VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

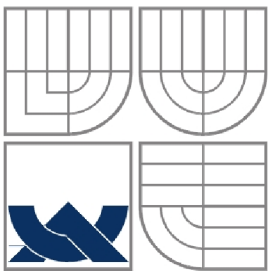
SYNCHRONIZACE A ŘÍZENÍ POHONŮ S VYUŽITÍM SBĚRNICE CAN

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

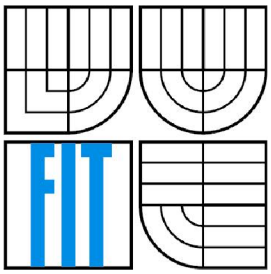
AUTOR PRÁCE
AUTHOR

LADISLAV DOBROVSKÝ

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SYNCHRONIZACE A ŘÍZENÍ POHONŮ S VYUŽITÍM SBĚRNICE CAN

SYNCHRONIZATION AND CONTROL OF ACTUATORS USING CAN BUS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LADISLAV DOBROVSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Miloš Eysselt, CSc.

BRNO 2010

Abstrakt

Systém pro ovládání pohonů po CAN sběrnici s USB-CAN převodníkem. Software je rozdělen do dvou částí, na serverovou aplikaci, která zpřístupňuje sběrnici klientským aplikacím prostřednictvím služby na schránkách (socket), protože hardwarový USB-CAN převodník fy IMF software nemohou jinak aplikace sdílet, a na aplikace ovládající po CAN různá zařízení. Je implementováno ovládání pozičního motoru IclA D065 fy Berger Lahr v aplikaci 3 osého robota, kde 3 motory pohybují každý s jedním vícesměrovým kolem fy Interroll. Tato aplikace má grafické uživatelské rozhraní s využitím knihovny GTK+. Praktická část práce vznikla na Odboru aplikované informatiky, ÚAI, FSI, VUT v Brně pod vedním konzultanta Ing. Radomila Matouška, Ph.D.

Abstract

System for motion control via CAN-Bus with USB-CAN converter. Software is divided to two parts. First is a server application that enables client appliactions to use CAN-Bus as a service available on sockets, because the IMF software's USB-CAN hardware converter can't be accessed directly by more applications. Client applications control different devices connected to CAN-Bus. Motion control using actuator Berger Lahr's IclA D065 in application of 3 axes omni-wheel robot has been impemented. Interroll's omni-wheels are used. This application has a graphical user interface done with the GTK+ library. The practical part of work has been developed in Dept. of Applied Computer Science, FME, Brno University of Technology under supervision of Ing. Radomil Matousek, Ph.D.

Klíčová slova

CAN sběrnice, CANopen, poziční řízení, synchronizace, řízení pohybu

Keywords

CAN bus, CANopen, actuator, synchronization, motion control

Citace

Ladislav Dobrovský: Synchronizace a řízení pohonů s využitím sběrnice CAN, bakalářská práce, Brno, FIT VUT v Brně, 2010

Synchronizace a řízení pohonů s využitím sběrnice CAN

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Miloše Eysselta, CSc .
Další informace mi poskytl Ing. Radomil Matoušek, PhD.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ladislav Dobrovský
19.5.2010

Poděkování

Tímto děkuji Ing. Radomilovi Matouškovi, PhD., že mi poskytl velké množství informací, volnost řešení a prostor k realizaci práce při ovládání fyzicky přítomných elektromechanických zařízení.

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

V průmyslové praxi se často setkáváme s problémy synchronizace pohonů zařízení, které ji vyžadují pro svoji funkčnost. Patří mezi ně různé výrobní a dopravní linky, výtahy (pokud používají více pohonů - elektronická hřídel) a v neposlední řadě aplikace z oblasti robotiky.

Práce je rozdělena prakticky do šesti kapitol, které popisují problematiku a navrhované řešení, implementací se zabývá kapitola poslední.

V kapitole 2 je popsán použitý standard průmyslové sběrnice CAN, která klade velký důraz na spolehlivost přenosu a synchronizaci v reálném čase. Nad CAN sběrnicí se používá aplikační protokol CANopen, který definuje komunikační profily různých typů zařízení.

Kapitola 3 se zabývá spojením mezi počítačem a CAN sběrnicí, které řeší v této práci hardwarový USB-CAN převodník firmy IMF soft. Ten se skládá z několika spolupracujících chipů a většinu své činnosti zajišťuje autonomně a šetří tak zatížení CPU počítače.

Kapitola 4 je zaměřena na ovládání a činnost pohonné jednotky IclA D065 firmy Berger Lahr. Jedná se o pohon s polohovým řízením (actuator) s vlastní elektronikou, která se ovládá relativně vysokoúrovňově protokolem CANopen v profilu DS-402. Elektronika ze zadaných dat automaticky reguluje procházející proud a tím točivé magnetické pole podle mechanického zatížení, které je na výstupní hřídeli.

Kapitola 5 uvádí problematiku integrace CAN sběrnice do aplikačního prostředí operačního systému. Pro efektivní a škálovatelné propojení převodníku s koncovými aplikacemi jsem navrhl použití modelu klient-server s vlastním protokolem, který reflektuje „filozofii“ CAN sběrnice a zároveň běžné chování převodníku z hlediska zařízení připojeného na port USB.

Kapitola 6 obsahuje praktické řešení koncové aplikace tříosého všesměrového robotu využívajícího trojici pohonů IclA D065. Je popsána základní kinematika rotace na místě a posunu přímým směrem. Související aplikace s grafickým uživatelským rozhraním ovládá robota umožňuje definovat cestu, kterou má robot projet.

Poslední kapitola popisuje implementaci ovládání USB-CAN převodníku IMF soft, CAN serveru a klientů a jejich protokolu, abstrakci IclA D065 a konečné aplikace s grafickým uživatelským rozhraním. To za použití programovacího jazyka C/C++a knihoven FTDI D2XX a GTK+ v prostředí operačního systému Linux.

2 Sběrnice a protokoly

2.1 Sběrnice CAN

Controller-area network (CAN) je standard průmyslové sběrnice. Je navržená tak, aby zařízení mohli mezi sebou komunikovat bez potřeby arbitra. Je užívána v distribuovaných řídících aplikacích, které běží v reálném čase. Původně byla CAN sběrnice vyvinuta pro potřeby automobilového průmyslu, ale rozšířila se i do průmyslové automatizace. Zprávy se po CAN posílají v rámcích, ale není zde podle OSI žádná transportní vrstva, protože realtimové aplikace vyžadují proud krátkých zpráv a nepotřebují zprávy dělit do více rámců.

Je snaha o možnost priority zpráv, možnosti konfigurace, konzistence dat v systému a zaručení zpoždění přenosu. Pro potřeby synchronizace zařízení zvládá CAN příjem dat všemi zařízeními se synchronizací času. Každé zařízení měří čas samo a k synchronizaci dochází vždy v první části přenosu bitu, viz časování bitu.

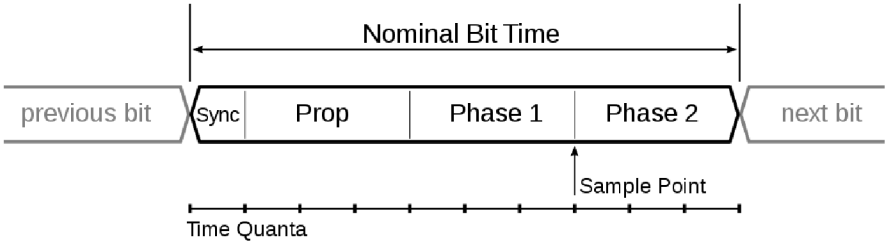
Tabulka 1: Vrstvená struktura CAN uzlu (dle [1])

Aplikační vrstva (CANopen) (v OSI L7)
Objektová vrstva (v OSI L2) • filtrování zpráv • zprávové a stavové ovládání
Přenosová vrstva(v OSI L2) • odstínění poruch • detekce chyb a signalizace • validace zpráv • potvrzování • rozhodování – arbitrace • rámcování zpráv • přenosová rychlost a časování
Fyzická vrstva (v OSI L1) • úrovně signálu a reprezentace bitu • přenosové médium

2.1.1 Fyzická vrstva

Informace se přenáší po dvojici vodičů terminovaných na obou koncích 120 kΩ rezistory. Hodnota bitu se odvozuje z rozdílu napětí na vodičích.
Přenos bitu se dělí na časová kvanta, která tvoří:

- Sync segment – využívá se pro synchronizaci, očekává se v něm změna hrany
- Prop segment – kompenzuje fyzické zpoždění sítě
- Phase 1,2 segment – kompenzují chyby fáze hrany, můžou být prodlouženy nebo zkráceny resynchronizací
- Sample point – okamžik vzorkování hodnoty bitu, hodnota zde se bere jako platná



Obrázek 1: Časování přenosu bitu a bod vzorkování

Synchroizace bitu

Tvrdá synchronizace – restartuje se interní časování bitu se sync segmentem. Tedy donutí hranu, která tvrdou synchronizaci způsobila, ležet uvnitř sync segmentu restartovaného časování.

Šířka resynchonizačního skoku – výsledkem resynchronizace může být Phase 1 segment prodloužen nebo Phase 2 segment zkrácen. Velikost prodloužení a zkrácení je limitovaná a měla by být programovatelná mezi 1 a min(4, Phase 1 segment). Informacem o časování by se měla odvodit ze změny hodnoty jednoho bitu na druhou. Toho se dosahuje tím, že je omezené množství bezprostředně následujících bitů se stejnou hodnotou.

Chyba fáze hrany – je pozice hrany vztažené ku sync segmentu a je měřená v časových kvantech. Znaménko chyby fáze je definováno:

- e = 0 pokud hrana leží v sync segmentu
- e > 0 pokud hrana leží před bodem vzorkování hodnoty bitu
- e < 0 pokud hrana leží za bodem vzorkování hodnoty bitu

Resynchronizace – projevuje se stejně jako tvrdá synchronizace, pokud je velikost chyby fáze hrany, která resynchronizaci způsobila, menší nebo rovna nastavené hodnotě šířky resynchonizačního skoku. Pokud je chyba fáze větší:

- a zároveň je chyba fáze kladná, pak Phase 1 segment se prodlouží o velikost šířky skoku
- a zároveň je chyba fáze záporná, pak Phase 2 segment se zkrátí o velikost šířky skoku

Pravidla synchroizace

Tvrdá synchronizace i resynchonizace jsou dvě formy synchronizačního procesu, které se podřizují těmto pravidlům:

1. pouze jedna synchroizace za jednu dobu bitu je dovolena
2. hrana je použita pro synchronizaci pouze, pokud se liší hodnota od předchozího bitu
3. tvrdá synchronizace je prováděna kdykoliv je změna hrany a sběrnice ve stavu čekání
4. všechny ostatní náběžné hrany spadající pod pravidla 1 a 2 se použijí k resynchonizaci s výjimkou uzlu vysílajícího dominantní bit.

Přenosová rychlost

Maximální přenosová rychlost je omezená délkou vodičů, aby byla zaručena doba odezvy a synchronizace přenosu bitu. Užívaná přenosová rychlost musí být u všech zařízení na sběrnici nastavená shodná, jinak je generována chyba CAN sběrnice zařízením, které to rozpozná.

Tabulka 2: Vztah přenosové rychlosti, délky vodičů a časování bitu

Přenosová rychlost Délka vodiče	**Nominální čas bitu t_b**	**Počet časových kvant bitu**	**Délka časového kvanta t_q**	**Poloha bodu vzorkování**
1 Mbit/s 25 m	1 µs	8	125 ns	6 t_q (750 ns)
800 kbit/s 50 m	1,25 µs	10	125 ns	8 t_q (1 µs)
500 kbit/s 100 m	2 µs	16	125 ns	14 t_q (1,75 µs)
250 kbit/s 250 m	4 µs	16	250 ns	14 t_q (3,5 µs)
125 kbit/s 500 m	8 µs	16	500 ns	14 t_q (7 µs)
50 kbit/s 1000 m	20 µs	16	1,25 µs	14 t_q (17,5 µs)
20 kbit/s 2500 m	50 µs	16	3,125 µs	14 t_q (43,75 µs)
10 kbit/s 5000m	100 µs	16	6,25 µs	14 t_q (87,5 µs)

2.1.2 Linková vrstva

Rámce obsahují 11-bitové COB-ID a maximálně 8 bytů uživatelských dat. Zbytek rámce tvoří bity řídící přenos a 15-bitové CRC (Tabulka 3). Rozsah COB-ID, kdy prvních 7 bitů je nastaveno na 1 jsou zakázány.

COB-ID přímo určuje prioritu zpráv, tedy pokud dvě zařízení chtějí ve stejném okamžiku vysílat, začne přenos to, jehož zpráva má nižší COB-ID.

Rámce mohou být:

- datové (data) – přenášejí uživatelská data
- vzdálené žádosti (remote) – žádost konzumenta o odeslání dat s daným COB-ID producentem, spíše se užívá aplikačního protokolu nad datovými rámci jako je SDO v CANopen
- chybové (error) – přítomnost chyby na sběrnici
- přetížení (overload) – nebyla dodržena dostatešná prodleva datových rámců
- mezirámcová výplň (interframe space) – odděluje datové rámce nebo rámce vzdálené žádosti

Tabulka 3: Formát datového rámce CAN 2.0A

Název části	Délka	Význam
Start-of-frame	1 bit	Určuje začátek rámce
Identifier	11 bitů	(Unikátní) identifikátor dat
Remote transmission request (RTR)	1 bit	Využíván k získání dat pokud není použit aplikační protokol jako je CANopen, běžně nastaven na 0
Identifier extension bit (IDE)	1 bit	Je nastaven na 0.
Reserved bit (r0)	1 bit	Vyhrazený bit, musí se nastavit na 0, ale příjemce akceptuje i hodnotu 1.
Data length code (DLC)	4 bity	Určuje délku přenášených uživatelských dat
Data field	0 až 8 bytů	Přenášená data, velikost určená DLC
CRC	15 bitů	Cyklický redundantní součet, ktd
CRC deliminier	1 bit	Oddělovač CRC, hodnota 1
ACK slot	1 bit	Potvrzování zprávy, vysílač nastavuje 1, přijímače můžou prohlásit nepřijetí a nastavit 0
ACK delimiter	1 bit	Oddělovač ACK, hodnota 1
End-of-frame (EOF)	7 bitů	Hodnota všech bitů 1

2.2 Protokol CANopen

CANopen je komunikační protokol a specifikace profilů zařízení pro vestavěné systémy v automatizaci. Podle klasifikace ISO/OSI modelu definuje síťovou vrstvu (L3) a transportní vrstvu (L4) pro SDO protokol. Většinou se používá nad sběrnicí CAN, přestože je možné jeho profily zařízení implementovat i nad komunikačními technologiemi jako Ethernet Powerlink nebo EtherCAT.

