ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ

DIPLOMOVÁ PRÁCE

CANOpen komunikace pro mobilního robota

Praha, 2008 **Autor: Jan Benda**

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 18.1.2008

Jan Benda

podpis

Poděkování

Na tomto místě bych chtěl zejména poděkovat vedoucímu diplomové práce Ing. Michalu Sojkovi, který mi poskytoval užitečné rady k této práci, ale i k jiným projektům, kterých jsem se na Katedře řídicí techniky během studia účastnil.

Můj veliký dík patří také mé rodině, která mě po celou dobu studia trpělivě podporovala.

Abstrakt

Předkládaná práce se zabývá problematikou návrhu pohonu a mechanické konstrukce autonomního robota pro soutěž Eurobot 2007. V další části pak popisuje problematiku komunikačních technologií pro distribuované řízení mechanismů s využitím CANOpen. Jsou zde popsány vytvořené knihovny a aplikace, umožňující řízení pohonů prostřednictvím sběrnice CAN a protokolu CANOpen se standardizovaným profilem pro řízení pohonů (*Motion Control Profile*). Práce je navázána na existující otevřený softwarový projekt CanFestival, implementující protokol CANOpen. Pro vlastní řízení motorů na platformě mikrokontroléru H8S/2638 je použita knihovna PXMC. Nadřazená řídící aplikace je vytvořena pro operační systém GNU-/Linux.

Abstract

This diploma thesis focuses on drive design and mechanical construction of mobile robot for Eurobot 2007 competition. Another part is devoted to problematics of communication technologies used for distributed control systems. As a part of this thesis, libraries and applications allowing drive control using CAN bus and CANopen protocol were created and described. This work is based on an existing open source project CanFestival, which is an implementation of CANopen protocol. The drive control using an H8S/2638 microcontroller is realized by means of PXMC library. The high level control application was created for GNU/Linux operating system.

České vysoké učení technické v Praze - Fakulta elektrotechnická

Katedra řídicí techniky ***Školní rok:*** 2006/2007

ZADÁNÍ D I P L O M O V É PRÁCE

Student: Jan B e n d a

Obor: Technická kybernetika

Název tématu: CANOpen komunikace pro mobilního robota

Z á s a d y p r o v y p r a c o v á n í :

1. Seznamte se pravidly soutěže Eurobot 2007, sběrnicí CAN a protokolem CANOpen. Dále prostudujte informace o použitých procesorech Renesas H8S/2638 a PPC5200.
2. Společně s ostatními členy týmu navrhněte mechaniku robotu pro soutěž Eurobot 2007.
3. Zvolte některou dostupnou implementaci CANOpen protokolu a použijte ji pro vytvoření CANOpen slave jednotky pro řízení motorů. Pro vlastní řízení motorů bude použita knihovna PXMC.
4. Zprovozněte CANOpen na "master" zařízení, které bude komunikovat s podřízenými jednotkami.
5. Vše otestujte a zdokumentujte.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Michal Sojka

