jednu otáčku
Úhel kroku	Úhel, o který se motor otočí při jednom kroku
Provozní moment [N.m]	Vyvíjený moment otáčejícího motoru
Statický moment [N.m]	Vyvíjený moment stojícího motoru
Odpor [Ω]	Odpor budícího vinutí
Indukčnost [mH]	Indukčnost budícího vinutí

1.1.4 Typická zapojení

Krokové motory rozdělujeme dle zapojení a řízení do dvou základních skupin, na unipolární a bipolární. Jednotlivé krokové motory se mohou lišit počtem výstupních vodičů, zde nastává omezení v tom, že nelze každý krokový motor zapojit libovolně v bipolárním nebo unipolárním zapojení. Obvyklé jsou motory se 4, 5, 6 nebo 8 vodiči.

Bipolární zapojení krokového motoru

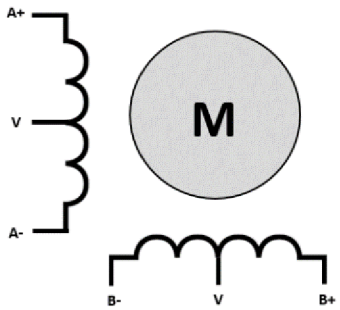
Pro bipolární řízení se využívají motory v 4, 6 a 8 vodičovém provedení. V dnešní době bipolární řízení převažuje and tím unipolárním. Mezi jeho přední vlastnosti patří vyšší moment a s tím spojená efektivita motoru. Bipolární zapojení dále dělíme na sériové a paralelní, přičemž pomocí sériového je možné dosáhnout vyššího momentu v nízkých otáčkách a pomocí paralelního vyšší rychlosti otáčení. Nevýhodou bipolárního řízení je složitější budič pro řízení, jelikož je nutné v jednotlivých fázích otáčet směr elektrického proudu. [3]



1.6 Bipolární zapojení krokového motoru [3]

Unipolární zapojení krokového motoru

Pro unipolární řízení lze využít motory v 5,6 nebo 8 vodičovém provedení. Unipolární řízení se vyznačuje především nižšími nároky na řízení motoru. Není třeba otáčet směr proudu při buzení fáze, z toho vyplívají jednodušší požadavky na řídící elektronický obvod. Nevýhodou oproti tomu je však nižší kroutící moment, proto se v dnešní době setkáme spíše s bipolárním zapojením. [3]



1.7 Unipolární zapojení krokového motoru [3]

1.1.5 Základní principy řízení

Jak již bylo uvedeno v úvodu kapitoly, krokové motory potřebují k vykonávání rotačního pohybu řízení v podobě sledu stejnosměrných impulzů. Pomocí následujících tabulek uvádíme nejzákladnější principy řízení krokových motorů. Symbol 1 značí kladné napětí na fázi a symbol -1 záporné napětí na fázi.

Princip jednofázového řízení spočívá v buzení pouze jedné fáze. V případě unipolárního řízení jsou jednotlivé fáze vyvedenou zemí rozděleny na dvě, tudíž jsou k dispozici čtyři fáze, které jsou buzeny kladnými impulzy proti zemi. Unipolární jednofázové řízení by podle **Tabulka 1.3** vypadalo jako sekvence kladných impulzů v pořadí A+(1), B+(1), A-(1), B-(1). Při bipolárním řízení již není k dispozici v rámci fáze vyvedená zem a je potřeba při řízení zajistit otáčení směru proudu, aby se změnila orientace magnetického pole. Bipolární jednofázové řízení by podle **Tabulka 1.3** vypadalo jako sekvence kladných a záporných impulzů A(1), B(1), A(-1), B(-1). [4]

Tabulka 1.3 Princip jednofázového řízení s plným krokem

Typ řízení	Jednofázové unipolární řízení s plným krokem			
A+	1	0	0	0
B+	0	1	0	0
A-	0	0	1	0
B-	0	0	0	1
Typ řízení	Jednofázové bipolární řízení s plným krokem			
A	1	0	-1	0
B	0	1	0	-1

Mnohem častějším typem řízení je dvoufázové řízení, které spočívá v tom, že jsou buzeny v jeden okamžik dvě fáze najednou, což má za důsledek větší provozní moment krokového motoru. Počet kroků na otáčku zůstává stejný, nicméně jsou kroky posunuty mezi póly. Sekvence pulzů již není uvedena, jelikož je zřejmá z **Tabulka 1.4** podle stejného principu jako v případě jednofázového řízení. [4]

Tabulka 1.4 Princip dvoufázového řízení s plným krokem

Typ řízení	Dvoufázové unipolární řízení s plným krokem			
A+	1		0	1
B+	1	1	0	0
A-	0	1	1	0
B-	0	0	1	1
Typ řízení	Dvoufázové bipolární řízení s plným krokem			
A	1	-1	-1	1
B	1	1	-1	-1

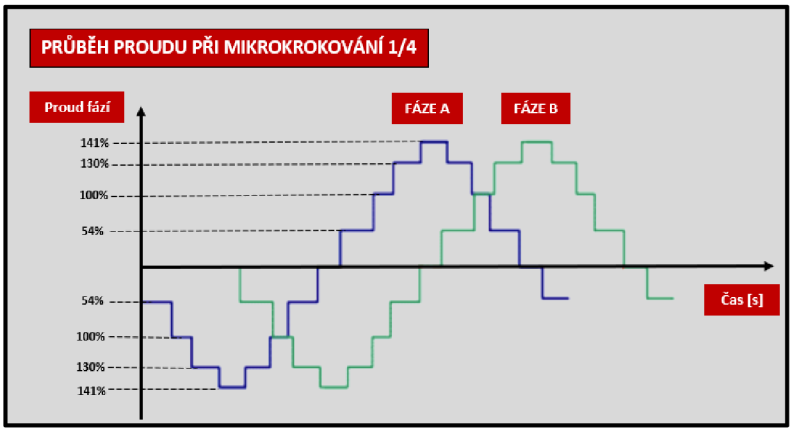
Řízení s polovičním krokem pracuje na principu, kdy se přechází mezi spínáním jedné fáze a následně dvou. Z této metody řízení plyne dvojnásobné rozlišení, nicméně je to za cenu menšího momentu oproti dvoufázovému řízení s plným krokem, jelikož zde nastává rozdílný moment ve chvíli, kdy je sepnuta jedna nebo dvě fáze. [4]

Tabulka 1.5 Princip dvoufázového řízení s polovičním krokem

Typ řízení	Dvoufázové unipolární řízení s polovičním krokem							
A+	1	1	0	0	0	0	0	1
B+	0	1	1	1	0	0	0	0
A-	0	0	0	1	1	1	0	0
B-	0	0	0	0	0	1	1	1
Typ řízení	Dvoufázové bipolární řízení s polovičním krokem							
A	0	-1	-1	-1	0	1	1	1
B	1	1	0	-1	-1	-1	0	1

Dalším důležitým pojmem v oblasti řízení krokových motorů je pojem mikrokrokování. Jedná se o schopnost donutit krokový motor krokovat po malých inkrementech mimo jeho přirozené polohy, které bývají standartní při řízení s plným krokem. Mikrokrokovat lze teoreticky donutit libovolný krokový motor, a to z důvodu, že mikrokroky nejsou vlastností krokového motoru, ale driveru (přesněji jeho výstupní částí nazývanou budič), který řídí proud jeho vinutími. Pro dosažení posunu o tzv. mikrokrok se využívá poměru proudu ve fázích. Průběh proudu v jednotlivých fázích a jejich poměr v každém okamžiku lze vidět na níže uvedeném obrázku **1.8**. [5]

Pro názornost je uveden konkrétní příklad. Máme dvoufázový bipolární motor, který disponuje krokem o velikosti 1,8°. Požadavkem je řízení pomocí mikrokrokování s poměrem 1/4, kde první číslice udává úhel plného kroku a druhá číslice počet mikrokroků na jeden krok. V tomto případě je jeden mikrokrok roven 0,45°. Na celou otáčku motoru o 360° je oproti původním 200 krokům potřeba 800 kroků. V praxi bývá nejvyšší hodnota pro použití mikrokrokování 1/256. [5]



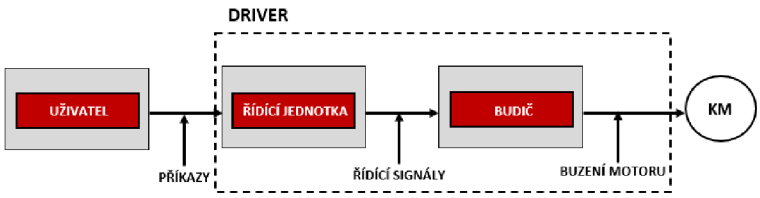
1.8 Průběh proudu při režimu mikrokrokování 1/4 [5]

1.2 Drivery pro řízení krokových motorů

V předešlé kapitole byly shrnuty základní informace o krokových motorech. Nyní již je známo, na jakém principu krokové motory pracují, jaké jsou typy i základní principy řízení. V tuto chvíli je důležité zmínit, že pro realizaci těchto principů v praxi je potřeba speciální elektronika zvaná jako driver krokového motoru. V této kapitole je uveden základní popis funkce a terminologii ohledně driverů, následně jsou rozebrány jejich základní parametry, rozdíly mezi řízením krokového motoru pomocí driveru v otevřené a uzavřené smyčce či základní rychlostní rozběhové profily Vzhledem k následujícímu návrhu vlastního zařízení je proveden také průzkum trhu a je zmíněno několik dostupných řešení, kde jsou porovnány funkce, technické parametry a možnosti ovládání skrze průmyslové komunikační sítě.

1.2.1 Popis základní funkce

Obecně lze definovat driver krokového motoru jako elektronické zařízení, které má za úkol bezpečně řídit provoz krokového motoru dle stanovených požadavků. Výstupem driveru je proud do jednotlivých fází motoru, přičemž driver zajistí, aby měl výstupní proud v čase požadovaný tvar, amplitudu a správný fázový posuv. Tímto je zajištěno otáčení motoru v požadovaném směru, požadovanou rychlostí a správné využití momentu motoru. Tímto byla shrnuta základní funkce, kterou by měl zajišťovat každý driver bez ohledu na konkrétní typ.



1.9 Základní princi řízení motoru pomocí driveru [3]

Dále je důležité ujasnit terminologii, která se bude opakovat v průběhu celé práce. Z průzkumu internetu plyne, že není zcela jednotná, jelikož se často zaměňují pojmy driver, budič, ovladač nebo kontrolér. V této práci bude k těmto pojmům přistupováno následujícím způsobem.

Driver krokového motoru

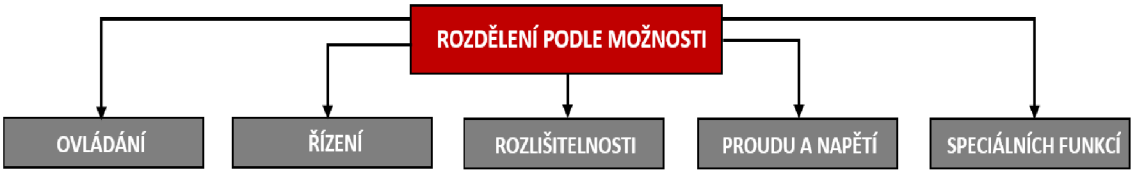
Tímto pojmem je rozuměno kompletní zařízení jako celek pro řízení krokového motoru, který má v sobě integrovanou řídící jednotku, budič či další potřebné příslušenství. Toto zařízení může být buďto plně samostatně funkční nebo zařazeno do hierarchie většího systému, kdy je vzdáleně ovládáno skrze ovládací rozhraní.

Budič krokového motoru

Tímto pojmem je rozuměna část driveru, která zajišťuje proudové buzení krokového motoru a funguje jako výkonový stupeň celého zařízení (driveru). Ke své činnosti potřebuje nadřazený ovládací systém, který bude buzení řídit.

1.2.2 Rozdělení driverů do specifických skupin

V technickém odvětví, které se věnuje řízení krokových motorů, neexistuje žádná norma, která by rozdělovala drivery krokových motorů do konkrétních skupin a popisovala specifické vlastnosti těchto skupin. Následující odstavce se pokusí rozdělit drivery do různých skupin podle průzkumu internetu.



1.10 Rozdělení driverů podle specifických vlastností

Dělení podle možnosti ovládání

Základním ovládáním je rozuměno jednoduché ovládání pomocí analogového nebo pulzního rozhraní. Drivery s těmito omezenými možnostmi jsou také nazývány jako step/dir, což je odvozeno od jejich standartních ovládacích vstupů. Ovládání je běžně prováděno pomocí základních ovládacích prvků, jako je například mechanické tlačítko nebo potenciometr. Tento typ driverů ve většině případů nelze uživatelsky programovat nebo napojit na vyšší řídící systém

Pokročilým ovládáním je rozuměn složitější typ ovládání pomocí jednoduchých standartních sběrnic jako jsou například USB, RS-485 nebo složitějších průmyslových sběrnic jako jsou například Can, Profibus nebo EtherCat. Tyto drivery se dají pomocí zmíněných sběrnic standartně napojit na vyšší řídící systém a fungovat jako zařízení typu slave ve složitějším systému, nebo jsou sběrnice využity pouze k přenesení uživatelského programu do paměti driveru a driver poté funguje jako nezávislé zařízení,

Kombinací ovládání je rozuměna možnost volby mezi možnostmi základním nebo pokročilým ovládáním. Takovéto drivery mají možnost uživatelské volby ovládání.

Dělení podle možnosti řízení

Řízením v otevřené smyčce je rozuměno typ řízení, kdy driver nepracuje s aktuální polohou a rychlostí motoru. Drivery pro tento typ řízení neobsahují vstup pro zpětnou vazbu a používají se pro jednoduché aplikace, kde není potřeba dodržet striktní požadavky na řízení.

Řízením v uzavřené smyčce je rozuměno typ řízení, kdy driver pracuje s aktuální polohou a rychlostí motoru. Drivery pro tento typ řízení obsahují vstup pro zpětnou vazbu, například rozhraní enkodéru. Naopak pro aplikace se striktními požadavky.

Kombinací řízení je rozuměno možnost uživatelské volby, jak bude driver použit.

Dělení podle možnosti rozlišitelnosti

Standartním rozlišením může být rozuměno takové rozlišení krokového motoru, kterého je dosaženo pomocí mikrokrokovacího režimu nejvýše 1/64. Jednoduché drivery mívají mikrokrokovací režimy volitelné od režimu celého kroku až po režim 1/32 nebo 1/64. Takovéto rozlišení mohou mít ale i složité drivery.

Vysokým rozlišením může být rozuměno takové rozlišení krokového motoru, kterého je dosaženo pomocí mikrokrokovacího režimu nejvýše 1/256. Složitější nebo přímo specializované drivery mívají mikrokrokovací režimy volitelné až po 1/256, což je v běžné praxi nejvyšší možná hodnota mikrokrokování.

Dělení podle možnosti proudu a napětí

Rozdělením podle proudu a napětí může být rozuměno takové rozdělení, kdy je driver schopen pracovat v provozních režimech vysokého napětí nebo vysokého výstupního proudu. Často je naopak požadována kombinace nízkonapěťového provozu a vysokého výstupního proudu. Vše je dáno požadavky na výslednou aplikace, ve které má být driver uplatněn.

Dělení podle možnosti speciálních funkcí

Speciálními funkcemi mohou být rozuměny takové funkce, které nebývají běžné v širokém rozsahu standartních driverů popsaných do této chvíle. Mezi takovéto funkce je možné zařadit přítomnost různých bezpečnostních vstupů, vstupů pro koncové spínače, automatickou detekci nulové polohy či mnoho softwarových nadstandartních funkcí. V tomto směru nelze provést jednoduché shrnutí

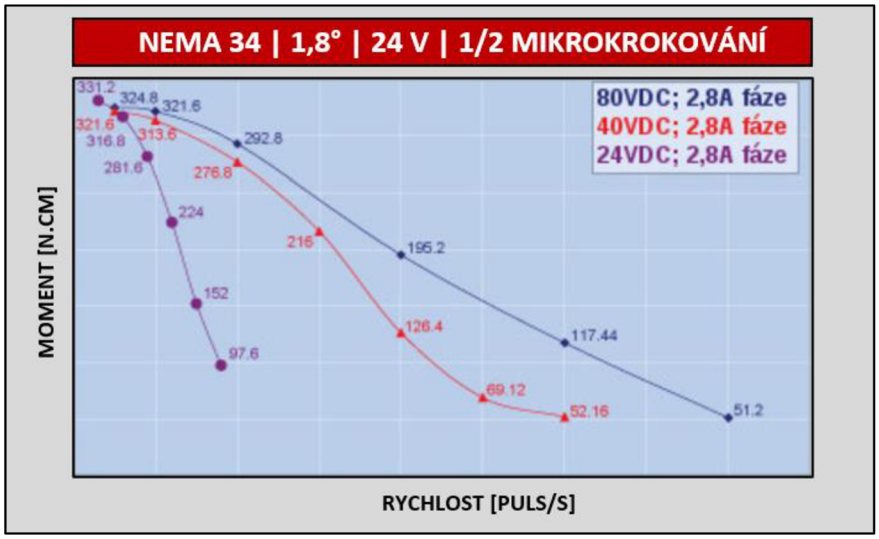
1.2.3 Typické parametry

Při výběru nebo návrhu driveru pro krokový motor je vždy nutné znát důležité parametry a vlastnosti, které mohou být rozhodující pro konkrétní aplikaci. Z tohoto jsou v této části uvedeny nejčastější parametry uváděné při výběru driveru a popsán jejich význam.

Maximální a minimální provozní napětí

Volba provozního napětí je důležitá pro správné využití výkonu motoru. Standartně je provozní napětí udáváno jako určitý interval, kde dolní hodnota symbolizuje minimální provozní napětí a horní hodnota maximální provozní napětí. Na obrázku **1.11** je možné

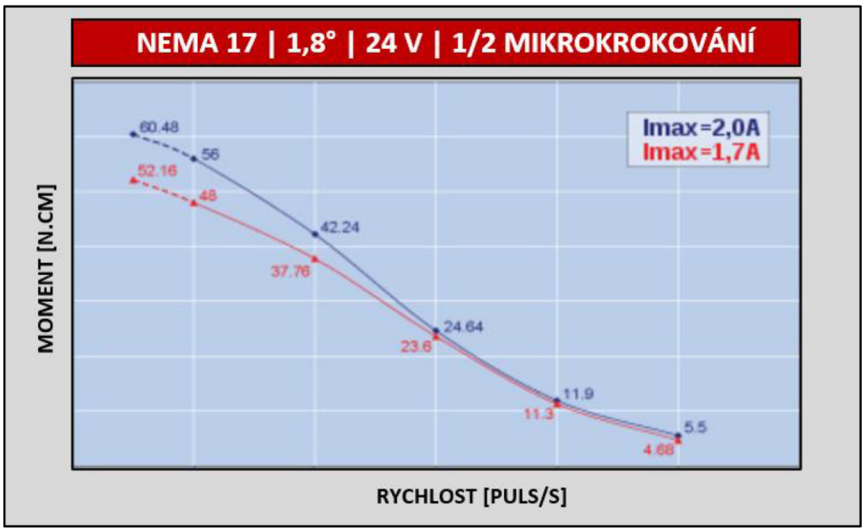
vidět změřenou momentovou charakteristiku, na které je vidět vliv napětí při větších rychlostech. Vyplývá z toho nutnost správné volby napětí, pokud má být využit maximální moment pro konkrétní rychlost motoru. [5]



1.11 Vliv napětí při velkých rychlostech na momentovou charakteristiku [5]

Maximální trvalý a špičkový proud fází

Maximální trvalý neboli jmenovitý proud, je takový proud, na který je krokový motor dimenzován. Pokud se jedná o budič, který pracuje s plným nebo polovičním krokem, tak špičkový proud odpovídá jmenovitému. Ve chvíli, kdy je využito mikrokrokování 1/4 a více, tak je hodnota špičkového proudu rovna 1,4 násobku jmenovitého proudu. Na obrázku **1.12** je možné vidět změřenou momentovou charakteristiku, kde je vidět vliv proudu při malých rychlostech. [5]



1.12 Vliv napětí při malých rychlostech na momentovou charakteristiku [5]

Režim mikrokrokování

Tento parametr je při výběru krokového motoru často rozhodující, jelikož ovlivňuje v aplikaci několik žádoucích i nežádoucích vlastností. Na vysoké mikrokrokování je kladen důraz v aplikacích, kde je potřeba zajistit zastavování motoru po malých inkrementech, hladký pohyb nebo nízký hluk motoru. Standartně jsou dostupné drivery krokových motorů s režimem mikrokrokování až 1/256. [5]

1.2.4 Řízení v otevřené a uzavřené smyčce

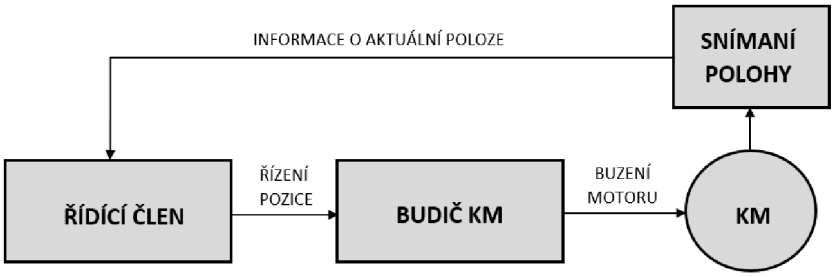
Krokové motory se používají ve dvou základních zapojeních, a to v zapojení s otevřenou nebo uzavřenou smyčkou. V následujících odstavcích porovnáme tyto dvě zapojení z hlediska jejich využití a vlastností, které nabízejí. Z těchto vyplývajících skutečností dále popíšeme fakt, že systémy s uzavřenou smyčkou zcela ale jistě vytlačují systémy s otevřenou smyčkou. [6]

Krokové motory v zapojení s otevřenou smyčkou jsou ceněny především díky možnostem řízení na přesnou polohu bez jakéhokoliv zpětnovazebního systému, čímž odpadají náklady na přídavnou elektroniku. Nicméně důležitým poznatkem je, že při použití tohoto zapojení je potřeba mít rezervu točivého momentu, jelikož v případě dosažení jeho špičkové hodnoty nebo při jejím překročení nastane rozhození krokování a tím je ztracena informace o poloze. Pokud shrneme celkové nedostatky řízení krokového motoru v otevřené smyčce, tak jde především o špatnou přesnost a spolehlivost, nemožnost odstranění poruchy za chodu, velký překmit, kroutící moment musí být využit pod jeho maximálními možnosti nebo pravděpodobnost ztráty kroku při velké rychlosti buzení motoru. [6]

1.13 Princip řízení v otevřené smyčce

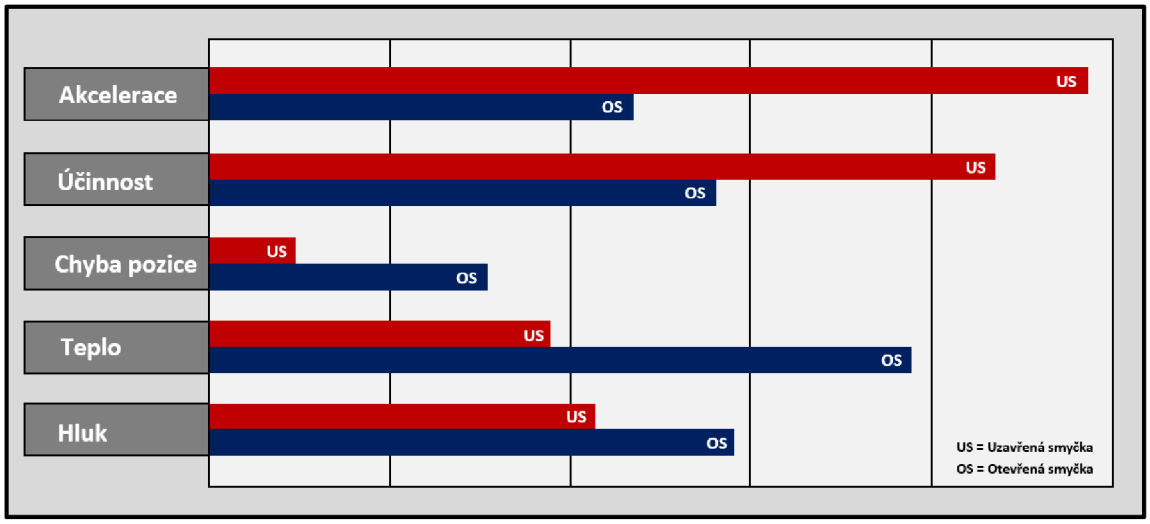
Aby bylo předejito špatným vlastnostem při řízení s otevřenou smyčkou, tak lze k řízení motoru přistupovat jako k řízení s uzavřenou smyčkou. Principem tohoto řízení je snímání polohy motoru pomocí enkodéru (nejčastější snímač otáček), z nějž je informace o poloze/rychlosti předávána do řídící jednotky tohoto uzavřeného systému, tato jednotka pak na základě předané informace upravuje polohu motoru dle požadavku. Tento princip je nazýván jako kompenzace kroku, nicméně úprava ztráty kroku není upravována v každém okamžiku běhu. Další možností tohoto řízení je kontinuální korekce polohy v reálném čase. Zde část, která se stará o řízení polohy zátěže, nepřetržitě sleduje polohu hřídele a generuje chybový signál pro řídící jednotku, která na základě tohoto signálu

upravuje řízení v reálném čase a takto je pohyb motoru sledován a korigován nepřetržitě po celou dobu. [6]



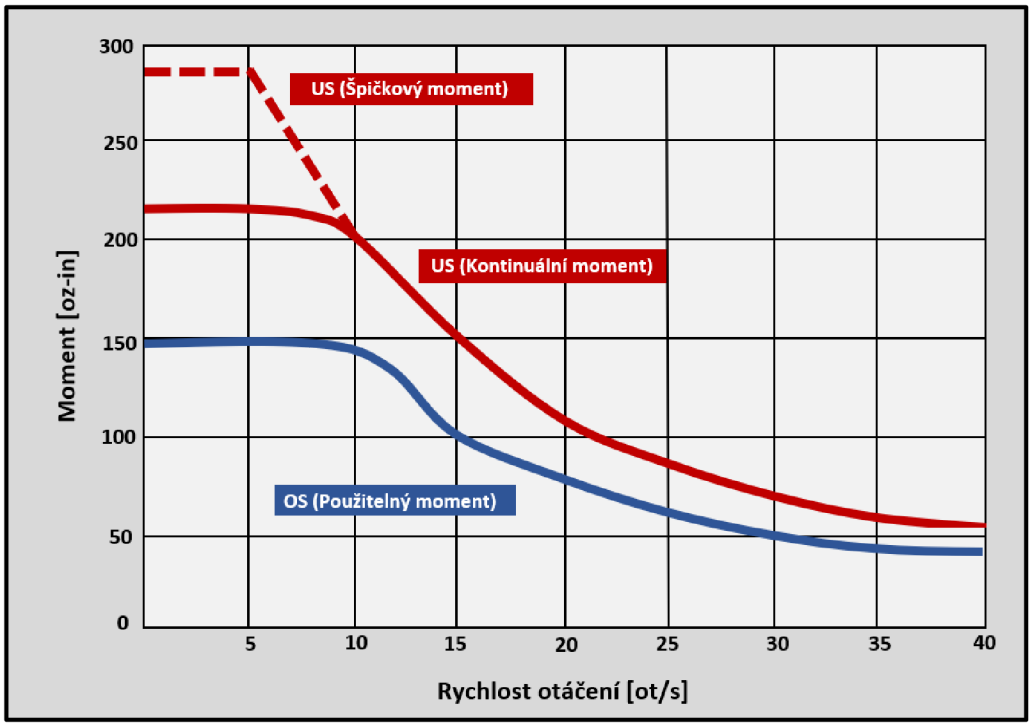
1.14 Princip řízení v uzavřené smyčce

Jakmile byly definovány základní principy, tak lze přistoupit ke konkrétnímu srovnání dle základních důležitých vlastností. Následující obrázek **1.15** ukazuje relativní srovnání vlastností řízení stejného motoru za stejných podmínek. Z grafu jasně vyplývají přednosti řízení s uzavřenou smyčkou, kde při porovnání převažuje lepší akcelerace (točivý moment), účinnost, přesnost, menší vykazování tepla a hladiny hluku. [6]



1.15 Porovnání vlastností při řízení v otevřené a uzavřené smyčce [6]

Pro další názorné porovnání vlastností je dobré porovnat dynamický točivý moment motoru v závislosti na rychlosti otáčení. Točivý moment je jedním ze základních faktorů vzhledem ke konečné aplikaci, kde je krokový motor v určitém zapojení použit. V grafu uvedeném na obrázku **1.16** je vidět jasné porovnání špičkového a nepřetržitého momentu uzavřené smyčky a dále pak použitelného momentu pro otevřenou smyčku. Z grafu momentové charakteristiky jasně vyplývá, že pro stejný krokový motor je mnohem výhodnější použití systému uzavřené smyčky. [6]



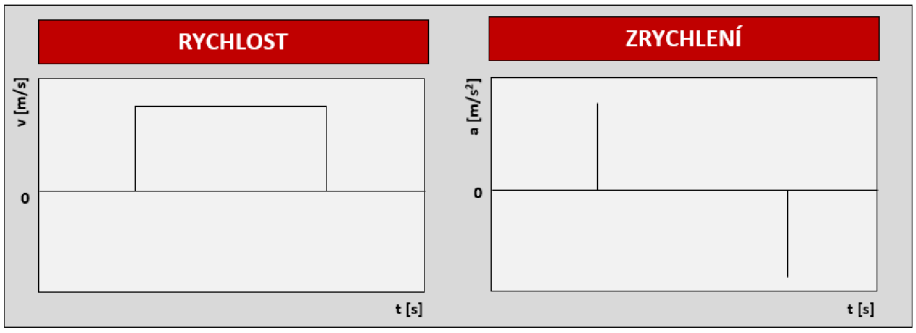
1.16 Porovnání průběhu dynamického momentu v závislosti na rychlosti motoru [6]

1.2.5 Rychlostní rozběhové profily

Další důležitou vlastností driverů je generování proudu do cívek na základě křivky, která se nazývá rychlostní rozběhový profil. Tato křivka definuje přesný průběh rychlosti krokového motoru v čase. Jak již bylo uvedeno, tak krokové motory slouží pro aplikace, kde je důležitá přesnost polohování po předem dané trajektorii či nastavení přesně definované rychlosti. Nutné je podotknout, že teorie těchto profilů je komplexní složitá disciplína a v této kapitole je uveden pouze velmi stručný přehled bez návaznosti na matematický aparát. Následující odstavce popisují typické základní profily a jejich možné využití. Názvy jednotlivých typů jsou povětšinou odvozeny od geometrického tvaru jejich rychlostní křivky.

Obdélníkový profil

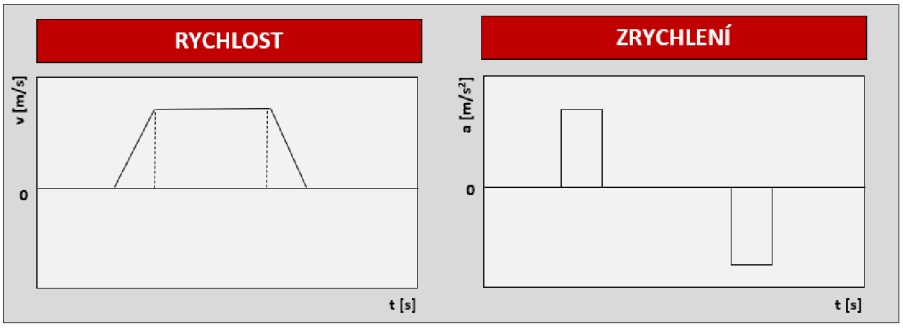
Základním užívaným typem rozběhového profilu je obdélníkový průběh. Tento typ je charakteristický tím, že při požadavku na novou rychlost je tato rychlost nastavena teoreticky v nulovém čase. Nevýhodou takto rychlého zrychlení, respektive zpomalení je velké trhnutí motoru nebo dokonce jeho zaseknutí. Téměř nepoužitelné při větším zatěžovacím momentu. Grafické vyjádření v čase vyobrazuje obrázek **1.17**. [7] [8]



1.17 Rozběhový profil typu obdélník [7]

Lichoběžníkový profil

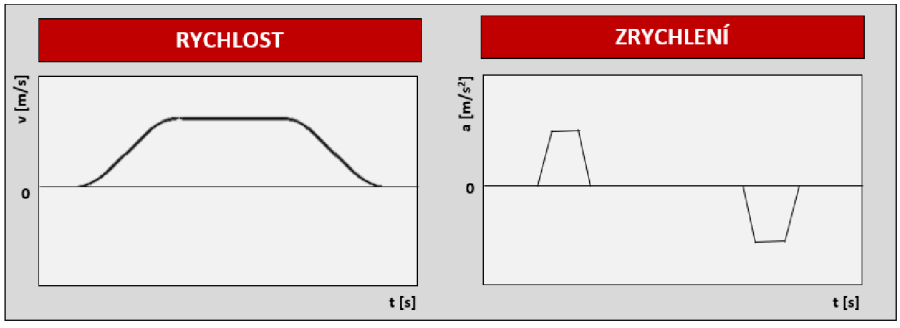
Tento profil je nejběžnějším užívaným typem. Užívá se v aplikacích, kde je potřeba pohyb s konstantní rychlostí po určitou dobu, ale zároveň je využito postupného zrychlení a zpomalení, což zaručuje plynulý pohyb. Hodnoty zrychlení a zpomalení lze matematickým výpočtem přizpůsobit konkrétní aplikaci. Profil odstraňuje problém s okamžitým zátěžovým momentem, nicméně stále zcela neodstraňuje problém s počátečním a koncovým trhnutím. Graficky je profil ukázán na obrázku **1.18**. [7] [8]



1.18 Rozběhový profil typu lichoběžník [7]

S-křivka

Tento profil je zařazen mezi pokročilejší a jeho použití by mělo být opodstatněno. Rozdíl oproti lichoběžníkovém typu je v tom, že i hodnoty zrychlení a zpomalení mají postupný nárust nebo pokles. Díky tomu je pohyb naprosto plynulý. Nevýhodou je potřeba většího výpočetního výkonu. Graficky je profil ukázán na obrázku **1.19**. [7] [8]



1.19 Rozběhový profil typu s-křivka [7]

1.2.6 Driver R272-42(80)-ETH

Driver R272-42-ETH spadá do skupiny driverů pro řízení dvoufázových hybridních krokových motorů a nachází se ve dvou verzích, které se liší maximální proudem na fázi – verze R272-42-ETH s maximálním proudem na fázi 4,2 A a verze R272-80-ETH s maximální proudem na fázi 8,0 A. Z hlediska ovládání, nabízí tento driver pro uživatele celkem až tři možnosti. Programové ovládání, analogové ovládání nebo pulzní ovládání.