CANopen určuje význam COB-ID u CAN zpráv a formát dat v rámcích. Pro zařízení definuje slovník objektů s parametry a informacemi o zařízení. Obsah slovníku se liší podle CANopen profilu zařízení a významu objektů rozšíření výrobce.

Význam objektů slovníku se ukládá v souborech Electronic Data Sheet (EDS), které mají buď strukturu ini souborů nebo od roku 2007 používají formát XML.

Každý objekt ve slovníku má datový typ, povolený rozsah hodnot, výchozí hodnotu, práva pro zápis a čtení a atribut, který určuje jestli se uchovává hodnota po vypnutí zařízení. Podle rozsahů COB-ID se zprávy dělí na různé typy, dále zpracovávané protokoly viz Tabulka 4.

Vícebytové datové typy jsou ukládány ve formátu little endian – nejvýznamnější byte je přenášen jako první.

Tabulka 4: Dělení zpráv v CANopen podle COB-ID na protokoly (dle [9])

Komunikační objekt	Funkční kód	Adresa uzlu (node–ID)	Celé COB-ID	Související objekty ve slovníku
NMT start/stop	0 0 0 0	0 0 0 0 0 0 0	0	
SYNC objekt	0 0 0 1	0 0 0 0 0 0 0	128 (80h)	1005h … 1007h
EMCY objekt	0 0 0 1	x x x x x x x	128 (80h) + node–ID	1014h
T_PDO1	0 0 1 1	x x x x x x x	384 (180h) + node–ID	1800h
R_PDO1	0 1 0 0	x x x x x x x	512 (200h) + node–ID	1400h
T_PDO2	0 1 0 1	x x x x x x x	540 (280h) + node–ID	1801h
R_PDO2	0 1 1 0	x x x x x x x	768 (300h) + node–ID	1401h
T_SDO	1 0 1 1	x x x x x x x	1408 (580h) + node–ID	1200h
R_SDO	1 1 0 0	x x x x x x x	1536 (600h) + node–ID	1200h
NMT řízení chyb	1 1 1 0	x x x x x x x	1792 (700h) + node–ID	100Ch...100Eh
LMT služby	1 1 1 1	1 1 0 0 1 0 x	2020 (7E4h), 2021 (7E5h)	
NMT identifikační služby	1 1 1 1	1 1 0 0 1 1 0	2022 (7E6h)	
DBT služby	1 1 1 1	1 1 0 0 x x x	2023 (7E7h), 2024 (7F8h)	
NMT služby	1 1 1 1	1 1 0 1 0 0 x	2025 (7E9h), 2026 (7EAh)	

2.2.1 Network Management (NMT) protocol

Hlásí bootování zařízení a nastavuje základní stavy jako aktivace a obecné pozastavení činnosti a reakce na zprávy dalších protokolů v CANopen.

Definuje také Heartbeat protocol, kdy podřízené zařízení periodicky generuje zprávy se svým stavem, aby se zajistilo zjištění zda je zařízení stále připojeno.

Zařízení (uzel) může být ve stavech:

- initialising: pouze boot-up zpráva; přechází do stavu pre-operational
- pre-operational: povolena komunikace NMT, SDO, EMCY
- stopped: povolena pouze komunikace NMT
- operational: povolena komuniace NMT, SDO, PDO, SYNC, EMCY

2.2.2 Layer Management (LMT) protocol

Zajišťuje nastavení NMT identifikátorů zařízení, změnu časování bitů a přenosové rychlosti. To se provádí buď v globálním režimu, kdy na sběrnici smí být připojeno pouze jedno zařízení, které se nastaví daty odeslanými po sběrnici. Nebo v selekčním režimu, kdy se nejprve vybere zařízení pomocí sériového čísla nebo názvu produktu nebo názvu výrobce a pak se nastavují parametry, které ostatní zařízení ignorují.

Zařízení zde jsou ve vztahu master-slave. Nejprve se tedy vybere zařízení, pak se odešlou nastavení, která chce uživatel změnit a nakonec se požádá o uložení a použití změněných hodnot. LMT zpráva má vždy velikost dat 8 bytů, ale ne všechny nesou užitečnou informaci.

2.2.3 Service Data Object (SDO) protocol

Používá se k nastavení a čtení hodnot ze slovníku objektů. Zápis vždy provází potvrzení o úspěšnosti s případným chybovým kódem. Objekty jsou ve slovníku vždy adresovány indexem a pod-indexem, takto může objekt být strukturou nebo polem. Ve zprávě SDO se odesílá index, podindex a kód, který určuje jestli se data čtou nebo zapisují a kolik bytů se očekává respektive posílá (viz tabulka 5). V poli command code (tabulka 7) je dána velikost dat (užitečná část, SDO zpráva má vždy velikost plné CAN zprávy), zda jde o žádost o zápis či čtení a v odpovědi určuje úspěch nebo chybu. U objektů delších než 4 byte se zbytek dat odesílá v rozšířeném formátu (tabulka 6), kdy další zpráva má na prvním byte command code (tabulka 7) a může následovat až 7 byte.

Tabulka 5: Formát SDO zpráv, objekty s velikostí do 4 byte (dle [9])

Význam	**COB-ID**	**Command code**	**Index**	**Pod-index**	**data**
Velikost	11 bitů	1 byte	2 byte	1 byte	4 byte
Hodnota	(580h nebo 600h) + node-ID	ccd	0 - FFFFh	0h - FFh	data

Tabulka 6: Formát SDO zpráv, objekty s velikostí nad 4 byte (dle [9])

Význam	**COB-ID**	**Command code**	**Data**
Velikost	11 bitů	1 byte	7 byte
Hodnota	(580h nebo 600h) + node-ID	ccd	0 - FFFFh

Tabulka 7: SDO Command code - určuje význam zprávy (dle [9])

Typ zprávy	**Délka dat (byte)**			
	4	**3**	**2**	**1**
Žádost o zápis	23h	27h	2Bh	2Fh
Odpověď na zápis	60h	60h	60h	60h
Žádost o čtení	40h	40h	40h	40h
Odpověď na čtení	43h	47h	4Bh	4Fh
Chybová odpověď	80h	80h	80h	80h

2.2.4 Process Data Object (PDO) protocol

Zaručuje mapování objektů ze slovníku na PDO. Toto nelze u všech objektů slovníku, ty musí mapování podporovat. Lze mapovat i více objektů slovníku do jednoho PDO, pokud se vejdou do jedné zprávy. Většinou zařízení podporují maximálně 4 PDO, přičemž každé má vysílací a přijímací kanál (R_PDO, T_PDO). Na rozdíl od SDO nenásleduje hned potvrzení, ale vysílací PDO se odešle, až když se změní stav nebo v případě chyby (to obvykle následuje i EMCY). Přijímací PDO může být nastaveno na asynchronní nebo synchronní chování, kdy se čeká na SYNC objekt. Poměr přijatých a odeslaných PDO tedy nemusí být 1:1. Některá zařízení využívají PDO jako asynchronní hlášení o změně některých parametrů, například teploměr by zasílal naměřenou hodnotu (periodicky nebo se změnou stavu). Zpráva obsahuje pouze COB-ID příslušného PDO a

Jde tedy o přístup jak snížit množství komunikace na sběrnici oproti SDO až o polovinu a navíc s možností synchronizace více zařízení. Zařízení ve stavu povolené funkce (NMT operational) může tyto zprávy produkovat bez dotazu.

2.2.5 Synchronization Object (SYNC) protocol

Spolupracuje s PDO protokolem. Některé PDO zprávy můžou být nastaveny na synchronní chování a jejich význam změny stavu se provede až po synchronizaci SYNC objektem. Takto lze souběžně spouštět více nastavených zařízení a realizovat aplikace, u kterých je současné započetí pohybu kritické. Příkladem můžou být soustavy dopravníků, výtahů a výrobních zařízení, u kterých bez synchronizace může dojít k poškození zařízení a ohrožení pracovníků.

U synchronizačního objektu je třeba pamatovat na to, že je shodný pro všechny zařízení připojené na sběrnici a jestli některé z nich na něj bude reagovat aniž by to bylo původním záměrem. Při použití synchronizace v CANopen je tedy vhodné mít synchronizované zařízení na vlastních CAN-Bus okruzích. Některá zařízení umožňují změnit COB-ID používané pro SYNC objekt a tím kolizím zabránit, nicméně s tím musí aplikace počítat neboť se dostávají do rozsahu příslušícího například EMCY objektům.

2.2.6 Emergency Object (EMCY) protocol

Využívá se pro hlášení fatálních chyb z hlediska prováděné operace nebo selhání zařízení. Chyby jsou rozdělené na třídy podle závažnosti. K chybovému kódu mohou být přidána další data, která ji popisují. Tyto chyby je nutné vždy zpracovat s ohledem na závažnost a nepokračovat před vyřešením problému v práci s jinými zařízeními, pokud jsou funkčně spojeny, případně může být nutné je ihned zastavit. Příkladem systémů kde hrozí poškození jsou synchronizované výrobní a dopravníkové linky. EMCY objekt bez chybového kódu je ohlášení o spuštění zařízení a jeho úspěšného připojení ke CAN sběrnici (Boot-Up message).

2.2.7 Slovník objektů CANopen

Je množina vlastností a parametrů zařízení, které jsou dostupné přes protokol SDO. Objekt ve slovníku je adresován indexem a může se skládat z jednoho či více pod-indexů. Tedy základním parametrem objektu je zda obsahuje pouze jednu hodnotu nebo více uložených v poli nebo záznamu. Datové typy samotné jsou také objekty slovníku.

Vlastnosti všech objektů slovníku jsou:

- index
- objekt (symbolické jméno) – proměnná (VAR), pole (ARRAY), záznam (RECORD)
- jméno
- datový typ (tabulka 10)
- přístup – čtení, zápis
- povinnost implementovat – každý profil zařízení definuje nutné objekty pro splnění specifikace a pak doporučené, které je vhodné implementovat u většiny zařízení

Na zařízení je uložena pouze hodnota a velikost dat přístupná přes kombinaci index a pod-index. Zbytek informací ke slovníku je k dispozici v manuálu zařízení a formou EDS souboru. Zde bývá uveden zevrubný popis funkce objektu, povolený rozsah hodnot, výchozí hodnota a zda u zapisovatelných objektů zůstává hodnota uložená po vypnutí zařízení.

Objekty ze skupiny komunikačního profilu umožňují identifikaci typu, výrobce a sériového čísla zařízení. Obsahují i další informace společné všem zařízením CANopen a kód poslední chyby, která byla odeslána jako EMCY. Výrobcem specifikované profily zpřístupňují nadstandardní nastavení nebo jiné přístupy než umožňuje profil standardizovaný. Některé zařízení jako je například IclA D065 neimplementují přímo ve slovníku datové typy.

Tabulka 8: Rozsahy indexů objektů ve slovníku CANopen (dle [9])

Rozsah indexů (hexa)	**Skupiny objektů**
0000h	vyhrazeno
0001h – 001Fh	statické datové typy
0020h – 003Fh	složené datové typy
0040h – 005Fh	datové typy specifikované výrobcem
0060h – 007Fh	statické datové typy pro profil zařízení
0080h – 009Fh	složené datové typy pro profil zařízení
00A0h – 0FFFh	vyhrazeno
1000h – 1FFFh	komunikační profil
2000h – 5FFFh	profily specifikované výrobcem
6000h – 9FFFh	standardizované profily zařízení
A000h – FFFFh	vyhrazeno

Tabulka 9: Objekt 1000h: typ zařízení (IclA D065 – actuator) (dle [9])

Popis objektu	*Index*	1000h
	Jméno objektu	Typ zařízení (device type)
	Kód objektu	Proměnná (VAR)
	Datový typ	Unsigned32 – 32 bitové bezznaménkové číslo
Popis hodnoty	*Pod-index*	00h – typ zařízení (device type)
	Význam	Typ zařízení a profil (Device type and profile)
	Přístup	Pouze pro čtení (ro)
	Hodnota	00020192h: bity 23–16 = 02h → actuator bity 15–0 = 0192h → profil DS-402 (402 = 192h)

Tabulka 10: Datové typy CANopen (dle [9])

Datový typ	**Rozsah hodnot**	**Délka dat**
Boolean	0=nepravda, 1=pravda	1 byte
Integer8	-128 … +127	1 byte
Integer16	-32768 … +32767	2 byte
Integer32	-2147483648 .. +2147483647	4 byte
Unsigned8	0 .. 255	1 byte
Unsigned16	0 .. 65535	2 byte
Unsigned32	0 .. 4294967295	4 byte
Visible String8	ASCII	8 byte
Visible String16	ASCII	16 byte

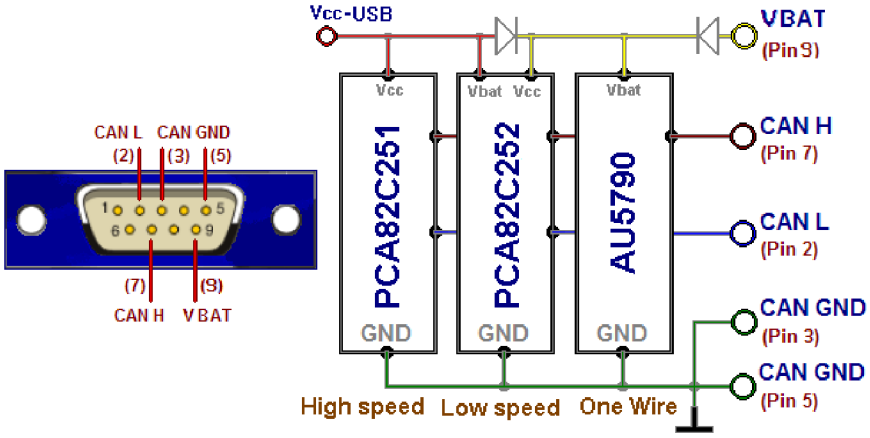
3 USB-CAN převodník fy IMFsoft

Umožňuje vysílání a přijímání zpráv na CAN sběrnici s použitím běžného PC S USB portem. Pomocí tří budičů CAN sběrnice ji umožňuje provozovat v různých režimech. Protože je primárně určen pro sledování a ladění sběrnice, jsou v procesoru převodníku přítomny funkce pro periodické nebo opožděné odesílání až 8 paralelních zpráv s rozlišením 1 ms. Implementuje vyrovnávací paměť o velikosti 256B a má tak nižší nároky na CPU PC a režii operačního systému. Čas přijetí zprávy se ukládá s rozlišením 1 ms. Nevyžaduje externí napájení, obsahuje ochranu proti přepjetí a napájení a činnost sběrnice signalizuje pomocí LED.