Termín zadání diplomové práce: zimní semestr 2006/2007

Termín odevzdání diplomové práce: leden 2008

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

prof. Ing. Zbyněk Škvor, CSc.
děkan

V
V Praze dne 21.02.2007

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

ANSI American National Standards Institute

ATA Advanced Technology Attachment

BLDC Brush-Less DC motor

CAN Controller Area Network

CiA CAN in Automation

CNC Computer Numerical Control

CRC Cyclic Redundancy Check

DC Direct Current

DDR Double Data Rate

DMA Direct Memory Access

GND electric Ground

GPL General Public License

HMI Human-Machine Interface

HTML HyperText Markup Language

I/O Input/Output

IIC Inter-Integrated Circuit

IDE Integrated Drive Electronics

IEC International Engineering Consortium

ISO International Organisation for Standartization

LAN Local Area Network

LLC Logical Link Control

MAC Medium Access Control

MCL Monte Carlo Localization

MIPS Million Instructions Per Second

NMT Network Management

OMK Ocera Make system

OSI Open System Interconnection

PDO Process Data Object

PCI Peripheral Component Interconnect

PID Proportional-Integral-Derivative controller

PLC Programmable Logic Controller

PXMC Pikron eXtensible Motion Control

ROM Read-Only Memory

RPDO Receive PDO

RTR Remote Transmission Request

SDO Service Data Object

SDRAM Synchronous Dynamic Random Access Memory

SPI Serial Peripheral Interface

SRAM Static Random Access Memory

SYNC Synchronisation Object

TPDO Transmit PDO

USB Universal Serial Bus

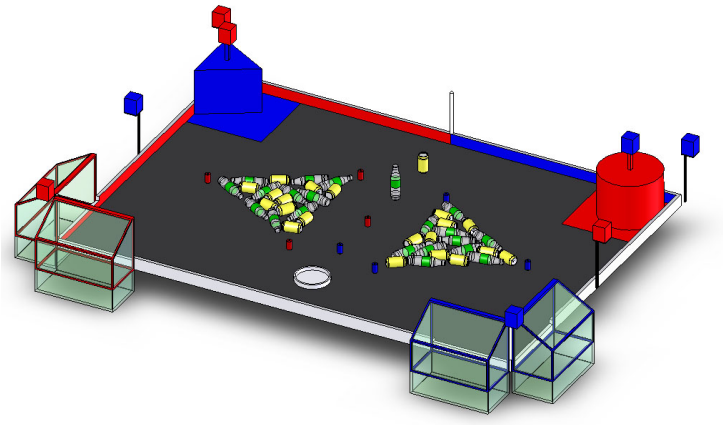
XML Extensible Markup Language

Kapitola 1

Úvod

V Evropě je každoročně pořádána mezinárodní soutěž autonomních robotů zvaná Eurobot [18]. Cílem této akce je popularizovat vědu a techniku mezi širokou veřejností. Členové univerzitních, středoškolských i mimoškolních týmů si mohou ověřit své technické dovednosti při stavbě robota a v soutěži změřit své síly s ostatními týmy. Soutěž má každoročně jiné zadání, ale většinou se jedná o sběr nejrůznějších předmětů na ploše hřiště. Úkolem robota je pak sebrané předměty roztřídit a dovézt do připravených košů. Roboti musí být zcela autonomní, to znamená, že všechny pohonné, senzorické, napájecí a řídící systémy mohou být pouze uvnitř robota a nesmí komunikovat s okolím. V zápasech se po hřišti pohybují vždy dva roboti soupeřících týmů.

Soutěž Eurobot 2007 nesla název *Robot Recycling Rally*. Konkrétní úkol robota byl sesbírat a do připravených košů roztřídit plastové láhve 0.5l, plechovky 0.33l a baterie velikosti velkého monočlánku. Hřiště má rozměry 210 $\times$ 300 cm a v jeho rozích jsou umístěny koše na roztříděné plechovky a láhve. Bílá miska na barevně roztříděné baterie má na hřišti pohyblivou pozici.



Na Katedře řídicí techniky vznikl tým studentů, jehož jsem členem, který se rozhodl autonomního robota podle pravidel postavit a soutěže se zúčastnit.

Předkládaná práce se zabývá několika tématy z oblasti řízení, použitelnými nejen v mobilní robotice. Rozebírá zejména problematiku návrhu pohonu a mechanické konstrukce robota, v další části pak popisuje problematiku komunikačních technologií pro distribuované řízení mechanismů.

V kapitole 2 je detailně popsán postup návrhu a dimenzování pohonu robota určeného pro soutěž Eurobot. Popisuje a dokumentuje se zde stavba speciálních mechanismů robota, jejichž účel vyplývá ze zadání soutěže. K této kapitole je také v příloze k dispozici protokol o výpočtu pohonu, obsahující jmenovité a maximální parametry vybraného motoru. K části kostry robota, u které se předpokládá i její použití v dalších ročnících soutěže, je také v příloze uvedena výkresová dokumentace. K ostatním součástem jsou zde uvedeny pouze situační nákresy.