Dále jsou uvedeny klíčové funkce, vlastnosti a tabulkA technických parametrů pro verzi R272-80-ETH. Další funkce a vlastnosti je možné najít v uživatelském manuálu.[9]

Funkce a možnosti [9]

1) Dálkové ovládání pomocí Ethernetu.
2) Samostatný běh motoru dle programu uloženého v uživatelské paměti – možnost až čtyř nezávislých programů, mezi kterými je možné volit.
3) Programování uživatelských programů pomocí lokální Ethernetové sítě nebo USB rozhraní.
4) Parametry pro řízení motoru (proud fází, přídržný proud či mikrokrokující režim) mohou být dále nastavovány pomocí menu kontroléru, k čemuž jsou využity vestavěné ovládací prvky a sedmisegmentový displej.
5) Pulzní řízení motoru je možné pomocí standartních vstupů ENABLE, STEP a DIR
6) Analogové řízení je možné pomocí vestavěných nebo externích potenciometrů, dále je možné využít analogový signál 0-5 VDC.
7) Z hlediska bezpečnostních funkcí je driver vybaven vstupem pro signál z bezpečnostního senzoru pro zastavení motoru, dále obsahuje regulátor napětí, aby bylo zamezeno poškození driveru, při překročení napájecího napětí. Z hlediska softwarového zabezpečení je využíváno 32 bitové heslo pro přístup pomocí lokální Ethernetové sítě. Pro signalizaci alarmů a chybových stavů je využito vestavěných alarmových výstupů a zmíněného sedmisegmentového displeje.

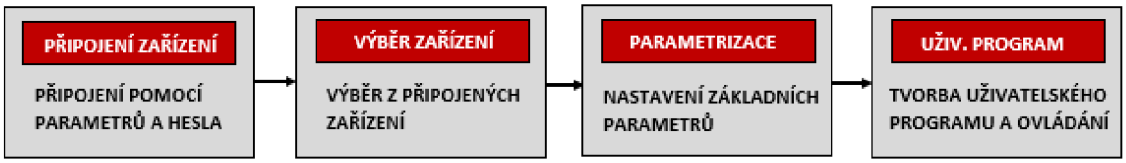
1.20 Ukázka driveru R272-42-ETH [9]

Tabulka 1.6 Technické parametry driveru R272-80-ETH [9]

Základní technické parametry	**Driver R272-80-ETH**
Napájecí napětí [V]	24-48
Maximální proud fází [A]	8
Minimální proud fází [A]	0,1
Mikrokrokující režimy	1/1 ; 1/2 ; 1/4 ; 1/8 ; 1/16 ; 1/32 ; 1/64 ; 1/128
Rozměry [mm]	120x110x45
Parametry ovládacích vstupů	**Driver R272-80-ETH**
High úroveň napětí [V]	4-5 (možnost úpravy na 24 V)
Low úroveň napětí [V]	0,1
Vstupní odpor [kΩ]	1
Parametry výstupů alarmů a chyb	**Driver R272-80-ETH**
Typ výstupu	Výstup optočlenu
Maximální úroveň napětí [V]	20
Minimální úroveň proudu [mA]	100
Parametry výstupního relé	**Driver R272-80-ETH**
Typ relé	Solidstate relé
Maximální úroveň napětí [V]	+- 350
Maximální úroveň proudu[mA]	+- 120

Z hlediska popisu driveru již další informace uvedeny nebudou. Informace o uživatelském rozhraní, elektrickém zapojení driveru a jeho vstupů/výstupů nebo připojení do lokální sítě je možné dohledat v uživatelské příručce.

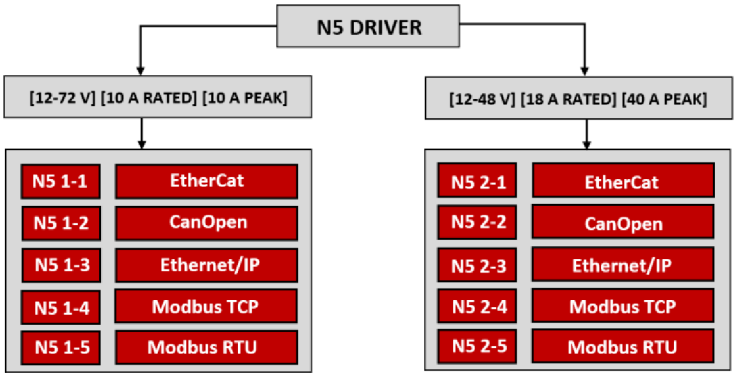
Jako poslední je důležité zmínit dostupný software pro vzdálené programování a ovládání driveru, kterým je SMC-Program. Tento software je přímo určen pro ovládání tohoto driveru. Nabízí snadné připojení driveru pomocí základních informací jako MAC adresa, IP adresa, příslušný PORT nebo heslo. SMC Program za nás řeší veškeré nastavení protokolu a zabezpečení, takže uživatel řeší pouze připojení konkrétního zařízení, parametrizaci, tvorbu řídícího programu pomocí příkazů a následné sledování či zásahy do programu. [9]



1.21 Princip softwaru SMC-Program [9]

1.2.7 Driver N5 s podporou průmyslových sítí

N5 driver je univerzální driver pro krokové i stejnosměrné motory od firmy Nanotec. Tento driver je vyráběn v několika verzích a toto rozdělení popisuje obrázek **1.22**. Hlavními specifiky pro jednotlivé verze je rozdíl výstupního proudu a také podpora konkrétní průmyslové komunikační sítě. Pro všechny uvedené verze platí následující specifické funkce.[10]



1.22 Rozdělení N5 driverů [10]

Funkce a možnosti [10]

1) Jednoduchá parametrizace pomocí vybrané průmyslové komunikační sítě.
2) Ovládání pomocí průmyslové komunikační sítě, digitálních pulzů a dalších digitálních/analogových vstupů.
3) Programovatelné pro samostatný provoz pomocí Plug & Drive Studio.
4) Možnost činnosti v uzavřené smyčce pomocí zpětné vazby z enkodéru.

1.23 Ukázka N5 driveru s podporou průmyslových sítí [10]

Tabulka 1.7 Tabulka technických parametrů pro driver N5 2-1 2-2 (CanOpen) [10]

Základní technické parametry	**N5 2-1**	**N5 2-2**
Napájecí napětí [V]	12-72	12-48
Jmenovitý proud fází [A]	10	18
Špičkový proud fází [A]	10	40
Rozměry	149x72x44	
Parametry ovládacích vstupů	**N5 2-1**	**N5 2-2**
Úroveň napětí [V] (digital vst. 1-4)	5/24 (programově nastavitelné)	
Úroveň napětí [V] (digital vst. 5-6)	5 až 24 (široky rozsah)	
Úroveň napětí [V] (2 analog. vst.)	-10 až 10 (programově nastavitelné)	
Úroveň proudu [mA] (2 analog. vst.)	0 až 20 (programově nastavitelné)	
Parametry výstupů	**N5 2-1**	**N5 2-2**
Typ výstupu	Otevřený kolektor	
Úroveň napětí [V]	0 až 24	
Úroveň proudu [mA]	0 až 500	
Parametry vstupního enkodéru	**N5 2-1**	**N5 2-2**
Typ signálu	Diferenciální nebo jednostranný	
Úroveň napětí [V]	5 nebo 24 (programově nastavitelné)	
Rozlišení enkodéru	65536 přírůstků na otáčku (16-bit)	

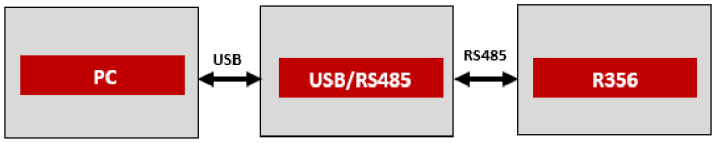
1.2.8 Mikroprogramovatelná jednotka R356

Mikroprogramovatelná jednotka R356 je univerzální driver krokových motorů, který je vhodný pro aplikace, kde je potřeba vysoká přesnost řízení. R356 umožňuje mikrokrokování s rozlišením až 256 mikrokroků na jeden krok. Dále jsou uvedeny klíčové funkce, vlastnosti a tabulka technických parametrů. [11]

Funkce a možnosti [11]

1) Možnost konfigurace a programování uživatelských programů.
2) Konfigurace a programování probíhá pomocí dvouvodičové sběrnice a sériové komunikace RS-485. Na této sběrnici může být připojeno až 16 jednotek R356. Pro převod konfigurace a programu je důležité zvolit vhodný převodník, nejčastěji pro převod USB na RS-458 nebo RS-232 na RS-485. Blokové zapojení je možné vidět na obrázku **1.24**.
3) Jednotka obsahuje uživatelskou paměť, ve které může být uloženo až 16 uživatelských programů.
4) Jednotka umožňuje samostatný běh, bez stálého připojení ke sběrnici.

5) Pro zpětnovazebního řízení s korekcí polohy je možnost připojení optického enkodéru.
6) Důležité parametry jako přídržný proud nebo proud během chodu motoru jsou zcela programovatelné. Dále je také podpora programovatelných ramp a rychlostí.



1.24 Blokové zapojení pro konfiguraci a programování R356 [11]

Dostupným softwarem pro konfiguraci a programování této jednotky je Lin Control. Software je velice intuitivním nástrojem, který umožňuje jednoduchou konfiguraci sériové komunikace a následně adresování konkrétních jednotek připojených na sběrnici. Parametrizace a programování je možné provádět dvěma způsoby. Umožněno je vykonání jednotlivých příkazů pomocí terminálu nebo je možnost sestavovat sekvence příkazů, které bude následně uloženy jako uživatelský program přímo do jednotky. [11]

1.25 Ukázka mikroprogramovatelné jednotky R356 [11]

Tabulka 1.8 Technické parametry jednotky R356 [11]

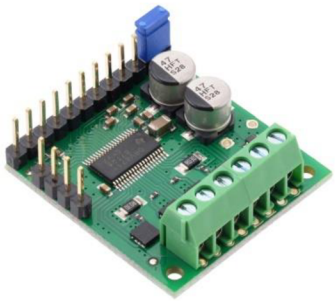
Základní technické parametry	**Jednotka R356**
Napájecí napětí [V]	12-40 V
Maximální proud fází [A]	3
Minimální proud fází [A]	0,1
Mikrokrokující režimy	1/1 ; 1/2 ; 1/4 ; 1/8 ; 1/16 ; 1/32 ; 1/64 ; 1/128 ; 1/256
Rozměry [mm]	27x26x10
Parametry ovládacích vstupů	**Jednotka R356**
Úroveň napětí [V]	0-5
Vstupní proud [mA]	700
Odpor pull-up rezistorů [kΩ]	20

1.2.9 Driver Pololu 36V4

Pololu 36V4 představuje driver, který se vyznačuje vysokým výkonem. Uživateli dodává možnost volby provozního napětí ve velkém rozsahu a také možnost nastavení výstupního proudu, který může dosahovat až 6A na fázi, přičemž je ale nutné přídavné chlazení. Tento driver můžeme zařadit do skupiny driverů, které neobsahují řídící jednotku a nelze je uživatelsky programovat. Vše je založeno na budiči DRV8711 v kombinaci s externími mosfet H můstky. Pomocí tohoto složení nabízí driver následující funkce a technické parametry. [12]

Funkce a možnosti [12]

1) Vysoký výkon s nastavitelným provozním napětím a výstupním proudem.
2) Základní ovládání pomocí STEP/DIR signálů nebo možnost konfigurace a ovládání pomocí sběrnice SPI.
3) Možnost vysokého krokového rozlišení až 1/256.
4) Volitelné výstupy STALL nebo BMF pro možnosti detekce zablokování a umožnění pokročilejších algoritmů řízení.
5) Bezpečnostní funkce jako blokování podpětí, ochrana proti nadproudu nebo ochrana proti zpětnému napětí.



1.26 Ukázka driveru Pololu 36V4 [12]

Tabulka 1.9 Technické parametry driveru Pololu 36V4 [12]

Základní technické parametry	**Driver Pololu 36V4**
Napájecí napětí [V]	8-50
Trvalý proud fází [A]	4 A
Maximální proud fází [A]	6 A
Mikrokrokující režimy	1/1 ; 1/2 ; 1/4 ; 1/8 ; 1/16 ; 1/32 ; 1/64 ; 1/128 ; 1/256
Rozměry [mm]	13x12x
Parametry ovládacích vstupů	**Driver Pololu 36V4**
Maximální úroveň napětí [V]	5,5
Minimální úroveň napětí [V]	1,5

1.3 Průmyslové komunikační sítě

Průmyslové komunikační sítě jsou v dnešní době naprostým standardem pro téměř všechny průmyslové aplikace. S jejich pomocí je možné rychle a efektivně odesílat povely jednotlivým komponentám v architektuře příslušného stroje či zařízení, a naopak jednotlivé komponenty jsou schopné reagovat na příslušné požadavky. V této kapitole je popsáno základní rozdělení, popis nejpoužívanějších sběrnic a protokolů, a dále možnost řešení v oblasti vestavěných systémů.

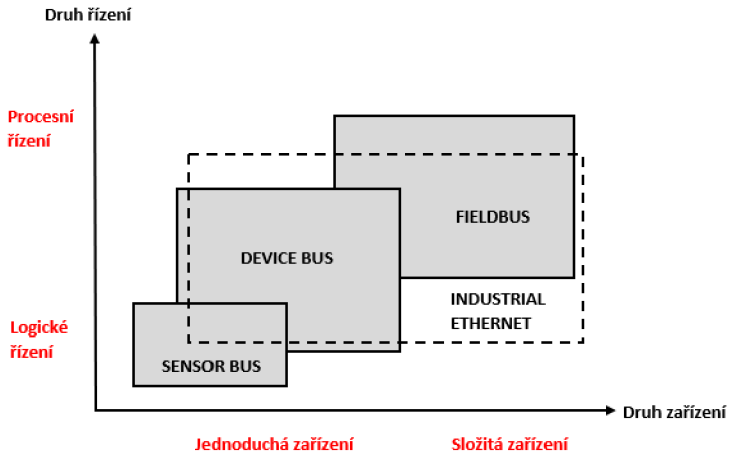
1.3.1 Rozdělení průmyslových sběrnic a jejich komunikačních protokolů

Průmyslové komunikační sítě (sběrnice) lze dělit v závislosti na druhu použitých zařízení, která jsou propojena a také na druhu řízení, které je použito. Na obrázku **1.27** je možné vidět základní grafické rozdělení na tři skupiny - sensor bus, device bus a field bus.

Rozdělení průmyslových sítí [15]

1) **Sensor bus** – As interface
2) **Device bus** – Profibus DP, DeviceNet, FIP, CANopen
3) **Fieldbus** – Profibus PA, Fieldbus Foundation
4) **Průmyslový ethernet** – Profinet, Powerlink, EtherCat

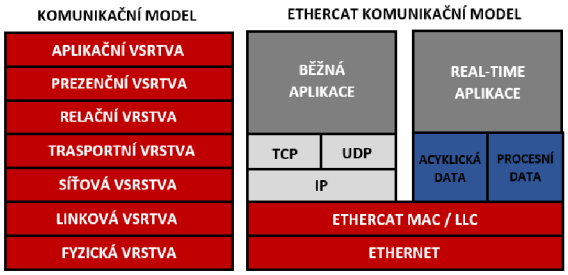
Poté, co bylo přiblíženo rozdělení průmyslových sítí, byla pozornost zaměřena na tři vybraná řešení. Cílem bylo udělat základní přehled mezi standarty EtherCat, Powerlink a Profibus. Tyto tři standarty byly na začátku návrhu určeny jako možné potencionální řešení. Následně byl vybrán jeden pro řešení navrhovaného driveru – vybrané řešení je podrobněji popsáno v praktické části kapitolou **2.3.**



1.27 Rozdělení průmyslových sítí dle zařízení a aplikace [15]

1.3.2 EtherCat

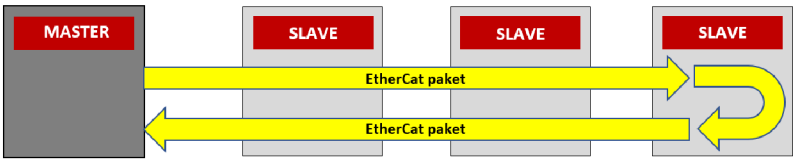
EtherCat je jedním z nejrozšířenějších standardů postaveném na průmyslovém ethernetu. Hlavní specifikum EtherCat je komunikace v reálném čase s velmi krátkými cykly a vysokou přesností synchronizace. Je vhodný pro centralizované i decentralizované systémy.



1.28 Komunikační model EtherCat [15]

Základní charakteristiky a parametry EtherCat [16]

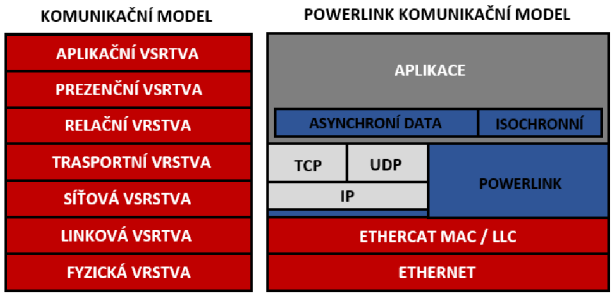
1) **Topologie** – využívají se standartní zapojení jako je linie, strom nebo kruh. Je však možné využít i zapojení jako sběrnice nebo sběrnice s odbočkami. Každá topologie tvoří v případě EtherCatu uzavřený komunikační kruh.
2) **Typ komunikace** – hlavní komunikačním modelem, který se využívá, je model master-slave.
3) **Rychlost komunikace** – je využíváno Ethernetu s přenosovou rychlostí 100 MB/s.
4) **Protokoly standartu** – zařízení se standartem EtherCat obsahují minimálně komunikační profil Coe (CanOpen over EtherCat) a dále volitelně protokoly EoE (Ethernet over EtherCat), SoE (Servodrive over EtherCat) nebo Foe(File acess over EtherCat).
5) **Základní princip funkce** – v rámci EtherCat sítě nejsou data od zařízení master odesílána jednotlivě k zařízením slave, ale rámce dat procházejí během jednoho cyklu všemi zařízeními slave. Oproti klasickému Ethernetu je eliminována kolize dat a je optimalizována rychlost přenosu. Zařízení slave mohu přidávat data do rámce nebo je z něj číst během jeho průchodu, nemohou však nové rámce sami vytvářet.
6) **Požadavky na hardware** – tyto požadavky se liší podle typu zařízení. Zařízení typu master nevyžaduje žádné speciální hardwarové požadavky a jeho realizace spočívá pouze v implementaci softwaru například na běžném PC nebo Ethernetovém kontroleru. Zařízení typu slave vyžaduje speciální hardware, se dvěma ethernetovými porty zajišťující velkou rychlost přenosu při průchodu dat.



1.29 Průběh EtherCat rámce sítí [16]

1.3.3 Powerlink

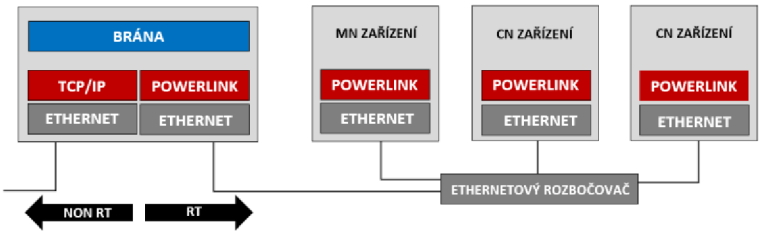
Standart průmyslového Ethernetu, který vychází z klasického Ethernetu. Stejně jako EtherCat se vyznačuje komunikací v reálném čase a krátkými komunikačními cykly, avšak na rozdíl od EtherCatu nepotřebuje žádný specializovaný hardware a dosahuje těchto vlastností odlišným způsobem, přesněji jde čistě o softwarové řešení.



1.30 Komunikační model Powerlink [15]

Základní charakteristika a parametry Powerlink [13] [14]

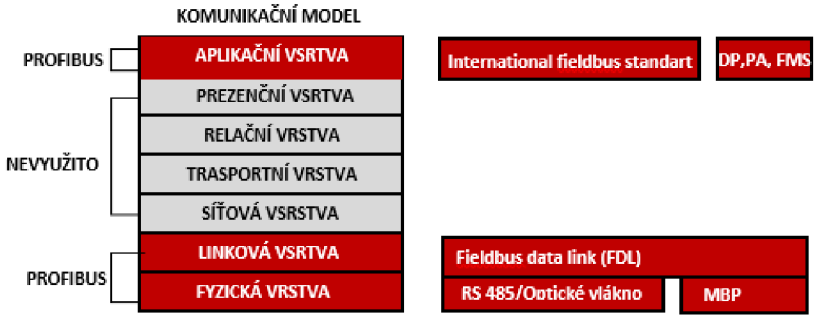
1) **Topologie** – možnost vytvořit libovolnou topologii, jelikož je zajištěna bezkoliznost pomocí arbitra přenosu (MN) a přiřazení časových oken pro jednotlivá zařízení (CN) v síti.
2) **Typ komunikace** – hlavním komunikačním modelem, který se využívá, je model producent-konzument.
3) **Rychlost komunikace** - je využíváno Ethernetu s přenosovou rychlostí 100 MB/s, přičemž délka jednoho segmentu je maximálně 100 m a na jeden segment maximálně 240 připojených zařízení.
4) **Základní princip funkce** – Ethernet Powerlink rozšiřuje klasický Ethernet o smíšeného dotazování a časového řazení. Je využíváno časového rozdělení přenosového cyklu na část přenosu časově kritických dat a část přenosu časově nekritických dat, o který se stará standartní protokol IP. V síti Ethernet Powerlink je dále dbáno na oddělení segmentů sítě pracujících v reálném čase a segmentů, které nepracují v reálném čase.
5) **Požadavky na hardware** – hardwarové prostředky se nijak neliší od klasického Ethernetu, jelikož řešení Ethernet Powerlink je čistě softwarové. V rámci této sítě je dovolenou použití zařízení hub, switch je zakázaný, protože je vyžadováno přesné časování sběrnice.



1.31 Základní architektura sítě Ethernet Powerlink [13]

1.3.4 Profibus

Profibus je průmyslová komunikační sběrnice určena především pro aplikace se zaměřením na automatizaci. Pro tuto sběrnici existují dvě varianty komunikačního protokolu, a to Profibus DP pro průmyslovou automatizaci a Profibus PA pro procesní automatizaci.



1.32 Komunikační model Profibus [15]

Základní charakteristiky a parametry Profibus DP [17] [18]

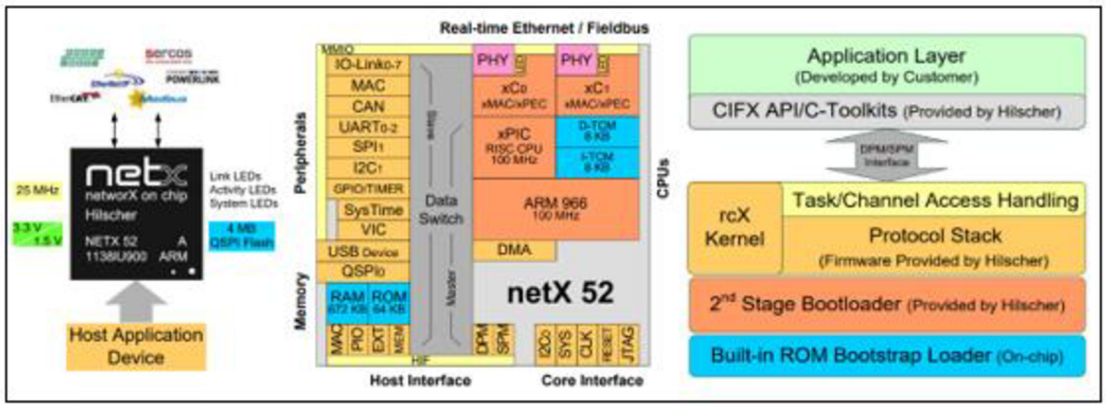
1) **Topologie** – podpora topologií sběrnice (preferováno), hvězda, strom a kruh
2) **Přenosová technologie** – pomocí stíněné kroucené dvojlinky a standartu sériové komunikace RS-485, pro který platí, že může být použito 32 stanic v jednom segmentu a celkem 127 stanic. Další možností je použití optického vlákna, kde můžeme realizovat přenos až na 80 km. Možností je i kombinace mezi RS-485 a optického vlákna, pomocí převodníku.
3) **Přenosová rychlost** – definované jsou určité maximální rychlosti pro vzdálenosti přenosového média. Nejnižší definovanou rychlostí je 9,6 kbit/s pro vzdálenost 1200 metrů a nejvyšší definovanou rychlost 12 Mb/s pro vzdálenost 100 metrů. Pro větší přenosové rychlosti již nelze zaručit. Přístup na sběrnici je deterministický, lze tedy garantovat cyklus sběrnice.
4) **Základní princip funkce** - sběrnice Profibus DP je využívaná pro cyklickou výměnu dat mezi řídícím PLC automatem a všemi jeho senzory a akčními členy. Využívána může být architektura monomaster-slave nebo multimaster-slave. Při využití multimaster-slave si musejí dávat jednotlivé řídící jednotky dávat přednost na sběrnici. Je umožněno jednoduše konfigurovat vzdálené periferie ve velkém množství a jednotlivé konfigurace mezi vstupy a výstupy PLC jsou cyklicky zapisovány a čteny. Dále je podporována acyklická komunikace mezi PLC a vzdálenými periferiemi jakou jsou např. HMI panely a acyklická komunikace pro konfigurační nástroje, které obstarávají změny nastavení různých hodnot či alarmů.

1.3.5 Průmyslový ethernet pomocí Netx52

Pro příklad, možné integrace podpory průmyslových komunikačních sítí do embedded zařízení, je uveden následující obvod. NetX 52 je komplexní integrovaný komunikační řadič pro protokoly průmyslového Ethernetu od společnosti Hilscher. Tento vysoce výkonný řadič umožňuje implementaci protokolů průmyslového ethernetu jako jsou ErherCat, Powerlink nebo Profinet. Fungovat může například jako doprovodný čip pro zprostředkování komunikace mezi průmyslovou sítí a řídícím mikrokontrolérem. Pro komunikaci disponuje SPI rozhraním pro rychlé čtení a zápis. [19]

Funkce a vlastnosti [19]

1) **Obecné informace** - komunikační rozhraní Netx52 je navrženo pro nízkou latenci, determinismus a flexibilitu. Je vhodné pro aplikace, které vyžadující práci v reálném čase. Postaveno na heterogenní vícejádrové architektuře obshaující jádro Arm spojené s řadičem periferního rozhraní (xPIC) a subsystémem komunikace (xC).
2) **Softwarové řešení protokolu** - nahráno v paměti čipu jako firmware. Dostupné zásobníky jsou pro protokoly EtherCat, Ethernet/IP, Modbus/TCP, Powerlink, Profinet nebo Sercos.
3) **Softwarové řešení pro hostitelské zařízení** - softwarové nástroje CIFX API / C.
4) **Průmyslové komunikační rozhraní** – dvoukanálové komunikační rozhraní s 10/100 Mbps Ethernetovými porty.
5) **Hostitelské komunikační rozhraní** – možnost bezproblémového připojení téměř jakékoliv hostitelské aplikace, buď pomocí paralelního rozhraní DPM, které poskytuje paralelní paměťovou sběrnici s daty šířky 8/16/32 bitů a adresami šířky 11 až 20 bitů, nebo pomocí sériového rozhraní SPM, které představuje vysokorychlostní sériovou sběrnici typu SPI nebo Quad SPI.



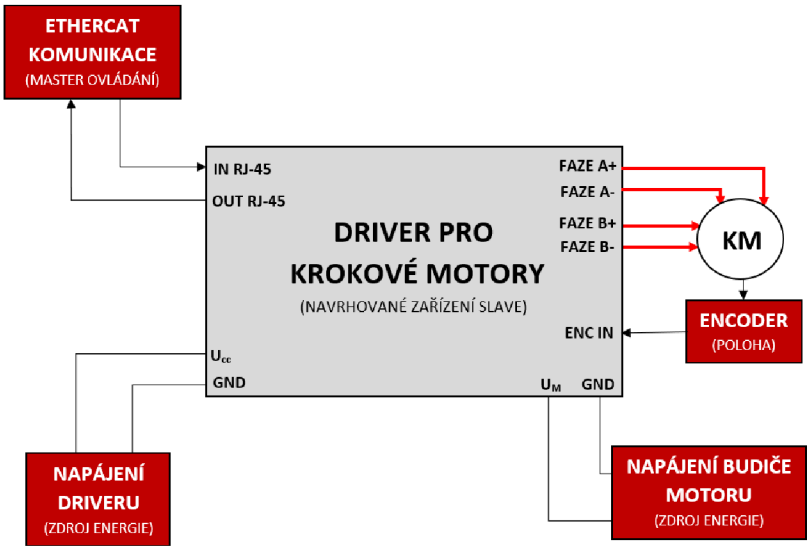
1.33 Venšjí a vnitřní popis obvodu NetX 52 [19]

2 PRAKTICKÁ ČÁST

Tato část práce se zabývá samotným návrhem a realizací driveru. Jednotlivé kapitoly jsou zde členěny dle posloupnosti jednotlivých prací. Cílem je definovat jasný a srozumitelný koncept, vybrat vhodné hardwarové komponenty, realizovat hardwarovou část, vytvořit softwarové vybavení a následně vše vhodným způsobem otestovat.

2.1 Popis a požadavky na výsledné zařízení

Před započetím samotného návrhu bylo nutné definovat konkrétní požadavky na výsledné zařízení, jako je popis hlavní funkcionality, vstupy, výstupy, výsledné parametry nebo specifické řídící funkce. Podle těchto požadavků se řídil celý návrh, nicméně s drobnými úpravami dle nově vzniklých skutečností. Na obrázku **2.1** je možné vidět představu výsledného driveru z vnějšího pohledu.



2.1 Vnější popis driveru

2.1.1 Hlavní požadavky

1) **Hlavní funkcionalita** – vytvoření driveru, který umožňuje řízení hybridních dvoufázových krokových motorů velikosti až NEMA23.

2) **Možnosti ovládání** – kompletní ovládání pomocí EtherCat standartu. Tento standart byl vybrán především z důvodu velkého rozmachu aplikací, ve kterých jsou využívány protokoly průmyslového ethernetu a také z požadavků velmi krátkého cyklu, který je při řízení podobných zařízení vyžadováno.

3) **Způsoby řízení** – vytvořit takové řešení, které bude podporovat řízení krokového motoru, jak v otevřené smyčce, tak ve smyčce uzavřené. Další důležitou části je možnost volby mezi řízením na rychlost nebo na polohu.

4) **Rychlostní profily** – možnost volby rozběhového rychlostního profilu typu obdélník, lichoběžník nebo s-křivka.

2.1.2 Požadované vnější vstupy a výstupy

Následující **Tabulka 2.1** uvádí požadované fyzické vstupy a výstupy a jejich základní popis vzhledem k funkcionalitě. Z uvedeného soupisu vyplývá jasný požadavek na ovládání pouze pomocí průmyslové komunikace, tudíž nebudou obsaženy žádné vstupní ovládací prvky.

Tabulka 2.1 Požadované vstupní a výstupní rozhraní driveru

Číslo	**Vstupy**	**Popis**
1	Napájení U_M	Vstup napětí pro buzení motoru
2	Napájení U_L	Vstup napájení pro logickou část driveru
3	RJ-45 IN	Vstupní port pro EtherCat
4	Enkodér IN	Vstupní rozhraní pro ABN enkodér
Číslo	**Výstupy**	**Popis**
1	Fáze A+, A-	Fáze bipolárního krokového motoru
2	Fáze B+, B-	Fáze bipolárního krokového motoru
3	RJ-45 OUT	Výstupní port pro EtherCat

2.1.3 Požadované parametry

Následující **Tabulka 2.2** uvádí základní výsledné parametry, které jsou očekávány od výsledného driveru. Uvedeny jsou pouze typické hlavní parametry.

Tabulka 2.2 Základní požadované parametry driveru

Číslo	**Parametr**	**Popis parametru**
1	Provozní napětí [V]	9 ž 40 V
2	Trvalý proud na fázi [A]	2 A
3	Maximální proud na fázi [A]	3 A
4	Režim mikrokrokování	1/1 až 1/256

2.2 Blokové schéma

Níže uvedené blokové schéma, na obrázku **2.2**, popisuje architekturu navrhovaného driveru z pohledu nejdůležitějších bloků a znázornění základního principu. Zařízení jako celek, je rozděleno na tři hlavní bloky, které pracují nezávisle na sobě, pouze si vzájemně vyměňují potřebné informace pro svoji činnost.