3.1 Hardware

Převodník je složen z komponent:

- Atmel T89C51CC01 - procesor řídící převodník s integrovanou CAN funkcionalitou
- PCA80C251 – budič CAN sběrnice – high speed (ISO11898)
- PCA80C252 – budič CAN sběrnice – low speed (ISO11519)
- AU5790 – budič CAN sběrnice – one wire (J2411)
- FTDI FT245RL - emulace sériového rozhraní přes USB

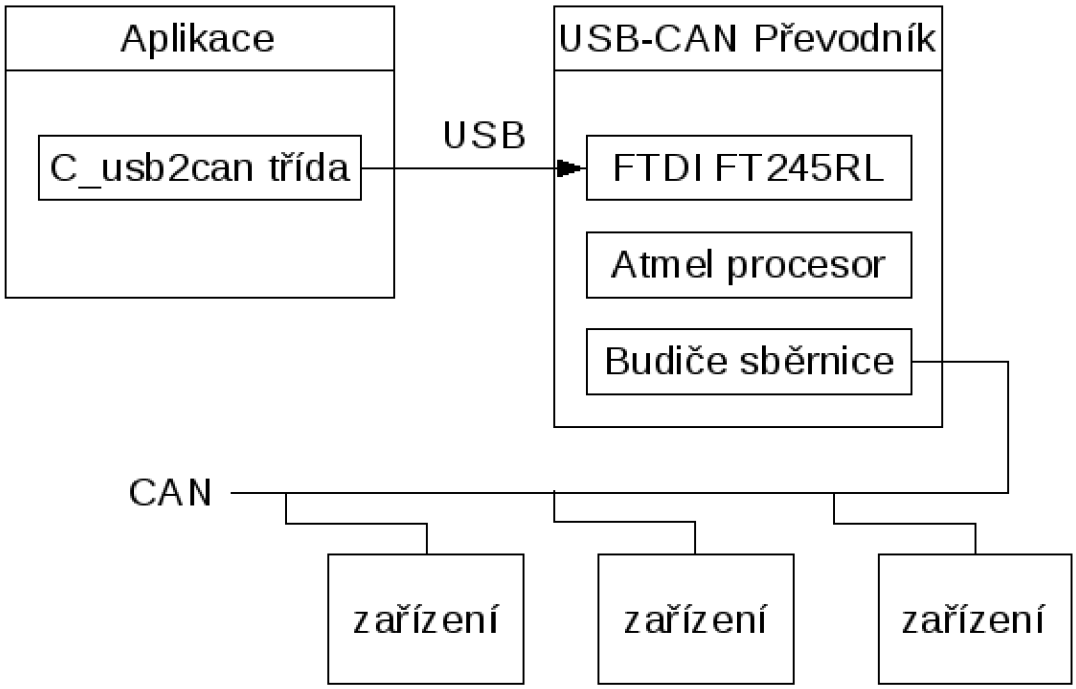


Obrázek 2: Zapojení budičů sběrnice v převodníku (dle [8])

3.2 Softwarové rozhraní

Celý převodník se ovládá přes chip FTDI FT245RL pomocí knihovny FTDI D2XX. Není tedy třeba instalace ovladače. Na převodník má takto přístup pouze jedna aplikace s přístupovými právy superuživatele. C++ třída (C_usb2can), která je abstrakcí převodníku a CAN sběrnice, má funkční kód k ovládání FTDI FT245RL a nepřímo Atmelu T89C51CC01 je založena na ukázkové aplikaci CANstart od fy IMFsoft. Ta neřeší všechny funkce CAN sběrnice, hlavně potvrzování odeslání zpráv a diagnostika chyb na sběrnici je omezená. V datech, která převodník posílá se vyhledávají užitečná data a "nezajímavé" informace se přeskakují, výrobce k nim nedodává žádnou dokumentaci a předpokládá se, že tyto funkce u převodníku sloužícího primárně k monitorování probíhajícího provozu na CAN sběrnici nejsou nezbytně nutné k implementaci. V práci s FTDI chipem byla v aplikaci výrobce chyba, což v některých situacích vedlo k nedeterminismu a ztrátě zpráv. Došlo tedy k úpravám, která zajišťují determinismus a zvyšují výkon. Pomocí dvojice převodníků byla testována konečná spolehlivost spojení.

Chip FTDI FT245RL se pod tímto označením a výrobcem hlásí na USB sběrnici. Dodavatel konečného hardwarového řešení tyto údaje může změnit. V připade USB-CAN převodníku IMF soft se starší verze hlásí jako FTDI FT245RL, ale novější už pod označením IMF soft. Pro zpětnou kompatibilitu je ponechána možnost identifikace podle pořadového čísla FTDI chipu připojeného na USB, což zabraňuje automatické detekci a klade zvýšené nároky na uživatele, který si musí být vědom používaných zařízení hlásících se jako FTDI chipy.



Obrázek 3: Užití USB-CAN převodníku aplikací (CAN_Server)

3.2.1 Zprávy ovládající převodník

Tabulka 11: Formát zprávy pro převodník

ID	START1	START2	CONTROL	LENGTH	DATA	END1	END2
Velikost	1	1	1	1	LENGTH	1	1
Hodnota	FAh	F5h	Podle příkazu	Délka dat	Podle příkazu	AHh	5Fh

Příkazy – typy zpráv

- Inicializace CAN sběrnice
  - CONTROL = 1
  - LENGTH = 6
  - DATA
    - BRP<<1 (1 byte)
    - PRS<<1 (1 byte)
    - (PHS1<<1) + (PHS2<<4) (1 byte)
    - 0Ah (1 byte)
    - 32h (1 byte)
    - 02h (1 byte)
  - koeficienty BRP, PRS, PHS1, PHS2 – viz tabulka 12

- Aktivace přijímacího zprávového centra (MC 0 – 7)
  - CONTROL = 2
  - LENGTH = 10
  - DATA:
    - číslo MC = [0 .. 7] (1 byte)
    - kódované ID=0 (4 byte)
    - maska - kódované ID=0 (4 byte)
    - CAN 2.0A resp. 2.0B : 08h resp. 18H

- Aktivace vysílacího zprávového centra (MC 8 – 15)
  - CONTROL = 3
  - LENGTH = 17
  - DATA:
    - číslo MC = [0 .. 7] (1 byte)
    - perioda [ms] – významější byte (nevyužito, posílá se 0) (1 byte)
    - perioda [ms] – nejméně významný byte (nevyužito, posílá se 0) (1 byte)
    - periodický nebo okamžitě odesílaný rámec (pouze okamžitý, hodnota 4) (1 byte)
    - kódované ID odesílané zprávy (4 byte)
    - délka dat CAN zprávy (1 byte)
    - data CAN zprávy i s případně nevyužitou kapacitou (8 byte)

- Nastavení lokálního času převodníku
  - CONTROL = 4
  - LENGTH = 5
  - DATA = hodiny, minuty, sekundy, 0, 0 (5 byte)

- Vypnutí CAN sběrnice
  - CONTROL = 6
  - LENGTH = 0

- Vypnutí přijímacího zprávového centra (MC 0 – 7)
  - CONTROL = 7
  - LENGTH = 1
  - DATA = číslo MC = [0 .. 7] (1 byte)

- Vypnutí vysílacího zprávového centra (MC 8 – 15)
  - CONTROL = 8
  - LENGTH = 1
  - DATA = číslo MC = [0 .. 7] (1 byte)
- Pauza v přenosech bez vypnutí budičů sběrnice
  - CONTROL = 10
  - LENGTH = 1
  - DATA = 0 (provoz) nebo 255 (pauza) (1 byte)

Tabulka 12: Kombinace BRP,PRS, PHS1 a PHS2 pro rychlosti CAN

Rychlost CAN	Sample point 50%	Sample point 62.5%	Sample point 75%	Sample point 87.5%
10 kb/s	49, 0, 5, 7	49, 0, 7, 5	49 ,2, 7, 3	49, 4, 7, 1
20 kb/s	24, 0, 5, 7	24, 0, 7, 5	24, 2, 7, 3	24, 4, 7, 1
33 kb/s	14, 0, 5, 7	14, 0, 7, 5	14, 2, 7, 3	14, 4, 7, 1
50 kb/s	9, 0, 5, 7	9, 0, 7, 5	9, 2, 7, 3	9, 4, 7, 1
83 kb/s	5, 0, 5, 7	5, 0, 7, 5	9, 2, 7, 3	5, 4, 7, 1
100 kb/s	4, 0, 5, 7	4, 0, 7, 5	5, 2, 7, 3	4, 4, 7, 1
125 kb/s	3, 0, 5, 7	3, 0, 7, 5	3, 2, 7, 3	3, 4, 7, 1
250 kb/s	1, 0, 5, 7	1, 0, 7, 5	1, 2, 7, 3	1, 4, 7, 1
500 kb/s	0, 0, 5, 7	0, 0, 7, 5	0, 2, 7, 3	0, 4, 7, 1
800 kb/s	0, 0, 2, 4	0, 0, 3, 3	0, 0, 4, 2	0, 4, 1, 1
1 Mb/s	0, 0, 1, 3	0, 0, 2, 2	0, 0, 3, 1	0, 3, 1, 0

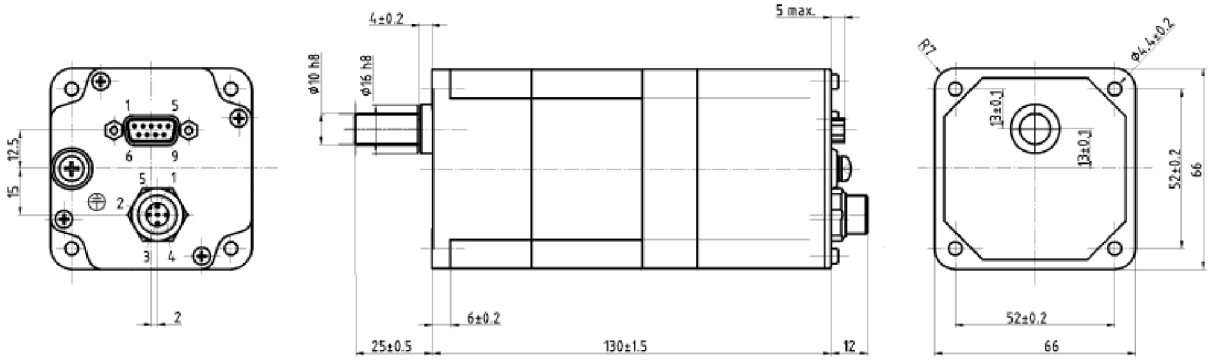
4 Polohově řízený IclA D065

IclA – Intelligence close to the Application – je řada pohonů s polohovým řízením fy Berger Lahr, které mají integrovanou ovládací elektroniku s enkodérem, vinutí motoru a převodovku v jednom. Zařízení komunikuje pomocí protokolu CANopen s profilem CiA DS-402. Ve slovníku CANopen objektů implementuje rozsahy indexů objektů pro komunikační profil, profily specifické výrobci a standardní profil (DS-402). Neobsahuje tedy ve slovníku datové typy ani další formální náležitosti celé normy. Důvodem je relativní zbytečnost a spojené náklady, když aplikace stejně bývají vázány na zařízení ve stroji použitém. Tedy běžné aplikace většinou nevyužívají obecnosti s jakou je CANopen definován.

Jsou použity 2 PDO s mapováním:

- R_PDO1 – provádí se asynchronně, zařízení přijímá objekt:
  - *Controlword* (*6040h*) [2 byte] – příkaz změny stavu
- T_PDO1 – provádí se asynchronně, zařízení vysílá objekt:
  - *Statusword* (*6041h*) [2 byte] – stav zařízení
- R_PDO2 – provádí se synchronně, zařízení přijímá objekty:
  - *Controlword* (*6040h*) [2 byte] – příkaz změny stavu
  - *Target position* (*607Ah*) [4 byte] – cílová pozice
- T_PDO2 – provádí se synchronně, zařízení vysílá objekty:
  - *Statusword* (*6041h*) [2 byte] – stav zařízení
  - *Position actual value* (*6064h*) [4 byte] – pozice zařízení.

Rozlišuje se mezi stavem (state) a operačním režimem (mode) zařízení.



Obrázek 4: Technický výkres IclA D065 DC024 (dle [10])

4.1 Operační režimy pozičního zařízení

V různých režimech reaguje zařízení jinak na stejnou změnu stavu. IclA D065 podporuje režimy:

- Manuální (specifický výrobci)– ovládání motoru digitálními vstupy na konektoru napájení
- Poziční (standardní) – ovládání pohybu přes sběrnici
- Poziční předdefinovaný (specifický výrobci a není dále popsán)
- Referenční (návratový) – ustanovení pozice referenční nuly (dimension settings)
- Konfigurační (specifický výrobci) – nastavování různých parametrů zařízení

Režim se nastavuje zápisem do objektu modes of operation (6060h) a zpětnou kontrolou přečtením objektu modes of operation display (6061h). Pro úspěšnou změnu režimu je třeba, aby zařízení bylo celkově v klidu (hřídel se neotáčí a neprovádí se změna nastaveníl).