V kapitole 3 bude čtenář seznámen s vlastnostmi sběrnice CAN [1] a CANOpen [12] protokolem, definujícím aplikační vrstvu komunikačního modelu. Je zde popsán mikrokontrolér H8S/2638 [20], v robotu použitý pro řízení pohonných mechanismů a 32 bitový mikrokontrolér MPC5200 s architekturou PowerPC^{TM} [14]. Ten je v robotu použit pro hlavní řídící aplikaci.

Kapitola 3 pak blíže popisuje standardizovaný profil zařízení, který specializuje komunikační protokol CANOpen pro řízení pohonů a polohovacích mechanismů. Jsou zde popsány proměnné a funkce definované standardem.

Kapitola 4 popisuje univerzální knihovnu PXMC, původně vyvinutou firmou PiKRON [5], určenou pro zpětnovazební řízení nejrůznějších typů motorů. V další části je popsán otevřený softwarový projekt CanFestival [22], který implementuje komunikační protokol CANOpen a jeho standardizované profily. Je zde uveden stručný návod na vytvoření aplikace v tomto projektu.

Dále jsou zde popsány aplikace, které implementují výše popsané hardwarové a softwarové prostředky komunikace po sběrnici CAN, protokol CANOpen a profil distribuovaného řízení pohonů. Je zde popsána vytvořená ukázková hlavní řídící aplikace, která je ovládána operátorem pomocí jednoduchých příkazů. Na podřízených uzlech komunikační sítě je pak spuštěna aplikace, která komunikuje s nadřazeným uzlem a přímo provádí řízení několika pohonných motorů. Je zde naznačena konfigurace takovéto sítě a popsány zdrojové kódy. Tím je dán návod pro řešení podobné problematiky v dalších konkrétních aplikacích nejen z oblasti robotiky.

K této práci je přiloženo CD se zdrojovými kódy a výkresy, jehož podrobnější obsah je uveden v příloze C.

Kapitola 2

Mechanika

Jedním z cílů mé diplomové práce bylo zkonstruovat a postavit autonomního mobilního robota. Poté se s sním společně s ostatními členy týmu zúčastnit soutěže Eurobot 2007. Nejprve bylo nutné prostudovat detailně pravidla soutěže [18], kde je specifikován úkol robota. Dále bylo nutno přihlédnout k technickým omezením na rozměry robota a na použité technologie. Rozhodování o funkci mechanických součástí robota, zejména těch týkajících se herní strategie, bylo diskutováno mezi všemi členy týmu. Většina dílů z hliníkového deskového materiálu byla vyfrézována na CNC stroji. Plastové desky zásobníku a ostatní drobné díly byly vyrobeny na CNC frézce vlastní výroby. Výrobní předpisy byly generovány z elektronických verzí výkresů uvedených v příloze B.

2.1 Návrh pohonu

Vzhledem k tomu, že jsme plánovali použít kostru[1] robota i s pohonem v dalších ročnících soutěže, rozhodli jsme položit velký důraz na kvalitu a životnost vybraného pohonu.

Jako pohonné motory jsme vybrali bezkomutátorové stejnosměrné motory (BLDC – *Brush–less DC motor*) od firmy MAXON^{TM}. Obecně se stejnosměrnými motory tohoto výrobce máme na Katedře řídicí techniky velmi dobré zkušenosti pro jejich vysokou kvalitu, dlouhou životnost a dobrou variabilitu ve výrobních řadách pohonů a jejich příslušenství. Výběr bezkomutátorových motorů byl jakýsi nadstandard, ale rozhodli jsme se pro ně pro jejich vyšší mechanickou odolnost a vyšší přetížitelnost. Jedinou nevýhodou je nutnost vytvoření třífázového elektrického proudu pro vytvoření rotačního magnetického pole ve statoru motoru. V dnešní době

[1] Kostru tvoří základna z 3 mm silné hliníkové desky a rám z profilů ITEM.