Komunikační jednotka

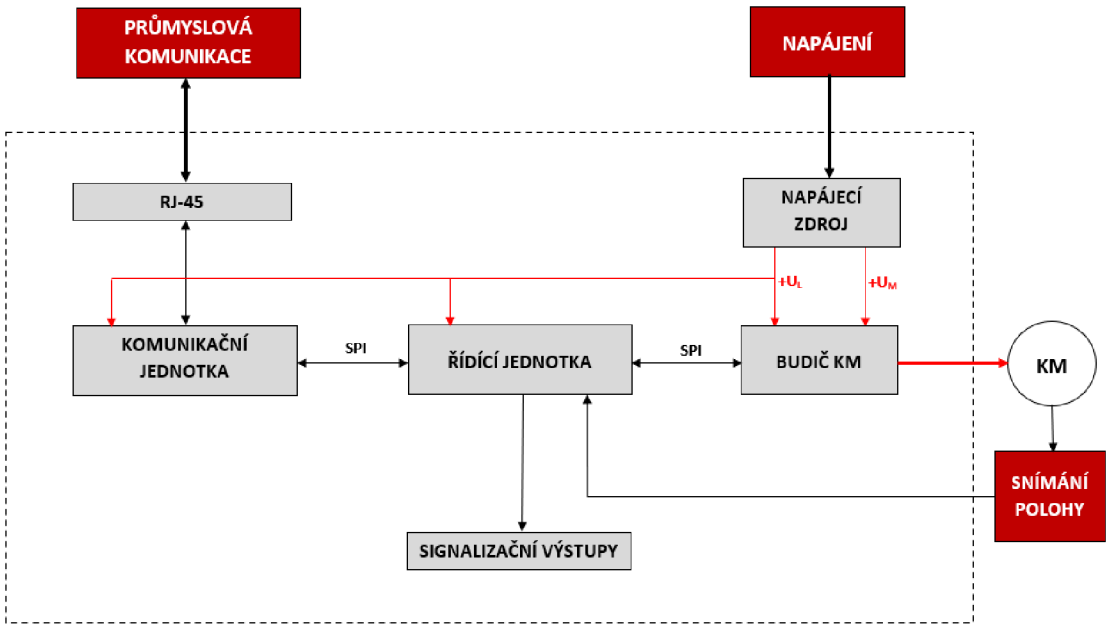
Tento blok slouží jako rozhraní, pro vybranou průmyslovou komunikaci, mezi řídící jednotkou a nadřazeným řídícím systémem. Z praktického hlediska bylo nutné vybrat vhodný komunikační kontrolér. Primárním cílem bylo vybrat kontrolér, který má vlastní řešení vybrané komunikace.

Řídící jednotka

Tento blok sdružuje všechny části, které slouží k řízení krokového motoru skrze jeho budič. Požadované parametry pro řízení řídící jednotka vyčítá skrze komunikační jednotku. Na základě těchto parametrů a zpětné informaci o poloze motoru jednotka vydává příslušné konfigurační a řídící povely pro budič krokového motoru.

Budič krokového motoru

Budič krokového motoru slouží jako výkonová část zařízení, která na základě řídících povelů dodává příslušné průběhy proudu do jednotlivých fází bipolárního motoru.



2.2 Základní blokové schéma

2.3 EtherCat komunikace podrobněji

V souhrnné kapitole **1.3.1** byly rozděleny, a základním způsobem popsány, často používané typy protokolů, které využívá průmyslový ethernet. Z definovaných požadavků na výsledný driver vyplynulo, že primární a také jedinou možností pro ovládání driveru je možnost pomocí nadřazeného systému, komunikujícího standartem EtherCat. Tato kapitola rozšiřuje základní informace o tomto standartu a opakují se zde některé základní informace z kapitoly **1.3.2.**

2.3.1 Možnosti topologie

Přednost EtherCat standartu spočívá především v principu zpracování EtherCat rámců. Oproti klasickému Ethernetu je zaručeno zpracování za běhu sítí, čehož je docíleno tím, že každé zařízení v síti obsahují minimálně dva porty. V síti nejsou používány žádné přepínače ani další podpůrné zařízení. Podporována je velká flexibilita topologie, jelikož na logické úrovni je vždy vytvořena topologie kruhová. Na úrovni fyzické je možné vytvořit topologii typu kruh, strom, linie, hvězda či další. Další důležitou vlastní je velká odolnost a libovolné zakončení sítě. Pojmem odolnost je myšleno to, že pokud je síť rozpojena například vinnou poruchy podřízeného zařízení nebo fyzické cesty, tak je toto místo detekováno, a pokud je to možné, tak je přizpůsobena cesta průchodu rámce. Pojmem zakončení sítě je myšlena detekce posledního zařízení v sítí, ve chvíli, kdy není dodržena kruhová topologie. [20] [21]

Pro shrnutí lze uvést následující. Výhody značně převyšují nad nevýhodami, zajištěna je výborná flexibilita, determinismus nebo absence přídavných hardwarových zařízení.

2.3.2 Požadavky na vybavení pro fyzickou vrstvu

EtherCat využívá standartní fyzickou vrstvu podle IEEE 802.3. Může tedy být využito standartního vedení pomocí kroucené dvojlinky, s označením 100BASE-TX, nebo optického vedením s označením 100BASE-FX. Propojení je možné vytvořit například i pomocí rozhraní MII. [20] [21]

2.3.3 Požadavky na HW vybavení master zařízení

Funkce master zařízení je dána především použitým softwarem. Z hardwarového hlediska stačí použít standartní PC, průmyslové PC, PLC automat nebo síťový kontroler. Podmínkou je přítomnost síťové karty a minimálně jednoho ethernetového portu. Všechny další vlastnosti jsou dány použitým softwarem a operačním systém. [20] [22]

2.3.4 Požadavky na HW vybavení slave zařízení

Na podřízená zařízení v EtherCat síti jsou kladeny ty nejvyšší požadavky na rychlé zpracování dat. Tyto požadavky jasně vyplývají již ze samostatného principu a topologie, jelikož průchozí rámce jsou zpracovávány během průchodu sítí. Obvody, které zajišťují podřízeným zařízením v síti tuto funkcionalitu se nazývají ESC (EtherCat slave controller). Obecnou blokovou strukturu takového obvodu ukazuje obrázek **2.3** a v následujícím textu kapitoly jsou hlavní bloky stručně popsány pomocí zdroje [22].

Process data interface

Rozhraní pro předávání procesních dat. Připojený mikrokontrolér může pro čtení dat využít SPI rozhraní nebo paralelní digitální sběrnici. Zároveň generuje přerušení.

FMMU

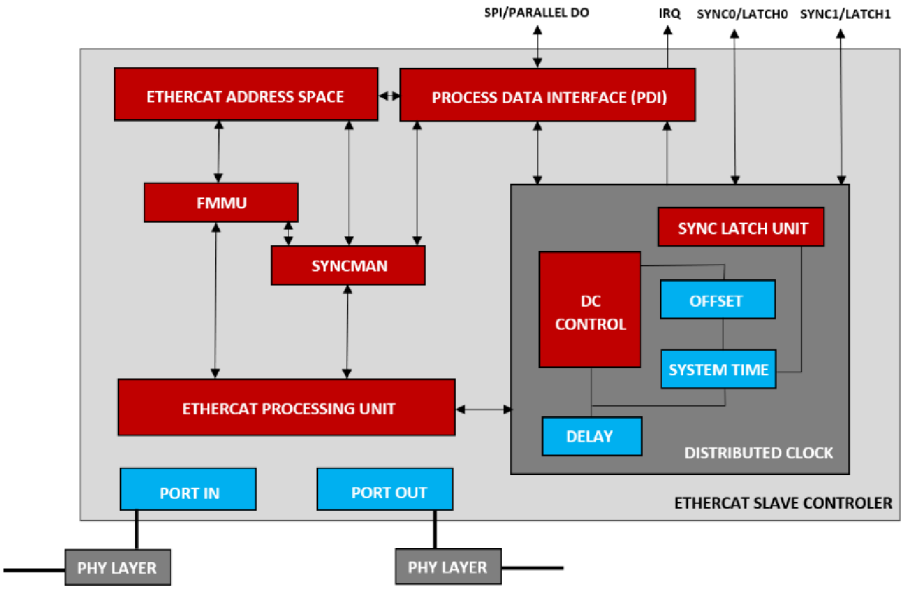
Jednotka správy paměti, která se stara o mapování logických adres na fyzické adresy.

Distributed clock

Jednotka slouží pro generování synchronizačních signálů. Těchto signálů je využito při synchronizované čtení/zapisování procesních dat a při synchronizaci celé sítě.

SyncManager

Obstarává mechanismus správné výměny dat s master zařízením. Může pracovat v buffer módu, který slouží pro cyklickou výměnu dat a k těmto datům není omezen přístup ani z jedné strany, nebo může pracovat v mailbox režimu, kde lze k datům přistupovat vždy až po dokončení přístupu jedné nebo druhé strany.



2.3 Obecná struktura ESC [22]

2.3.5 Struktura EtherCat rámce

Pro komunikaci pomocí standartu EtherCat jsou využívány standartní ethernetové rámce IEE802.3. Obrázek **2.4** ukazuje rozvětvenou strukturu od základního ethernetového rámce až po strukturu jednotlivých EtherCat datagramů. Rámec je opatřen standartní Ethernet hlavičkou a pro identifikaci rámec typu EtherCat slouží část s názvem EtherType, kde musí být obsažena hodnota 0x884A. Část týkající se EtherCat protokolu je uložena v rámci Ethernet data, a skládá se z hlavičky a za sebou uložených EtherCat datagramů. Struktura hlavičky a datagramu je následně popsána podrobněji. [24]

EtherCat hlavička

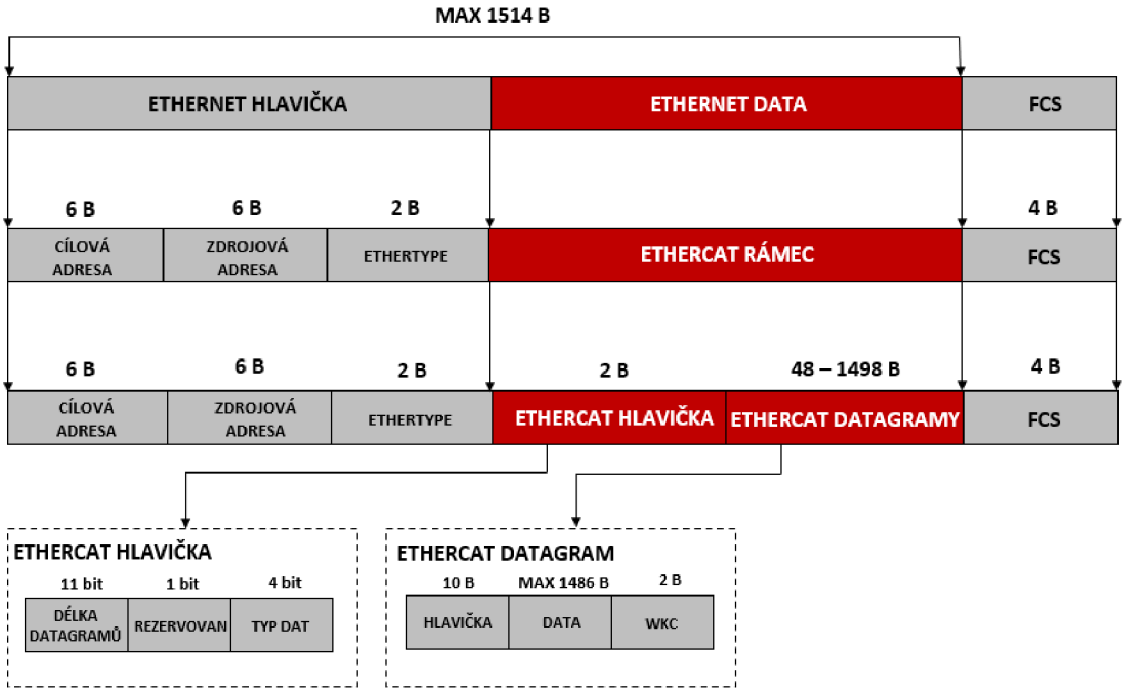
Hlavička obsahuje celkem 16 bitů a je rozdělena na tři datové úseky.

1) **11 bitů** – Část, která obsahuje informaci o celkovém počtu datagramů.
2) **1 bit** – Rezervován.
3) **4 bity** – Část, která určuje typ protokolu.

EtherCat datagramy

Část určená pro přenos procesních dat. Každý datagram má opět definovanou strukturu.

1) **10 bytů -** Hlavička datagramu obsahuje všechny potřebné provozní informace.
2) **Max 1486 bytů** – Datový prostor pro přenášená procesní data.
3) **2 byty** – Pracovní čítač.



2.4 Struktura EtherCat rámce [24]

2.3.6 Možnosti adresování

Master zařízení používá pro práci s podřízenými zařízeními v topologii sítě tři základní typy adresování, jejichž význam je odlišen účelem použití. V následujícím textu jsou krátce popsány jednotlivé typy a obrázek **2.5** ukazuje odlišnost ve struktuře adres.[22]

Poziční adresování

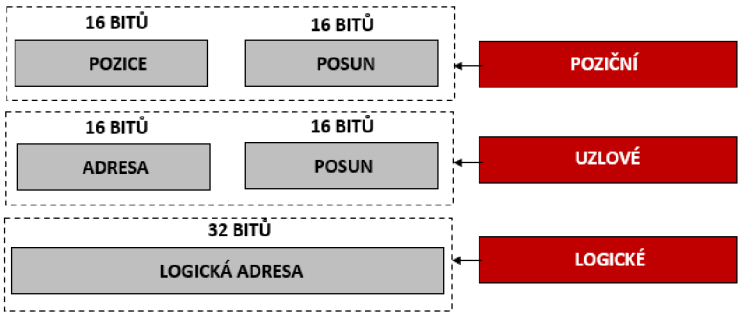
Tento typ adresování je využit především při prvotním skenování sítě, kdy master provede automatické skenování dostupných zařízení v síti a každému přiřadí poziční adresu podle pořadí v dané topologii. Adresa každého zařízení odpovídá jeho pozici. [22]

Adresování pomocí uzlů

Tento typ adresování je využit pro přístup k registrům podřízených zařízení. Vyplývají podmínkou pro použití tohoto typu je úspěšná předchozí identifikace. Žádaný příkaz obsaženy v EtherCat datagramu je vykonán, pokud se adresa uzlu shoduje konfigurovatelnou adresou nebo jejím aliasem uloženým v ESI-EEPROM. [22]

Logické adresování

Tento typ adresování umožňuje bitové přiřazení dat. Všechna ESC čtou data ze stejného logické prostoru o velikosti 4 GB. Každé podřízené zařízení obsahuje jednotku FMMU, která se stará o převedení logických adres na místní fyzické adresy. Při startu sítě master zařízení konfiguruje u každého podřízeného zařízení právě jednotku FMMU, takže zařízení ví, jaký logický prostor mu náleží. Mechanismus logického adresování značně snižuje režii komunikace a je efektivně využíváno při přístupu k procesním datům. [22]



2.5 EtherCat - typy adresování a struktura adres [22]

2.3.7 Možnosti synchronizace

EtherCat je často využíván v aplikacích, kde je kladen požadavek na synchronizaci, tedy na vykonávání více úkonů v přesně synchronizovaném čase. Pro účel synchronizace obsahují ESC distribuované hodiny. Podřízené zařízení má schopnost zapsat do datagramu čas příchodu, a naopak čas odchodu. Díky těmto informacím může master zařízení vyrovnat zpoždění v celé síti. Po vypočítání všech potřebných parametrů na synchronizaci je vyslán tzv. vysílací předpis pro synchronizaci hodin všech zařízení. [24]

2.3.8 EtherCat stavový automat

Mechanismus nazývaný jako ESM (EtherCat state machine) je v rámci sítě využíván pro jasný přehled o stavu podřízených zařízení. Každé podřízené zařízení pracuje vždy v jednom definovaném stavu. Jednotlivé stavy umožňují přístup ke konkrétním funkcím zařízení. Pro přechod mezi jednotlivými stavy musí být odeslán požadavek od master zařízení, a pokud jsou splněny požadavky na přechod do dalšího stavu, tak je tak učiněno a zpětně odesláno potvrzení o úspěšném přechodu. Pokud ne, tak je vyvolána chyba. Stavový automat jako celek, včetně závislosti stavů, je vyobrazen na obrázku **2.6.** a následující odstavce popisují jednotlivé stavy pomocí zdroje [23].

Init

Výchozí stav každého podřízeného zařízení po zapnutí napájení. Ve stavu inicializace není umožněn žádný datový přenos. Master má přístup pouze k registrům, které slouží pro nastavení mailbox režimu a správce synchronizace.

Pre-operational

Po přechodu do toho stavu je možná komunikace v mailbox režimu. Stále není možné zpracovávat žádná procesní data. Master provede všechna potřebná nastavení pro správné použití jednotky FMMU a další. Zároveň jsou vyčtena všechna nastavení z EEPROM.

Safe-operational

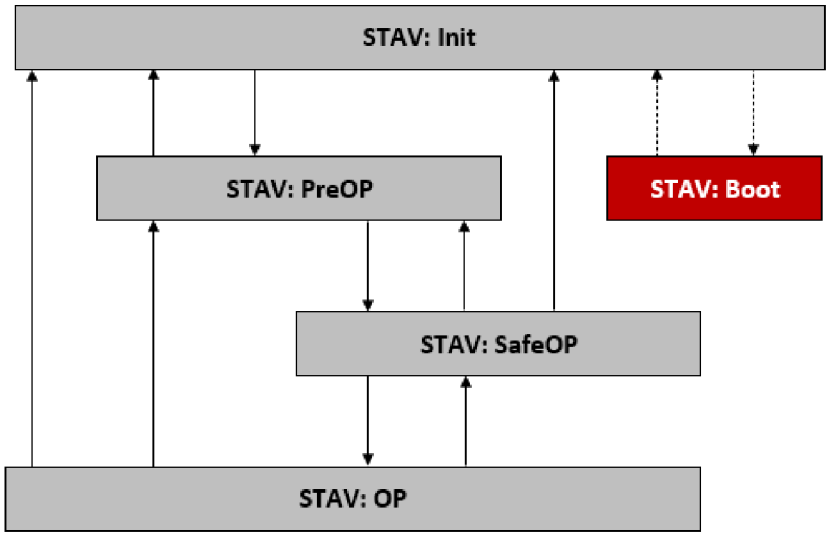
V tomto stavu je již možný i přenos a zpracování procesních dat. Vstupní data jsou cyklicky aktualizována, výstupní data nelze nastavit, jelikož výstupy podřízeného zařízení jsou udržovány v tzv. bezpečném stavu. V tomto stavu jsou zároveň kontrolována nastavení.

Operational

Tento stav umožňuje již neomezenou mailbox komunikaci a přenos procesních dat. Nutnou podmínkou pro přechod ze stavu safe-oprational je nastavení platných dat na výstupech, které byly do té doby nastaveny v bezpečném stavu.

Bootstrap

Stav, který je nazývaný jako volitelný, ale ve většině případů je využíván. V tomto stavu je využíván pouze mailbox režim a není umožněn přenos procesních dat. Využit je konkrétně rozšiřující protokol FoE, který slouží pro přenos souborů sítí a využíváno toho bývá například při aktualizaci firmwaru ESC.



2.6 EtherCat stavový automat [23]

2.3.9 Možnosti rozšířených profilů

Pro snadnou obsluhu specifických zařízení nebo činností jsou pro EtherCat implementovány tzv. rozšířené profily, které je možné využít v mailbox režimu. Tyto profily jsou níže vyjmenovány a je popsán jejich význam z hlediska uplatnění pomocí zdroje [20].

Ethernet over EtherCAT (EoE)
Profil pro umožnění ethernetového datového přenosu skrze EtherCat síť. Ethernet zařízení jsou v síti připojena pomocí switchportu.

CAN application layer over EtherCAT (CoE)
Profil umožňuje komunikovat se zařízeními pomocí standartního přístupu CanOpen, kde jsou využíváno PDO a SDO.

File Access over EtherCAT (FoE)
Profil pro možnost přístupu k souborům podřízených zařízení v síti. Využíván bývá pro nahrávání nových či aktualizovaných konfiguračních souborů.

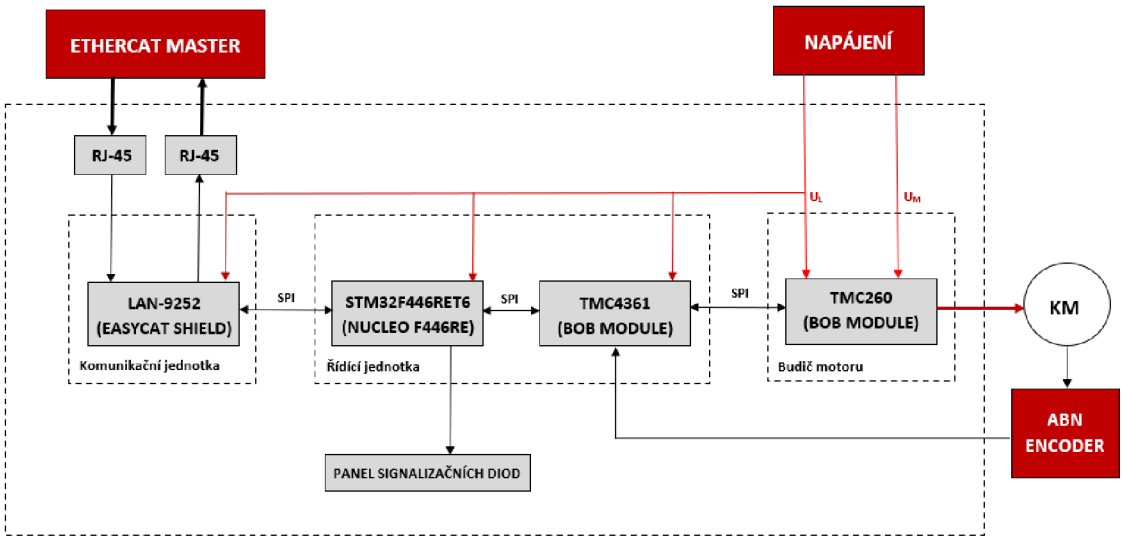
Servo Profile over EtherCAT (SoE)
Profil vhodný pro řízení servopohonů, kde jsou kladeny nejvyšší požadavky, na řízení v reálném čase. Tento profil je také známý pod názvem SERCOS.

2.4 Návrh hardwarové realizace

Tato kapitola popisuje návrh hardwarové části driveru. Základním předpokladem pro úspěšnou realizaci byl výběr vhodných komponent splňujících koncept, popsaný v kapitole **2.1** a graficky vyjádřen pomocí blokového schématu na obrázku **2.2**. Cílem návrhu bylo vytvořit funkční prototyp, kde není dbáno na integraci celého hardwaru na jeden plošný spoj, ale jsou použity vhodně propojené modulové komponenty.

2.4.1 Podrobné blokové schéma

Následující obrázek **2.7** rozvíjí blokové schéma do podrobnější struktury. Obecné bloky jsou již nahrazeny konkrétními obvody, respektive vývojovými shieldy, jenž tyto obvody obsahují.



2.7 Podrobné blokové schéma

Následující **Tabulka 2.3** ukazuje přehled hlavních komponent, které byly pro realizaci vybrány. Tyto komponenty jsou podrobněji popsány v následujících kapitolách.

Tabulka 2.3 Hlavní použité komponenty pro hardwarovou realizaci

P.	Popis	Použitý obvod	Použití vývojový shield
1.	Řídící mikrokontrolér	STM32F446RET6	NUCLEO-F446RE
2.	Servo-kontrolér	TMC4361	TMC4361-BOB
3.	Budič motoru	TMC262	TMC262-BOB40
4.	Komunikační kontroler	LAN9252	EasyCat

2.4.2 Řídící mikrokontrolér STM32F446RET6

Obecně lze konstatovat, že základem každého embedded zařízení je řídící mikrokontrolér, který obsahuje řídící software pro vykonávání specifické činnosti. Při výběru vhodného mikrokontroléru bylo důležité nejprve definovat úkony, které budou pomocí jeho činnosti prováděny a následně na základě těchto požadavků provést vhodný výběr.

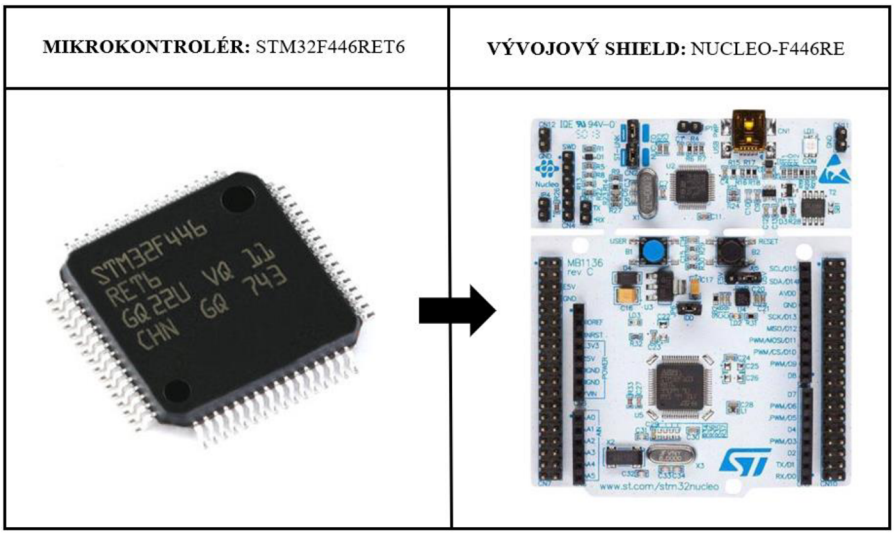
Z blokového schématu uvedeném na obrázku **2.7** jasně vyplývá, že mikrokontrolér je rozhraním mezi ECS a částí driveru, která se stará o řízení pohybu motoru. Mikrokontrolér má za úkol zapisovat a vyčítat procesní data EtherCat komunikace a na základě těchto dat nastavovat parametry, spouštět řízení, hlídat mezní stavy nebo vyčítat provozní hodnoty. Tyto požadavky jsou v kombinaci s EtherCat sítí náročné na vykonávání v reálném čase, takže je potřeba, aby byl procesor mikrokontroléru dostatečně rychlý. Dalším důležitým požadavkem z hlediska periferií je požadavek na minimálně dvě hardwarové SPI rozhraní. Pokud by probíhala vnější SPI komunikace na stejných hardwarových výstupech, byla by omezena rychlost mezi vyčítáním či odesíláním procesních dat a řízením motoru, jelikož by bylo vždy nutné přepínat pomocí fyzického výstupu využívané zařízení.

Pro vývoj prototypových zařízení jsou v dnešní době hojně využívány vývojové shieldy, kde mezi nejznámější patří desky Arduino, EPS nebo stále rozšiřující se shieldy založené na rodině mikrokontroléru STM32. V případě navrhovaného driveru byla pozornost zaměřena na desky typu NUCLEO, které nabízí mikrokontrolér s velmi dobře přizpůsobenými periferiemi a rozhraním pro rychlé programování. Konkrétně byla vybrána deska NUCLEO-F446RE s mikrokontrolérem STM32F446RET6, jehož základní parametry a vnější periferie shrnuje následující **Tabulka 2.4**.

Tabulka 2.4 Důležité parametry a periferie mikrokontroléru STM32F446RET6 [25]

P.	Parametr/Periferie	Hodnota/Počet
1.	Procesor	32-bit ARM Cortex-M4 s FPU
2.	Frekvence	180 MHz (max)
3.	Napájení	1,7 až 3,6 V
4.	FLASH	512 kB
5.	SRAM	128 kB
6.	SPI	4x
7.	USART	4x
8.	UART	2x
9.	GPIO	50x (možnost externího přerušení)
10.	CAN	2x

Uvedená **Tabulka 2.4** uvádí pouze nejzákladnější přehled. Je nutné podotknout, že ve chvíli, kdy by bylo cílem vytvořit zařízení integrované na společném plošném spoji s jasnými nároky na cenu a další parametry, tak by bylo důležité provést důkladnější výběr. Pro vyvíjený prototyp byla však tato varianta zvolena jako vhodná. Vybraný mikrokontrolér a vývojový shield jsou vyobrazeny na obrázku **2.8**



2.8 MCU STM32F446RET6 a příslušný vývojový shield NUCLEO-F446RE [25]

2.4.3 Řídící servo-kontrolér TMC4361

V dnešní době se na elektroniku, věnující se řízení motorů, specializuje velké množství firem a tím narůstá i počet rozmanitých možství, kterými je možné konkrétní aplikace řešit. Z požadavků uvedených v kapitole **2.1.1** je jasné, že stěžejní částí bude zpětnovazební řízení v uzavřené smyčce, kde potřeba vhodně ošetřit zpracování zpětné vazby, a dále generování různých rozběhových profilů. Nejdříve bylo tedy nutné definovat, kterou cestou se vydat.

Softwarové řešení

Pro řízení je využit mikrokontrolér a koncový budič krokového motoru. Veškeré generování rychlostních profilů (rampy), zpracování zpětné vazby a další výpočty probíhají v rámci řídícího softwaru, který je uložen v paměti mikrokontroléru.

Hardwarové řešení

Mezi mikrokontrolér a koncový budič je vložena další obvodová část nazývaná jako ovladač pohybu nebo také servo-kontrolér. Tato část má zpravidla uvedené funkce pro generování ramp a zpětnovazební řízení implementovány na hardwarové úrovni a jejich funkčnost je možné využít pomocí definovaného protokolu a vnější komunikace.

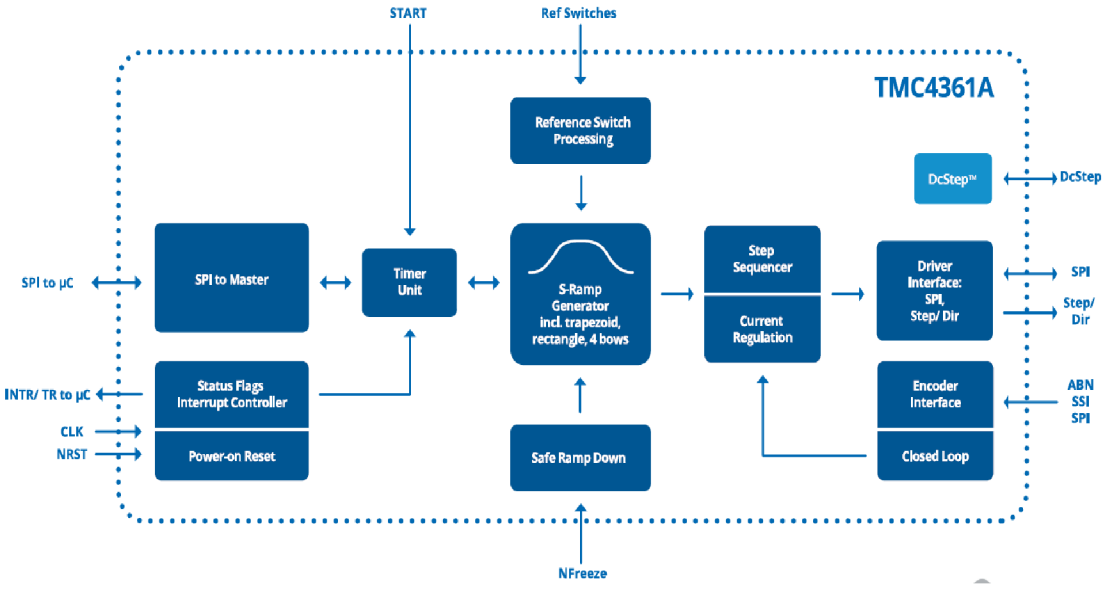
V rámci této práce byl zvolen postup pomocí hardwarového řešení. Hlavním důvodem bylo to, že driver jako celek má být zařazen v EtherCat síti, kde je kladen důraz na řízení v reálném čase s krátkou dobou cyklu, takže špatně implementované řešení na softwarové úrovni by mohlo způsobit spíše problémy.

Jednou z předních firem zaměřujících se na elektroniku pro řízení motorů je firma Trinamic, na jejíž obvody bylo dále zacíleno. Z hlediska ovladačů pohybu je firmou nabízeno skutečně mnoho možností, pro účel navrhovaného driveru byl vybrán obvod TMC4361, jehož funkce a parametry budou shrnuty v následujícím textu této podkapitoly.

Hlavní funkce TMC4361 [27]

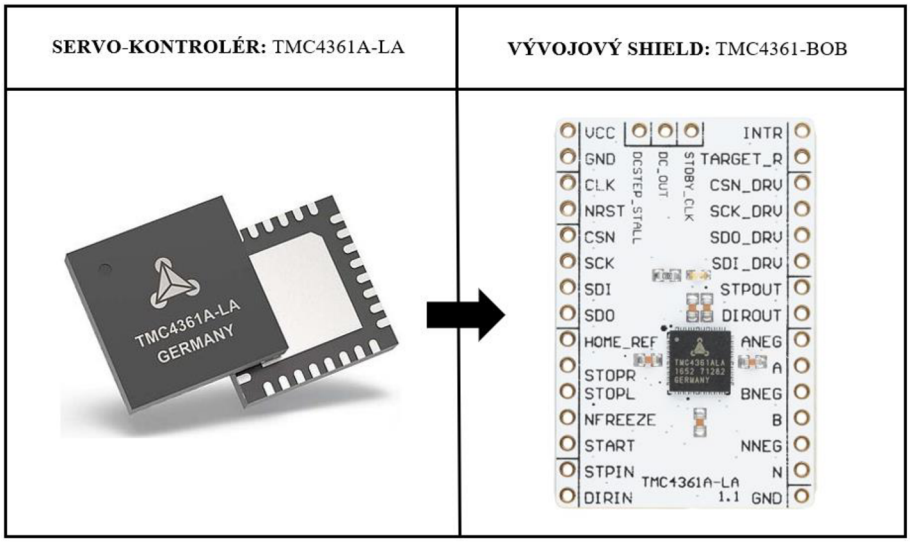
1) Jednoduché ovládací rozhraní funkcí pomocí SPI. Ovládání probíhá pomocí zápisu a čtení vnitřních registrů.
2) Výstupní SPI a STEP/DIR rozhraní pro koncové budiče krokového motoru. Z hlediska podpory je nejvhodnější kombinace opět s TMC obvodem.
3) Vnitřní generátor rampy s možností volby a parametrizace různých rozběhových profilů, jako je například S-křivka.
4) Podpora zpětnovazebního řízení a rozhraní pro připojení inkrementálních, absolutních či SPI enkodérů pro snímaní aktuální polohy a rychlosti motoru.
5) Podpora specializovaných funkcí jako jsou ChopSync nebo DcStop

Z pohledu možností vstupně-výstupního rozhraní obvodu a jeho hlavních funkcí je uveden obrázek **2.9**, který popisuje kompletní blokovou strukturu obvodu.