Tabulka 13: Objekt Modes of operation (6060h) (dle [9])

Popis objektu	*Index, kód objektu*	6060h	Proměnná (VAR)	
	Jméno objektu	Žádost o nastavení režimu (modes of operation)		
Popis hodnoty	*Pod-index, typ, přístup*	00h – controlword	Unsigned16	wo
	Význam	Nastavuje režim, provedení nastavení je třeba ověřit		
	Hodnota	-128: konfigurační (specifický výrobci) -2: poziční předdefinovaný (specifický výrobci) -1: manuální (specifický výrobci) 1: poziční (standardní) 6: referenční (návratový)		

Tabulka 14: Objekt Modes of operation display (6061h) (dle [9])

Popis objektu	*Index, kód objektu*	6061h	Proměnná (VAR)	
	Jméno objektu	Nastavený režim (modes of operation display)		
Popis hodnoty	*Pod-index, typ, přístup*	00h – controlword	Unsigned16	ro
	Význam	Po pokusu o změnu režimu je třeba zde zkontrolovat úspěch		
	Hodnota	Viz hodnoty objektu 6060h (Tabulka 13)		

4.1.1 Manuální režim

Zařízení reaguje na digitální vstupy MAN_N a MAN_P, které způsobují otáčení motoru v negativním respektive pozitivním směru s parametry pohybu danými hodnotami v objektu 2011h (Tabulka 15). Před kontinuálním pohybem může nastat krok (impuls), kdy se posune motor o několik inkrementů a, až pokud signál trvá, započne se kontinuální otáčení.

Tento režim nevyžaduje žádné nastavení stavu zařízení, lze jej použít hned po zapnutí zařízení.

Tabulka 15: Objekt Manual mode settings (2011h) (dle [9])

Popis objektu	*Index, kód objektu*	2011h	Záznam (RECORD)	
	Jméno objektu	Nastavení manuálního režimu (manual mode settings)		
Popis hodnoty	*Pod-index, typ, přístup*	00h – number of elements	Unsigned32	ro
	Význam	Počet záznamů v objektu		
	Hodnota	6		
Popis hodnoty	*Pod-index, typ, přístup*	01h – increments	Unsigned16	rw
	Význam	Počet inkrementů prvního skoku		
	Hodnota	2, (rozsah [0, 2..65535])		
Popis hodnoty	*Pod-index*	02h – velocity	Unsigned32	rw
	Význam	Rychlost otáčení [inkrementy/s]		
	Hodnota	200, (rozsah 0..999)		
Popis hodnoty	*Pod-index*	03h – acceleration	Unsigned32	rw
	Význam	Zrychlení [inkrementy/s^2]		
	Hodnota	500, (rozsah 0..5000)		
Popis hodnoty	*Pod-index*	04h – deceleration	Unsigned32	rw
	Význam	Zpomalení při brždění [inkrementy/s^2]		
	Hodnota	2000, (rozsah 0..2000)		
Popis hodnoty	*Pod-index*	05h – max current	Unsigned16	rw
	Význam	Maximální poměrný proud [promile]		
	Hodnota	1000, (rozsah 0..1000)		
Popis hodnoty	*Pod-index*	06h – release time	Unsigned16	rw
	Význam	Doba tvrání pulsu [ms]		
	Hodnota	500, (rozsah 0..65535)		

4.1.2 Poziční režim

Nastavuje se cílová pozice (objekt 607Ah, tabulka 16) a rampa zrychlení (objekt 6083h, tabulka 17), limitní rychlosti (objekt 6081h, tabulka 18) a zpomalení při brždění (objekt 6084h, tabulka 19). Pro spuštění pohybu je třeba, aby zařízení bylo ve stavu „F – operation enabled“, tedy kdy je motor pod proudem a regulátor je připraven na řízení přívodu proudu vzhledem k zátížení. Konečné spuštění je nastavením bitu *CW_NEW_SETPOINT* v objektu controlword (6040h, tabulka 20). Změna pozice může být relativní k aktuální pozici nebo absolutní k referenční nule, rozhoduje o tom nastavení bitu *CW_RELATIVE* v objektu controlword (6040h, tabulka 20).

Tabulka 16: Objekt Target position (607Ah) (dle [9])

Popis objektu	*Index, kód objektu*	607Ah	Proměnná (VAR)	
	Jméno objektu	(Target position)		
Popis hodnoty	*Pod-index, typ, přístup*	00h	Integer32	rw
	Význam	Aktuální pozice natočení hřídele motoru [inkrementy]		

Tabulka 17: Objekt Profile acceleration (6083h) (dle [9])

Popis objektu	*Index, kód objektu*	6083h	Proměnná (VAR)	
	Jméno objektu	Zrychlení (Profile acceleration)		
Popis hodnoty	*Pod-index, typ, přístup*	00h	Unsigned32	rw
	Význam	Zrychlení otáček motoru[inkrementy/s^2]		
	Hodnota	500 (rozsah 0..5000)		

Tabulka 18: Objekt Profile velocity (6081h) (dle [9])

Popis objektu	*Index, kód objektu*	6081h	Proměnná (VAR)	
	Jméno objektu	Rychlost (Profile velocity)		
Popis hodnoty	*Pod-index, typ, přístup*	00h	Unsigned32	rw
	Význam	Regulovaná rychlost otáček motoru [inkrementy/s]		
	Hodnota	200 (rozsah 0..999)		

Tabulka 19: Objekt Profile deceleration (6084Ah) (dle [9])

Popis objektu	*Index, kód objektu*	6084h	Proměnná (VAR)	
	Jméno objektu	(Profile deceleration)		
Popis hodnoty	*Pod-index, typ, přístup*	00h	Unsigned32	rw
	Význam	Zpomalení otáček motoru při brždění [inkrementy/s^2]		
	Hodnota	500 (rozsah 0..2000)		

4.1.3 Referencovací režim

IclA D065 podporuje pouze metodu nastavení referenční nuly vzhledem k aktuální pozici (homing method dimension settings). Jiná zařízení umožňují i automatické vychledání referenčního bodu pomocí signálu čidla, na to IclA nemám digitální vstupy.

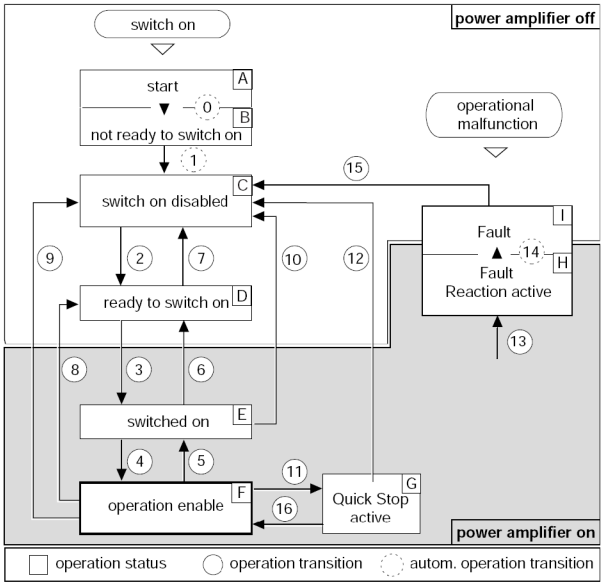
Jo přechodu do referencovacího režimu je možné nastavovat bezpečnostní limity pozic v objektu *Software position safety limit* (*2008h*), limity pozic zařízení v objektu *Software position limit* (*2009h*) a pracovní limity pozic v objektu *Software position limit* (*607Dh*). Limity se ukládají do paměti prostřednictvím objektu *Store parameters* (*1010h*).

Je třeba vybrat nastavování referenční nuly hodnotou *FFh* objektu *Homing method* (*6098h*), nastavit hodnotu aktuální pozice vzhledem k nové referenční nule v objektu *Position assigment value* (*200Bh*) a pomocí změny bitu *CW_START_HOMING* v objektu *Controlword* (*6040h*, tabulka 20) spustit referenční operaci.

Aplikace robotu nemá pro referenční režim využití, pracuje vždy relativně k aktuální pozici.

4.2 Stavový automat pozičního zařízení

Tento stavový automat určuje další reakce zařízení v závislosti na režimu a možnosti aktivace požadovaných operací.



Obrázek 5: Stavový automat pro DS-402 (dle[9])

Stavy zařízení
B – nepřipraven ke spuštění
C – spuštění nepovoleno
D – připraven ke spuštění
E – spuštěný
F – operace povolena
G – nouzové zastavení
I – chybový stav

přechody – příkaz		→ stav
2,6,8	– Shutdown	→ D
3	– Switch on	→ E
7,9,10,12	– Disable Voltage	→ C
7, 10	– Quick Stop	→ C
11	– Quick Stop	→ G
5	– Disable operation	→ E
4,16	– Enable operation	→ F
15	– Fault reset	→ C

Tabulka 20: Objekt Controlword (6040h) (dle [9])

Popis objektu	*Index, kód objektu*	6040 h	Proměnná (VAR)	
	Jméno objektu	Řídící slovo (controlword)		
Popis hodnoty	*Pod-index, typ, přístup*	00h – controlword	Unsigned16	wo
	Význam	Význam bitů (viz tabulka 21 a tabulka 22)		

Tabulka 21: Příkaz změny stavu - význam bitů v Controlword (6040h) (dle [9])

Bit	**Identifikátor (v implementaci)**	**význam**
0	CW_SWITCH_ON	Zapnutí zařízení
1	CW_DISABLE_VOLTAGE	Odpojení napájení
2	CW_QUICK_STOP	Zastavení zařízení
3	CW_ENABLE_OPERATION	Povolení provedení operace
4	CW_START_HOMING	Započetí referencování (referencovací mód)
	CW_NEW_SETPOINT	Započetí poziční operace (poziční mód)
6	CW_RELATIVE	Cílová pozice je relativní k aktuální
7	CW_RESET_FAULT	Zotavení z chybového stavu
8	CW_STOP	Pozastavení pohybu motoru

Stav zařízení je určen prvními 7 bity objektu *Statusword* (*6041h*, tabulka 22). Na bitech, které u stavu nejsou uvedené nezáleží pro platnost stavu (tabulka 24). Stavy B až I určují základní reakce na určité příkazy. Pro spuštění pohybu motoru je nutné nastavit stav F, ve kterém lze aplikovat bity závislé na režimu.

Ostatní bity se týkají významově jednotlivých režimů nebo udávají různé chyby, hlavně překročení pracovních limitů (běžný rozsah pozic, ve kterém zařízení pracuje), limitů nepoškození zařízení (po překročení bývá potřeba servisní zásah) a bezpečnostních limitů (oblasti pozic, kdy by mohlo dojít k poškození člověka anebo strojů).

Tabulka 22: Objekt Statusword (6041h) (dle [9])

Popis objektu	*Index, kód objektu*	6041 h	Proměnná (VAR)	
	Jméno objektu	Stavové slovo (statusword)		
Popis hodnoty	*Pod-index, typ, přístup*	00h – statusword	Unsigned16	ro
	Význam	Význam bitů (tabulka 23)		
	Hodnota	Podle stavu a chyb (tabulka 24)		

Tabulka 23: Stav zařízení - význam bitů objektu Statusword (6041h)

Bit	**Identifikátor (v implementaci)**	**význam**
0	SW_READY_TO_SWITCH_ON	Příprava ke spuštění
1	SW_SWITCHED_ON	Indikace spuštění
2	SW_OPERATION_ENABLED	Povolení operace podle režimu
3	SW_FAULT	Chybový bit
4	SW_VOLTAGE_DISABLED	Odpojení napájení
5	SW_QUICK_STOP	Zastavení zařízení
6	SW_SWITCH_ON_DISABLED	Znemožnění zapnutí
7	SW_WARNING	Varování – opravitelná chyba, motor se může nacházet v nepovoleném stavu
8	SW_RIGHT_OUT_OF_DRIVE_AREA	0 → Pozice překračuje limit W1 1 → Pozice překračuje limit W0
9	SW_REMOTE	
10	SW_TARGET_REACHED	Dosáhnutí cíle
11	SW_INTERNAL_LIMIT_ACTIVE	Pozice překračuje limit W0 nebo W1
12	SW_SETPOINT_ACKNOWLEDGE	Potvrzení nastavení nového cíle (poziční režim)
	SW_HOMING_ATTAINED	Vykonání referencování (referencovací režim)
13	SW_HOMING_ERROR	Chyba při referencování zařízení
14	SW_OUT_OF_DRIVE_AREA	Pozice překračuje limit D0 nebo D1
15	SW_OUT_OF_SECURITY_AREA	Pozice překračuje limit S0 nebo S1

Tabulka 24: Rozpoznání stavu podle kombinací bitů ve Statusword (6041h)

stav	Vypnuté bity	Zapnuté bity
B	SW_READY_TO_SWITCH_ON SW_SWITCHED_ON SW_OPERATION_ENABLED SW_FAULT SW_SWITCH_ON_DISABLED	
C	SW_READY_TO_SWITCH_ON SW_SWITCHED_ON SW_OPERATION_ENABLED SW_FAULT	SW_SWITCH_ON_DISABLED
D	SW_SWITCHED_ON SW_OPERATION_ENABLED SW_FAULT SW_SWITCH_ON_DISABLED	SW_READY_TO_SWITCH_ON SW_QUICK_STOP
E	SW_OPERATION_ENABLED SW_FAULT SW_QUICK_STOP SW_SWITCH_ON_DISABLED	SW_READY_TO_SWITCH_ON SW_SWITCHED_ON
F	SW_FAULT SW_SWITCH_ON_DISABLED	SW_READY_TO_SWITCH_ON SW_SWITCHED_ON SW_OPERATION_ENABLED SW_QUICK_STOP
G	SW_FAULT SW_QUICK_STOP SW_SWITCH_ON_DISABLED	SW_READY_TO_SWITCH_ON SW_SWITCHED_ON SW_OPERATION_ENABLED
I		SW_FAULT