Tabulka 2.1: Zvolené parametry pro výpočet pohonu robota

počet pohonných motorů (diferenční řízení)	2
celková hmotnost robota	$m = 10kg$
maximální rychlost	$v = 1ms^{-1}$
maximální zrychlení	$a = 5ms^{-2}$
průměr pohonného kola	$D = 80\ mm$

je však dostatek elektronických součástek i řídících obvodů pro podobné střídače.

Parametry pro dimenzování pohonu jsou uvedeny v tabulce 2.1. Návrh mezních hodnot jsem volil s ohledem na předpokládanou hmotnost robota, na velikost hrací plochy a rychlost, kterou lze na hřišti dosáhnout. Volba vhodného průměru a šířky pohonného kolečka vycházela z dostatečné styčné plochy s povrchem hrací plochy.

Pro parametry zadané v tabulce 2.1 lze spočítat maximální otáčky kola jako

$$n_k = \frac{v.60}{\pi.D} = 238.7min^{-1}.$$

Síla působící na robota při akceleraci a přímé jízdě je

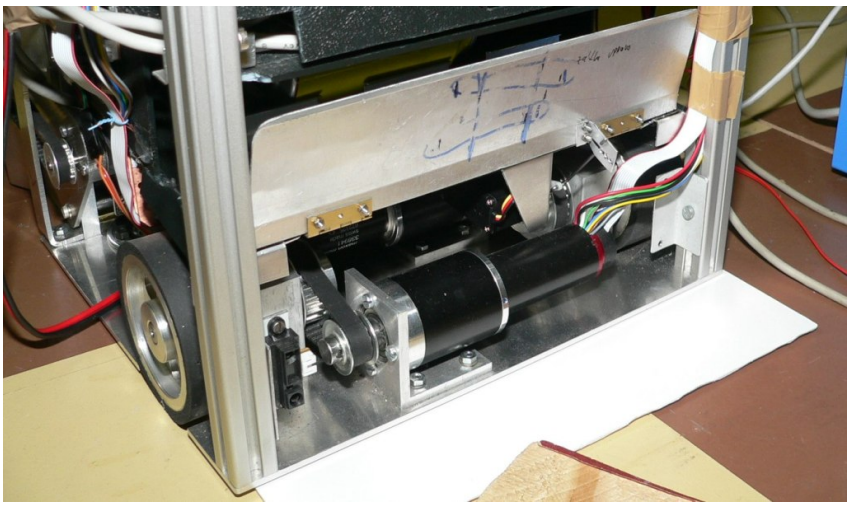
$$F = m.a = 10kg.5ms^{-2} = 50N,$$

z čehož je pro každý ze dvou pohonných motorů polovina, tedy $F_{1/2} = 25N$. Při uvažování poloměru kola (ramena síly) je kroutící moment na hřídeli kola

$$M_k = \frac{F_{1/2}.D}{2} = 1Nm.$$

Pro vypočtené parametry n_k a M_k jsem vybíral vhodný motor s převodovkou. Ve výrobní řadě požadovaných výkonů mají všechny sestavy motorů s planetovými převodovkami větší délku než je polovina šířky kostry robota. Z důvodu těchto rozměrů sestavy enkodér-motor-předovka bylo nutné osu tohoto pohonu umístit mimo osu kola. Zvolil jsem proto řešení použít motor s planetovou převodovkou a převodem ozubeným řemenem mezi hřídelí kola a převodovky. Nespornou výhodou je i možnost částečné změny převodového poměru volbou různého průměru ozubených řemenic (v použité sestavě nakonec v rozsahu cca 1:1 až 1:4). Převod ozubeným řemenem neproklouzne a dovoluje kompenzovat mírnou různoběžnost hřídelů a jiné nepřesnosti ve výrobě.

K nalezení vhodné sestavy pohonu mi velice dobře posloužil Maxon Selection Program 1.2, který je možno volně stáhnout na internetových stránkách fy. Uzimex [9]. Dokumentace k sestavě pohonu včetně pracovních charakteristik je v příloze A této práce.