2.9 Bloková struktura obvodu TMC4361 [27]

Pro rychlé prototypování je obvod TMC4361 nabízen v několika verzích vývojových shieldů. Z nabízených možností byla vybrána verze shieldu typu BOB. Tato verze nabízí obvod na malém plošném spoji s přizpůsobenými vnějšími součástkami a vyvedenými vstupy a výstupy. Integrace obvodu TMC4361 do vývojového shieldu je vyobrazena obrázku **2.10**.



2.10 Servo kontrolér TMC4361 a příslušný vývojový shield TMC4361-BOB [27][26]

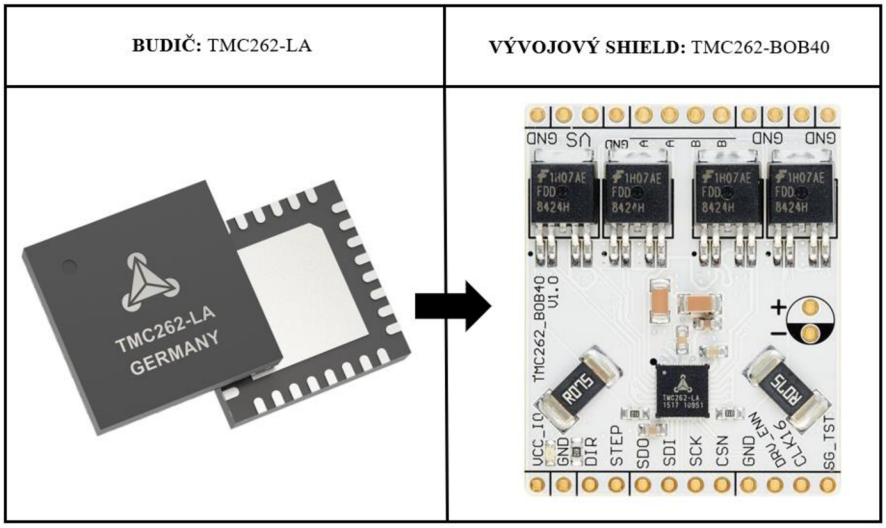
2.4.4 Budič krokového motoru TMC262 s externím mosfet H-můstkem

Jak již bylo uvedeno při popisu ovladače pohybu TMC4361, tak nejlepší variantou pro koncový budič krokového motoru je opět TMC obvod. Z technického datového listu vyplynulo jako nejvhodnější řešení použití budiče TMC262. Jedná se o univerzální budič dvoufázových bipolárních krokových motorů s rozsáhlými možnostmi nastavení a specializovanými funkcemi. Tyto funkce a možnosti jsou opět shrnuty v následujícím textu.

Hlavní funkce TMC262 [28]

1) Vysoký výkon při připojení externího H-můstku s MOSFET tranzistory. Možnost dosáhnutívýstupníhou proudu až 10 A, a napětí až 60 V.
2) Možnost vysokého rozlišení pomocí volby režimu mikrokrokování. Režim mikrokrokování lze nastavit až na hodnotu 256 mikrokroků na jeden krok.
3) Vstupní SPI rozhraní pro konfiguraci a následné ovládání pohybu motoru. Pro možnost ovládání pohybu je však obsaženo i klasické S/D rozhraní.
4) Specializována funkce StallGuard2, CoolStep, MicroPlyer nebo SpreadCycle, což jsou patentované funkce firmy Trinamic.

Obvod TMC262 je opět možné nalézt ve více typech vývojový shieldů, nicméně pro tuto aplikaci byl z hlediska kompaktnosti zvolen opět typ BOB. Výběr byl ze tří možností, které se liší možností výstupního výkonu, jelikož jednotlivé varianty v sobě obsahují již zabudovaný externí H-můstek. Vybrána byla varianta TMC262-BOB40, kde koncová číslovka značí maximální možné napětí pro výstup motoru, a v tomto případě je to 40V. Maximální výstupní proud jednou fází motoru je 2,8 A RMS. Integrace obvodu TMC262 do vývojového shieldu je vyobrazena na obrázku **2.11**.



2.11 Budič TMC-262 a příslušný vývojový shield TMC262-BOB40 [28][29]

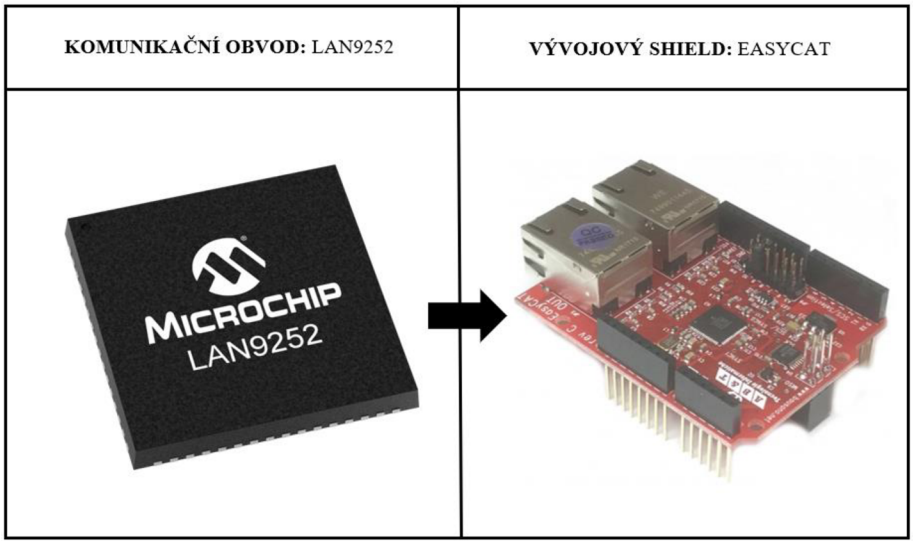
2.4.5 Komunikační obvod LAN9252

V kapitole **2.3.4** byly shrnuty požadavky na hardwarové vybavení pro EtherCat podřízené zařízení a byla popsána základní struktura obecného ESC. Na základě těchto teoretických poznatků byly uvažovány a otestovány dvě varianty obvodů. První variantou byl obvod NetX52 ve vývojovém NSHIELD 52-RE a druhou variantou obvod LAN9252 ve vývojovém shieldu EasyCat. Pro výsledné řešení návrhu driveru byl vybrán obvod LAN9252. Tento řadič síťové EtherCat komunikace je přímo určený pro aplikace, kde má být cílové zařízení nasazeno jako podřízené zařízení v EtherCat síti. Samotný katalogový list LAN9252 uvádí vhodnost pro aplikace jako je řízení motorů či další odvětví automatizace.

Hlavní funkce a vlastnosti LAN9252 [30]

1) 2/3 portový EtherCat řadič s integrovanými ethernetovými PHY. Je podporován standart 100BASE TX a rychlost 100Mbps. Fyzické RJ-45 konektory obsahují LED diody pro indikaci přenosu.
2) Obvod odpovídá standartní struktuře ESC. Obsahuje tři jednotky FMMU, čtyři jednotky SyncManager, distribuované hodiny pro synchronizaci a DPRAM paměť o velikosti 4kB.
3) Podpora standartního buffer a mailbox režimu pro výměnu procesních dat.
4) Podpora vnější výměny dat pomocí SPI rozhraní nebo až 16-bitového digitálního rozhraní. Dále jsou k dispozici synchronizační výstupy a výstupy pro přerušení.

Vývojový shield EasyCat využívá zapojení LAN9252 v tzv. režimu mikrokontroléru, kde je pro komunikaci ze strany podřízeného zařízení využita externí komunikace pomocí SPI rozhraní. Integrace obvodu LAN9252 do vývojového shieldu ukazuje obrázek **2.12.**

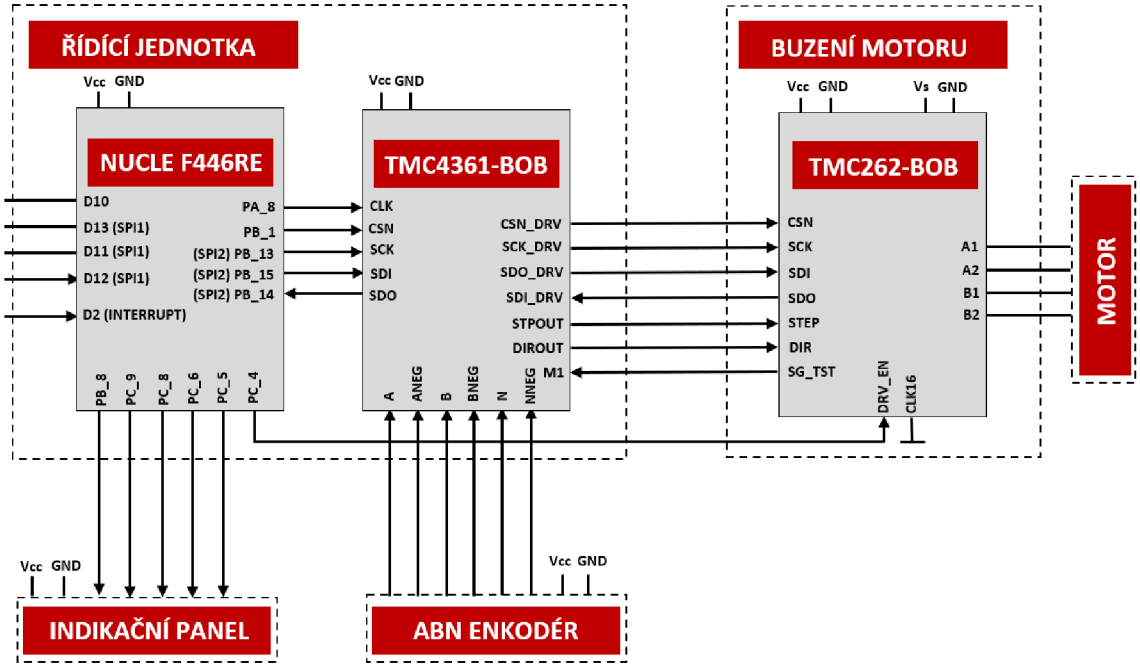


2.12 Komunikační obvod LAN9252 a příslušný vývojový shield EASYCAT [30][31]

2.4.6 Kompletní elektrické zapojení

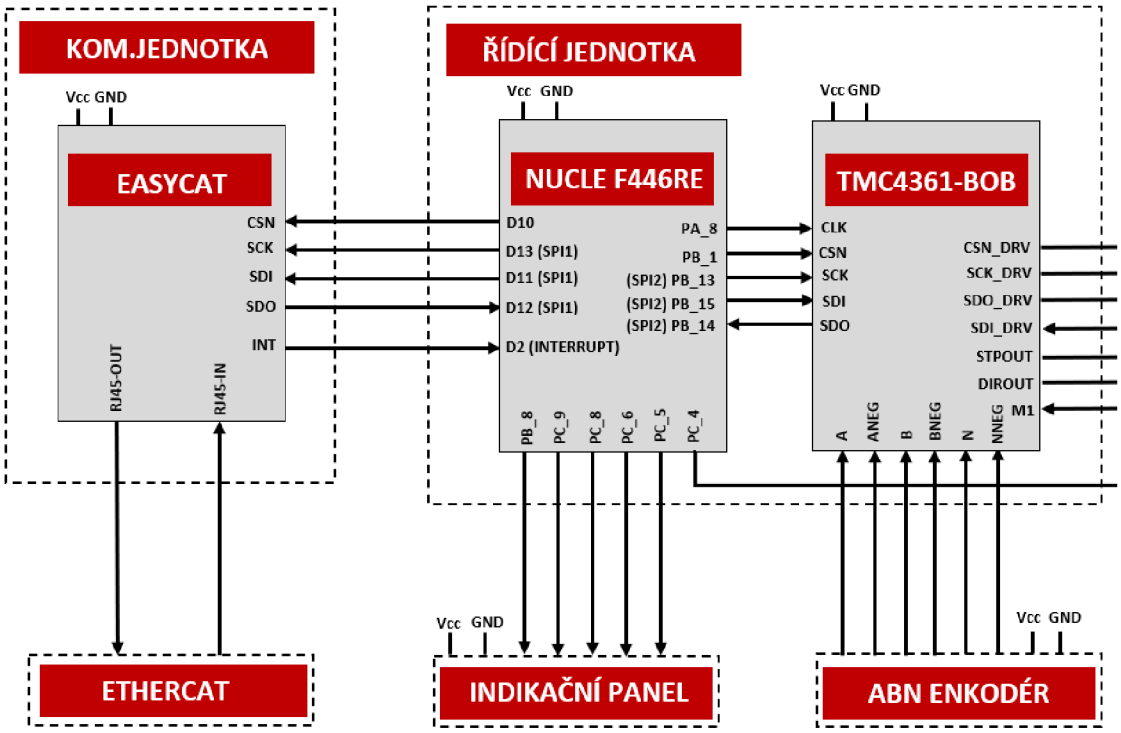
Při propojování jednotlivých komponent bylo postupováno v souladu s doporučenými zapojeními od výrobců. Zprvu je dobré zmínit základní společné informace. Napájení pro všechny komponenty prototypu je na standartní úrovni +5V a pro výstupní krokový motor je možnost připojit napětí v rozmezí 9-36V s možností přetížení na 40V. V rámci prototypu nejsou řešeny žádné ochrany proti vysokému napětí či přepólování, proto bylo důležité pracovat s kvalitním externím zdrojem, který těmito vlastnostmi disponuje.

Elektrické propojení rozšiřuje rozvětvené blokové schéma z obrázku **2.7** na úroveň propojení jednotlivých vstupů, výstupů a sběrnic. Při grafickém popisu elektrického propojení je opět respektováno rozložení na hlavní bloky. Shieldy NUCLEO F446RE a TMC4361-BOB tvoří řídící jednotku a komunikují pomocí SPI rozhraní s označením SPI2. Pro správnou funkci TMC4361 je generován taktovací signál o frekvenci 16 MHZ. Řídící jednotka nadále obstarává stav indikačního panelu pomocí digitálních výstupů a zpracování signálu z ABN enkodéru pomocí vstupního rozhraní TMC4361. Komunikace s budičem TMC262 je prováděna pomocí výstupního SPI a S/D rozhraní TMC4361. Pro povolení práce s výstupním budičem je použit digitální výstup. Motor je připojen pomocí standartního připojení dvoufázových hybridních krokových motorů. Propojení řídící jednotky, buzení motoru a dalších částí ukazuje následující obrázek **2.13**.



2.13 Elektrické propojení řídící jednotky a budiče

Komunikace mezi řídící a komunikační jednotkou probíhá také pomocí SPI rozhraní, tentokrát s označením SPI1. Toto propojení ukazuje obrázek **2.14**. Byl tedy splněn požadavek na oddělení komunikace pomocí využití více hardwarových SPI. Celkové zapojení je uvedeno v příloze **A.6.**

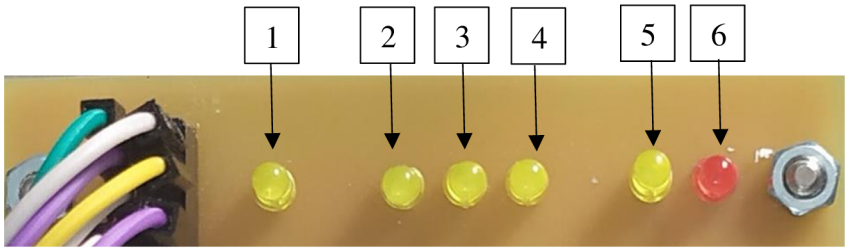


2.14 Elektrické propojení řídící jednotky a komunikační jednotky

2.4.7 Indikační panel s diodami

Pro základní přehled stavu driveru byl vytvořený jednoduchý indikační panel s osazenými LED diodami. Diody dávají rychlý přehled o stavu driveru. Panel je ukázaný na obrázku **2.15** a elektrické schéma se nachází v příloze **A.3.**

1) Signalizace přítomnosti napájení driveru.
2) Signalizace inicializované komunikace mezi master zařízením a driverem.
3) Signalizace připojení vstupního portu pro EtherCat komunikaci.
4) Signalizace připojení výstupního portu pro EtherCat komunikaci.
5) Signalizace rozběhnutého motoru.
6) Signalizace zastaveného motoru.

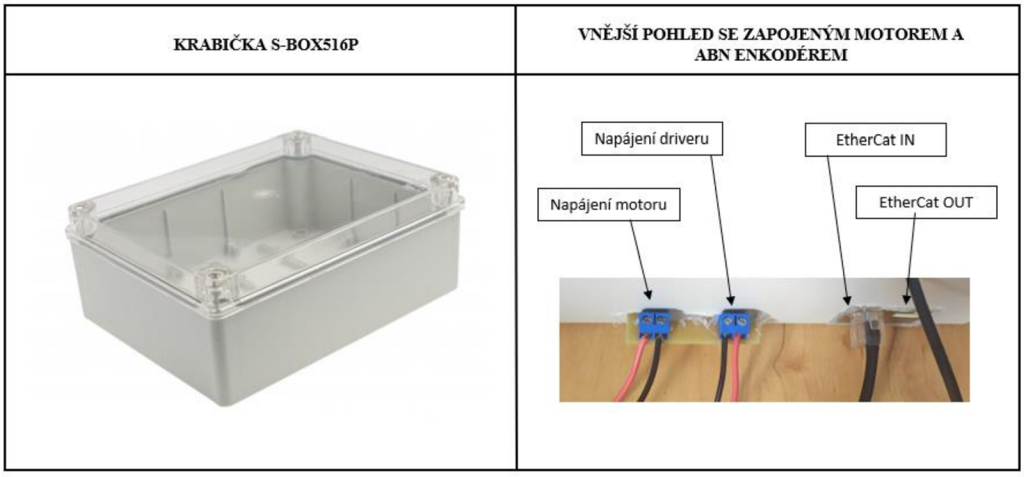


2.15 Indikační panel

2.4.8 Konstrukční uspořádání prototypu

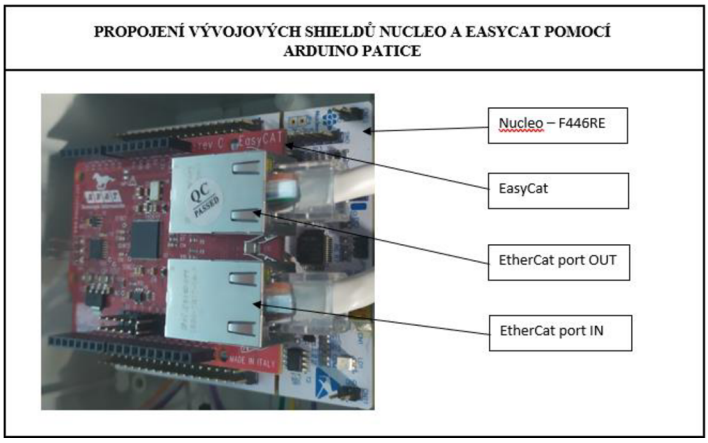
Na začátku kapitoly byl vznesen požadavek na hardwarové funkcionality zařízení jako celku. Cílem bylo vytvořit prototyp, který bude splňovat definované požadavky na funkčnost a vstupně-výstupní rozhraní, ale jak již bylo vícekrát zmíněno, tak nebude splňovat všechny náležitosti na zařízení integrované na plošném spoji.

Pro sestavení prototypu byla vybrána univerzální krabička na elektroniku S- BOX516P a jednotlivé komponenty byly upevněny pomocí distančních sloupků. Vstupní rozhraní, v podobě napájecích svorkovnic a konektorů RJ-45, je umístěno pomocí vytvořených prostupů do krabičky. Přívodní vodiče pro krokový motor a zpětnou vazbu v podobě ABN enkodéru jsou přivedeny přímo do krabičky. Použitá krabička a vstupně-výstupní rozhraní je ukázáno na obrázku **2.16**.



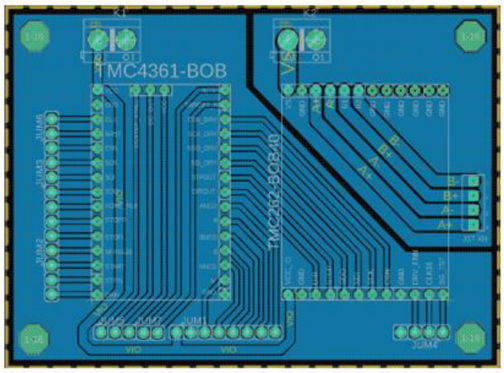
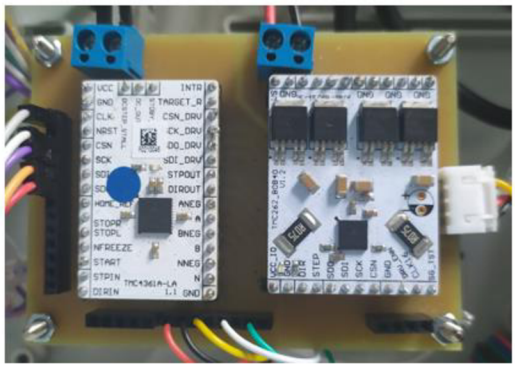
2.16 Univerzální krabička S-BOX516P a vsazené vstupy/výstupy

U vývojového shieldu Nucleo F446RE a vývojového shieldu EasyCat bylo vhodné využít jejich společnou konstrukční část, a to je dostupnost tzv. Aduino UNO patice, díky které mohou být zasazeny desky přímo do sebe, jak je ukázáno na obrázku **2.17** . V tuto chvíli již bylo potřeba vše pouze správně nakonfigurovat z pohledu softwaru.



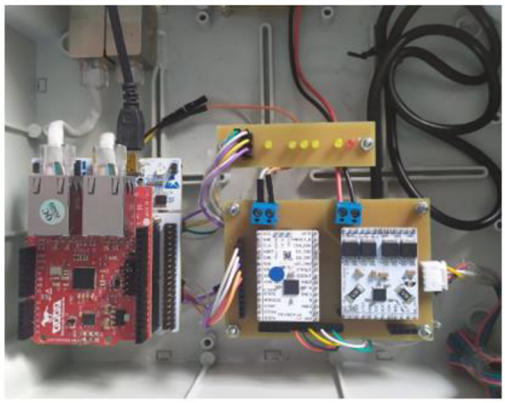
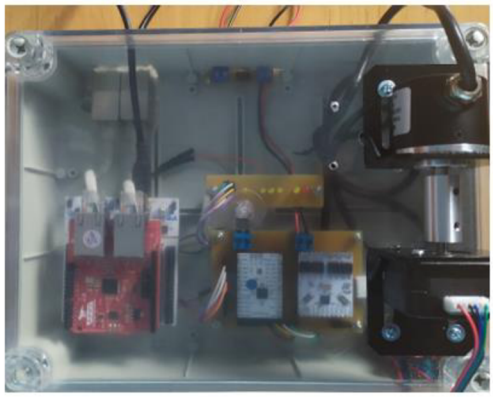
2.17 Propojení vývojových shieldů Nucleo F446RE a EasyCat

Obvody pro řízení motoru ve formě vývojových shieldů typu BOB neobsahují žádné otvory pro přímé usazení ani vhodně vyvedené piny. Z tohoto důvodu byl vytvořen jednoduchý plošný spoj, který propojuje žádané vstupy a výstupy těchto obvodů, jak je popsáno v předchozí kapitole **2.4.6.** Dále jsou vyvedeny vstupy a výstupy ve formě pin lišty, napájecí svorkovnice a konektor pro připojení motoru. Návrh desky plošného spoje a fyzickou realizaci ukazuje obrázek **2.18**. Elektrické schéma a návrh plošného spoje jsou dále uvede v přílohách **A.1** a **A.2.**

NÁVRH DPS	OSAZENÁ DPS
	

2.18 Plošný spoj pro moduly TMC4261-BOB a TMC262-BOB

Jednotlivé části v rámci vnitřku krabičky jsou propojeny univerzálními spojovacími vodiči. Na obrázku **2.19** je možné vidět kompletní vnitřek krabičky se všemi propojenými částmi. Dále je vidět krabičku z vnějšího pohledu se zapojeným napájením, vstupním EtherCat portem, krokovým motorem a ABN enkodérem. V tomto stavu byl prototyp považován, z hardwarového pohledu, za dostatečný a bylo přistoupeno k návrhu a implementaci softwarového vybavení.

VNITŘEK KRABIČKY	VNĚJŠÍ POHLED SE ZAPOJENÝM MOTOREM A ABN ENKODÉREM
	

2.19 Konstrukční uspořádání vytvořené krabičky prototypu driveru

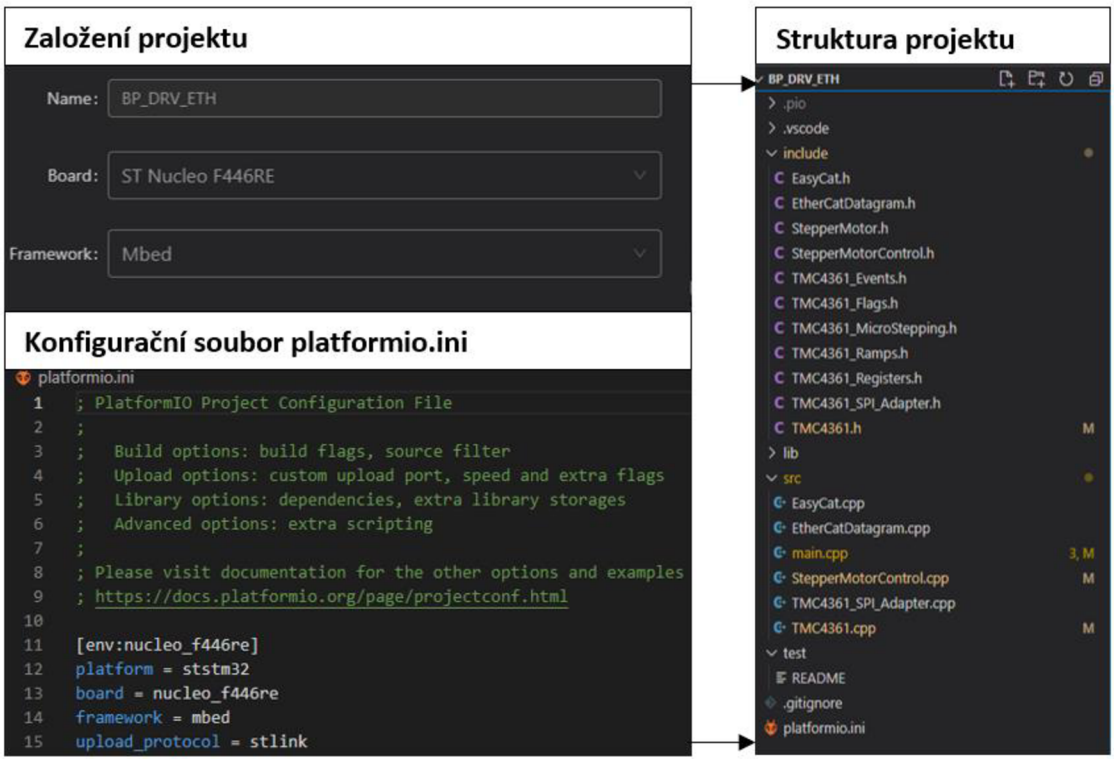
2.5 Návrh řídícího firmwaru

Tato kapitola popisuje návrh řídícího firmwaru, jenž je uložen v paměti mikrokontroléru, který tvoří část mezi ESC a obvody pro řízení motoru. Cílem je popsat hlavní rutinní smyčku a strukturu programu z pohledu propojení jednotlivých tříd či datového toku. Dále podrobněji popsat implementaci jednotlivých částí a jejich možnosti pro sestavení firmwaru jako celku.

2.5.1 Prostředí PlatformIO a framewrok Mbed

Pro vývoj softwarového vybavení bylo zvoleno prostředí VSCode s rozšířením o nástroj PlatformIO. Tento rozšiřující nástroj umožňuje snadné generování projektů, včetně nastavení všech periferií, pro vývojové desky Nucleo či další jiné typy jako jsou Arduino nebo ESP. PlatformIO dále nabízí integrovaný ladící systém, mnoho dostupných knihoven a také podporu pro snadnou kompilaci a import vytvořeného programu. [32]

Dalším důležitým faktorem, který byl zohledněn při výběru, je podpora frameworků. Pro vybranou vývojovou desku Nucleo F446RE je možnost zvolit mezi frameworky Arduino, Mbed, CMSIS, STM32Cube, Linoopencm3 nebo ZephyrRTOS. Po důkladném prostudování jednotlivých možností byl zvolen framewrok Mbed. Strukturu projektu, včetně jeho založení, je možné vidět na obrázku **2.20**. [32]

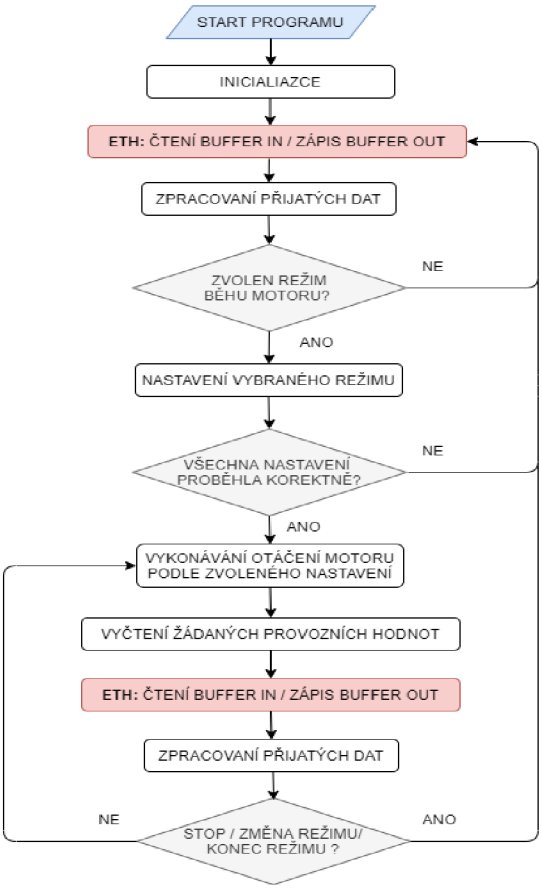


2.20 Ukázka projektu pomocí PlatformIO

2.5.2 Popis hlavní smyčky

Cílem bylo vytvořit řídící firmware, který je schopen na základě přijatých dat nastavovat zvolené parametry řízení, řídit krokový motor a v reálném čase reagovat na požadavky, jako je zasílání stavových provozních hodnot do master zařízení nebo naopak změna akčních provozních hodnot, jako jsou například změna rychlosti nebo směru motoru.

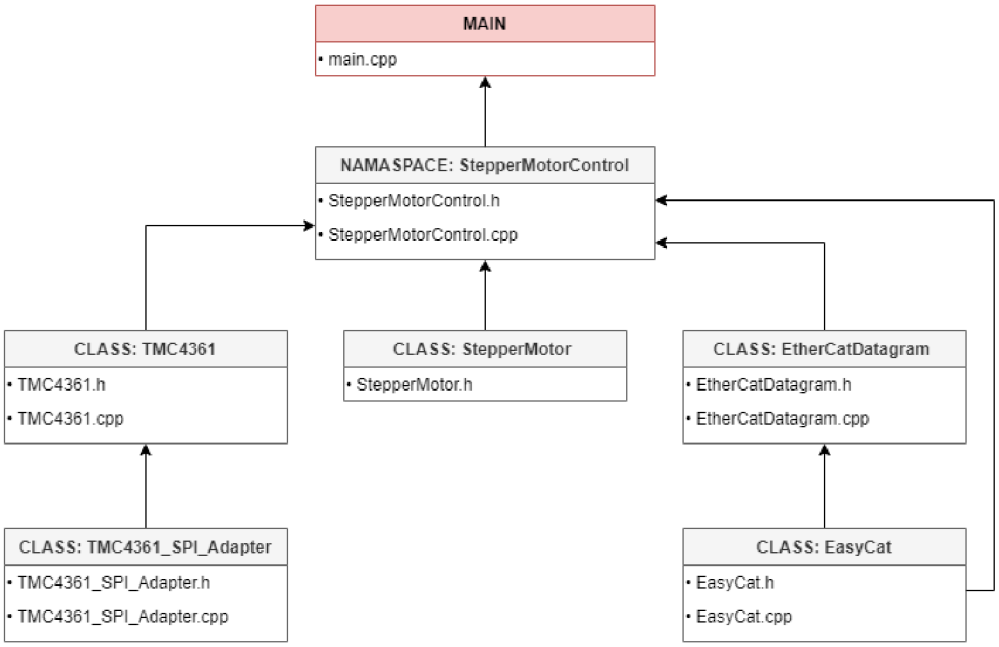
Princip hlavní smyčky je následující – po startu a inicializaci programu probíhá cyklické vyčítání dat ze vstupního bufferu ESC a zápis příslušných dat výstupního bufferu ESC. Vyčtená data jsou zpracována a pokud je zvolen režim, ve kterém má probíhat provoz motoru, tak proběhne nastavení všechno hodnot, které jsou nezbytné pro daný režim a bezpečný provoz motoru. Pokud vše proběhne v pořádku, tak je přistoupeno k vykonávání otáčení motoru. V této části je cyklicky prováděno otáčení motoru, vyčítání žádaných provozních hodnot a následné čtení vstupního bufferu ESC a zápis do výstupního bufferu ESC. Na základě vyčtených a zpracovaných dat je možné ovlivňovat parametry otáčení motoru, přijmout požadavek na zaslání nových provozních hodnot a další. Konec této části smyčkou může být způsoben tím, že režim doběhl do svého konce, přišel požadavek na stop motoru nebo je stop motoru vyvolán z bezpečnostních důvodů.



2.21 Vývojový diagram hlavní smyčky

2.5.3 UML diagram rozvržení tříd

Následující obrázek **2.22** popisuje strukturu firmwaru z pohledu jednotlivých tříd a hlavní řídící aplikace. Je snaha o jasné a logické rozčlenění jednotlivých částí do vlastních souborů a tříd, aby bylo možné k jednotlivým částem přistupovat jako k samostatnému celku, který nabízí vlastní rozhraní, ale není možné zasahovat do jeho definice.



2.22 UML diagram rozvržení tříd

Následující **Tabulka 2.5** shrnuje hlavní části firmwaru a základní popis jejich funkce.