4.3 Monitorování funkce zařízení

Motor umožňuje monitorovat důležité ukazatele jako je teplota, odběr proudu, napětí zdroje, aktuální pozice a aktuální rychlost. Vše je popsáno v následujících tabulkách:

Tabulka 25: Objekt Velocity actual value (606Ch) (dle [9])

Popis objektu	*Index, kód objektu*	606Ch	Proměnná (VAR)	
	Jméno objektu	Aktuální hodnota rychlosti (Velocity actual value)		
Popis hodnoty	*Pod-index, typ, přístup*	00h	Integer32	ro
	Význam	Rychlost otáčení [inkrementy/s]		

Tabulka 26: Objekt Position actual value (6064h) (dle [9])

Popis objektu	*Index, kód objektu*	6064h	Proměnná (VAR)	
	Jméno objektu	Aktuální hodnota pozice (Position actual value)		
Popis hodnoty	*Pod-index, typ, přístup*	00h	Integer32	ro
	Význam	Natočení hřídele motoru vůči referenční nule [inkrementy]		

Tabulka 27: Objekt Current actual value (6078h) (dle [9])

Popis objektu	*Index, kód objektu*	6078h	Proměnná (VAR)	
	Jméno objektu	Aktuální hodnota proudu (Current actual value)		
Popis hodnoty	*Pod-index, typ, přístup*	00h	Integer16	ro
	Význam	Přepočítané proudové zatížení [promile]		

Tabulka 28: Objekt DC link circuit voltage (6079h) (dle [9])

Popis objektu	*Index, kód objektu*	6079h	Proměnná (VAR)	
	Jméno objektu	Napětí DC na obvodu (DC link circuit voltage)		
Popis hodnoty	*Pod-index, typ, přístup*	00h	Unsigned32	ro
	Význam	Napětí [mV]		

Tabulka 29: Objekt Temperature actual value (200Dh) (dle [9])

Popis objektu	*Index, kód objektu*	200Dh	Proměnná (VAR)	
	Jméno objektu	Aktuální hodnota teploty (Temperature actual value)		
Popis hodnoty	*Pod-index, typ, přístup*	00h	Integer16	ro
	Význam	Teplota zesilovače [°C]		

4.4 Sekvence zpráv pro běžné spuštění

Tento příklad ukazuje nastavení a spuštění pohonu v pozičním režimu asynchronně. Synchronní spuštění se liší v odeslání posledního příkazu přes PDO2 a SYNC.

Nastavení pozice 1000 [inc] s rychlostí 500 [inc/s], zrychlením 3000 [inc/s^2] a zpomalením 1500 [inc/s^2]. ID motoru jako NMT uzlu je 4.

Nastavení rychlosti na 500 – SDO (*profile velocity 6081h:00h*):
Tx COB-Id 604h data 23h 81h 60h 0h F4h 1h 0h 0h
potvrzení SDO:
Rx COB-Id 584h data 60h 81h 60h 0h 0h 0h 0h 0h

Nastavení zrychlení na 3000 – SDO (*profile acceleration 6083h:00h*):
Tx COB-Id 604h data 23h 83h 60h 0h B8h Bh 0h 0h
potvrzení SDO:
Rx COB-Id 584h data 60h 83h 60h 0h 0h 0h 0h 0h

Nastavení zpomalení na 3000 – SDO (*profile deceleration 6084h:00h*):
Tx COB-Id 604h data 23h 84h 60h 0h DCh 5h 0h 0h
potvrzení SDO:
Rx COB-Id 584h data 60h 84h 60h 0h 0h 0h 0h 0h

Změna stavu NMT na Operational:
Tx COB-Id 0h data 1h 4h
odpověď PDO1 (*statusword 6041h:00h*) – stav C:
Rx COB-Id 184h data 40h 0h

Pokud aktuání stav nepřišel do 2 sekund, požádá se o něj – SDO (*statusword 6041h:00h*)
Tx COB-Id 604h data 40h 41h 60h 0h
odpověď SDO (*statusword 6041h:00h*) – stav C:
Rx COB-Id 584h data 4Bh 41h 60h 0h 40h 0h 0h 0h

Příkaz Shutdown – PDO1 (*controlword 6040h:00h*)
Tx COB-Id 204h data 6h 0h
odpověď PDO1 (*statusword 6041h:00h*) – stav D:
Rx COB-Id 184h data 21h 0h

Příkaz Switch on – PDO1 (*controlword 6040h:00h*)
Tx COB-Id 204h data 7h 0h
odpověď PDO1 (*statusword 6041h:00h*) – stav E:
Rx COB-Id 184h data 23h 0h

Příkaz Enable operation – PDO1 (*controlword 6040h:00h*)

Tx COB-Id 204h data Fh 0h
odpověď PDO1 (*statusword 6041h:00h*) – stav F:
Rx COB-Id 184h data 27h 0h

Nastavení pozičního módu – SDO (*modes of operation 6060h:00h*)

Tx COB-Id 604h data 2Fh 60h 60h 0h 1h
potvrzení SDO:
Rx COB-Id 584h data 60h 60h 60h 0h 40h 0h 0h 0h
odpověď (*statusword 6041h:00h*) – stav F
Rx COB-Id 184h data 27h 2h

Nastavení příkazu (*controlword 6040h:00h*) **s příznaky**: *CW_ENABLE_OPERATION, CW_QUICK_STOP, CW_DISABLE_VOLTAGE,CW_SWITCH_ON*
Aby se zrušil případný *CW_NEW_SETPOINT* – PDO1:

Tx COB-Id 204h data Fh 0h
bez odpovědi, protože se nezměnil stav

Nastavení cílové pozice 1000 – SDO (*statusword 6041h:00h*)

Tx COB-Id 604h data 23h 7Ah 60h 0h E8h 3h 0h 0h
povrzení SDO:
Rx COB-Id 584h data 60h 7Ah 60h 0h 40h 0h 0h 0h

Nastavení příkazu (*controlword 6040h:00h*) **s příznaky**: *CW_ENABLE_OPERATION, CW_QUICK_STOP, CW_DISABLE_VOLTAGE,CW_SWITCH_ON, CW_NEW_SETPOINT*
Žádost o spuštění pohybu – PDO1:

Tx COB-Id 204h data 1Fh 0h
odpověď PDO1 (*statusword 6041h:00h*) – stav F +SW_SETPOINT_ACKNOWLEDGE, pozice cíle byla potvrzena jako dosažitelná a motor započal pohyb:
Rx COB-Id 184h data 27h 12h

(... motor provádí změnu pozice ...)

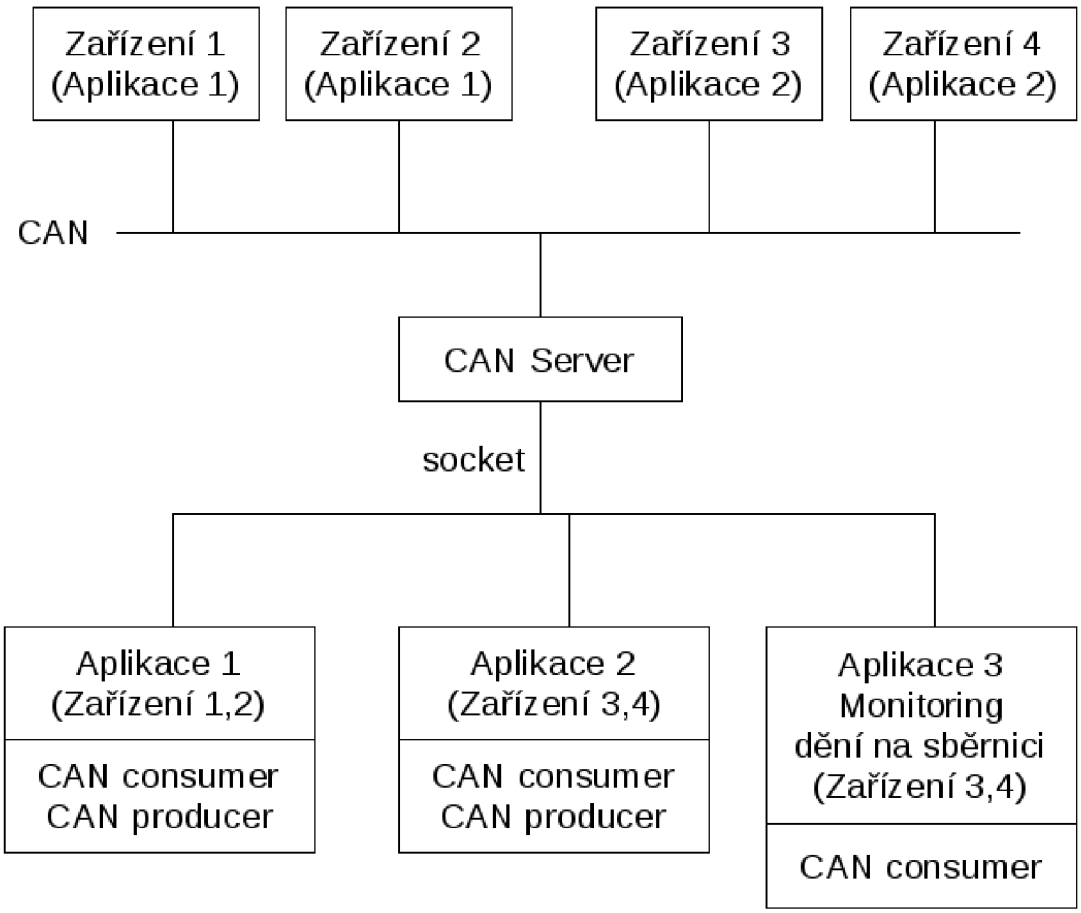
Dosažení cíle – odpověď PDO1 (*statusword 6041h:00h*)
stav F s příznaky *SW_SETPOINT_ACKNOWLEDGE* a *SW_TARGET REACHED*

Rx COB-Id 184h data 27h 16h

5 CAN jako služba na socketu

Tento koncept umožňuje škálovatelnost řešení systémů aplikací, které ovládají zařízení na CAN sběrnici a monitorují provoz na sběrnici. A snižuje náklady na taková řešení využitím jediného USB-CAN převodníku, aby ho aplikace efektivně a konkurentně sdílely. I v rámci jediná aplikace je vhodné použít zvlášť klienta, který přijímá PDO a EMCY, a klienta pro SDO komunikaci. Socket může být lokální (Unix domain sockets) nebo síťový (TCP/IP), to přidává do řešení další škálovatelnost, jako je vzdálená správa CAN zařízení.

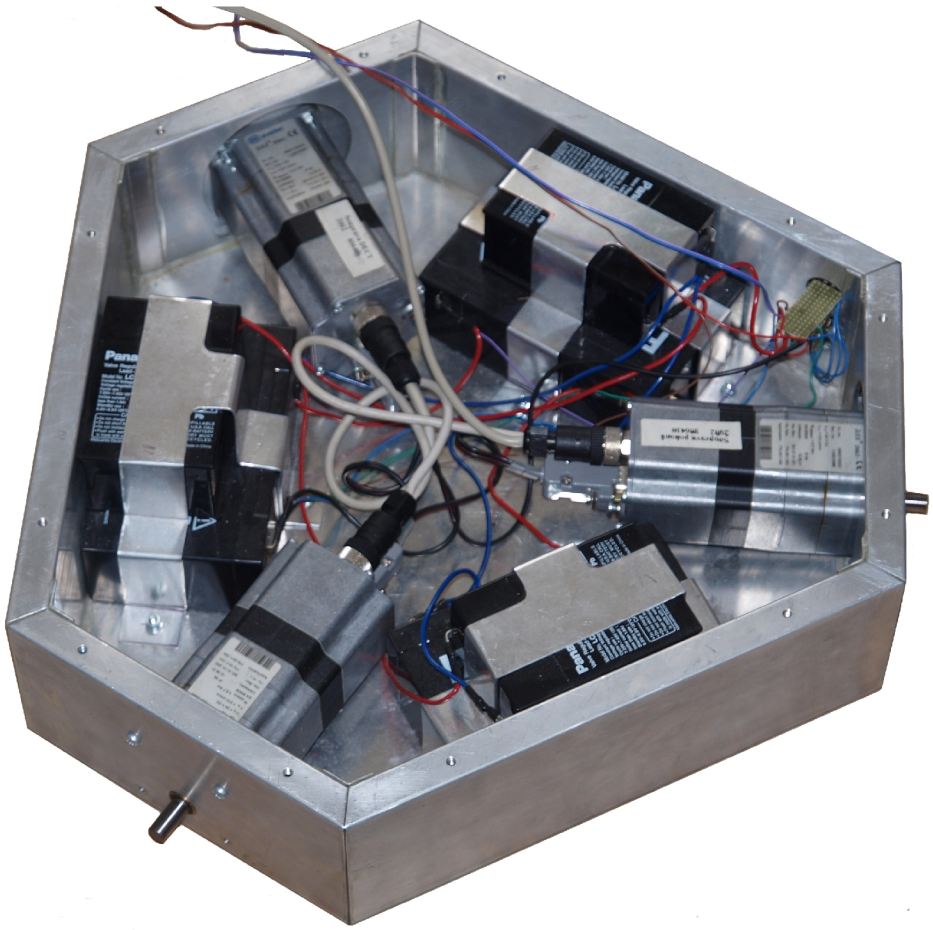
Před použitím tohoto konceptu bylo testováno přímé užití převodníku jednou aplikací. Objevily se obtíže při testování a nebylo možné zároveň spustit pomocné nástroje pro kontrolu nastavení parametrů a chybových stavů zařízení. Také inicializace a vypnutí převodníku jsou zdlouhavé operace trvající několik sekund. Změnou přístupu se zjednodušila uživatelská rozhraní aplikací, u kterých nyní není třeba zadávat přenosovou rychlost sběrnice a spuštění převodníku. V neposlední řadě došlo k odstranění závislosti koncových aplikací na FTDI D2XX knihovně, která působila některé problémy při práci s vlákny, které vnitřně využívá.