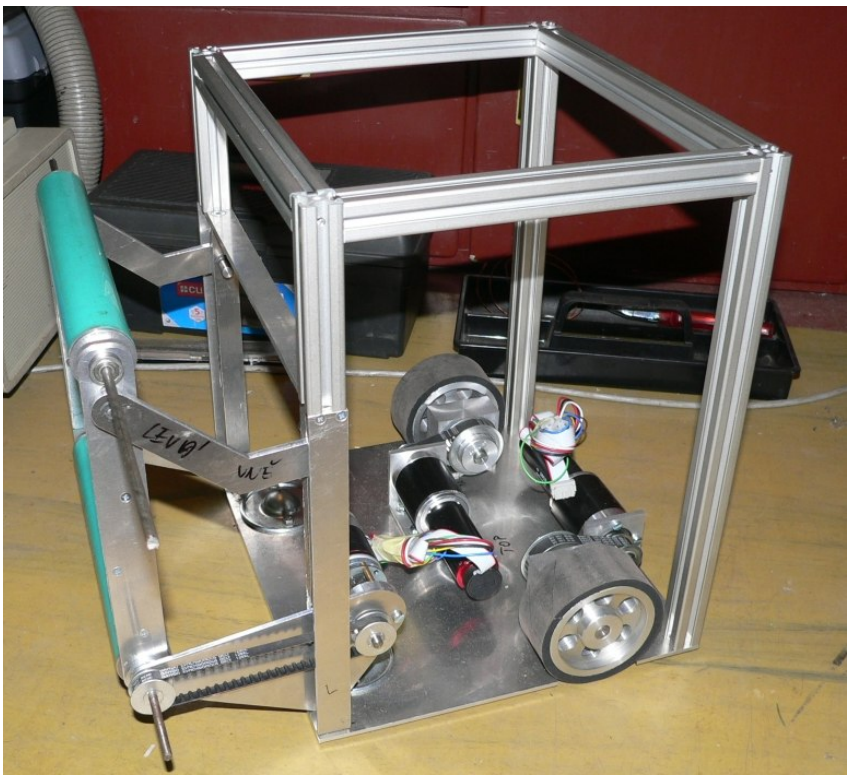


Obrázek 2.1: Motor zabudovaný v robotu

Při volbě vhodného výkonu pohonné jednotky jsem vycházel z předpokladu, že robot se bude v průběhu zápasu neustále rozjíždět a zastavovat. Návrhový kroutící moment na hřídeli kola M_k tedy bude vyvozován většinu pracovního času. Nelze počítat s tím, že se motor po rozjetí robota na jmenovitou konstantní rychlost odlehčí a bude pouze překonávat jízdní odpory. Pro vypočtené parametry n_k a M_k jsem hledal parametry motoru jmenovité, ve kterých se předpokládá nepřetržitý provoz, nikoliv mezní, kde je nutno uvažovat i dobu pohybu pracovního bodu v oblasti nad jmenovitými parametry. Jednoduše řečeno, maximální výkon potřebný k rozjezdu robota na navrhovaných parametrech nepřekročí jmenovitý výkon motoru a nemělo by tak při provozu dojít k jeho nadměrnému tepelnému zatěžování. V příloze A je uvedena momentová charakteristika vybraného motoru. Hodnoty n_k a M_k jsou položeny do jmenovitých hodnot sestavy motor-převodovka-řemenový převod.

Výpočet vlastností řemenového převodu jsem prováděl pomocí programu, který pracuje přímo s jednotlivými výrobními řadami ozubených řemenů a řemenic podle katalogu. Program pro výpočet řemenových převodů je zdarma ke stažení na internetových stránkách firmy Tyma [7].

Výkres celé sestavy pohonu je v příloze B na obrázku B.1. Malá řemenice má 12 zubů a nese označení 12XL037, velká řemenice má 24 zubů a je označena 24XL037. Použitý ozubený řemen je 76XL037.