Tabulka 2.5 Seznam implementovaných částí a jejich základní popis

P.	Typ	Název	Popis
1.	Class	EasyCat	Obsluhu EasyCat shieldu.
2.	Class	EtherCatDatagram	Práce s procesními buffery EasyCat.
3.	Class	TMC4361_SPI_Adapter	Formátování a odesílání SPI datagramu podle požadavků TMC4361.
4.	Class	TMC4361	Ovládání TMC4361 a příslušného driveru pomocí jejich registrů.
5.	Class	StepperMotor	Uchovává vlastnosti řízeného motoru a aktuální nastavené parametry řízení.
6.	Namespace	StepperMotorControl	Obsluha řízení motoru podle jednotlivých režimů.

2.5.4 Struktura z pohledu jednotlivých vrstev a datového toku

Z hlediska pohledu na firmware, jako celek, je vhodné uvést graficky znázorněný tok dat skrze jednotlivé vrstvy. Tento tok dat je reprezentován obrázkem **2.23**.

EtherCat datagram

V kapitole **2.3.5** byla popsána struktura EtherCat rámce. V rámci rámce je přidělen každému zařízení přidělovány EtherCat datagramy. Tento datagram je zpracováván na úrovni ESC, kdy jsou vyčítána, a naopak zapisována procesní data.

SPI datagram

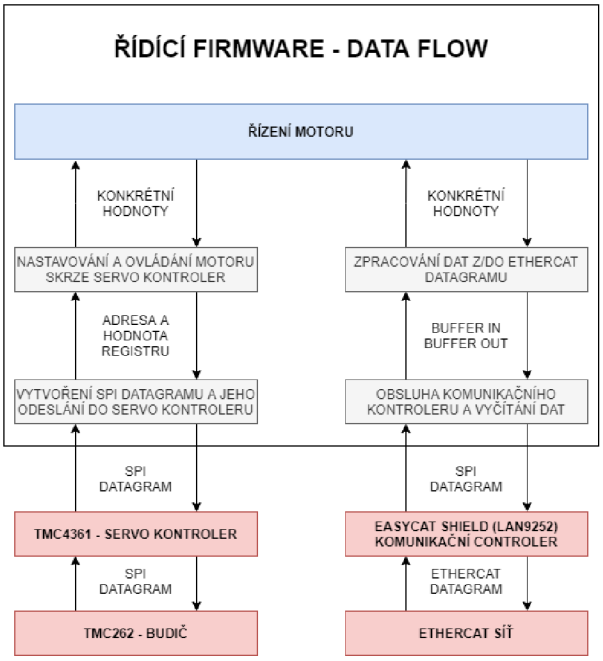
Program uložený v paměti kontroléru přijímá a zapisuje procesní data z/do ESC pomocí SPI datagramů. Stejným způsobem je komunikováno opačným směrem s obvody TMC4361 a TMC262, které slouží pro řízení pohybu motoru.

Buffer In / Buffer Out

Část programu, která se stará o obsluhu ESC, umožňuje přístup k procesním datům ve formě pole, v němž jsou uložené příslušné počty a hodnoty bytů.

Konkrétní hodnoty

Na úrovní řízení motoru jsou vyčítány parametry jako konkrétní hodnoty, například požadovaná rychlost motoru je vyčtena v otáčkách za sekundu a stejným způsobem je zapsán požadavek na změnu rychlosti. Až nižší vrstva programu se dále postará o převedení na počet pulsů za sekundu a následné naformátování SPI datagramu.

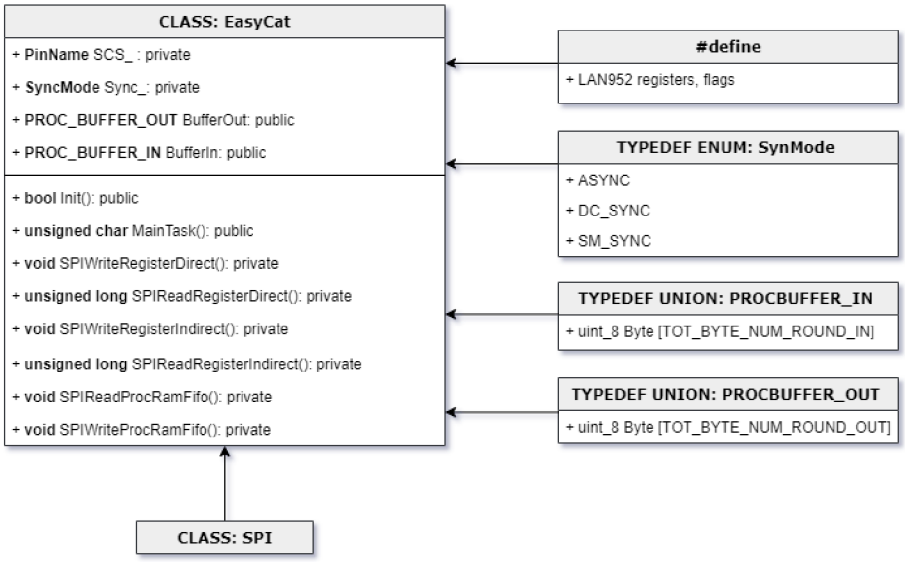


2.23 Digram jednotlivých vrstev a datového toku firmwaru

2.5.5 Class – EasyCat

V kapitole **2.4.5** byl popsán obvod LAN9252 ve vývojovém shieldu EasyCat. Dále byl uveden způsob vnější komunikace s mikrokontrolérem pomocí standartní SPI sběrnice. Firma AB&T nabízí pro obsluhu shieldu již implementovanou třídu s totožným názvem EasyCat, a ta byla využita při návrhu celkového obslužného firmwaru. Tato kapitola popisuje funkčnost a možnosti třídy, které jsou stěžejní pro navrhovaný firmware. [33]

Nabízen je režim ve standartním režimu, kde je fixně nastavená velikost pro vstupní a výstupní buffer, jenž slouží pro výměnu dat, nebo pokročilý režim, kde je možné za běhu rozsah měnit. Při dodání shieldu byl v EEPROM paměti nahrán konfigurační binární soubor, který definuje standartní režim s velikostní vstupního a výstupního bufferu procesních dat 32 bytů. . Tento režim byl při dalším návrhu zachován Jako synchronizační režim byl vybrán typ asynchroní s enum označením *ASYNC*. Obsah třídy popisuje UML diagram na obrázku **2.24**.



2.24 UML diagram pro třídu EasyCat

Inicializace, komunikace a uchovávání procesních dat

Metoda pro ověření komunikace a inicializaci EasyCat shieldu.

```
bool Init();
```

Metoda pro pravidelnou obsluhu EasyCat shieldu a čtení/zápis procesních dat. Tato metoda musí být v rámci firmwaru volána cyklicky.

```
void MainTask();
```

Struktury bufferů pro ukládání vyčtených a zapisovaných procesních dat.

```
PROCBUFFER_IN; PROCBUFFER_OUT;
```

2.5.6 Class – EtherCatDatagram

Při prvním popisu firmwaru byl vznesen požadavek na určitou strukturu a datový tok skrze jednotlivé úrovně, jak popisuje obrázek **2.23**. Cílem bylo, aby nejvrchnější část architektury firmwaru, pro řízení motoru, pracovala s hotovými částmi, které podporují dostatečnou abstrakci a díky tomu je značně zvýšena přehlednost programu. Pro tento účel byla vytvořena třída *EtherCatDatagram*, které slouží jako rozhraní mezi procesními buffery, jenž jsou součástí instance třídy *EasyCat,* a řídící třídou *StepperMotorControl*.

Při implementaci bylo bráno v potaz následující – procesní buffery mají pevnou neměnnou velikost a při návrhu je jim určen přesně definovaný formát pro jednotlivé režimy řízení, které jsou popsány až v pozdější kapitole **2.5.10**. V rámci struktury třídy byly vytvořeny privátní metody pro vyčítání různých datových typů. Přehled těchto typů uvádí **Tabulka 2.6**.

Tabulka 2.6 Přehled datových typů

P.	Datový typ pročtení/zápis	Popis
1.	bool	Bit
2.	uint8_t	Konkrétní byte
3.	uint16_t	Dva byty v pořadí
4.	uint32_t	Čtyři byty v pořadí

Příklad metody zápisu hodnoty bitu – v parametru je předána informace o konkrétním bytu a bitu, a na tuto pozici je zapsána cílová bitová hodnota.

```
void SetBitInByte(const BufferByte &aByte, const ByteBit &aBit, const
                  bool &aValue);
```

Příklad metody čtení hodnoty o velikost čtyř bytů – v parametru je předána informace o prvním bytu a následně je přečtena a vrácena celková hodnota typu uint32_t.

```
uint32_t ReadLongValue(const BufferByte &aStartByte);
```

Veřejné metody využívají ve své definici privátní metody a jsou pojmenovány podle hodnoty nebo instrukce, kterou mají číst, respektive zapsat. Všechny metody pracují na stejném principu a vzhledem k velkému množství není uveden UML diagram třídy.

Příklad metody vyčtení pokynu ke startu pohybu motor – metoda vrací hodnotu bitu z předem daného místa vstupního bufferu a tento bit představuje pokyn ke startu.

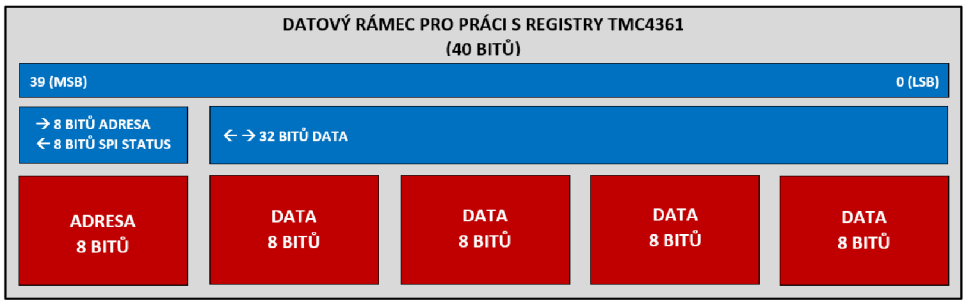
```
bool ReadStartInstruction();
```

Příklad metody pro zápis aktuální rychlosti motoru – v parametru je předána aktuální vyčtená rychlost a ta je zapsaná na předem dané místo ve výstupním bufferu.

```
void SendActualVelocity(const RunMode &aRunMode, const uint8_t
                        &aVelocity);
```

2.5.7 Class– TMC4361_SPI_Adapter

Z blokového schématu, jenž popisuje základní architekturu navrhovaného driveru, je patrné, že komunikace mezi jednotlivými bloky probíhá pouze za pomocí standartní SPI sběrnice. Třída *TMC4361_SPI_Adapter* zprostředkovává komunikaci mezi řídícím mikrokontrolérem a servo-kontrolérem TMC4361. Řízení či vyčítání provozních dat probíhá výhradně pomocí zápisu nebo naopak čtení registrů servo-kontroléru. Komunikace musí být přizpůsobena vyžadovanému formátu datového rámce. [27]



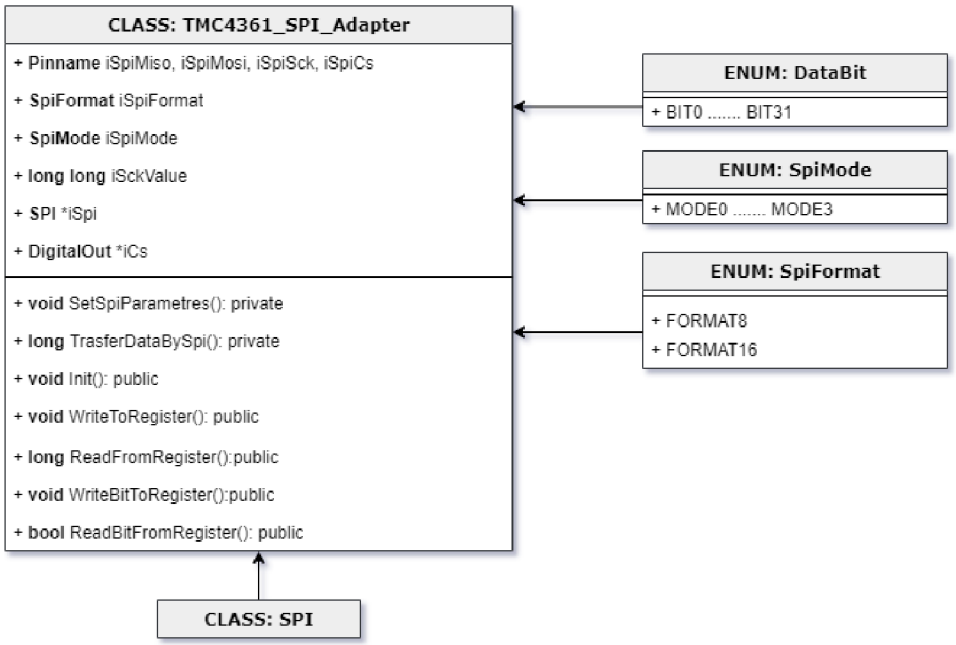
2.25 Datový rámec TMC4361

V oficiálním technickém listu servo-kontroleru TMC4361 [27] je uvedena struktura datového rámce, jenž je graficky vyobrazena na **2.25**. Obecně se SPI sběrnice využívá nejčastěji v osmibitovém režimu. V tomto případě se místo zápisu nebo čtení určí na začátku osmibitovou adresou a následně jsou ve čtyřech dalších cyklech sběrnice odeslány další čtyři bloky dat. Ze základního principu přenosu dat po SPI je známo, že při každém odeslaném datovém bloku z řídícího master zařízení, je také zpětně přijat datový blok o stejné velikosti. Této skutečnosti je využito pro udržování provozních informací samotné SPI sběrnice, a především pro efektivní čtení dat. Další úskalí pro úspěšnou komunikaci jsou popsána v technickém listu a byl jimi řízen celý návrh.

1. **Zápis do registru** – Při odeslání adresy je nejvyšší bit nastaven jako 1.

2. **Čtení registru** – Při odeslání adresy je nejvyšší bit nastaven jako 0. Na obsahu následujících dat nezáleží, ale musí být odeslány.

Před implementací samotné třídy bylo důležité definovat požadavky, které by měla třída splňovat pro její další využití ve vyšších vrstvách celé aplikace. SPI komunikace jako taková, byla implementována již nesčetněkrát, tudíž bylo využito třídy SPI, která je obsažena v knihovně v rámci *mbed.h*. Úkolem této třídy je komunikaci přizpůsobit zmíněnému datovém rámci a umožnit zápis či čtení dat registrů na konkrétních adresách. Mnoho provozních nastavení probíhá pouze pomocí nastavení jednoho bitu registru, takže bylo vhodné zařadit i funkcionalitu zápisu či čtení konkrétních bitu registru.



2.26 UML diagram pro třídu TMC4361_SPI_Adapter

Atributy třídy

Z důvodu jisté univerzálnosti třída obsahuje kromě metod také atributy, které umožňují při vytváření instance definovat a udržovat informaci o použitých pinech sběrnice, nastavení formátu, režimu přenosu, nebo také rychlosti přenosu. Výčet všech atributů je uveden v UML diagramu na **2.26**.

Vytvoření a inicializace

Pro vytváření instancí *TMC4361_SPI_Adapter* jsou definovány tři typy konstruktorů, jenž umožňují vnější inicializaci výše zmíněných privátních atributů. Dále je možné nechat parametry nastavit implicitně dle defaultních hodnot.

Instance vytvořena s defaultními hodnotami.

```
TMC4361_SPI_Adapter();
```

Instance vytvořena s volbou SPI rozhraní.

```
TMC4361_SPI_Adapter(const PinName &aSpiMosi, const PinName &aSpiMiso,
                    const PinName &aSpiSck, const PinName &aSpiCs);
```

Instance vytvořena s volbou SPI rozhraní a parametry přenosu.

```
TMC4361_SPI_Adapter(const PinName &aSpiMosi, const PinName &aSpiMiso,
                    const PinName &aSpiSck, const PinName &aSpiCs,
                    const long &aSpiSckValue, const SpiFormat
                    &aSpiFormat, const SpiMode &aSpiMode);
```

Nastavení a přenos dat

Metoda pro nastavení SPI sběrnice dle předaných dat v parametru konstruktoru, nebo základních defaultních hodnot.

```
void SetSpiParametres();
```

Metoda pro přenos dat po, konstruktorem nastavené, SPI sběrnici.

```
long TransferDataBySPI(const uint8_t &aAdress, const long &aData = 0);
```

Přímá práce s registry

Metoda pro zjištění platné komunikace mezi řídícím mikrokontrolérem a servo kontrolérem TMC4361.

```
bool Init();
```

Metoda pro zápis dat do registru. Parametrem musí být předána adresa registru a data pro zápis.

```
void WriteToRegister(const uint8_t &aAdress, const long &aData);
```

Metoda pro čtení aktuálně zapsaných dat na adresa předané parametrem. Parametrem musí být předána adresa a čtená data jsou předána ve formě návratové hodnoty.

```
long ReadFromRegister(const uint8_t &aAdress);
```

Metoda pro zápis hodnoty bitu na konkrétní datové místo registru. Parametrem musí být předána adresa registru, zvolený datový bit a hodnota pro nastavení.

```
void WriteBitToRegister(const uint8_t &aAdress, const SPIDataBit
                        &aBitNumber, const bool &aSetValue);
```

Metoda pro čtení hodnoty bitu na konkrétním datovém místu registru. Parametrem musí být předána adresa registru a zvolený datový bit.

```
bool ReadBitFromRegister(const uint8_t &aAdress, const SPIDataBit
                         &aBitNumber);
```

2.5.8 Class – TMC4361

V předchozí kapitole **2.5.8** bylo popsáno softwarové řešení komunikace mezi řídícím mikrokontrolérem a servo-kontrolérem TMC4361. Pro řízení motoru byla vytvořena stejnojmenná třída *TMC4361*. Nastavování parametrů řízení, zasílání řídících pokynů nebo vyčítání provozních hodnot probíhá zmíněným zápisem nebo čtením registrů. Nutné je podotknout, že obvod TMC4361 [27] obsahuje velké množství registrů s velmi rozmanitými možnostmi nastavení a použití pro specifické aplikace. Nejprve bylo tedy důležité ujasnit jaké možnosti nastavení a řízení bude potřeba implementovat, aby byly splněny prvotní stanovené požadavky.

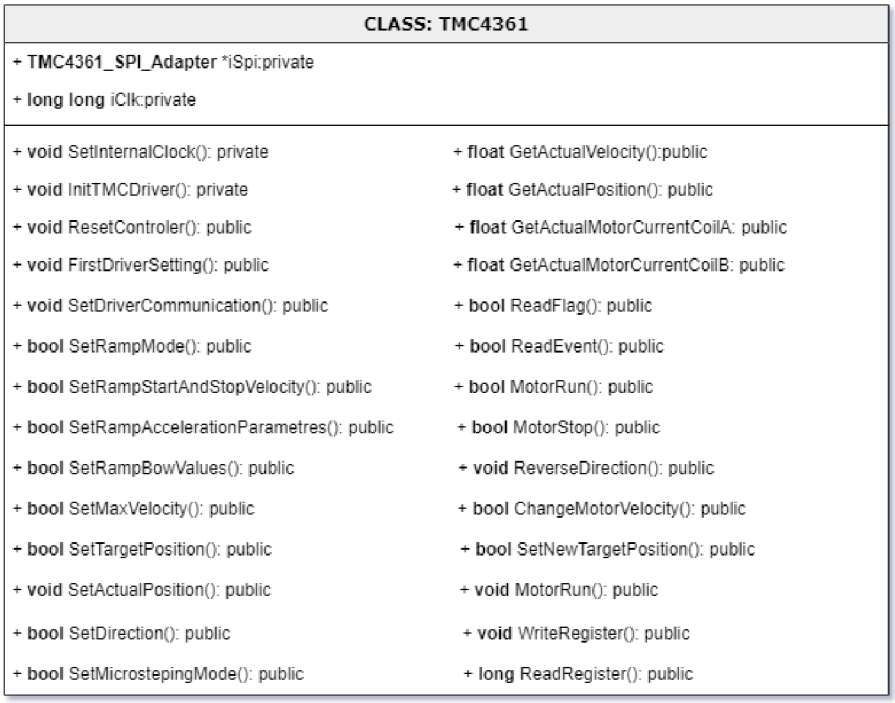
Požadavky na funkcionalitu třídy

1) Nastavení vnějšího SPI rozhraní pro konfiguraci koncového budiče TMC262 a nastavení vnějšího S/D rozhraní pro generování průběhu rychlosti a směru motoru.
2) Prvotní nastavení koncového budiče TMC262 pomocí jeho konfiguračních registrů.
3) Nastavení režimu rychlostního profilu a možnost jeho parametrizace.
4) Nastavení režimu mikrokrokování.
5) Nastavení požadované prvotní rychlosti a směru otáčení.
6) Možnost volby nastavení řízení v otevřené nebo uzavřené smyčce.
7) Možnost konfigurace připojeného ABN enkodéru, jenž bude sloužit pro zpětnou vazbu při řízení v uzavřené smyčce.
8) Možnost volby řízení na rychlost nebo na polohu.
9) Možnost vyčítání provozních hodnot jako je aktuální rychlost nebo poloha.
10) Nastavení pokynů pro start, stop nebo reverzaci motoru.
11) Možnost měnit parametry řízení jako je rychlost nebo směr za běhu motoru.
12) Možnost přímého zápisu nebo čtení libovolných registrů.

Pro přehled jsou uvedeny pomocí **Tabulka 2.7** hlavní stěžejní použité registry. Ve skutečné implementaci jsou využity i další, které se v uvedené tabulce nevyskytují.

Tabulka 2.7 Hlavní využité registry TMC4361 [27]

P.	Registr	Adresa	P.	Registr	Adresa
1	GENERAL_CONF	0x00	15	VSTOP	0x26
2	SPI_OUT_CONF	0x04	16	VBREAK	0x27
3	CURRENT_CONF	0x05	17	AMAX	0x28
4	SCALE_VALUES	0x06	18	DMAX	0x29
5	ENC_IN_CONF	0x07	19	ASTART	0x2A
6	STEP_CONF	0x0A	20	DFINAL	0x2B
7	EVENTS	0x0E	21	BOW1,2,3,4	0x2D..30
8	STATUS	0x0F	22	VIRT_STOP_LEFT	0x33
9	STP_LENGTH_ADD	0x10	23	VIRT_STOP_RIGHT	0x34
10	RAMP_MODE	0x20	24	XRANGE	0x36
11	XACTUAL	0x21	25	XTARGET	0x37
12	VACTUAL	0x22	26	ENC_CONST	0x54
13	VMAX	0x24	27	ENC_POS	0x52
14	VSTART	0x25	28		



2.27 UML diagram pro třídu TMC4361

Atributy třídy

Z hlediska dat třída uchovává ukazatel na instanci *TMC4361_SPI_Adapter,* pomocí které je realizován přenos dat do/z registrů, a dále hodnotu vnější taktovací frekvence. Hodnota frekvence slouží uvnitř třídy například pro výpočty mezních parametrů jako je maximální generovatelná rychlost nebo další. Prozatím je možné pracovat pouze s defaultní hodnotou 16 MHz, je zde však ponechán prostor pro budoucí rozšíření.

Vytvoření a inicializace

Při vytvoření instance je provedeno veškeré nastavení, aby byl obvod připravený pro obsluhu. Postupně je nastavena na výstupu mikrokontroléru taktovací frekvence pro TMC4361, vytvořena instance *TMC4361_SPI_Adapter,* pro možnost komunikace a následně inicializován příslušný koncový budič.

```
TMC4361::TMC4361()
{
    iClkInt = TMC4361_INTERNAL_CLK; //Uložení nastavené frekvence
    SetInternalClock();

    iSpi = new TMC4361_SPI_Adapter(parametry…); //Komunikace pomocí SPI
    ResetControler(); //Reset kontroleru do prvnotního nastavení
    InitTMCDriver(aDriver); //Inicializace koncového budiče
}
```

Nastavení komunikace a koncového budiče

Aby byly obvody pro řízení motoru připraveny k okamžitému použití, je třeba nastavit jejich součinnost a prvotní nastavení již při vytváření instance. V rámci konstruktoru je volána následující metoda *InitTMCDriver,* která nastaví komunikaci a nakofiguruje koncový TMC budič.

```
void InitTMCDriver(const TMCMotorDriver &aDriver);
```

Komunikace s koncovým budičem probíhá pomocí vnějšího SPI rozhraní, které je součástí TMC4361. Režim a parametry komunikace jsou nastaveny pomocí registru *SPI_OUT_CONF*. Řízení pohybu je realizováno pomocí výstupního S/D rozhraní, jemuž bylo potřeba nastavit časové parametry pomocí registru *STP_OUT_LENGTH*.

```
void TMC4361::SetDriverCommunication()
{
    iSpi->WriteToRegister(STP_LENGTH_ADD, 0x00060004);
    iSpi->WriteToRegister(SPI_OUT_CONF, 0x848002EB);
};
```

Koncový budič TMC262 obsahuje celkem pět konfiguračních registrů. Zápis do těchto registru probíhá pomocí krycích datagramů skrze registr *COVER_LOW*. Datagramy mají délku 20 bitů a obsahují adresu registru a přenášená data.

```
void TMC4361::FirstDriverSetting()
{
    iSpi->WriteToRegister(COVER_LOW, 0x090131); // REGISTR: DRVCTRL
    iSpi->WriteToRegister(COVER_LOW, 0x0D0F0F); // REGISTR: SGCSCONF
    iSpi->WriteToRegister(COVER_LOW, 0x0EF040); // REGISTR: DRVCONF
    iSpi->WriteToRegister(COVER_LOW, 0x0); // REGISTR: CHOPCONF
};
```

Nastavení parametrů řízení

Před zahájením pohybu motoru musí být vždy provedeno korektní nastavení žádaných parametrů. Následující metody slouží pro nastavení všech parametrů, které jsou důležité pro splnění definovaných požadavků. Všechny tyto metody mají návratovou hodnotu typu bool. Ta symbolizuje správně provedené nastavení. Obecně je princip všech metod stejný – na začátku proběhne kontrola správného rozsahu hodnot předaných parametrem, poté je proveden zápis do příslušných registrů a jako poslední proběhne kontrola čtením registrů pro ověření správně zapsané hodnoty.

Metoda pro nastavení režimu rampy pomocí registru *RAMPMODE*. Režimy jsou rozděleny odlišeny principem řízení a rozběhových profilem rychlosti. Ve zdrojovém kódu je pro přehlednost všech dostupných režimů vytvořena enum definice *TMC4361_RampMode*.

```
bool SetRampMode(StepperMotor *aMotor, const TMC4361_RampMode
                 &aRampMode);
```

Metoda pro nastavení parametrů rampy. Pomocí registrů *VSTART* a *VSTOP* je možné nastavit úroveň počáteční a koncové rychlosti, respektive takové rychlosti, při které je rychlost nastavena s obdélníkovým profilem.

```
bool SetRampStartAndStopVelocity(StepperMotor *aMotor, const uint32_t
                        &aVelocityStart, const uint32_t &aVelocityStop)
```

Metoda pro nastavení parametrů rampy. Pomocí registrů *AMAX, DMAX, ASTART, AFINAL* a *ABREAK* jsou nastaveny všechny parametry zrychlení.

```
bool SetRampAccelerationParametres(StepperMotor *aMotor, const uint32_t
&aMaxAcceleration, const uint32_t &aMaxDeccelerationn, const uint32_t
&aStartAcceleration, const uint32_t &aFinalDeccelarition, const uint32_t
&aVelocityBreak);
```

Metoda pro nastavení parametrů rampy. Pomocí registrů *BOW1, BOW2, BOW3* a *BOW4* jsou nastaveny parametry pro vymodelování tvaru rampy typu s-křivka.

```
bool SetRampBowValues(StepperMotor *aMotor, const uint32_t &aBow1, const
uint32_t &aBow2, const uint32_t &aBow3, const uint32_t &aBow4);
```

Metoda pro nastavení cílové rychlosti pomocí registru *VMAX*. Rychlost je nastavována v otáčkách za sekundu.

```
bool SetMaxVelocity(StepperMotor *aMotor, const long &aVelocity);
```

Metoda pro nastavení cílové pozice pomocí registru *XTARGET*. Možné využití při řízení na polohu, kde je poloha nastavována v milimetrech.

```
bool SetTargetPosition(StepperMotor *aMotor, const long &aPositon);
```

Metoda pro nastavení nové referenční pozice pomocí registru *XACTUAL*. Při použití této metody je aktuální poloha nastavená na hodnotu předanou parametrem.

```
bool SetActualPositon(StepperMotor *aMotor, const long &aPositon);
```

Metoda pro nastavení počátečního směru otáčení pomocí registru *GENERAL_CONF*, kde je pro určení směru využit bit 28.

```
bool SetDirection(StepperMotor *aMotor, const MotorDirection
                  &aDirection);
```

Metoda pro nastavení režimu mikrokrokování. Mikrokrokování je možné nastavit od režimu plného kroku až po režim 1/256.

```
bool SetMicrosteppingMode(StepperMotor *aMotor, uint8_t
                          aMicroSteppingMode);
```

Čtení provozních hodnot

Pro vyhodnocování stavu motoru bylo třeba implementovat metody, které umožní vyčítat provozní hodnoty za běhu motoru. Jako velmi užitečné lze považovat přítomnost stavového registru *STATUS* a událostního registru *EVENTS*, pomocí kterých je možné jednoduše určit chování motoru v průběhu řízení.

Metoda pro vyčítání aktuální rychlosti pomocí registru *VACTUAL*. Rychlost je opět vyčítána v otáčkách za sekundu.

```
float GetActualSpeed(StepperMotor *aMotor);
```

Metoda vyčtení aktuální pozice pomocí registru *XACTUAL*. Hodnota pozice je vztažená k nastavené referenční pozici.

```
float GetActualPosition(StepperMotor *aMotor);
```

Metody pro vyčtení aktuálního dodávaného proudu cívkami motoru.

```
float GetActualMotorCurrentCoilA(StepperMotor *aMotor);
float GetActualMotorCurrentCoilB(StepperMotor *aMotor);
```

Metoda vyčtení aktuální hodnoty zvoleného příznaku obsaženého v registru *STATUS*. Pro přehlednost zdrojového kódu byla vytvořena enum definice TMC4361_Flags.

```
bool ReadFlag(const TMC4361_Flags &aFlag);
```

Metoda vyčtení aktuální hodnoty zvolené události obsažené v registru *EVENTS*. Pro přehlednost zdrojového kódu byla vytvořena enum definice TMC4361_Events.

```
bool ReadEvent(const TMC4361_Events &aEvent);
```

Ovládání pohybu

Pro ovládání pohybu motoru byly implementovány následující metody. Respektovány byly požadavky na možnosti startu, zastavení a změnu parametrů za běhu motoru.

Metoda pro zahájení nastaveného režimu pohybu. Pokud je řízení na rychlost, tak metoda pro start využívá zápis nové maximální rychlosti, pokud je řízení na polohu, tak využívá zápis nové cílové pozice.

```
bool MotorRun(StepperMotor *aMotor, const bool &aTypeStart, const long
              aVelOrPos );
```

Metoda pro zastavení pohybu motoru. Zastavení je provedeno pomocí nastavení maximální rychlosti na hodnotu nula. Uvnitř metody je čekáno až bude motor skutečně na nulové rychlosti.

```
bool MotorStop(StepperMotor *aMotor);
```

Metoda pro otočení směru pohybu motoru. Rychlost je nastavena na nulu a po dosáhnutí nulové rychlosti je provedena je změna směru a opětovně nastavena cílová rychlost.

```
void ReverseDirection(StepperMotor *aMotor);
```

Metoda pro změnu rychlosti motoru při běhu motoru. Změna probíhá pomocí zápisu nové maximální rychlosti.

```
bool ChangeMotorVelocity(StepperMotor *aMotor, const long &aVelocity);
```

Metoda pro změnu pozice. Změna probíhá pomocí zápisu nové cílové pozice.

```
bool SetNewTargetPosition(StepperMotor *aMotor, const long &aPositon);
```

Řízení v uzavřené smyčce

Všechny dosud popsané metody přepokládali řízení v otevřené smyčce. Pro práci v uzavřené smyčce je důležité provést nastavení zpětnovazebního ABN enkodéru a nastavit parametry řízení. Pro tento účel byly naimplementovány následující metody a při jejich návrhu bylo vycházeno z příkladů v příručce odkázané na zdroji [39].