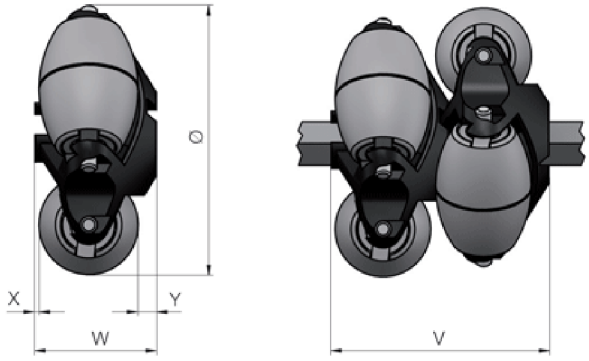
Obrázek 6: CAN jako služba na socketu

6 Aplikace – tříosý všesměrový robot

Všesměrového pohybu robot dosahuje použitím kol, které kolmo na směr otáčení mají valivé tělíska („soudečky“). Běžně pracuje tak, že se pohybem všech tří kol natočí do směru žádaného pohybu a pak současnou aktivací dvojice motorů zahájí přesun. Účinná uražená dráha je tedy menší než ta, co museli kolečka ve směru svého otáčení urazit na obvodu.

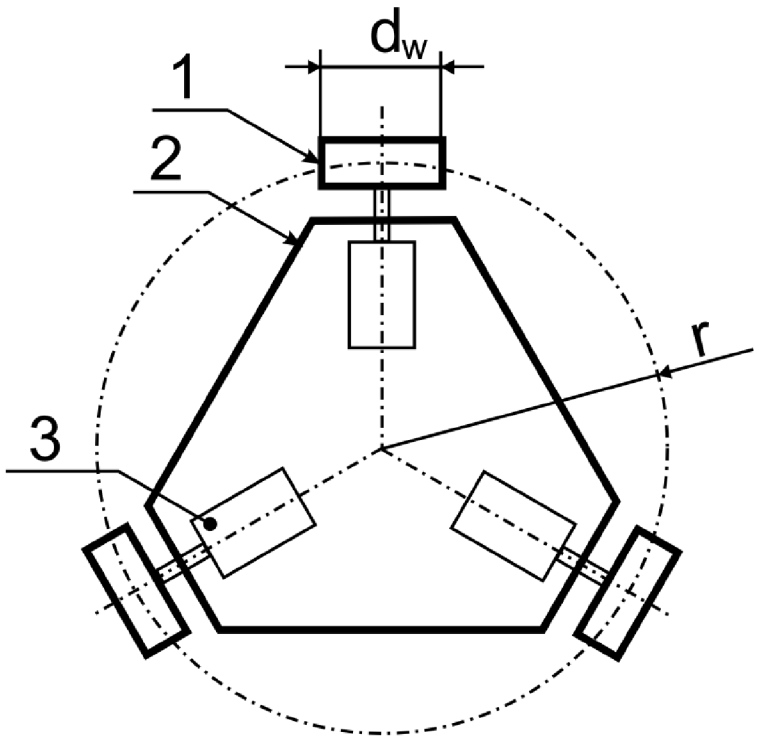


Obrázek 7: Podvozek robotu s pohony a akumulátory



Obrázek 8: Nákres kola Interroll Omniwheels Series 2500

Poziční zařízení IclA D065 DC024 S018 instalované v robotu má převodový poměr G_R=160:9. Pro vstup převodovky je rozlišení enkodéru 12 inkrementů na jednu otáčku I_{pr}=12.G_R.
Kola Interroll mají vnější průměr 120 mm.



Obrázek 9: Schematický nákres robotu

Popis nákresu robotu (obrázek 9):

1 – všesměrové kolo fy Interroll
2 – podvozek robotu
3 – pohon IclA D065
r – vzdálenost bodu dotyku kol s podložkou od středu robotu [m]
d_w – průměr kola [m]

6.1 Kinematika

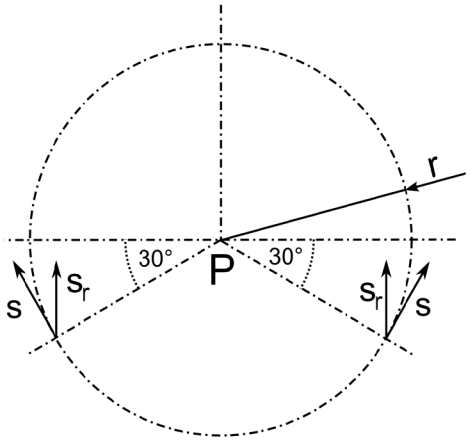
Pro dráhu na obvodu kola ve směru otáčení platí vztah pro *i* inkrementů enkoderu:

$$s = i \cdot I_{pr} \cdot \pi \cdot d_w \quad [m]$$

Pohyb vpřed

Kola se otáčí vzájemně opačně, aby se bod P pohyboval v ose předního kola. Při pohybu dopředu se konstrukcí robotu uražená dráha s_r obvodu kola redukuje kosinem 30° na dráhu výsledného pohybu.

$$s_r = s \cdot \cos 30° \quad [m]$$

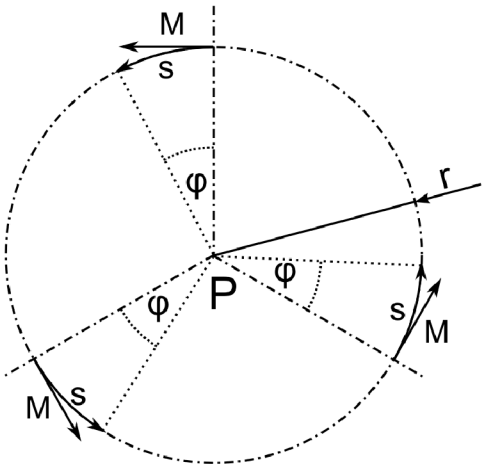


Obrázek 10: Kinematika pohybu vpřed

Rotace robotu

Všechna kola se točí souhlasně a stejně rychle, aby se celý robot otočil kolem svého středu, bodu P. Rotace do směru žádaného pohybu o úhel φ je dána vztahem:

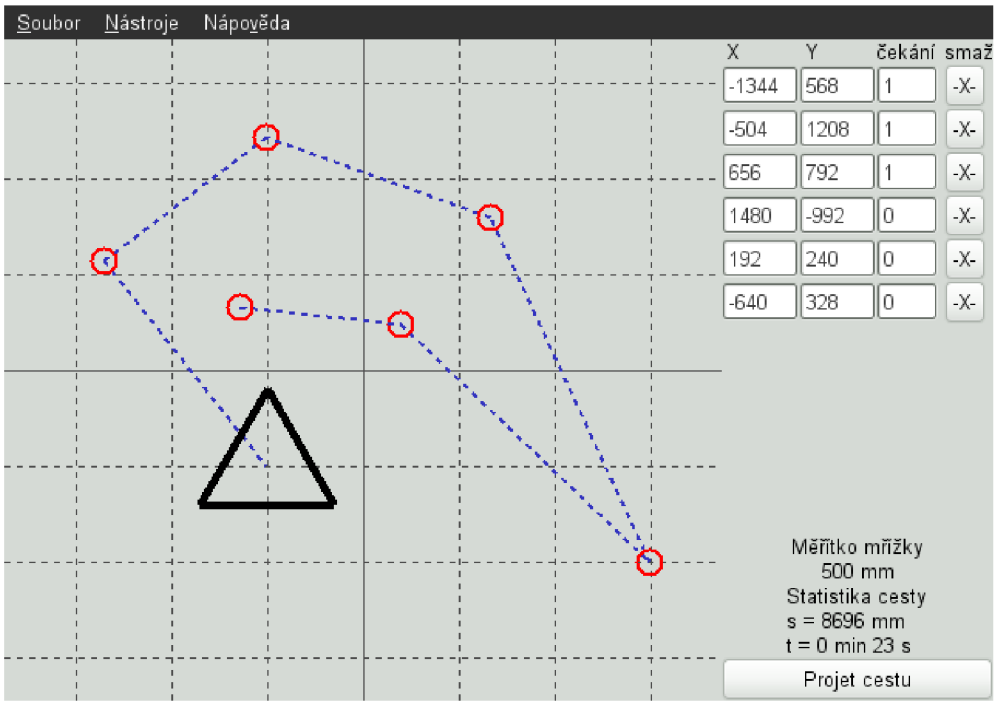
$$\phi = \frac{s}{r} \quad [rad]$$



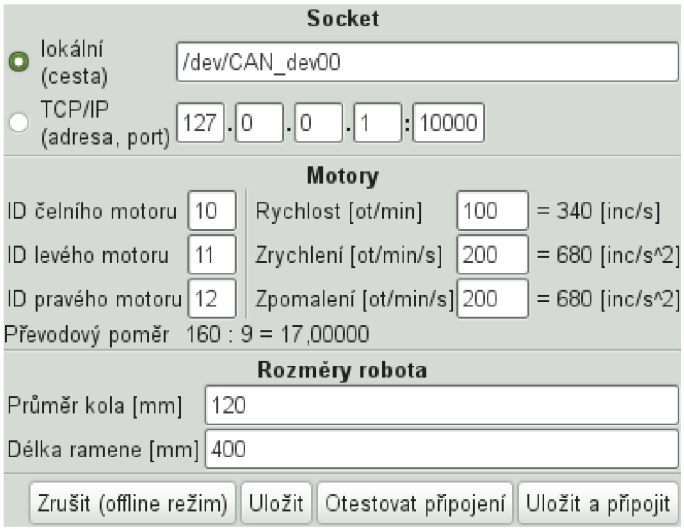
Obrázek 11: Kinematika rotace

6.2 Ovládací aplikace s GUI

Umožňuje ovládání robotu v grafickém uživatelském rozhraní. Při spuštění se zobrazí dialog pro připojení ke CAN jako službě na socketu, nastavení rozměrů robotu, ID motorů jako NMT uzlů a rychlosti výsledného pohybu. Diagnostika je v samostatném dialogu přístupná z hlavního menu. Hlavní okno aplikace obsahuje plochu, kde je zakreslena momentální poloha robotu a body cesty (waypoint). Na pravé straně hlavního okna se nachází seznam všech bodů cesty s možností jejich úpravy. Body cesty se přidávají stiskem tlačítka myši v prostoru hlavní plochy a je možno jejich polohu upravovat tahem myši.



Obrázek 12: Hlavní okno aplikace



Obrázek 13: Okno nastavení

7 Implementace

K implementaci je použitý jazyk C/C++ a knihovny FTDI D2XX pro komunikaci s převodníkem a soubor knihoven GTK+ pro uživatelské rozhraní. Implementace je určena a testována v prostředí operačního systému Linux a kompilátoru G++ z projektu GCC (GNU Compiler Collection). Přenesení do prostředí jiných operačních systémů jako je MS Windows a Apple Mac OS by mělo být možné a poměrně snadné díky přenositelnosti použitých knihoven a stylu psaní kódu vyhovujícího normě jazyka C++ (ISO/IEC 14882:2003).

7.1 Protokol mezi klientem a serverem

Protokol je založený tak, že server se chová jako buffer a sám neodesílá žádná data dokud si je client nevyžádá, ten tedy kontroluje zda data nepřišla, pokud je ve stavu, že by nějaká očekával. Měl by samozřejmě kontrolovat i průběžně, jestli zařízení nehlásí kritický chybový stav. U klientů typu consumer se nastavují filtry CAN zpráv podle COB-ID, tyto filtry fungují jako „white list“, tedy ukládají se do fronty pouze zprávy s COB-ID ve filtru obsaženým.

7.1.1 Zprávy

Na prvním byte zprávy je vždy typ zprávy, který určuje význam následujících bytů. Struktura s rozhraním zpráv protokolu *CAN_protMessage* poskytuje metody pro vytváření všech typů zpráv a metodu *extractData*, která rozšifruje data přijaté zprávy do členských proměnných. To zjednodušuje práci se zprávami a zároveň přidává škálovatelnost řešení. Je třeba se mít na pozoru, není to třída, ale struktura a z hlediska čistoty objektového návrhu je porušeno zapouzdření. Pro odeslání zprávy na socket a pokusu o přijetí ze socketu jsou implementovány metody *send* a *recv*.

Typy zpráv a jejich data

- responsePositive (CMD_RESPONSE_POSITIVE)
  - pozitivní odpověď
- responseNegative (CMD_RESPONSE_NEGATIVE)
  - negatiní odpověď
- clientConnect (CMD_CLIENT_CONNECT)
  - žádost o připojení klienta
  - data: typ klienta (1 byte)
- clientConnectResponse (CMD_CLIENT_CONNECT_RESPONSE)
  - odpověď na připojení klienta
  - data: režim serveru (1 byte), přijatý typ klienta (1 byte)
- controllerSetBitrate (CMD_CONTROLLER_SET_BITRATE)
  - žadost controllera o změnu rychlosti CAN sběrnice
  - data: rychlost (4 byte)
- controllerGetBitrate (CMD_CONTROLLER_GET_BITRATE)
  - žádost controllera o rychlost sběrnice

- controllerGetBitrateResponse (CMD_CONTROLLER_GET_BITRATE_RESPONSE)
  - odpověď s rychlostí sběrnice
  - data: rychlost (4 byte)
- producerSend (CMD_PRODUCER_SEND)
  - odeslání CAN zprávy na sběrnici producentem
  - data: délka (1 byte), COB-ID (2 byte), data CAN zprávy (0 až 8 byte)
- consumerRecv (CMD_CONSUMER_RECV)
  - žádost o přijetí zprávy z fronty (pokud nějaká zpráva ve frontě je)
- consumerRecvResponse (CMD_CONSUMER_RECV_RESPONSE)
  - odpověď se zprávou z fronty pro klienta
  - data: délka (1 byte), COB-ID (2 byte), data CAN zprávy (0 až 8 byte)
- consumerFilterAdd (CMD_CONSUMER_FILTER_ADD)
  - přidání COB-ID do filtru pro frontu consumera
  - data: COB-ID (2 byte)
- consumerFilterClear (CMD_CONSUMER_FILTER_CLEAR)
  - zrušení filtru fronty consumera, ten pak nemůže přijímat žádné zprávy
- clientDisconnect (CMD_CLIENT_DISCONNECT)
  - oznámení odpojení klienta, na zprávu server neodpovídá, protože klient zavírá socket

7.1.2 Traksakce

Server reaguje na zprávy bezkontextově (význam zprávy se nemění podle situace). Transakce jsou vždy iniciované klientem:

- Připojení
  - žádost: clientConnect
  - odpověď při úspěchu: clientConnectResponse
  - odpověď při neúspěchu: responseNegative
- Odpojení
  - žádost: clientDisonnect
- Zjištění ryclosti sběrnice
  - žádost: controllerGetBitrate
  - odpověď při úspěchu: controllerGetBitrateResponse
- Odeslání CAN zprávy
  - žádost: producerSend
  - odpověď při úspěchu: responsePositive
  - nezdařilo se odeslání: responseNegative
- Příjem CAN zprávy z fronty
  - žádost: consumerRecv
  - odpověď při úspěchu: consumerRecvResponse
  - fronta je prázdná: responseNegative
- Rozšíření filtru CAN zpráv
  - žádost: consumerFilterAdd
  - odpověď při úspěchu: responsePositive
  - odpověď při neúspěchu: responseNegative

- Zrušení filtru CAN zpráv
  - žádost: consumerFilterClear
  - odpověď při úspěchu: responsePositive
  - odpověď při neúspěchu: responseNegative

7.2 CAN server

Implementuje koncept CAN sběrnice jako služby na socketu. Nebylo však možno použít démona služeb (xinetd), protože hardware převodníku musí být spuštěn a kontrolován serverem nepřetržitě. Což odpovídá realtimovým aplikacím, které zařízení na CAN sběrnici používají. Využití socketů řeší postupné provádění požadovaných operací. Server není konkurentní na úrovní vláken ani procesů, protože CAN sběrnice jako zdroj přenáší pouze jednu zprávu (podobné jako half-duplex, ale v poměru 1:N). Přepínání a souběžnost komunikace přesouvá na stranu operačního systému, který funkčnost socketů zajišťuje. Pro připojené konzumenty zavádí oddělené fronty, ve kterých jim ukládá data dokud si je nevyžádají. V tomto kopíruje chování USB-CAN, kde též zařízení odpovídá pouze pokud je dotázáno. Je možné zapnout odpojení klientů při pasivitě, hlavně když se jim hromadí nevybrané zprávy. To může značit i socket, který nebyl správně ukončen a z druhého konce tedy už žádná informace nepřijde. Sockety kvůli konkurentnosti v rámci jediného vlákna jsou nastaveny tak, že se neohlášené ukončení socketu ze strany klienta (např. pád aplikace) v některých případech nepodaří detekovat.

Režimy serveru:

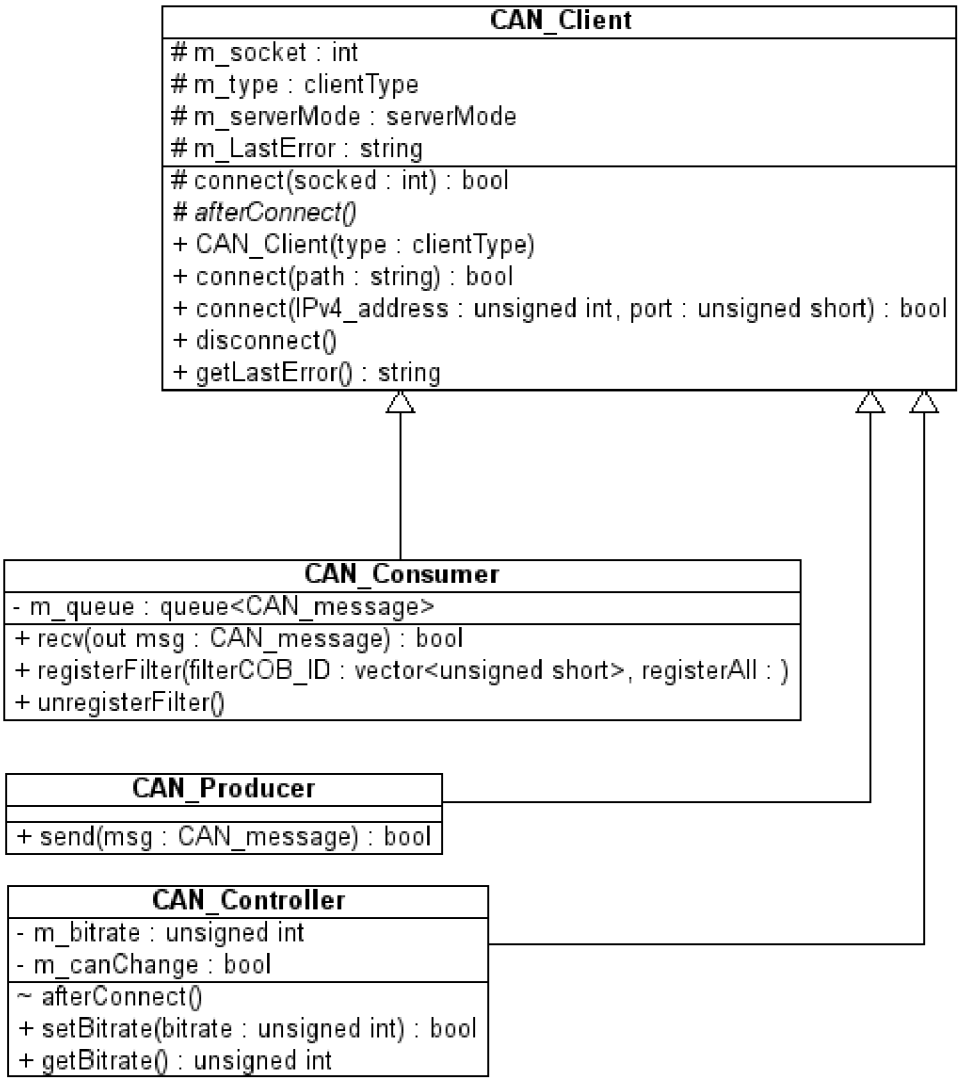
- constant – rychlost sběrnice CAN je nastavená pevně a žádný klient ji nemůže změnit
- first-master – rychlost sběrnice CAN může měnit pouze první připojený controller klient
- free – rychlost sběrnice CAN může změnit kterýkoliv připojený controller klient

Server vypisuje události na standardní výstup. U přenášených CAN zpráv, které odpovídají profilu CANopen navíc rozlišuje, o jaký typ objektů jde a vypisuje k němu relevantní informace, například u SDO index objektu ve slovníku nebo u NMT, kterých uzlů se příkaz týká. To rozšiřuje možnosti ladění komunikujících aplikací.

Jako aplikace je CAN server spouštěn z prostředí příkazové řádky a veškerá nastavení jsou předána jako parametry příkazové řádky. Tak lze spustit více CAN serverů u řešení s více USB-CAN převodníky.

7.3 CAN client

Pro připojení na CAN server se využívá třídy *CAN_Client*, která je bází dalších tříd, které plní specifické úkoly. Přetížená metoda *connect* se připojuje na lokální socket nebo na TCP/IP socket. Metoda *disconnect* odpojuje od serveru a zavírá socket na straně klienta. Chybovou hlášku lze získat voláním *getLastError* po volání metod, které vrací *false* a tím přítomnost chyby signalizují. Rozhraní umožňuje potomkům implementovat virtuální metodu *afterConnect*, která se volá s dynamickou vazbou po úspěšném připojení k serveru.



Obrázek 14: UML diagram tříd CAN clientů

7.3.1 Controller – řízení sběrnice

Používá se k nastavení a čtení přenosové rychlosti CAN sběrnice. Oddělení této funkcionality od hlavního klienta reflektuje, že většinou aplikace ovládající zařízení rychlost sběrnice znát nemusí a nepotřebují ji měnit. Nastavení rychlosti nemusí být controlleru umožněno, závisí to na režimu v jakém je server a jestli jiný controller si nevyžádal výlučný přístup.

Implementace je ve třídě *CAN_Consumer*, která je potomkem třídy *CAN_Client*. Poskytuje rozhraní pro získání rychlosti sběrnice voláním metody *getBitrate* nebo její nastavení metodou *setBitrate*. Třída využívá možnosti virtuální metody *afterConnect* a implementuje ji jako načtení rychlosti sběrnice hned po připojení k serveru.

7.3.2 Producer – odesílání zpráv

Oddělení od hlavního rozhraní klienta umožňuje sdílení pruducera v rámci aplikace, která pro ovládání více zařízení nepoužívá vlákna a zároveň nechce řešit producenta CAN zpráv jako sdílený zdroj. Implementace ve třídě *CAN_Producer* obsahuje pouze metodu *send*, která odesílá CAN zprávu.

7.3.3 Consumer – příjem zpráv

Pro většinu aplikací se používají 2 na zařízení. Jeden pro CANopen SDO a druhý pro všechny ostatní zprávy, hlavně PDO a EMCY. Není totiž žádoucí, aby do metod, které s SDO pracují mohli vniknout zprávy jako je EMCY. V takové situaci je nevýhodné, že aplikace čeká na vnitřní timeout pro SDO odpověď a nezpracuje EMCY okamžitě, pokud nepoužívá další vlákno jako „watchdog“. Implementace rozhraní třídy *CAN_Consumer* má metody *recv*, která přijímá CAN zprávu, *registerFilter*, která přidává COB-ID zpráv do filtru a ty pak server pro klienta ukládá, a *unregisterFilter* pro zrušení filtru.

7.4 IclA D065

Abstrakce pozičního zařízení IclA D065 je implementovaná v knihovně icla_d065.a ve třídě C_IclA_D065 s dále popsaným veřejným rozhraním (pouze běžně užívaná funkcionalita).

7.4.1 Veřejné rozhraní třídy C_IclA_D065

Připojení

- init(producer, consumer, consumerCANopen, dictionary): bool
  - pomocí připojených klientů a slovníku se pokusí komunikovat se zařízením
- setID(id)
  - nastaví ID motoru v rámci abstrakce (nemění ID skutečného motoru přes LMT)
- getID: unsigned int
  - vrací nastavené ID motoru
- testConnection: bool
  - otestuje připojení na nastaveném ID, jestli odpovídá pozičnímu zařízení DS-402

Informace a diagnostika

- getGearMotorRevolotions() : unsigned int
  - počet otáček motoru na vstupu převodovky (pro výpočet převodového poměru)
- getGearShaftRevolutions() : unsigned int
  - počet otáček na výstupu převodovky (pro výpočet převodového poměru)
- getPosition : int
  - načte ze zařízení aktuální pozici
- getCurrent : int
  - načte ze zařízení aktuální pozici proud motoru [mA]
- getVoltage : unsigned int
  - načte ze zařízení aktuální napětí zdroje [mV]
- getTemperature : short int
  - načte ze zařízení aktuální teplotu [°C]

Nastavení

- NMT_setOperational
  - přepne zařízení do stavu Operational z hlediska NMT
- setVelocity(v)
  - nastaví konečnou rychlost motoru (inc/s)
- setAcceleration(a)
  - nastaví zrychlení motoru (inc/s^2)
- setDeceleration(d)
  - nastaví zpomalení (brždění) motoru (inc/s^2)

- setHomingMode
  - nastaví motor do referencovacího režimu, automaticky provede i potřebné změny stavu
- setPositionMode
  - nastaví motor do pozičního režimu, automaticky provede i potřebné změny stavu
- setManualMode
  - nastaví manuální režim; je doporuřené do něj motor přepnout při ukončení práce

Poziční režim

- setTargetPosition(target)
  - nastaví cílovou pozici, která bude odeslána přes PDO2
- startMotion
  - spustí pohyb motoru asynchronně přes PDO1, před tím odešle cílovou pozici jako SDO
- prepareSyncMotion
  - odešle PDO2 pro nastavení cílové pozice a stavu k započetí pohybu před SYNC
- syncMotion
  - odešle SYNC objekt, volá se pouze u jednoho motoru, synchronizuje všechny
- stopMotion
  - zastaví pohyb motoru
- isMoving : bool
  - vrací příznak otáčení motoru podle stavu (motor přímo nedotazuje)
- relativePositioning
  - nastaví příznak relativního pohybu, který se při spuštění pohybu odesílá
- absolutePositioning
  - zruší příznak relativního pohybu, který se při spuštění pohybu odesílá
- processMessages
  - přijme a zpracuje přijaté zprávy, převede je na stav a nastaví příznaky

Příkazy změny stavu (odpovídají obrázku 5)

- switchOn : bool
- enableOperation : bool
- disableOperation : bool
- shutdown : bool
- quickStop : bool
- disableVoltage : bool
- faultReset : bool

7.4.2 Příklad použití veřejného rozhraní

Načte se CANopen slovník, připojí klienty k serveru na lokální socket. Nastaví se rychlostní rampa, přepne se do pozičního režimu (automaticky nastaví NMT Operational, aby se mohli přijímat PDO zprávy a postupně přepne motor do stavu F – operational enabled), nastaví se relativní pozicování a cílová pozice, spustí se pohyb (zde asynchronně) a dokud se motor pohybuje, kontrolují se přijaté zprávy. Tento zjednodušený příklad reaguje na meúspěch pouze opuštěním funkce bez ukončení spojení klientů nebo zotavujících kroků.

```
// načtení slovníku pro CANopen
C_dictionary icla_d065_dic;
if(!icla_d065_dic.load("icla_d065.dic"))
     return; // neúspěch

// klienti
CAN_Consumer consumer, consumerCANopen;
CAN_Producer producer;

// připojení klientů na socket
if(!consumer.connect(path))
     return; // neúspěch
if(!consumerCANopen.connect(path))
     return; // neúspěch
if(!producer.connect(path))
     return; // neúspěch

unsigned deviceID=2;
C_IclA_D065 motor;

if(motor.init(&producer,&consumer,&consumerCANopen,&icla_d065_dic))
{
     motor.setID(deviceID); // NMT nodeID motoru
     // test spojení
     if(motor.testConnection())
          return; // neúspěch
     // rychlostní rampa
     motor.setVelocity(100); // dosažená rychlost [Inc/s]
     motor.setAcceleration(3000); // zrychlení [Inc/s^2]
     motor.setDeceleration(2000); // brždění [Inc/s^2]

     motor.setPositionMode(); // volá NMT_setOperational a mění stav
     motor.relativePositioning(); // relativní pozicování

     motor.setTargetPosition(400); // cílová pozice (relativně)
     motor.startMotion(); // asynchronní spuštění pohybu
     while(motor.isMoving()) // smyčka dokud se motor pohybuje
     {
          motor.processMessages();// aktualizace stavu ze zpráv
     }
}
```

7.5 GUI Aplikace

Grafické uživatelské rozhraní je vytvořeno pomocí knihovny GTK+ a RAD nástroje Glade, který umožňuje efektivní uložení popisu rozhraní ve formátu XML. V programu se pak přímo pracuje pouze s GTK+ widgets (prvky rozhraní), které interaktivně zasahují do běhu aplikace, tedy se z nich získávají vstupy nebo slouží k výstupu. Ostatní widgets jsou buď kontejnery, tedy organizují ostatní widgets do struktury, nebo neinteraktivní prvky jako nápisy nebo dekorační rámečky. Signály událostí interaktivních prvků se spojují s callback funkcemi, které činnost provedou. Protože callback funkce nemohou být členské proměnné a zároveň je dobré se vyhnout globálním proměnným, je použitý návrhový vzor singleton. Takto je celá instance aplikace zabalená do jedné třídy. V některých ohledech nebyl z praktických důvodů dodržen kompletně čistý objektový návrh z hlediska zapouzdření. To vychází vtříc nepravděpodobnosti znovupoužití těch částí kódu, které jsou velmi úzce spojeny s konkrétním uživatelským rozhraním v XML souboru.