Metoda pro nastavení zpětnovazebního ABN enkodéru. Hlavním parametrem pro nastavení je rozlišení enkodéru.

```
bool SetABNEncoder(const uint16_t &aResolution);
```

Metoda pro nastavení parametrů řízení v uzavřené smyčce. Metoda připraví řízení na uzavřenou smyčku a nastaví potřebné parametry.

```
bool SetCloosedLoopControl(const uint32_t &aMaxDeviation);
```

Přímá práce s registry

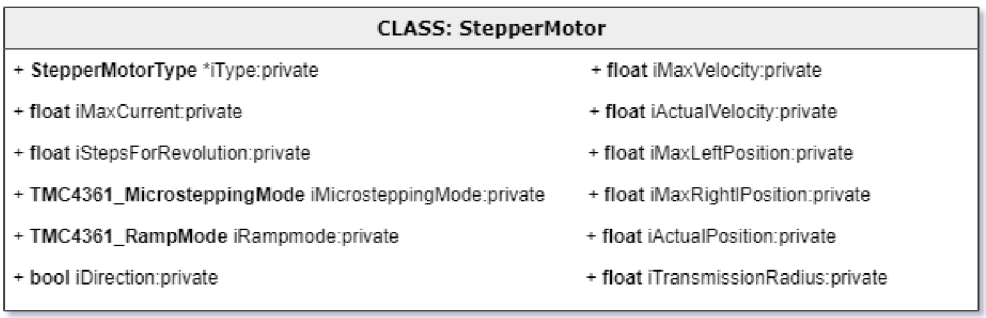
Pro přímou práci s registry byly implementovány metody, které v podstatě kopírují výstupní rozhraní *TMC4361_SPI_Adapter*, a pouze jsou zapouzdřeny v nové definici.

```
void   WriteRegisterBit(const   uint8_t   &aAdress,   const   SPIDataBit
                        &aBitNumber, const bool &aSetValue);

bool   ReadRegisterBit(const   uint8_t   &aAdress,   const   SPIDataBit
                        &aBitNumber);
```

2.5.9 Class - StepperMotor

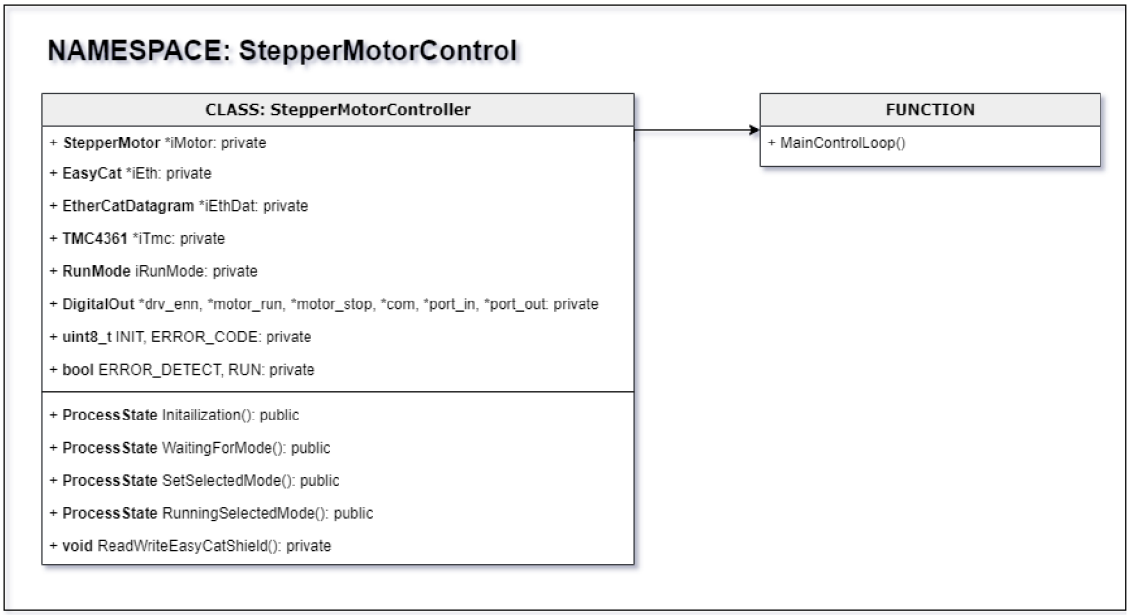
V rámci struktury zdrojového kódu třída představuje model krokového motoru. Úloha třídy je pouze k uchovávání dat ohledně parametrů motoru a parametrů řízení. Tyto parametry jsou ukládány z důvodu univerzálnosti výpočtů a rychlé dostupnosti aktuálního nastavení řízení. Jelikož je třída pouze datového charakteru a obsahuje pouze metody typu SET/GET, tak je její implementace provedena pouze v rámci .h souboru. Přehled ukládaných parametrů ukazuje UML diagram na obrázku **2.28**.



2.28 UML diagram pro třídu StepperMotor

2.5.10 Namespace – StepperMotorControl

Jmenný prostor *StepperMotorControl* zapouzdřuje vše potřebné k sestavení hlavní řídící smyčky popsané obrázkem **2.21**. Struktura je popsána UML diagramem na obrázku **2.29**.

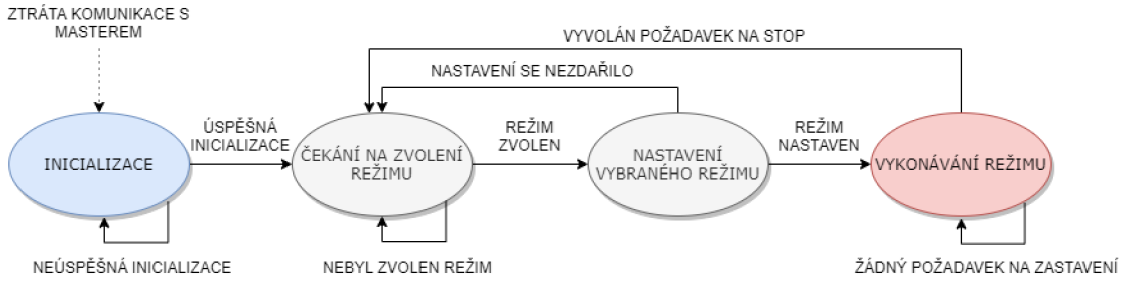


2.29 UML diagram pro jmenný prostor StepperMotorControl

Hlavní funkce MainControlLoop

Tato funkce je považována za hlavní řídící smyčku a je v ní implementován algoritmus, popsaný vývojový diagramem na obrázku **2.21**. Cílem bylo udržovat tuto hlavní část zdrojového kódu, co nejjednodušší a nejpřehlednější. Pro implementaci byl zvolen standartní návrhový vzor typu stavový automat. Ten využívá celkem čtyři stavy a pro jejich vykonávání volá metody třídy *StepperMotorController*. Stavy a přechody jsou rozlišeny pomocí enum definice *ProcessState*. Pro přehlednost byl vytvořený názorný přechodový diagram stavového automatu vyobrazený na obrázku **2.30**.

```
void MainControlLoop();
```



2.30 Přechodový diagram hlavního stavového automatu

Inicializace

Stav inicializace má několik po sobě jdoucích prioritních kroků. Prvním krokem je úspěšné nastavení komunikace s EasyCat shieldem a jeho uvedení do prvotního stavu. Pokud proběhne tato část úspěšně, je přepokládáno, že je zařízení připraveno ke komunikaci v rámci topologie sítě. Dalším krokem je ověření správné výměny dat s master zařízením, kde dojde k požadavku ověření pomocí kontrolního bitu a následným zápisem ověřovacího bitu. Posledním krokem je žádost o povolení koncového budiče motoru, který je povolován pomocí digitálního výstupu. Po žádosti je provedeno povolení a ověření, zda je na výstupu fyzicky připojen krokový motor. Pokud ano, je odeslán potvrzovací bit a následně je proveden přechod do následujícího stavu. Do stavu inicializace se driver dostane ve chvíli, kdy je během procesu ztracena komunikace s master zařízením. Pokud tato situace nastane, je třeba provést inicializaci znovu.

```
ProcessState Initialization();
```

Čekání na volbu režimu

Stav pro čekání na zvolený režim provádí cyklické vyčítání procesních dat a ve chvíli, kdy je na vstupním bytu 30 hodnota, která odpovídá platnému režimu, tak je tento režim nastaven jako aktuální a následuje přechod do dalšího stavu. Dostupné režimy jsou ukázány v **Tabulce 2.8**. Do tohoto stavu je zpětně přecházeno, když je režim ukončen nebo nastane v dalších částech chyba. V případě výskytu chyby musí být tento stav vyresetován pomocí master zařízení, které tím dává najevo, že chybu zpracovalo.

```
ProcessState RunningSelectedMode();
```

Tabulka 2.8 Přehled řídících režimů

P.	Režim	Byte 30	Popis
1.	NO_MODE	0	Nebyl zvolen žádný režim
2.	MODE_1	1	Základní režim řízení s možností volby rychlosti, směru a režimu mikrokrokování.
3.	MODE_2	2	Pokročilý režim řízení s možností volby profilu rychlost a jeho parametrů.
4.	MODE_3	3	Zpětnovazební řízení s možností volby řízení na rychlost nebo na polohu.
5.	MODE_4	4	Režim přímého zápisu/čtení registrů. Master posílá adresy a hodnoty registrů.

Nastavení vybraného režimu

Každý z režimu vyžaduje odlišné nastavení, tedy i odlišný formát přijatých procesních dat, které obsahují hodnoty žádaných parametrů. Při vstupu do tohoto stavu je podle nastaveného režimu vybrán příslušný blok nastavení, následně proběhne postupné nastavení a pokud není detekována žádná chyba, může nastat přechod do dalšího stavu. Formát vstupního a výstupního bufferu je uveden v **Příloze D.**

```
ProcessState SetSelectedMode();
```

Vykonávání režimu

Po úspěšném nastavení následuje stav vykonávání režimu. Po přechodu do tohoto stavu je čekáno na příchozí instrukci pro start. Po úspěšně provedeném startu jsou cyklicky vyčítána nově příchozí data, a naopak zapisována nově vyčtená žádaná data. Na základě přijatých dat může být za běhu prováděna změna směru nebo rychlosti, popřípadě polohy. Zároveň je hlídána přítomnost instrukce stop, při jejímž příchodu je provedeno zastavení motoru. Po požadavku na zastavení je počkáno až rychlost opravdu klesne na nulu a následuje přechod do stavu pro čekání na volbu režimu.

```
ProcessState RunningSelectedMode();
```

Následující úsek zdrojového kódu ukazuje jednoduchou implementaci popsaného stavového automatu. Mezi stavy se přechází pomocí proměnné *ActualState* a její hodnota je v každém cyklu určována návratovou hodnotou metody aktuálního stavu.

```
while (PROGRAM){
        Millis = t1.read_ms();
        if (Millis - PreviousCycle >= 1)
        {
            switch (ActualState)
            {

            case INITIALIZATION:
                ActualState = iController.Initialization();
                break;

            case WAIT_FOR_MODE_SELECTION:
                ActualState = iController.WaitingForMode();
                break;

            case SETTING_SELECTED_MODE:
                ActualState = iController.SetSelectedMode();
                break;

            case RUNNING:
                ActualState = iController.RunningSelectedMode();
                break;
            }
            PreviousCycle = t1.read_ms();
        }
    }
```

2.6 Ovládací EtherCat master

Pro otestování funkčnosti zrealizovaného driveru bylo třeba k dispozici řídícího EtherCat masteru. V kapitole **2.3.3** je zmíněn fakt, že pro realizaci EtherCat master zařízení není třeba žádný speciální hardware. Postačí tedy obyčejný PC se síťovou kartou a vhodným softwarovým vybavením. V této kapitole jsou uvedeny možnosti dostupného softwaru a následně pomocí nejvhodnější možnosti popis implementace jednoduché řídící aplikace.

2.6.1 Dostupný software a jeho možnosti

V následujících odstavcích jsou uvedeny tři možné softwary pro realizaci master zařízení. Nejedná se však o detailní popis, pouze o krátký přehled dostupných možností.

TwinCat

TwinCat je oficiální software od společnosti Bechoff, která je také autorem EtherCat standartu. Obsahuje vše, co je potřeba pro vytvoření EtherCat sítě, vytváření EtherCat rámců nebo diagnostiku sítě. Tento software je placený a v této práci nebyl využit, nicméně je vhodné jej zmínit. [34]

Simple open EtherCat master

SOEM (Simple open EtherCat master) je volně dostupná knihovna napsaná v jazyce C pro vytváření EtherCat master zařízení. Knihovna je dostupná pro operační systém Linux nebo Windows. Umožněn je základní přenos procesních dat, ale i pokročilá nastavení nebo diagnostika sítě. Tato knihovna byla využita při testování různých variant ESC v prvopočátku návrhu, kdy bylo třeba ověřit funkčnost výměny dat. [35]

Codesys

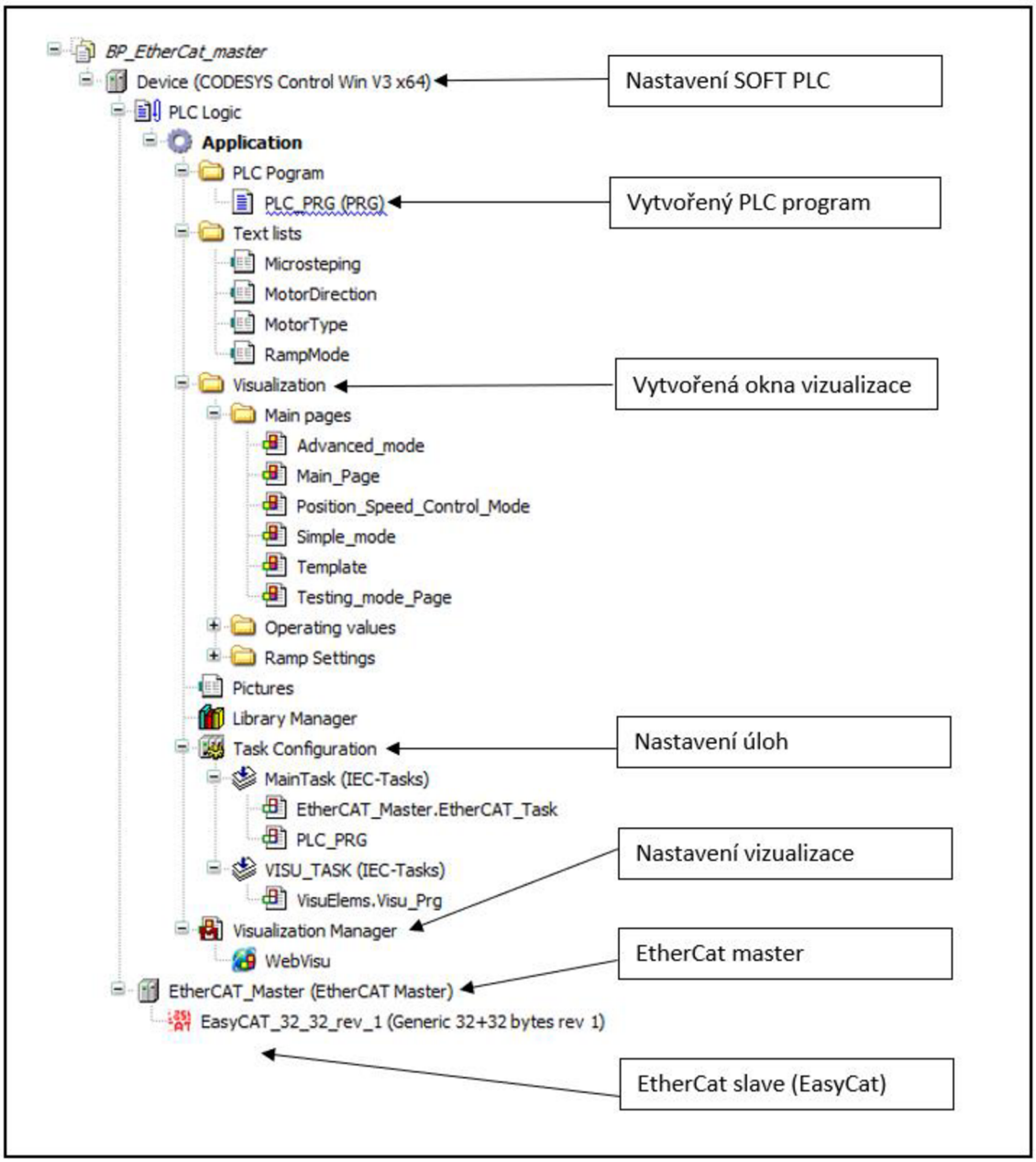
Vývojové prostředí Codesys je známo především jako prostředí pro programování PLC automatů. Dostupná je však velká podpora různých nadstandartních funkcí a knihoven. Podporována je celá škála průmyslových sítí, včetně rozšířených profilů. V dalším průběhu práce bylo využito právě této varianty, jako varianty primární. [36]

2.6.2 Řídící master aplikace pomocí prostředí Codesys

Z popsaných možností bylo pro účel testování vybráno prostředí Codesys. Důvodem je jednoduchá integrace EtherCat master zařízení pomocí zmíněných knihoven, následné propojení procesních bufferů s PLC programem a také možnost využití integrovaného soft PLC. Cílem bylo vytvořit takové ovládací rozhraní, pomocí kterého je možné inicializovat podřízený driver, nastavit parametry řízení motoru, odesílat řídící příkazy, nebo naopak přijímat žádané provozní veličiny či zprávy ohledně chyb.

Struktura Codesys projektu

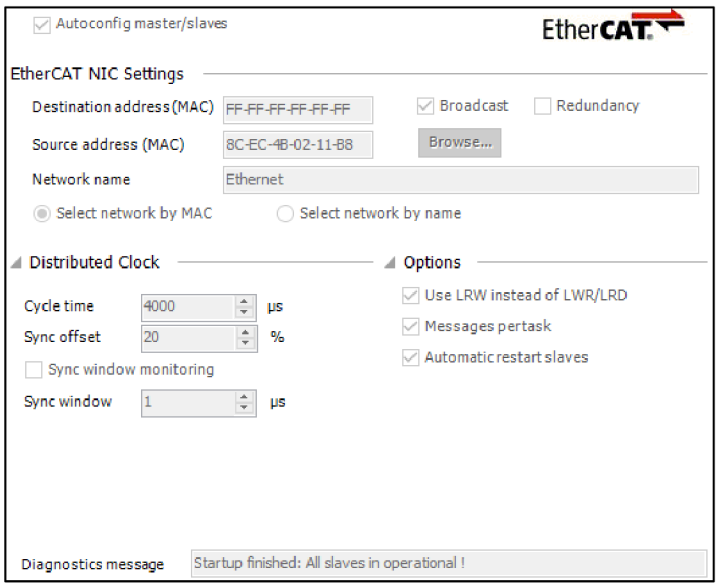
Struktura celého projektu je graficky vyobrazena na obrázku **2.31**. Pomocí Codesys Win V3 x64 je vytvořeno virtuální soft PLC a následně jsou nadefinovány další části. PLC program, napsaný v jazyce ST, tvoří logickou strukturu pro odesílání a přijímání dat pomocí EtherCat standartu. Proměnné jsou přiřazeny pomocí adres na předem určené byty procesních bufferů. Jednotlivé parametry a pokyny lze zadávat pomocí vytvořených vizualizačních oken, stejně tak zobrazovat přijímané provozní hodnoty.



2.31 Struktura Codesys projektu

Nastavení EtherCat komunikace

Na obrázku **2.31** je v dolní části struktury možné vidět položku EtherCAT_master, která je naimportována z přenosné knihovny, a pomocí které je vytvořena pro soft PLC podpora EtherCat komunikace, přesněji je ze zařízení vytvořen EtherCat master. Hlavní nastavení ukazuje obrázek **2.32**.



2.32 Konfigurace EtherCat master zařízení pomocí Codesys

EtherCat master vytvořený Codesysem podporuje možnost automatického skenování slave zařízení zapojených v aktuální topologie sítě. Aby mohlo být detekováno zařízení, musí být v knihovně obsažen jeho konfigurační .xml soubor, jenž odpovídá binárnímu ESI souboru nahranému v EEPROM paměti ESC. V případě použitého EasyCat shieldu nabízí firma AB&T připravený konfigurační soubor, který popisuje zařízení pro výměnu procesních dat o velikosti 32 vstupních a 32 výstupních bytů. Úspěšně naskenované zařízení je možné opět vidět ve stromové struktuře na obrázku **2.31**. Obrázek **2.33** dále ukazuje pomocí diagnostických zpráv úspěšný start EtherCat sítě a připojení zařízení.

Severity	Time Stamp	Description
ⓘ	05.05.2021 20:28:54.833	Startup finished: All slaves in operational !
ⓘ	05.05.2021 20:28:54.829	All slaves operational
ⓘ	05.05.2021 20:28:54.827	Set operational mode
ⓘ	05.05.2021 20:28:54.760	All slaves safe-operational
ⓘ	05.05.2021 20:28:54.748	Set safe operational
ⓘ	05.05.2021 20:28:54.748	Synchronize Slaves
ⓘ	05.05.2021 20:28:54.747	Configure distributed clock settings
ⓘ	05.05.2021 20:28:54.740	All slaves pre-operational
ⓘ	05.05.2021 20:28:54.732	prepare slaves
ⓘ	05.05.2021 20:28:54.728	All slaves init mode
ⓘ	05.05.2021 20:28:54.713	Set physical addresses
ⓘ	05.05.2021 20:28:54.627	Read slave informations
ⓘ	05.05.2021 20:28:50.613	Preparation successfull
ⓘ	05.05.2021 20:28:50.612	Networkadapter opened

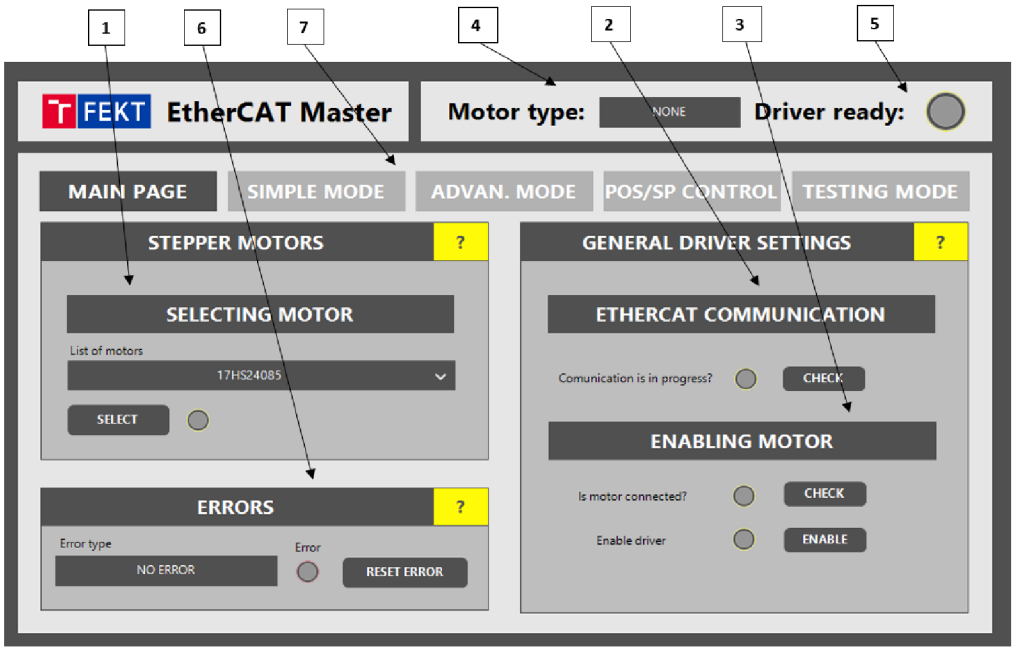
2.33 Diagnostické zprávy k úspěšném startu EtherCat sítě a připojení slave zařízení

2.6.3 Vizualizace - Hlavní okno

Rozhraní pro komunikaci s driverem, kdy jsou předávány parametry nebo řídící povely, bylo vytvořeno pomocí standartní vizualizace. Pro tvorbu vizualizace je prostředím Codesys nabízena komplexní sada vývojových nástrojů. Po spuštění PLC aplikace je pomocí webového rozhraní možné zobrazit vytvořené grafické rozhraní. Jako defaultní okno po spuštění je zobrazeno okno s názvem *MAIN PAGE*.

Před zahájením řízení motoru pomocí jednoho z dostupných režimů je vyžadováno provedení jednoduché inicializace. Nejprve je nutné zvolit řízený krokový motor, následně ověřit, zda výměna procesních dat s driverem pomocí EtherCat komunikace probíhá korektně a následně zjistit, zda je motor fyzicky skutečně připojen na výstupu, a pokud ano, tak je možné povolit práci s koncovým budičem driveru. Vytvořené okno je zobrazeno na obrázku **2.34** a dále je uveden popis jednotlivých částí v očíslovaných bodech.

1) Výběr dostupného motoru. Pro účel testování dostupný pouze typ 17HS24085.
2) Ověření výměny procesních dat mezi EtherCat masterem a podřízeným driverem.
3) Ověření připojeného motoru a následné povolení koncového budiče driveru.
4) Název vybraného motoru, jehož výběr byl potvrzen.
5) Indikace stavu podřízeného driveru. Pokud inicializace proběhne korektně, svítí žlutě.
6) Pokud nastane při běhu chyba, tak je v tomto okně zobrazen kód chyb. Při výskytu chyby je třeba tento stav vyresetovat.
7) Menu s výběrem řídících režimů. Pokud není provedena inicializace, je zakázán přístup.

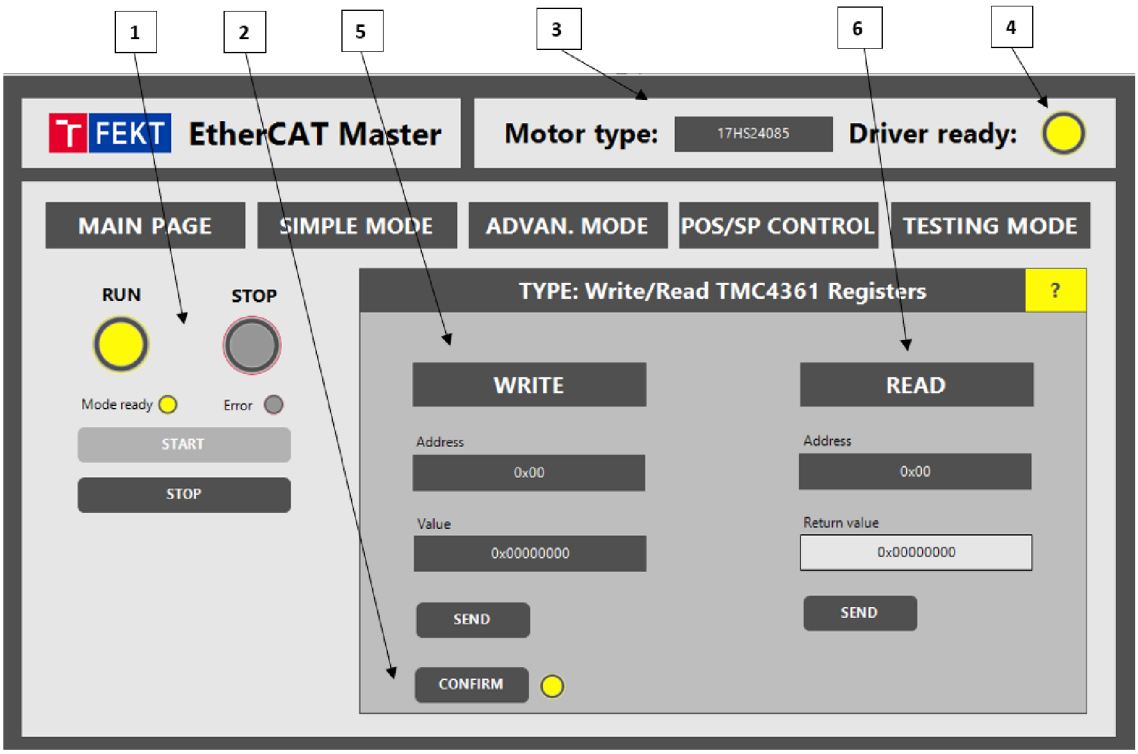


2.34 Codesys EtherCat master - hlavní okno vizualizace

2.6.4 Vizualizace - Testovací režim

Režim nazvaný jako testovací byl v rámci řídící ovládací aplikace zvolen ze dvou důvodů. Ve chvíli, kdy návrh driveru došel do bodu, že bylo potřeba začít nastavovat a ovládat pohyb motoru pomocí řídích registrů obvodu TMC4361 a TMC262, jak je popsáno v kapitole **2.5.8**, tak bylo potřeba tyto jednotlivá nastavení vhodně testovat. K tomu byl využit tento režim, kdy je možné zapsat na jakýkoliv dostupných registr novou hodnotu nebo naopak z libovolného registru hodnotu číst. Tato možnost byla velice důležitá při návrhu softwarového vybavení driveru. Druhým důvodem byla skutečnost, že mnohdy by celá řídící aplikace mohla probíhat na straně PLC a řídící mikrokontrolér v podřízeném zařízení by mohl sloužit pouze jako interpret pro zápis a čtení. Vytvořené okno vizualizace s názvem *TESTING MODE* je zobrazeno na obrázku **2.35** a následně je uveden popis obsahu okna v očíslovaných bodech.

1) Aktuální stav režimu. Po úspěšném nastavení režimu a stisku start je možné zapisovat nebo číst registry.
2) Potvrzení nastaveného režimu. Bez tohoto nastavení není možné provést start režimu a zapisovat do ovládacích prvků vizualizace.
3) Název vybraného motoru, jehož výběr byl potvrzen.
4) Indikace stavu podřízeného driveru. Pokud inicializace proběhla korektně, svítí žlutě.
5) Zápis hodnoty na adresu registru.
6) Čtení hodnoty z adresy vybraného registru.

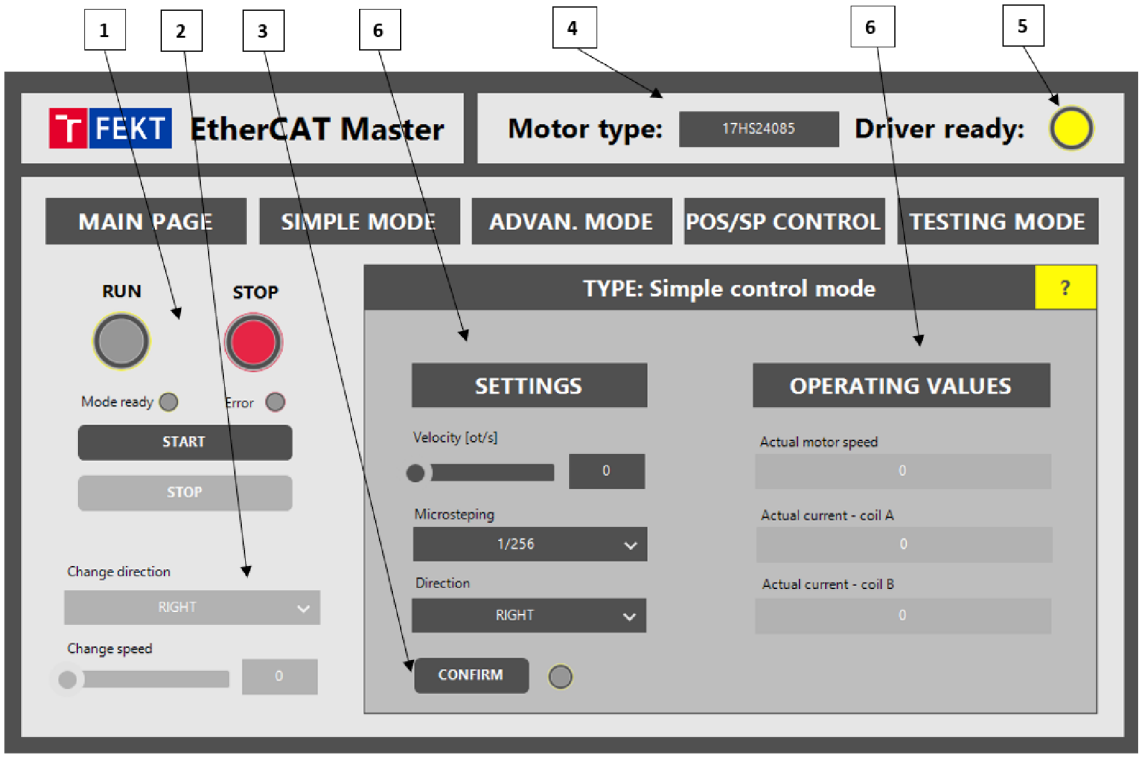


2.35 Codesys EtherCat master - okno vizualizace pro testování registrů TMC4361

2.6.5 Vizualizace - Základní režim řízení

Režim nazvaný jako základní dává k dispozici možnost nastavení základních parametrů, kterými disponuje většina běžných driverů. Jako vstupní parametry je třeba zvolit počáteční rychlost, režim mikrokrokování a počáteční směr otáčení. Po úspěšném potvrzení nastavení lze řízení motoru odstartovat a následně měnit provozní parametry v reálném čase. Z provozních hodnot je zobrazována aktuální rychlost a proud jednotlivými vynutími motoru. Výstupní profil rychlosti je nastaven jako obdélníkový, což způsobí při velké počáteční rychlosti nebo prudké změně rychlosti seknutí motoru. Vytvořené okno vizualizace, s názvem *SIMPLE MODE,* je zobrazeno na obrázku **2.36** a následně je uveden popis obsahu okna v očíslovaných bodech.

1) Aktuální stav režimu. Po úspěšném nastavení režimu a stisku start je spuštěn běh motoru a možnost změny provozních parametrů.
2) Ovládací prvky pro možnost změn za běhu motoru v reálném času.
3) Potvrzení nastaveného režimu. Bez tohoto nastavení není možné provést start režimu a provádět změny.
4) Název vybraného motoru, jehož výběr byl potvrzen.
5) Indikace stavu podřízeného driveru. Pokud inicializace proběhla korektně, svítí žlutě.
6) Ovládací prvky pro nastavení prvotních parametrů režimu.
7) Zobrazovací prvky pro vyčtené provozní hodnoty.

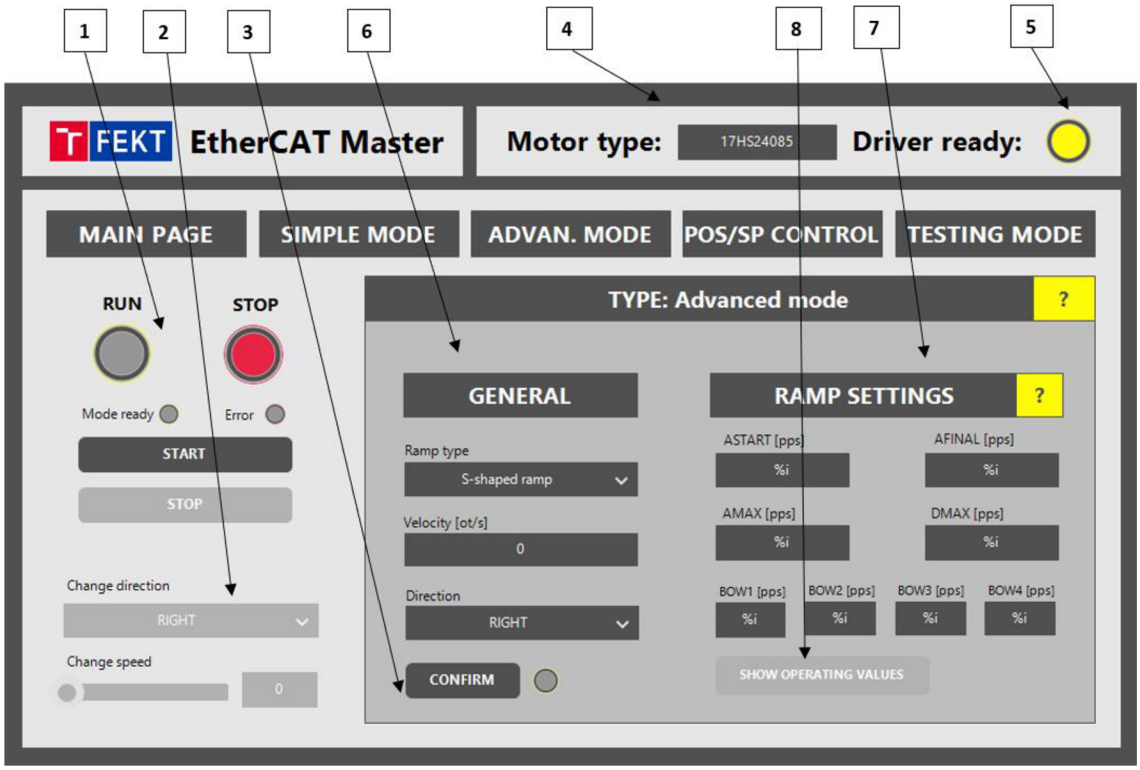


2.36 Codesys EtherCat master - okno vizualizace pro řízení motoru v základním režimu

2.6.6 Vizualizace - Pokročilý režim řízení

Režim nazvaný jako pokročilý dává k dispozici možnost volby rychlostního profilu a také editaci jeho parametrů, což prakticky umožňuje modelování vybrané křivky. K dispozici je volba profilu typu lichoběžník, v základním nebo pokročilém šestibodém tvaru, a dále s-křivka. Ze základních parametrů je nutné zvolit rychlost a směr otáčení motoru, režim mikrokrokování je defaultně nastaven na 1/256. Každý typ rychlostního profilu má odlišné parametry nastavení, které se nastavují v jednotkách PPS. Vytvořené okno vizualizace s názvem *ADVANCED MODE* je zobrazeno na obrázku **2.37** a následně je uveden popis obsahu okna v očíslovaných bodech.

1) Aktuální stav režimu. Po úspěšném nastavení režimu a stisku start je spuštěn běh motoru a možnost změny provozních parametrů.
2) Ovládací prvky pro možnost změn za běhu motoru v reálném času.
3) Potvrzení nastaveného režimu. Bez tohoto nastavení není možné provést start režimu a provádět změny.
4) Název vybraného motoru, jehož výběr byl potvrzen.
5) Indikace stavu podřízeného driveru.
6) Ovládací prvky pro nastavení prvotních parametrů režimu.
7) Ovládací prvky pro nastavení parametrů zvoleného rychlostního profilu.
8) Tlačítko pro přechod na okno s aktuálními provozními hodnotami.

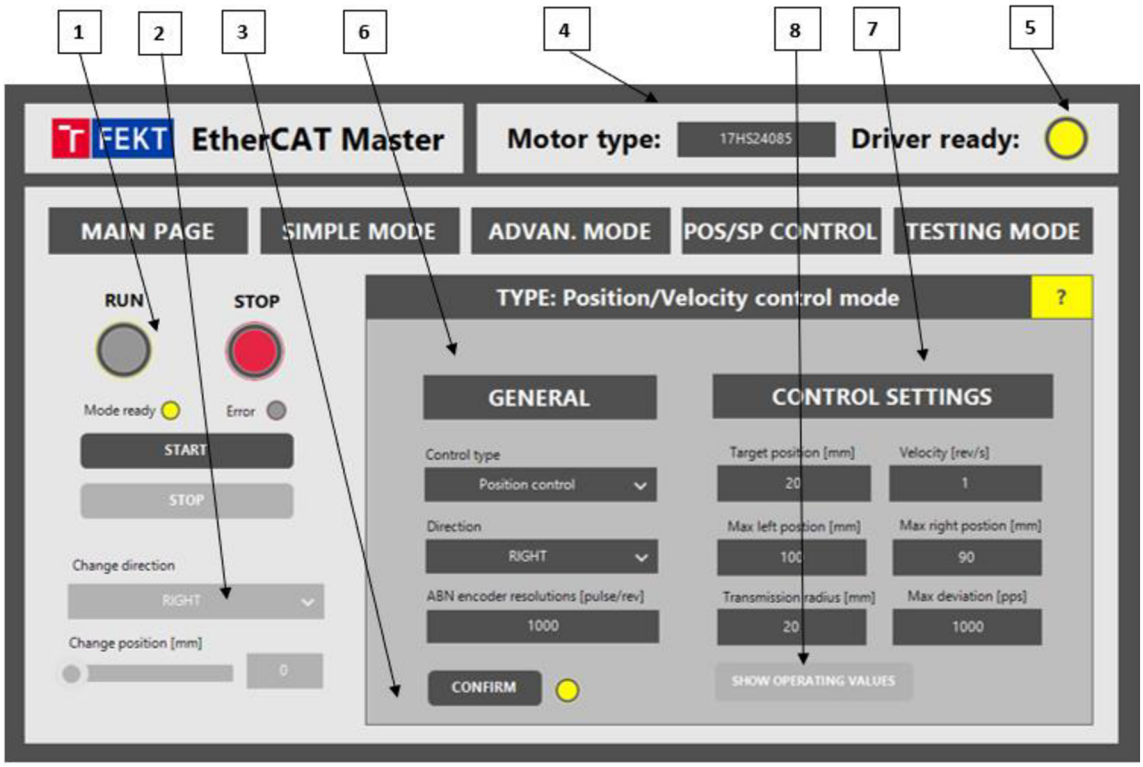


2.37 Codesys EtherCat master - okno vizualizace pro řízení motoru v pokročilém režimu

2.6.7 Vizualizace - Režim řízení na rychlost nebo polohu v uzavřené smyčce

Režim nazvaný jako řízení na rychlost nebo na polohu dává k dispozici možnosti využití řízení v režimu uzavřené smyčky. Obecně by šlo tento režim velice pokročile parametrizovat, nicméně v tomto případě jsou prozatím defaultně nastaveny parametry rampy či režim mikrokrokování, a možnosti volby jsou následující. Pro oba režimy lze nastavit rozlišení zpětnovazebního enkodéru, směr nebo maximální povolenou odchylku řízení. Při režimu řízení na polohu lze zvolit mezní parametry polohy a také poloměr převodovky. Vytvořené okno vizualizace s názvem *POS/SP CONTROL* je zobrazeno na obrázku **2.38** a následně je uveden popis obsahu okna v očíslovaných bodech.

1) Aktuální stav režimu. Po úspěšném nastavení režimu a stisku start je spuštěn běh motoru a možnost změny provozních parametrů.
2) Ovládací prvky pro možnost změn za běhu motoru v reálném času.
3) Potvrzení nastaveného režimu. Bez tohoto nastavení není možné provést start režimu a provádět změny.
4) Název vybraného motoru, jehož výběr byl potvrzen.
5) Indikace stavu podřízeného driveru.
6) Ovládací prvky pro nastavení prvotních parametrů režimu.
7) Ovládací prvky pro nastavení parametrů zvoleného typu řízení
8) Tlačítko pro přechod na okno s aktuálními provozními hodnotami.

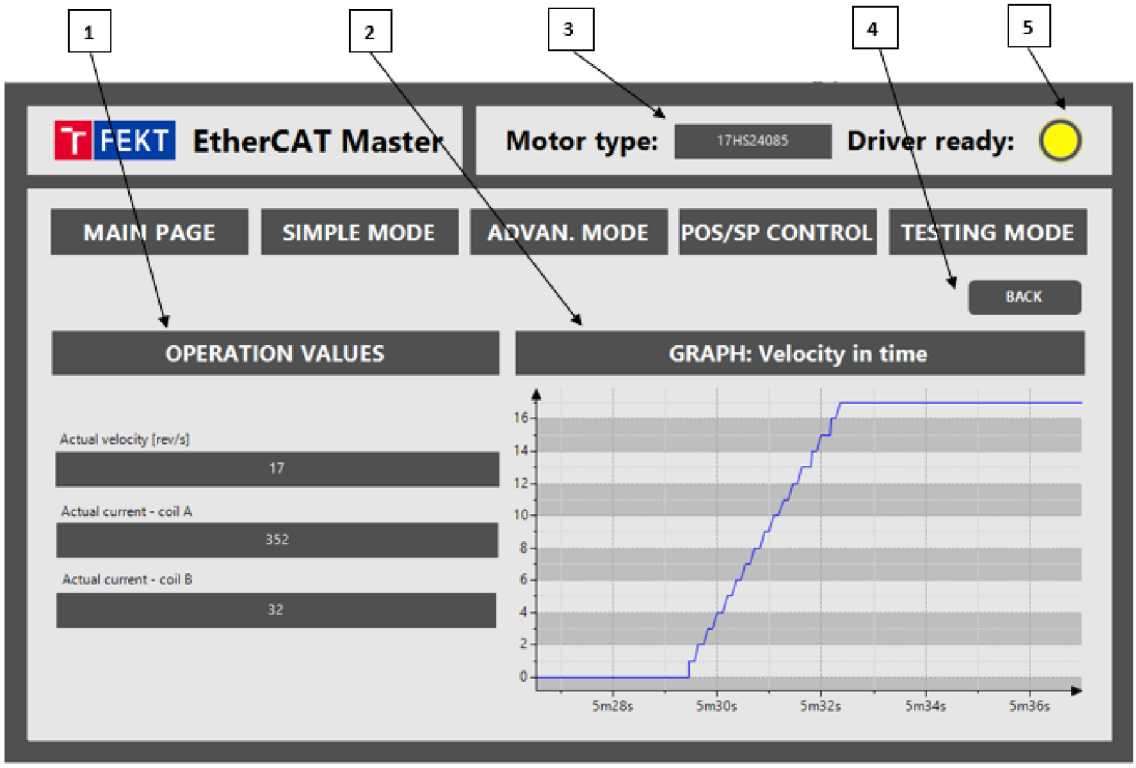


2.38 Codesys EtherCat master - okno vizualizace pro řízení motoru v pokročilém režimu

2.6.8 Vizualizace – Zobrazování provozních hodnot

Každý režim je opatřený vlastním oknem pro zobrazování aktuálních vyčtených hodnot. Pro názornost jsou vyčítány hodnoty v podobě aktuální rychlosti a hodnoty proudu, které jsou generované ve vinutích motoru. Okno zachycující vyčítání a zobrazování hodnot je ukázáno na obrázku **2.39**. Na tomto snímku je také zachycen grafický průběh nárustu rychlosti. Režim byl nastaven na *ADVANCED MODE* s rozběhovým profilem typu lichoběžníková rampa a cílovou rychlostí 17 otáček za sekundu. Vidět je odpovídají nárust rychlosti, pouze je zde lehký defekt v podobě schodovitého tvaru průběhu. Tento tvar je způsoben tím, že rychlost je vyčítána pouze po jednotkách otáček, což není zcela vhodné.

1) Prvky pro zobrazování provozních hodnot.
2) Grafický průběh rychlosti, popřípadě polohy, zachycené v čase.
3) Název vybraného motoru, jehož výběr byl potvrzen.
4) Tlačítko pro přechod na okno s nastavením režimu.
5) Indikace stavu podřízeného driveru.



2.39 Codesys EtherCat master - okno vizualizace pro vyčtené provozní hodnoty

2.7 Testování výsledného zařízení

Ve chvíli, kdy byl zhotovený navrhovaný driver se všemi náležitostmi, a vytvořeno ovládací EtherCat master zařízení, tak nastal okamžik, kdy bylo třeba vše vhodným způsobem otestovat. Tato kapitola uvádí zprvu požadavky na testování a následně zvolenou metodiku a zhodnocené výsledky.

2.7.1 Požadavky na testování driveru

V rámci požadavků na otestování funkce zrealizovaného driveru byly stanoveny požadavky, které jsou uvedeny v následující **Tabulce 2.9.**

Tabulka 2.9 Přehled provedených testů

P.	Typ testu
1.	Měření doby propagace EtherCat rámce
2.	Zachycení a analýza vyslaného EtherCat rámce
3.	Měření skutečných otáček motoru

2.7.2 Krokový motor 17HS2408S a ABN enkodér S3806

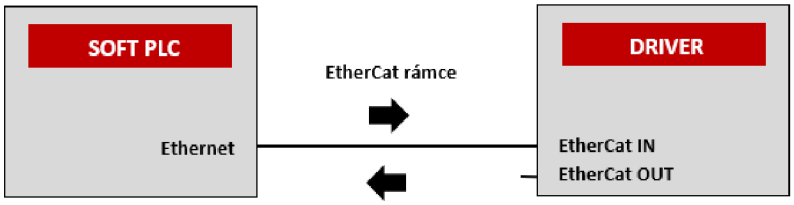
Jako testovaný motor byl využit krokový motor HS2408S velikosti NEMA 17. Jedná se o standartní krokový motor využívaný například v 3D tiskárnách. Pro měření otáček byl využit rotační inkrementální ABN enkodér typu S3806. Základní parametry využitých komponent uvádí **Tabulka 2.10**.

Tabulka 2.10 Parametry motoru 17HS2408S a enkodéru S3806 [37] [38]

	Krokový motor 17HS2408S		**ABN enkodér S3806**	
P.	**Parametr**	**Hodnota**	**Parametr**	**Hodnota**
1.	Velikost	Nema17	Typ enkodéru	Inkrementální
2.	Trvalý proud [A]	0,60	Průměr pouzdra [mm]	38,00
3.	Maximální proud [A]	1,00	Napájecí napětí [V]	5-24
4.	Úhel na jeden krok [°]	1,80	Výstupní napětí [V]	5-26 (open col.)
5.	Přídržný moment [N.m]	0,25	Průměr hřídele [mm]	6,00
6.	Odpor vinutí [Ω]	8,00	Výstupní frekvence [kHz]	Max.300
7.	Indukčnost vinutí [mH]	10,00	Rozlišení [impuls/ot]	1000

2.7.3 Doba propagace EtherCat rámce

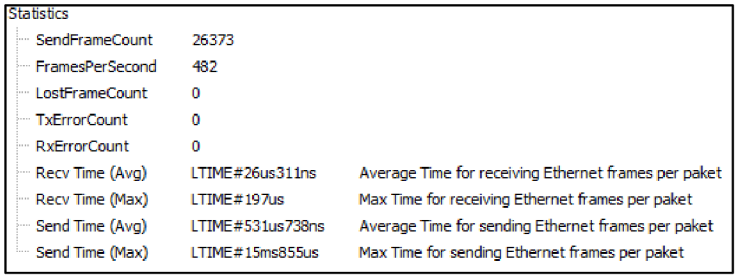
Ohledně parametru nazvaného jako doba propagace EtherCat rámce bylo cílem změřit dva parametry. Prvním parametrem je doba propagace celou sítí, tedy doba od odeslání a zpětného přijetí EtherCat rámce, a druhým parametrem je doba propagace od odeslání EtherCat rámce po přijetí provozních dat mikrokontrolérem podřízeného driveru. Při měření byla použita topologie sítě ukázaná na obrázku **2.40**. Vytvořený program pro EtherCat master zařízení byl spuštěn pomocí PC, kde bylo vytvořeno soft PLC. Jelikož PC obsahuje pouze jeden ethernetový vstup, tak nebylo možné vytvořit kruhovou topologii, která by byla pro měření vhodnější. Pro tento typ úlohy byl zvolen program Wireshark, který umožňuje sledovat veškerý přenos paketů skrze fyzické ethernetovém rozhraní, včetně filtrace těch, která jsou typu EtherCat, tedy označeny pomocí EtherType hodnotou 0x88A4.



2.40 Topologie EtherCat sítě při měření

Propagace EtherCat rámce v rámci sítě

V teoretickém rozboru EtherCat protokolu byla zmíněna možnost dosažení velmi krátkého cyklu až v jednotkách mikrosekund. Vzhledem k nastavení sítě, typu operačního systému, na kterém běží master zařízení, nevhodné topologii a vybavení fyzické vrstvy nešlo očekávat takto skvělé výsledky. Prostředí Codesys nabízí v rámci statistiky během komunikace parametry, které ukazuje obrázek **2.41**. Pomocí programu WireShark byl zaznamenán průběh komunikace na ethernetovám portu. Každý odchozí a příchozí rámec byl opatřen časovým razítkem a pomocí těchto hodnot byl stanoven cyklus sítě na hodnotu 600 mikrosekund. Druhý zmíněný parametr nakonec změřen nebyl, jelikož pro cílovou metodu, kde měl být využit hardwarový výstup z master zařízení nebyly dostupné prostředky a jiná metoda nebyla z časových důvodů zvolena.

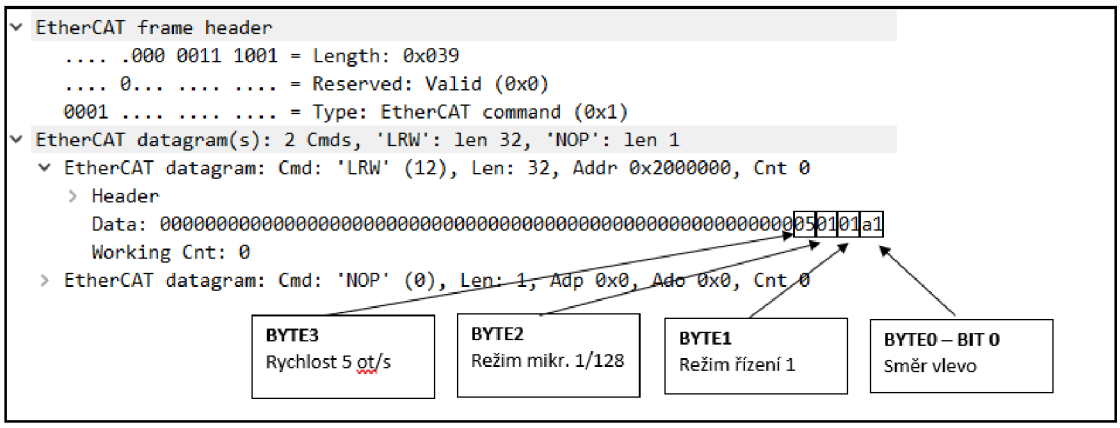


2.41 Codesys EtherCat master status

2.7.4 Analýza EtherCat rámce

V kapitole **2.3.5** byla teoreticky rozebrána struktura EtherCat rámce, který je v rámci EtherCat standartu využíván pro přenos procesních a dalších provozních dat. Cílem této kapitoly je zachytit vysílané, respektive přijímané rámce a zanalyzovat jejich obsah z hlediska obecné struktury a očekávaného obsahu procesních dat. Pro tuto úlohu byl zvolen opět program Wireshark, který kromě zachycení rámců dokáže zobrazit i jejich celkovou strukturu a datový obsah.

Cílem bylo detekovat odesílaný rámec, ve kterém je obsažen datagram s procesními daty, a v sobě má obsažené informace o nastavení základního režimu. Parametry byly při nastavení režimu nastaveny následujícím způsobem – rychlost 5 otáček za sekundu, režim mikrokrokování 1/128 a směr otáčení na levou stranu.



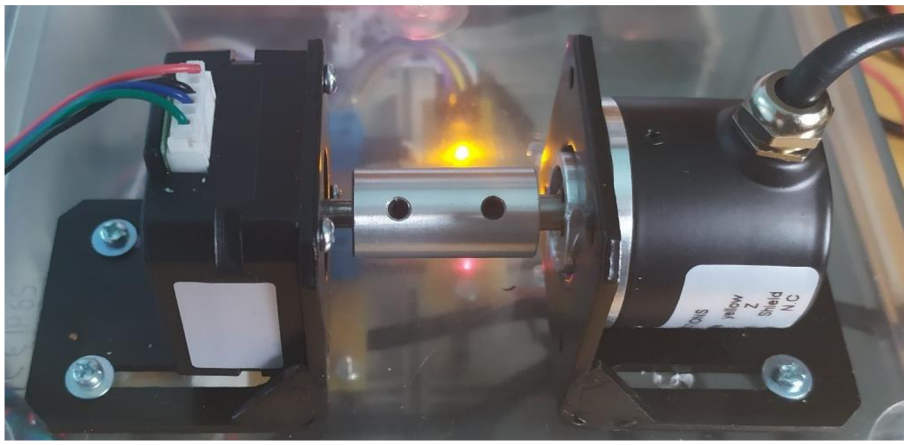
2.42 Analyzovaný EtherCat datagram pomocí WireShark

Obrázek **2.42** ukazuje zachycený EtherCat rámec odesílaný směrem z EtherCat master zařízení a je jasně vidět standartní struktura. Datagram označený příkazem LRW obsahuje odesílaná procesní data pro podřízené zařízení opatřené shieldem EasyCat. Zachycená data odpovídají přepokládaným výsledkům. Rychlost je zapsána v bytu 3 a odpovídá dekadické hodnotě 5.Režim mikrokrokování odpovídá hodnotě 1, ta symbolizuje režim 1/128. Zvolený režim řízení odpovídá hodnotě 1, ta symbolizuje základní režim. Směr otáčení je symbolizován nultým bitem bytu 0. Pro tento případ je nastaven na hodnotu 1, která odpovídá směru otáčení doleva. Další bity bytu 0 nejsou analyzovány, jedná se o další řídící bity. Nutné je podotknout, že procesní buffery na straně mastera i podřízeného zařízení fungují na principu FIFO paměti, takže v přenášeném datagramu je směr otočený, respektive byte 31 odpovídá bytu 0 a tak dále.

Celá zaznamenaná komunikace, pro měření doby cyklu a analýzu EtherCat rámce, je uvedena v příloze na CD ve složce naměřená data.

2.7.5 Měření skutečných otáček

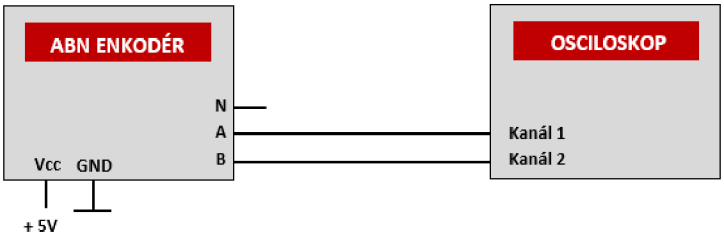
Pro ověření funkčnosti řízení otáček krokového motoru bylo zvoleno měření, kdy je motor řízen v režimu otevřené smyčky a pomocí inkrementálního enkodéru je snímán průběh otáčení. Motor a enkodér jsou spojeny mechanickou spojkou, jak je ukázáno na obrázku **2.43** a výstupy enkodéru jsou připojeny na jednotlivé kanály osciloskopu, pomocí kterého byly změřeny průběhy a parametry.



2.43 Mechanické spojení hřídelí krokového motoru a enkodéru

Inkrementální ABN enkodér obsahuje celkem tři výstupy. Výstupy A, B slouží pro výstupní impulsní průběh, kde počet impulsů pro jednu otáčku odpovídá rozlišení enkodéru. V tomto případě má vybraný enkodér rozlišení 1000 impulsů na jednu otáčku. Rychlost motoru je úměrná frekvenci zmíněného impulsního průběhu na výstupech A,B. Směr otáček je dán fázovým posuvem mezi průběhy, a jeho hodnota je buď +90° nebo -90°. Otáčky byly nastavovány pomocí vytvořeného ovládacího rozhraní od 1 ot/s po 20 ot/s. Krok změny byl stanoven na 1 ot/s. Výstup N dává jednou za reálnou mechanickou otáčku enkodéru na svůj výstup impuls, což se dá využít například pro otočení hřídele motoru do referenční polohy, tento výstup v rámci měření nebyl využit.

Pro měření bylo tedy využito závislosti frekvence na otáčkách, pomocí digitálního osciloskopu byly zobrazeny jednotlivé průběhy, byla detekována perioda a odečtena frekvence. Jednoduchý měřící obvod ukazuje schéma na obrázku **2.44**.

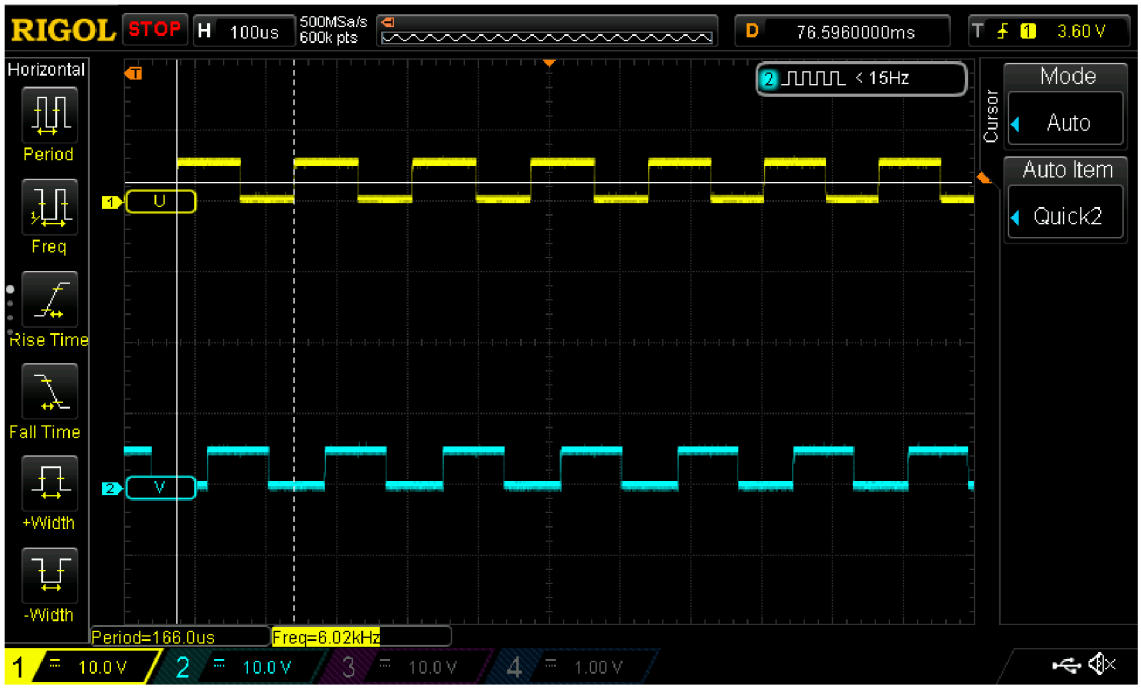


2.44 Měření otáček pomocí osciloskopu

Pro ověření principů měření enkodérem je na obrázku **2.45** ukázán záznam obrazovky osciloskopu. Na tomto záznamu je vidět totožný průběh na obou kanálech, pouze s rozdílem fázového posuvu. Tento konkrétní záznam odpovídá otáčkám 6 ot/s. Výpočet byl proveden podle vztahu **2.1.** Všechny naměřené a vypočtené hodnoty jsou uvedeny v příloze **C.1.**

$$n_s = \frac{f}{Rozlišení\ enkodéru} = \frac{6020}{1000} = 6{,}02\ \left[\frac{ot}{s}\right] \qquad (2.1)$$

Z naměřených hodnot lze konstatovat úspěšné ověření nastavených otáček a funkce použitého enkodéru. Nutné je podotknout, že při zařazení enkodéru do zpětnovazebního obvodu je signál vyhodnocován na principu čítání, nikoliv měření frekvence. Čítání impulsů je pro zpracování výstupů přesnější a používanější metoda. Pro každou dvojici nastavené a naměřené hodnoty byla vypočítána procentní chyba. Tato chyba se v naměřeném intervalu hodnot pohybovala v jednotkách procent, nicméně je důležité ještě jednou zmínit, že do měření vstoupila celá řada vnějších vlivů. Účel byl však splněn.



2.45 Záznam obrazovky osciloskopu při měření otáček pomocí enkodéru

3 ZÁVĚR

Cílem této bakalářské práce bylo navrhnout driver pro řízení krokových motorů, který podporuje ovládání pomocí vybrané průmyslové komunikace. Před zahájením návrhu bylo třeba vytvořit teoretický základ informací. Pro tuto potřebu byla vytvořena teoretická rešerše na téma krokové motory, drivery pro řízení krokových motorů a průmyslové komunikační sítě, včetně možnosti implementace v embedded zařízeních.

Na základě teoretických poznatků byl pro ovládání driveru vybrán moderní a rozvíjející se protokol průmyslového ethernetu s názvem EtherCat, jenž podporuje velmi krátký cyklus, kvalitní synchronizaci a obecně je považován za vhodnou variantu při řízení motorů. Dalším důvodem byla obecně snadná integrace do embedded zařízení pomocí obvodů ESC. Po zvolení této varianty bylo potřeba se seznámit s tímto protokolem podrobněji, základní teorie byla rozšířena o pokročilý souhrn informací jako jsou požadavky na hardwarové vybavení master/slave zařízení nebo možnosti adresování, synchronizace či topologie.

Po důkladném prostudování teoretických informací a vytvoření příslušných rešerší bylo přistoupeno k praktické části práce. Základem úspěchu bylo vytvoření celkového konceptu a prvotní definice požadavků. Hardwarová část driveru byla vytvořena jako prototyp pomocí modulových komponent, respektive populárních vývojových shieldů. Vytvořený prototyp splňuje požadavky na všechny definované vstupy, výstupy, parametry, podporu EtherCat protokolu a SPI komunikaci mezi všemi vnitřními částmi. Při konstrukci bylo využito univerzální krabičky, základních mechanických součástí, vybraných komponent i jednoduchého plošného spoje.

Při návrhu řídícího firmwaru byly, před započetím programováním, vytvořeny a graficky vyjádřeny diagramy z hlediska hlavní řídící smyčky, objektové struktury nebo datového toku. Tyto diagramy usnadňovaly práci během celého vývoje, jelikož byl předem daný jasný a strukturovaný cíl. Následně byly úspěšně implementovány jednotlivé třídy. Pouze pro obsluhu vybraného ESC LAN9252 byla převzata již hotová třída EasyCat a pomocí jejího snadného veřejného rozhraní zakomponována do celkové struktury. Primárním cílem při návrhu bylo úspěšné zprovoznění komunikace z jedné strany s EtherCat masterem a z druhé strany s TMC obvody pro řízení motoru. Poté následovala část implementace konkrétních funkcí pro ovládání motoru, podle předem stanovených požadavků. V poslední části byla pomocí vytvořených součástí implementována hlavní řídící smyčka a stanovené řídící režimy. Díky dodržené objektové struktuře je výsledný program poměrně názorný a přehledný.

Po zhotovení prototypu z hlediska hardwaru i softwaru bylo nutné provést vhodný způsob testování, k tomu bylo zapotřebí vytvořit vhodné master zařízení. Pro tento účel byl zvolen klasický PC s řídící PLC aplikací vytvořenou pomocí vývojového prostředí Codesys. V rámci řídícího firmwaru byly naimplementované zmíněné řídící režimy, které vyžadující odlišný specifický formát procesních dat a možnost změny hodnot parametrů.

Z tohoto důvodu byly vytvořeny přehledná uživatelská vizualizační okna s možností volby parametrů a řízení běhu programu pomocí standartních povelů jako jsou například start, stop, změna rychlosti nebo reverzace otáčení. Vytvořený PLC program s přívětivým uživatelským rozhraním byl hojně využíván při ladění firmwaru a následném testování požadovaných funkcí.

Poslední část práce se věnuje zmíněnému testování funkčnosti driveru. Pomocí vytvořeného ovládacího zařízení byly otestovány všechny implementované režimy či změny parametrů za běhu motoru. Pro ověření správné funkčnosti samotného driveru byly měřeny otáčky v režimu otevřené smyčky pomocí osciloskopického měření. Úspěšně byla ověřena funkcionalita správného nastavení rychlosti a směru otáčení. Dále byly ověřeny pomocí programu WireShark teoretické poznatky ohledně EtherCat protokolu. Ověřen byl princip průchodu EtherCat rámců, provedena analýza struktury a předpokládaného datového obsahu rámce, a na základě změřených hodnot času, při odesílání/přijímání rámců, určena průměrná hodnota cyklu sítě.

Závěrem lze konstatovat následující. Byl úspěšně zrealizován prototyp driveru s podporou ovládání pomocí EtherCat protokolu. Úspěšně bylo vytvořeno hardwarové vybavení a implementován řídící firmware. Pro otestování driveru bylo vytvořeno EtherCat master zařízení a všechny funkcionality byly otestovány.

Na úplný závěr je nutné objektivně zhodnotit úspěšně implementované funkce a zároveň nedostatky vytvořeného zařízení. Jako pozitivní je dobré zmínit úplné, nebo v některých případech alespoň částečné, dodržení všech bodů zadání práce a vlastních postupně definovaných požadavků. Výsledný driver může být zapojen a detekován v topologii EtherCat sítě, s master zařízením je schopen výměny procesních dat v buffer režimu, je schopen přijímat nastavení a podle toho vykonávat otáčení motoru s vybranou rychlostí, polohou směrem či specifickým rozběhovým profilem. Vybrané parametry mohou být měněny za běhu motoru, a naopak vybrané provozní hodnoty mohou být za běhu motoru odesílány zpět do master zařízení pro přehled a vizualizaci. Driver může být zapojen a provozován v režimu otevřené i uzavřené smyčky, nicméně je nutné zmínit, že většina funkcionalit byla testována spíše v režimu otevřené smyčky a režim uzavřené smyčky byl implementován až v samotném závěru, kde byla ověřena jen funkčnost zpracování zpětné vazby a dosažení žádané rychlosti. Mezi hlavní nedostatky patří především nedostatečné využití potenciálu vybraných TMC obvodů, které nabízejí mnohem více možností, funkcionalit a parametrizace. Dále by bylo potřeba celý řídící firmware zuniverzálnit a vylepšit. Pro EtherCat komunikaci by bylo vhodné využít mailbox režim s rozšířeným profilem. Pokud by mělo být zařízení využito v praxi, je nutné vše integrovat na plošný spoj a přizpůsobit potřebným normám a legislativě.

LITERATURA

[1] Krokové motory 1 – typy motorů. *Robodoupě* [online]. 11.9.2013 [cit. 2020-12-28]. Dostupné z: http://robodoupe.cz/2013/krokove-motory-1-typy-motoru/

[2] KUBIN, Jiří. *Krokové motory a jejich řízení* [online]. [cit. 2020-12-28]. Dostupné z:https://www.pslib.cz/jiri.kubin/SIZ/Elektrick%C3%A9%20stroje/Krokov%C3%A9_motory.pdf

[3] ARM, Jakub. *Motory*. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2019.

[4] MADARA, Daniel. *Řízení krokového motoru pomocí platformy TI*. 2017. Bakalářská práce. ČVUT, Fakulta elektrotechnická. Vedoucí práce Ing. Jan Bauer, Ph.D.