Callbackové funkce provádějí z většiny reakce na stisk tlačítek a volbu položek v menu. Trochu zvláštní jsou callbacky od *waypoints_eventbox*, které reagují na stisk tlačítek a pohyb kurzoru myši nad plochou pro vykreslení cesty, aby se umožnily její úpravy.

Oknům s interaktivními prvky jsou v singletonu přiřazeny třídy, které pak redundantně obsahují validní hodnoty z GUI. Tedy pokud je v některém vstupním widgetu invalidní hodnota, může se vypsat chyba, ale v žádném případě se neprojeví ve vnitřní logice aplikace.

7.5.1 Struktura widgets v aplikaci

Struktura widgets hlavního okna

- **GtkWindow** *main_window* – hlavní okno aplikace
  - **GtkVBox** – kontejner s vertikálním rozložením
    - **GtkMenuBar** – pruh hlavního menu
      - **GtkMenuItem** – položka menu Soubor
        - **GtkMenu** – rozbalovací menu Soubor
          - **GtkImageMenuItem** *new_item* – položka menu Nový
          - **GtkImageMenuItem** *open_item* – položka menu Otevřít
          - **GtkImageMenuItem** *save_item* – položka menu Uložit
          - **GtkImageMenuItem** *save_as_item* – položka menu Uložit jako
          - **GtkSeparatorMenuItem** – odělovač v menu
          - **GtkImageMenuItem** *quit_item* – položka menu Ukončit
      - **GtkMenuItem** – položka menu Nástroje
        - **GtkMenu** – rozbalovací menu Nástroje
          - **GtkImageMenuItem** *settings_item* – položka menu Nastavení
          - **GtkImageMenuItem** *diagnostics_item* – položka menu Diagnostika
      - **GtkMenuItem** – položka menu Nápověda
        - **GtkMenu** – rozbalovací menu Nápověda
          - **GtkImageMenuItem** *about_item* – položka menu O aplikaci
    - **GtkHBox** – kontejner s horizontálním rozložením

- **GtkEventBox** *waypoints_eventbox* – kontejner zachycující zprávy, na které potomek nemůže reagovat
  - **GtkDrawingArea** *waypoints_drawingarea* – plocha pro kreslení cesty
- **GtkVBox** – kontejner s vertikálním rozložením
  - **GtkTable** *points_table* – tabulka s hodnotami bodů cesty
  - **GtkLabel** *way_stats_label* – popisek pro statistiku cesty
  - **GtkLabel** *grid_scale_label* – popisek měřítka mřížky
  - **GtkButton** *way_run_button* – tlačítko pro spuštění pohybu robotu

Struktura widgets okna nastavení

- **GtkWindow** *settings_window* – modální okno s nastavením
  - **GtkVBox** – kontejner s vertikálním rozložením
    - **GtkLabel** – popisek „Socket“
    - **GtkTable** – kontejner s tabulkovým rozložením
      - **GtkRadioButton** *SET_socket_local_radiobutton* – přepínací tlačítko pro volbu lokálního socketu
      - **GtkRadioButton** *SET_socket_TCPIP_radiobutton* – přepínací tlačítko pro volbu vzdáleného socketu s IPv4 adresou
      - **GtkHBox** – kontejner s horizontálním rozložením
        - **GtkEntry** *SET_socket_address_1_entry* – první číslo IP adresy
        - **GtkLabel** – popisek s odělující tečkou adresy
        - **GtkEntry** *SET_socket_address_2_entry* – druhé číslo IP adresy
        - **GtkLabel** – popisek s odělující tečkou adresy
        - **GtkEntry** *SET_socket_address_3_entry* – třetí číslo IP adresy
        - **GtkLabel** – popisek s odělující tečkou adresy
        - **GtkEntry** *SET_socket_address_4_entry* – čtvrté číslo IP adresy
        - **GtkLabel** – popisek s odělující dvojtečkou adresy a portu
        - **GtkEntry** *SET_socket_port_entry* – port socketu
    - **GtkHSeparator** – horizontální oddělovač
    - **GtkLabel** – popisek „Motory“
    - **GtkTable** – kontejner s tabulkovým rozložením
      - **GtkEntry** *SET_front_motor_ID_entry* – ID předního motoru
      - **GtkEntry** *SET_left_motor_ID_entry* – ID levého motoru
      - **GtkEntry** *SET_right_motor_ID_entry* – ID pravého motoru
      - **GtkLabel** – popisek „Rychlost [ot/min]“
      - **GtkLabel** – popisek „Zrychlení [ot/min/s]“
      - **GtkLabel** – popisek „Zpomalení [ot/min/s]“
      - **GtkEntry** *SET_velocity_entry* – rychlost otáčení motoru
      - **GtkEntry** *SET_acceleration_entry* – zrychlení motoru
      - **GtkEntry** *SET_decelaration_entry* – zpomalení motoru
      - **GtkVSeparator** – vertikální oddělovač
      - **GtkLabel** – popisek „ID čelního motoru“
      - **GtkLabel** – popisek „ID levého motoru“
      - **GtkLabel** – popisek „ID pravého motoru“

        - **GtkLabel** – popisek „ [inc/s]“
        - **GtkLabel** – popisek „ [inc/s^2]“
        - **GtkLabel** – popisek „ [inc/s^2]“
        - **GtkLabel** – popisek „ = “
        - **GtkLabel** – popisek „ = “
        - **GtkLabel** – popisek „ = “
        - **GtkLabel** *SET_velocity_inc_label* – popisek hodnoty rychlosti v inc/s
        - **GtkLabel** *SET_acceleration_inc_label* – popisek hodnoty zrychlení v inc/s^2
        - **GtkLabel** *SET_deceleration_inc_label* – popisek hodnoty zpomalení v inc/s^2
        - **GtkLabel** – popisek „Převodový poměr“
        - **GtkLabel** *SET_gear_ratio_label* – popisek převodového poměru „x : y = r“
    - **GtkHSeparator** – horizontální oddělovač
    - **GtkLabel** – popisek „Rozměry robotu“
    - **GtkTable** – kontejner s tabulkovým rozložením
        - **GtkEntry** *SET_wheel_diameter_entry* – průměr kola v mm
        - **GtkEntry** *SET_arm_length_entry* – rameno robotu v mm (od středu po kolo)
        - **GtkLabel** – popisek „Průměr kola [mm]“
        - **GtkLabel** – popisek „Délka ramene [mm]“
    - **GtkHSeparator** – horizontální oddělovač
    - **GtkHBox** – kontejner s horizontálním rozložením
        - **GtkButton** *SET_save_and_connect_button* – tlačítko Uložit a připojit
        - **GtkButton** *SET_test_button* – tlačítko Otestovat připojení
        - **GtkButton** *SET_save_button* – tlačítko Uložit
        - **GtkButton** *SET_cancel_button* – tlačítko Zrušit (offline režim)

Struktura widgets okna diagnostiky

- **GtkWindow** *diagnostics_window* – okno s monitorováním
    - **GtkLabel** – popisek „Teplota [°C] “
    - **GtkLabel** – popisek „Napětí zdroje [mV] “
    - **GtkLabel** – popisek „Procházející proud [mA] “
    - **GtkButton** *DIAG_reload_button* – tlačítko Obnovit
    - **GtkLabel** *DIAG_temperature_label* – popisek pro hodnotu teploty
    - **GtkLabel** *DIAG_voltage_label* – popisek pro hodnotu napětí
    - **GtkLabel** *DIAG_current_label* – popisek pro hodnotu proudu

Struktura widgets okna o aplikaci

- **GtkWindow** *about_window* – okno O aplikaci
    - **GtkLabel** – popisek s informacemi o aplikaci

8 Závěr

Pro ovládání tříosého robotu byl použit USB-CAN převodník od firmy IMF soft a k němu bylo vyvinuto ovládací rozhraní a socketový server, aby se stala sběrnice CAN síťovou službou přístupnou přes běžná rozhraní operačního systému. Dále byly vyvinuty jednotlivé typy klientů pro spojení aplikace se serverem. Pro tento účel se používá vlastního protokolu, který reflektuje potřeby aplikací na CAN sběrnici a respektuje zároveň USB-CAN převodník jako sdílený zdroj. Také při jeho návrhu bylo zohledněno, že zpřístupňuje USB zařízení, a tak server pouze reaguje na dotazy a příkazy, nikdy však data neodesílá z vlastní iniciativy.

K pozičnímu zařízení IclA D065 se vyvinula další vrstva využívající rozhraní klientů. V aplikaci je pak IclA objekt s vlastním řízením stavu a rozhraním, které zjednodušuje běžně prováděné úkony a odstiňuje zpracování chyb.

Tak bylo možné efektivně implementovat řízení robotu a koncovou aplikaci s grafickým uživatelským rozhraním, na kterém není abstrakce robotu přímo závislá.

Celek tvoří systém spolupracujících zařízení a programů. Škálovatelnost spolupráce více programů odpovídá filozofii vytváření software v prostředí operačních systémů typu UNIX. Konkrétně je použit operační systém Linux.

Použití nástroje Glade pro uložení popisu GUI ve formátu XML umožňuje efektivní překlad uživatelského rozhraní aplikace do jiných jazyků a reorganizaci prvků rozhraní v existujících oknech bez zásahu do zdrojového kódu programu.

Jako možné rozšíření a pokračování práce by bylo možné uvažovat hlubší analýzu kinematiky a dynamiky robotu, návrh simulačního modelu (například v prostředí Matlab/Simulink) a jeho ovládání při pohybu po křivých drahách. Dále rozšíření řízení o prvky umělé inteligence a informací ze senzorů, které budou na robot v budoucnu instalovány. Tak by aplikace mohla provádět korekce polohy a pracovat s nečekanými překážkami v cestě. Tímto způsobem se do budoucna počítá s podstatným rozšířením funkcionality robotu.

Literatura

[1] BOSCH: CAN Specification [online]. Verze 2.0. Stuttgart: Robert Bosch GmbH, 1991 . Dostupné z WWW: <http://www.semiconductors.bosch.de/pdf/can2spec.pdf>

[2] CAN in Automation. Application Layer and Communication Profile: CiA Draft Standard 301 [online]. Verze 4.02. Erlangen 13.2.2002 .
Dostupné z WWW: <http://www.can-cia.org/index.php?id=440> (formulář žádosti o standard)

[3] CAN in Automation. CAN in the OSI Reference Model: CiA Draft Standard 201 [online]. Erlangen, únor 1996.
Dostupné z WWW: <http://www.can-cia.org/index.php?id=440> (formulář žádosti o standard)

[4] CAN in Automation. NMT Serivce specificationl: CiA Draft Standard 203-1 [online]. Erlangen, únor 1996.
Dostupné z WWW: <http://www.can-cia.org/index.php?id=440> (formulář žádosti o standard)

[5] CAN in Automation. NMT Protocol specificationl: CiA Draft Standard 203-2 [online]. Erlangen, únor 1996.
Dostupné z WWW: <http://www.can-cia.org/index.php?id=440> (formulář žádosti o standard)

[6] CAN in Automation. LMT Service specificationl: CiA Draft Standard 205-1 [online]. Erlangen, únor 1996.
Dostupné z WWW: <http://www.can-cia.org/index.php?id=440> (formulář žádosti o standard)

[7] CAN in Automation. LMT Protocol specificationl: CiA Draft Standard 205-2 [online]. Erlangen, únor 1996.
Dostupné z WWW: <http://www.can-cia.org/index.php?id=440> (formulář žádosti o standard)

[8] IMFsoft. USB-CAN Adapter: TRIPLE drivers [online]. Verze 4.5. Ostrava: 6.7.2006.
Dostupné z WWW:
<http://imfsoft.cz/hardware/produkty/usb-can-adapter/download/cz/usb-can-adapter.pdf>

[9] SIG Positec Automation GmbH. IclA: Fieldbus manual. Verze 4.01. Lahr.

[10] SIG Positec Automation GmbH. IclA Catalogue: Integrated Positioning Drives. Vydání 04/2003. Lahr.

Seznam příloh

Příloha 1. CD s elektronickou verzí, zdrojovými kódy a uživatelským manuálem aplikace