[5] *Servo-drive* [online]. [cit. 2020-12-28]. Dostupné z: www.servo-drive.cz

[6] KORDIK, Jeff. Open-loop stepper motor versus closed-loop stepper motor systems. Motion control tips [online]. 2016 [cit. 2020-12-28]. Dostupné z: https://www.motioncontroltips.com/open-loop-stepper-motor-versus-closed-loop-stepper-motor-systems/

[7] COLLINS, DANIELLE. What is a motion profile. Motion control tips [online]. 2.10.2019 [cit. 2021-5-22]. Dostupné z: https://www.motioncontroltips.com/what-is-a-motion-profile/

[8] LEWIN, Chuck. Mathematics of Motion Control Profiles. *Performance Motion Devices* [online]. [cit. 2021-5-22]. Dostupné z: https://www.pmdcorp.com/resources/type/articles/get/mathematics-of-motion-control-profiles-article

[9] PROGRAMMABLE STEP MOTOR CONTROLLER R272-42-ETH and R272-80-ETH: Manual. *Servo-drive* [online]. 2018 [cit. 2020-12-28]. Dostupné z: https://www.servo-drive.cz/pdf_catalog/controller_R272-42-ETH.pdf

[10] N5 – MOTOR CONTROLLERS FOR CANOPEN, ETHERCAT, ETHERNET/IP OR MODBUS RTU/TCP. Nanotec [online]. [cit. 2020-12-28]. Dostupné z: https://en.nanotec.com/products/1597-1597-n5-motor-controller-a-vdc

[11] R356 Controller: User manual. *Servo-drive* [online]. [cit. 2020-12-28]. Dostupné z: https://www.servo-drive.cz/pdf_catalog/Manual_R356.pdf

[12] Pololu High-Power Stepper Motor Driver 36v4. Pololu Robotics & Electronics [online]. [cit. 2020-12-28]. Dostupné z: https://www.pololu.com/product/3730

[13] ZEZULKA, František a Ondřej HYNČICA. Průmyslový Ethernet VIII: Ethernet Powerlink, Profinet. Ústav automatizace a měřící techniky [online]. UAMT FEKT VUT v Brně, 28.122008 [cit. 2020-12-29]. Dostupné z: https://www.uamt.feec.vutbr.cz/~zezulka/download/KPPA/A05_08s62_VIII.pdf

[14] BÍLEK, Karel. Ethernet Powerlink – komunikace v reálném čase. *Automa* [online]. [cit. 2020-12-29]. Dostupné z: https://automa.cz/cz/casopis-clanky/ethernet-powerlink-komunikace-v-realnem-case-2001_03_33500_2769/

[15] ARM, Jakub. *Komunikační sítě: Prostředky průmyslové automatizace (BPC PPA)*. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2019.

[16] ZEZULKA, František a Ondřej HYNČICA. Průmyslový Ethernet IX: EtherNet/IP, EtherCAT. *Ústav automatizace a měřící techniky* [online]. UAMT FEKT VUT v Brně, 28.12.2008 [cit. 2020-12-29]. Dostupné z: https://www.uamt.feec.vutbr.cz/~zezulka/download/KPPA/A10_08s60_IX.pdf

[17] HANZÁLEK, Zdeněk a Pavel BURGET. Průmyslová sběrnice Profibus. *Vývoj.HW.cz* [online]. https://vyvoj.hw.cz/, 17. 1. 2004 [cit. 2020-12-28]. Dostupné z: https://vyvoj.hw.cz/navrh-obvodu/rozhrani/prumyslova-sbernice-profibus.html

[18] ZÁKLADNÍ INFORMACE O PRŮMYSLOVÉ SBĚRNICI PROFIBUS – ČÁST I. *Foxon* [online]. 19.7. 2013 [cit. 2020-12-28]. Dostupné z: https://www.foxon.cz/blog/prakticka-teorie/162-zakladni-informace-o-prumyslove-sbernici-profibus-cast-i

[19] Multiprotocol Chip Interface for Industrial Communication. *SmartIndustry* [online]. [cit. 2021-01-02]. Dostupné z: https://www.smartindustry.com/assets/Uploads/SI-WP-Hilscher-MPC-Chip.pdf

[20] EtherCAT - Ethernet Fieldbus. *ETHERCAT Technology Group* [online]. [cit. 2021-5-22]. Dostupné z: https://www.ethercat.org/en/technology.html

[21] What Is EtherCAT Protocol and How Does It Work? *DEWESoft* [online]. 26.7.2020 [cit. 2021-5-22]. Dostupné z: https://dewesoft.com/daq/what-is-ethercat-protocol#time-stamped-data

[22] EtherCat - General [online]. [cit. 2021-5-22]. Dostupné z: https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_io_intro/1257993099.html&id=3196541253205318339

[23] EtherCAT State Machine. *BECKHOFF* [online]. [cit. 2021-5-22]. Dostupné z: https://infosys.beckhoff.com/english.php?content=../content/1033/ax5000_usermanual/html/Bt_EcBasics_EcStateMachine.htm&id=

[24] EtherCAT introduction. *IOC Robotic's Lab* [online]. [cit. 2021-5-23]. Dostupné z: https://sir.upc.edu/wikis/roblab/index.php/Development/Ethercat

[25] STM32F446RE. STMicroelectronics [online]. [cit. 2021-5-22]. Dostupné z: https://www.st.com/en/microcontrollers-microprocessors/stm32f446re.html

[26] TMC4361A-BOB. *TRINAMIC* [online]. [cit. 2021-5-22]. Dostupné z: https://www.trinamic.com/support/eval-kits/details/tmc4361a-bob/

[27] TMC4361 DATASHEET. *Trinamic* [online]. [cit. 2021-01-02]. Dostupné z: https://www.trinamic.com/fileadmin/assets/Products/ICs_Documents/TMC4361_Datasheet_Rev3.10.pdf

[28] TMC262C-LA DATASHEET. *TRINAMIC* [online]. [cit. 2021-5-22]. Dostupné z: https://www.trinamic.com/products/integrated-circuits/details/tmc262c-la-1/

[29] TMC262-BOB. *TRINAMIC* [online]. [cit. 2021-5-22]. Dostupné z: https://www.trinamic.com/support/eval-kits/details/tmc262-bob/

[30] LAN9252. *MICROCHIP* [online]. [cit. 2021-5-22]. Dostupné z: http://ww1.microchip.com/downloads/en/devicedoc/00001909a.pdf

[31] EasyCAT Shield for Arduino. AB&T [online]. [cit. 2021-5-22]. Dostupné z: https://www.bausano.net/en/hardware/ethercat-e-arduino/easycat.html

[32] PLATFORMIO [online]. [cit. 2021-5-22]. Dostupné z: https://platformio.org/

[33] EasyCAT_lib. *MBED* [online]. [cit. 2021-5-22]. Dostupné z: https://os.mbed.com/users/EasyCAT/code/EasyCAT_lib/

[34] EtherCAT master in TwinCAT. *BECKHOFF* [online]. [cit. 2021-5-22]. Dostupné z: https://infosys.beckhoff.com/english.php?content=../content/1033/ethercatsystem/2469087755.html&id=

[35] SOEM. *Open EtherCAT Society* [online]. [cit. 2021-5-22]. Dostupné z: https://openethercatsociety.github.io/doc/soem/

[36] CODESYS ETHERCAT MASTER. *Codesys* [online]. [cit. 2021-5-22]. Dostupné z: https://www.codesys.com/products/codesys-fieldbus/industrial-ethernet/ethercat.html

[37] Krokový motor Nema 17 HS2408S. *GIGA-PC* [online]. [cit. 2021-5-22]. Dostupné z: https://www.giga-pc.cz/elektro-dily/krokovy-motor-nema-17-hs2408s-2/

[38] S3806 rotačni inkrementální enkodér. *CNCSHOP* [online]. [cit. 2021-5-22]. Dostupné z: http://www.cncshop.cz/s3806-rotacni-inkrementalni-enkoder

[39] AN032: TMC4361A closed-loop motor control for stepper motor drivers. *TRINAMIC* [online]. 10.3.2020 [cit. 2021-5-24]. Dostupné z: https://www.trinamic.com/products/integrated-circuits/details/tmc4361a-la/

SEZNAM SYMBOLŮ A ZKRATEK

CoE	Can Appliaction Layer over EtherCat
EEPROM	Electrically Erasable Programmable Read-Only Memory
EoE	Ethernet over EtherCat
ESC	EtherCat Slave Controller
ESI	EtherCat Slave Information
ESM	EtherCat State Machine
FCS	Frame Check Sequence
FEKT	Fakulta elektrotechniky a komunikačních technologií
FIFO	First in, First out
FMMU	FieldBus Memory Mangement Unit
FoE	File Acess over EtherCat
HW	Hardware
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
LED	Light-Emitting Diode
MAC	Media Acces Control
MCU	Microcontroller Unit
MII	Media-Independent Interface
PC	Personal Computer
PDO	Process Data Object
PHY	Physical layer
PLC	Programmable Logic Controller
PPS	Pulse Per Second
SDO	Service Data Object
SoE	Servo Profile over EtherCat
SPI	Serial Peripheral Interface
ST	Structured text
SW	Software
UML	Unified Modeling Language
USB	Universal Serial Bus
VUT	Vysoké učení technické v Brně
XML	Extensible Markup Language

SEZNAM PŘÍLOH

PŘÍLOHA A – ELEKTRICKÁ SCHÉMATA A PLOŠNÉ SPOJE **99**

PŘÍLOHA B – FOTO ZHOTOVENÉHO PROTOTYPU **103**

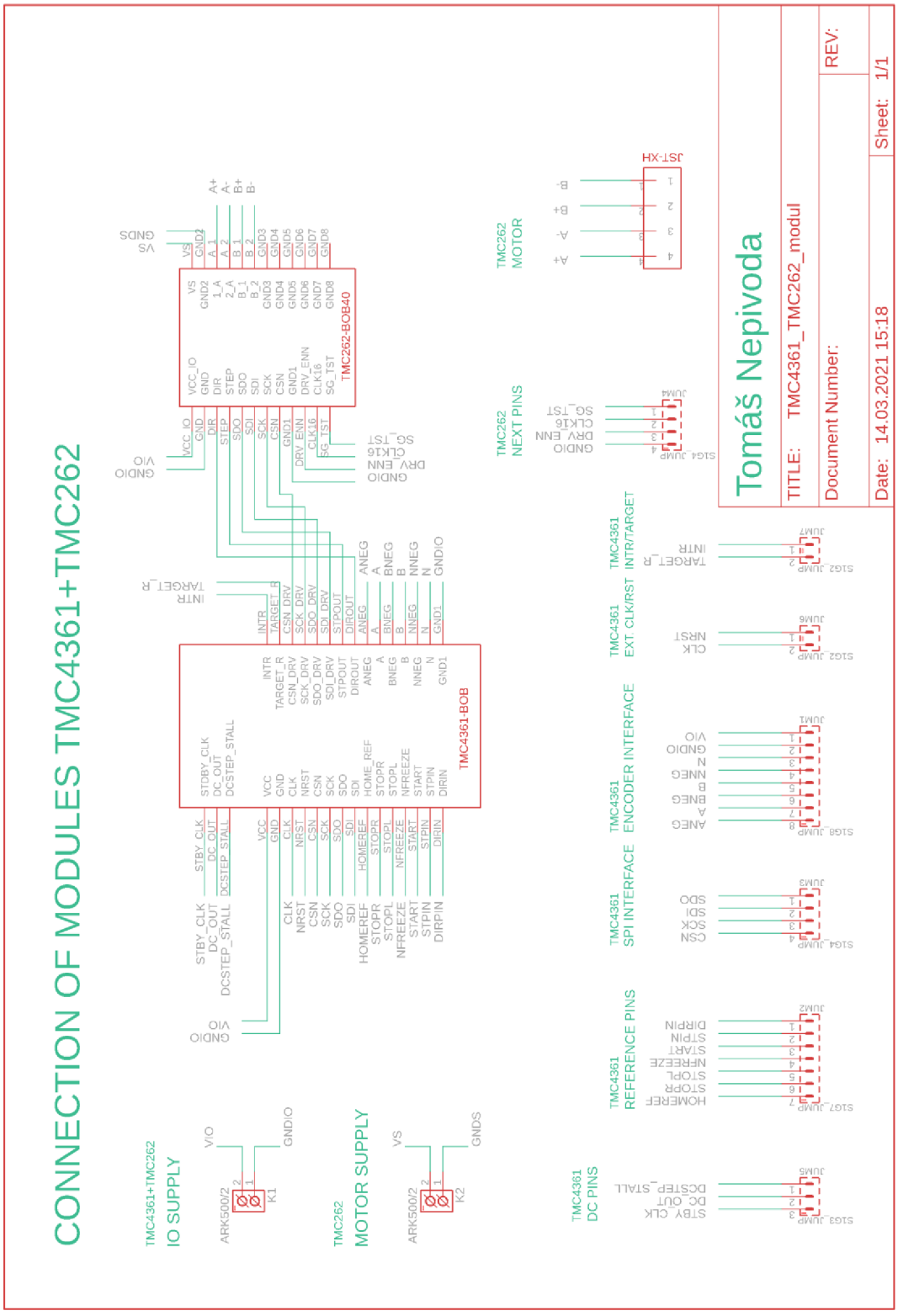
PŘÍLOHA C – MĚŘENÍ SKUTEČNÝCH OTÁČEK **104**

PŘÍLOHA D – FORMÁT PŘENÁŠENÝCH PROCESNÍCH DAT **105**

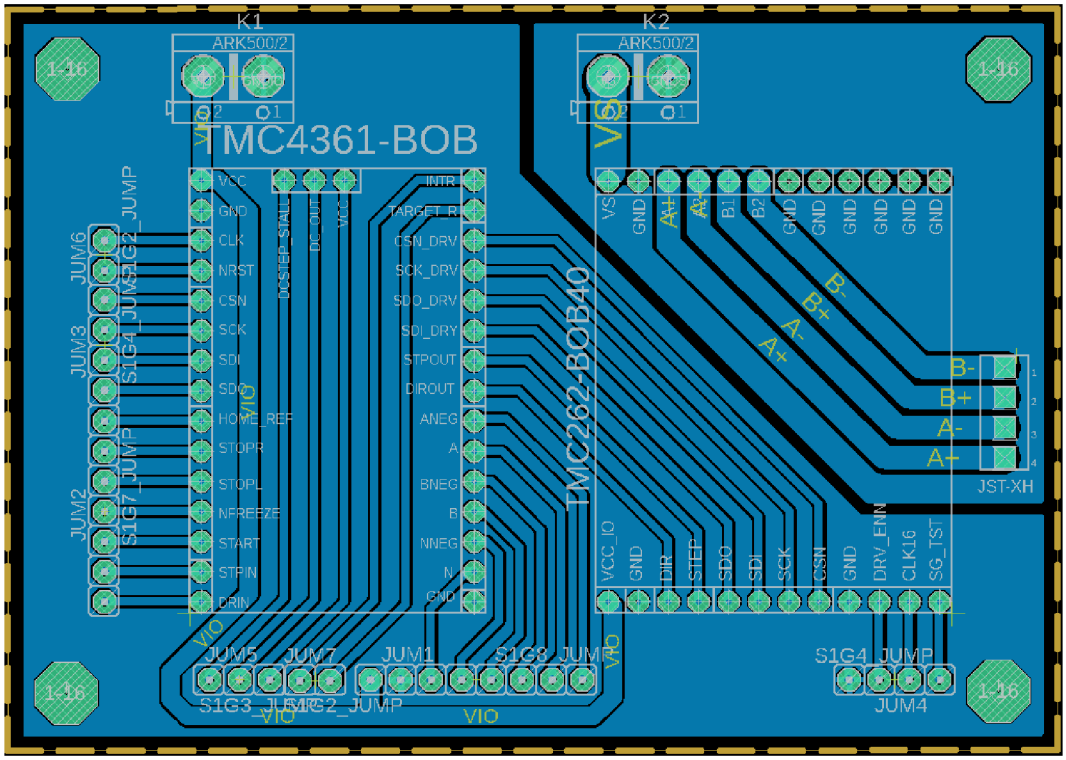
PŘÍLOHA E – SEZNAM PŘÍLOH NA PŘILOŽENÉM CD **107**

Příloha A - Elektrická schémata a plošné spoje

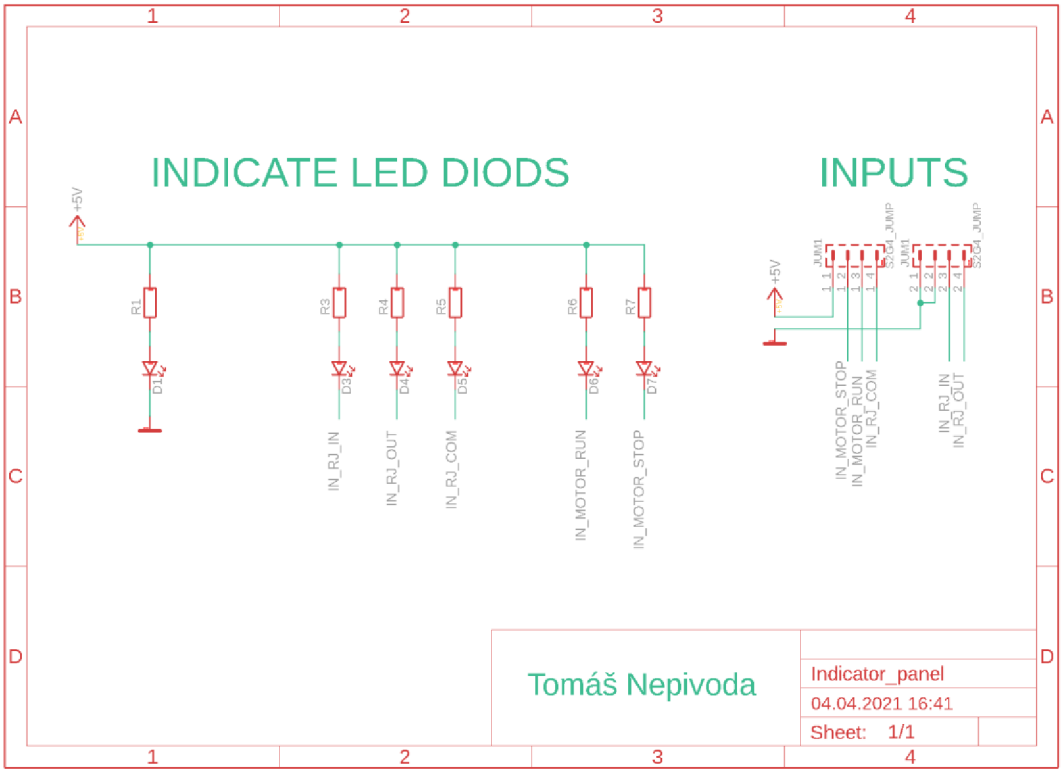
A.1 Schéma propojení TMC modulů



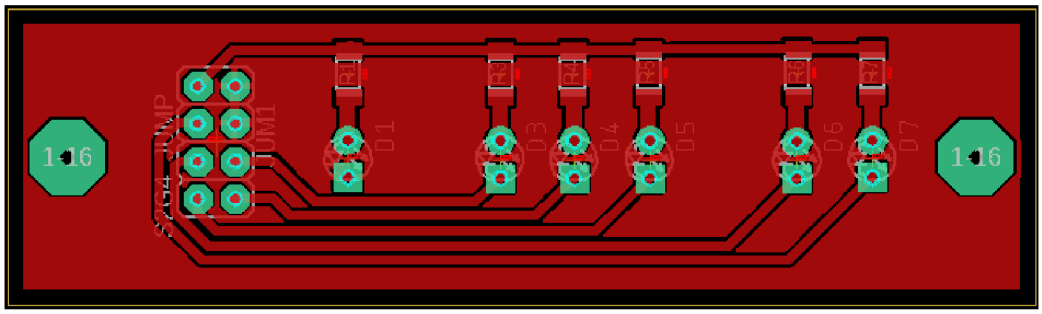
A.2 Plošný spoj TMC modulů



A.3 Elektrické schéma indikačního panelu



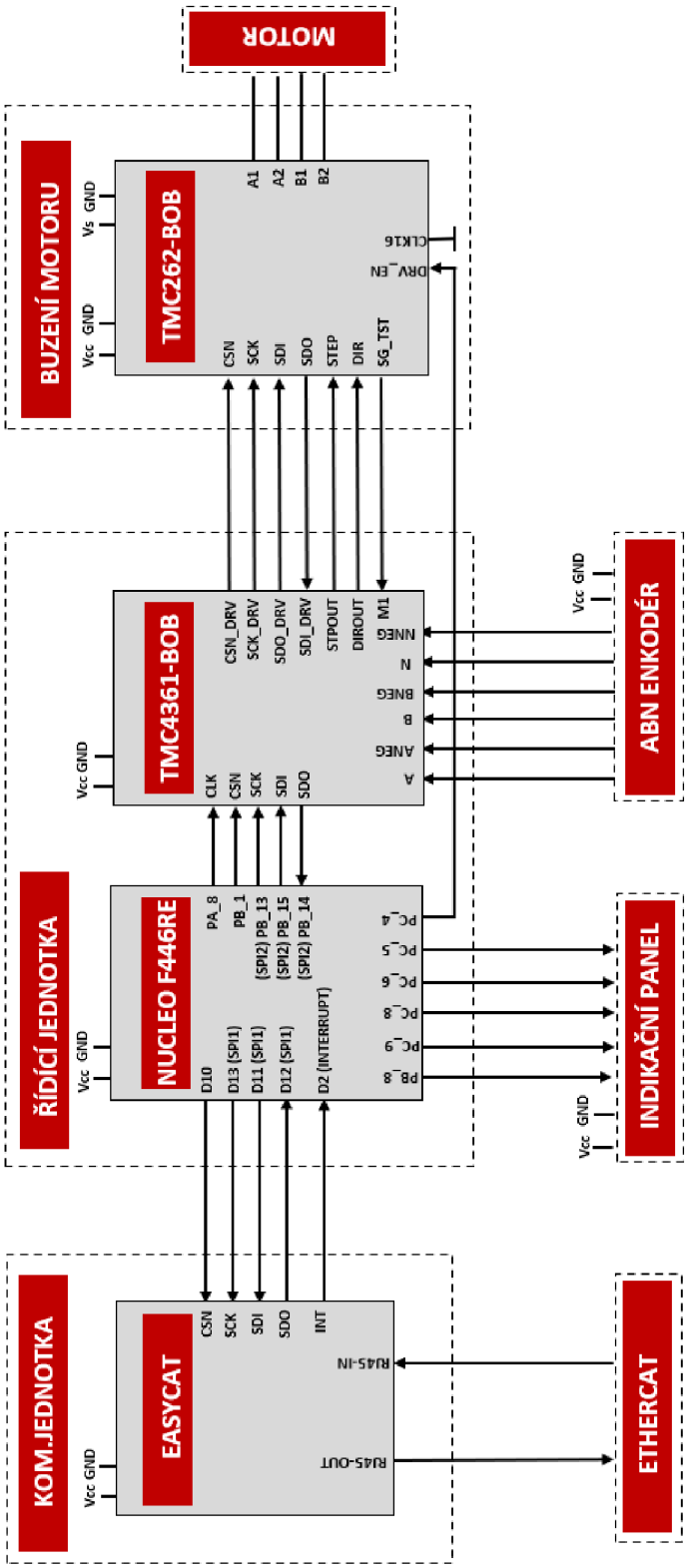
A.4 Plošný spoj indikačního panelu



A.5 Seznam použitých součástek

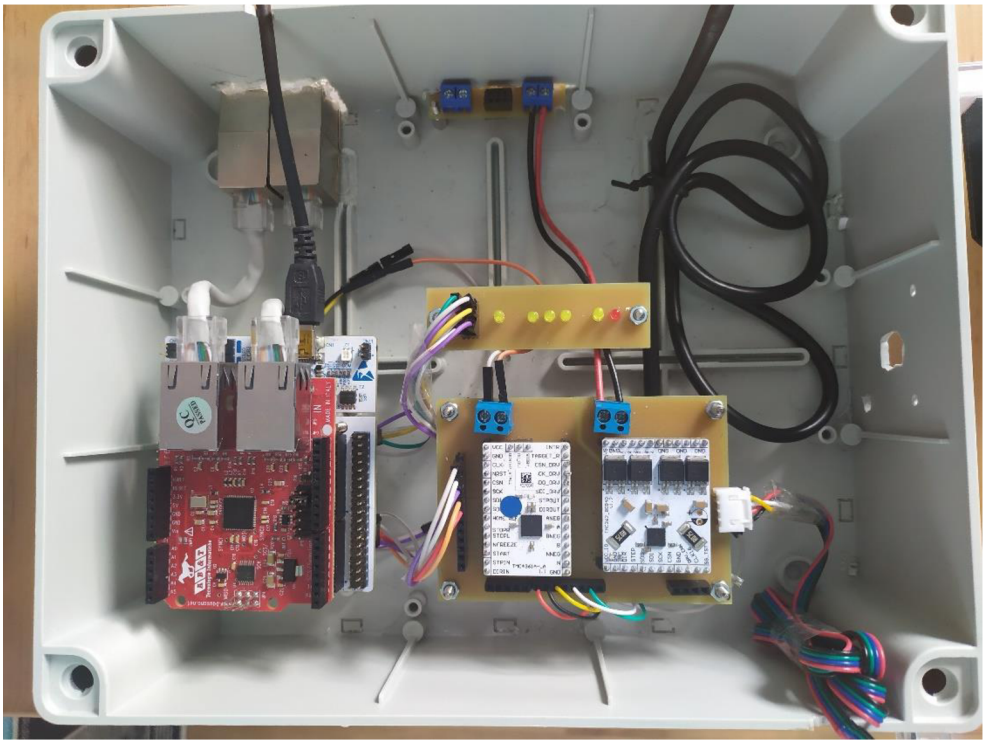
P.	Název	Počet	Odkaz
1	TMC4361A-BOB	1x	https://1url.cz/aKi57
2	TMC262-BOB40	1x	https://1url.cz/IKibM
3	EasyCat shield	1x	https://1url.cz/4Ki5c
4	NUCLEO-F446RE	1x	https://1url.cz/vKibK
5	Svorkovnice KF301-2P	4x	https://1url.cz/8KwN7
6	Dutinková lišta 20 pin, 2,54 mm	2x	https://1url.cz/pKwHL
7	Konektor JST-XH, 2,54 mm	1x	https://1url.cz/mKwHt
8	Rezistor, SMD, 220R	6x	https://1url.cz/bKwHe
9	Led dioda, 3mm	6x	https://1url.cz/LKwHz
10	Distanční sloupek	9x	https://1url.cz/KKwHK
11	Krokový motor, HS2408S	1x	https://1url.cz/EKwHr
12	ABN enkodér, S3806	1x	https://1url.cz/0KwHu
13	Mechanická spojka, 5 na 6 mm	1x	https://1url.cz/YKwH1
14	Krabička S-BOX516P	1x	https://1url.cz/ZKwHJ

A.6 Schéma propojení všech HW komponent

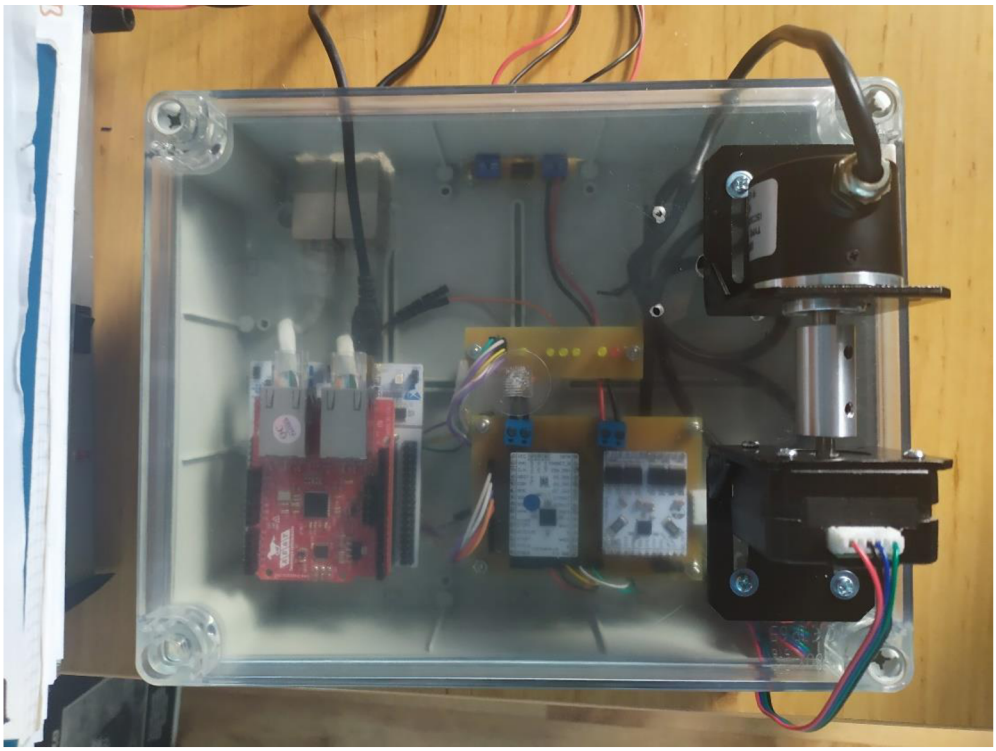


Příloha B - Foto zhotoveného prototypu driveru

B.1 Vnitřní pohled



B.2 Vnější pohled



Příloha C - Měření skutečných otáček

C.1 Naměřené a vypočtené hodnoty

n_N [ot/s]	1	2	3	4	5	6	7	8	9	10
f [kHz]	1,01	2,06	3,05	4,17	5,10	6,02	7,14	8,20	9,09	10,20
n_s [ot/s]	1,01	2,06	3,05	4,17	5,10	6,02	7,14	8,20	9,09	10,20
δ_r [%]	-1,00	-3,00	-1,66	-4,25	-2,00	-0,33	-2,00	-2,50	-1,00	-2,00
n_s [ot/s]	11	12	13	14	15	16	17	18	19	20
f [kHz]	11,10	12,20	13,10	14,30	15,10	16,10	17,20	18,50	19,50	20,50
n_n [ot/s]	11,10	12,20	13,10	14,30	15,10	16,10	17,20	18,50	19,50	20,50
δ_r [%]	-0,90	-1,60	-0,76	-2,14	-0,66	-0,62	-1,17	-2,77	-2,63	-2,5

Výpočet skutečných otáček:

$$n_s = \frac{f}{Rozlišení\ enkodéru} = \frac{6020}{1000} = 6{,}02\ \left[\frac{ot}{s}\right]$$

Výpočet procentní chyby:

$$\delta_r = \frac{n_N - n_s}{n_N}.100 = \frac{6 - 6{,}02}{6} = -0{,}33\ [\%]$$

Použitý osciloskop pro měření:

1) Výrobce – RIGOL
2) Typ - DS1054

Příloha D - Formát přenášených procesních dat

• Přijímaná data od master zařízení | Volné pozice pro další rozšiřování

BYTE	MODE1	MODE2	MODE3	MODE4
31	Provozní data [7: Komunikace 4:Reset chyb 3:Start 2:Stop 0: Směr]			
30	Zvolený režim			
29	Mikrokrokování	Typ rampy	Typ řízení	Adresa registru
28	Rychlost	Rychlost	Rychlost	Hodnota registru
27	-	VSTART (SixPoint Ramp) BOW14 (S-shaped)	Poloha	
26	-		Poloměr převodu	
25	-		Max. poloha levá	
24	-		Max. poloha pravá	-
23	-	VSTART (SixPoint Ramp) BOW14 (S-shaped)	Maximální odchylka	-
22	-			-
21	-			-
20	-			-
19	-	ASTART	Rozlišení ABN enkodéru	-
18	-			-
17	-		-	-
16	-		-	-
15	-	DFINAL	-	-
14	-		-	-
13	-		-	-
12	-		-	-
11	-	AMAX	-	-
10	-		-	-
9	-		-	-
8	-		-	-
7	-	DMAX	-	-
6	-		-	-
5	-		-	-
4	-		-	-
3	-	VBREAK	-	-
2	-		-	-
1	-		-	-
0	-		-	-

• Odesílaná data do master zařízení | Volné pozice pro další rozšiřování

BYTE	MODE1	MODE2	MODE3	MODE4
31	Provozní data [7:Komunikace 6: 0: Chyba]			
30	Kód chyby			
29	-			Hodnota registru
28	Rychlost			
27	-		Poloha	
26	Aktuální proud cívkou A			
25				-
24	Aktuální proud cívkou B			-
23				-
22	-	-	-	-
21	-	-	-	-
20	-	-	-	-
19	-	-	-	-
18	-	-	-	-
17	-	-	-	-
16	-	-	-	-
15	-	-	-	-
14	-	-	-	-
13	-	-	-	-
12	-	-	-	-
11	-	-	-	-
10	-	-	-	-
9	-	-	-	-
8	-	-	-	-
7	-	-	-	-
6	-	-	-	-
5	-	-	-	-
4	-	-	-	-
3	-	-	-	-
2	-	-	-	-
1	-	-	-	-
0	-	-	-	-

Příloha E - Seznam příloh na přiloženém CD

SLOŽKA: 1_Textové_soubory

└── **SOUBOR:** Bakalářská_práce.pdf

SLOŽKA: 2_Řídící_firmware

└── **SLOŽKA:** BP_DRV_ETH

SLOŽKA: 3_Návrhy_plošných spojů

└── **SOUBOR:** TMC4361_TMC262_module.sch
SOUBOR: TMC4361_TMC262_module.brd
SOUBOR: Indicator_panel.sch
SOUBOR: Indicator_panel.brd

SLOŽKA: 4_EtherCat_master

└── **SOUBOR:** BP_EtherCat_master.project
SOUBOR: EasyCat.xml

SLOŽKA: 5_Naměřená data

└── **SOUBOR:** 1_EtherCat_komunikace_WireShark
SLOŽKA: 2_Mereni_otacek_zaznamy_osciloskop

SLOŽKA: 6_Multimédia

└── **SLOŽKA:** 1_Fotodokumentace
SLOŽKA: 2_Demonstrační video