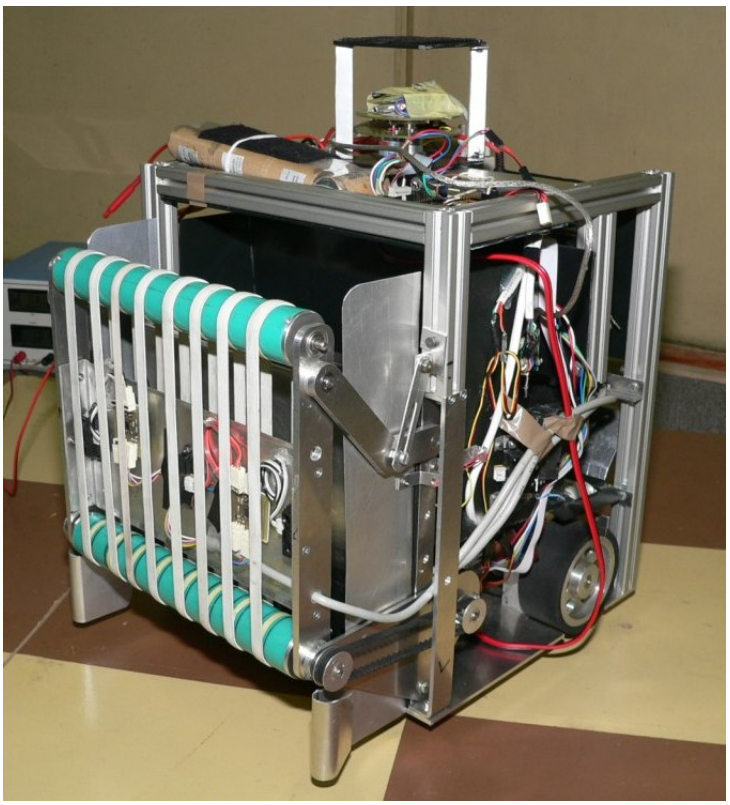
Obrázek 2.2: Kostra robota s namontovanými motory a částí sběrače

2.2 Kostra robota

Soutěž autonomních robotů Eurobot má dlouholetou tradici. Náš tým by se chtěl této soutěže účastnit nejen v roce 2007 ale i v ročnících následujících. Vzhledem k tomu, že pravidla se v omezení na maximální rozměry robota rok od roku téměř nemění, rozhodli jsme se sestrojit kostru univerzální, použitelnou i v dalších letech. Přirozeně, že vnitřní mechanismy robota se každoročně mění podle úkolu a zadání soutěže. Jako základní desku kostry jsem zvolil 3 mm silný plech z hliníkové slitiny. Její velikost byla odvozena od plánovaného půdorysu robota. Na základní desce jsou připevněny oba pohonné motory, domky s ložisky pohonných kol a dvě odvalovací kovové kuličky nahrazující přední kola. Na základně je potom vystavěn rám ze stavebnicových hliníkových profilů ITEM, určených pro stavbu jednoúčelových strojů. Výhodou je nesporně jednoduchá montáž s použitím dodávaných spojovacích prvků.

Nahrazení předních kol kuličkami umožňuje otočit robota na místě při diferenčním řízení pohonu kol. Kuličky se mohou v rovině hrací plochy odvalovat v libovolném směru.

Pohonná kola jsou vyrobena z hliníkové slitiny, soustružena a odvrtána pro odlehčení. Na jejich obvod je nalepena 6 mm silná mechová guma zajišťující přilnavost k povrchu hřiště.



Obrázek 2.3: Kompletně sestavený sběrací mechanismus na robotu

V místě kontaktu kola s plochou hřiště se guma na kole zmáčkne na tloušťku cca 4 mm. Každé z kol má svoji hřídel uloženou ve dvou jednořadých kuličkových ložiskách s prachovkami proti nečistotám. Ložiskové domky jsou taktéž vyrobeny z hliníkové slitiny. Výkresy těchto součástí jsou uvedeny v přílohách této práce. Na obrázku B.1 je základní deska s namontovaným rámem, oběma pohony, koly a odvalovacími kuličkami v přední části robota.

2.3 Sběrací mechanismus

Jak již bylo řečeno v úvodu, úkolem robota v letošním kole soutěže je sesbírat láhve, plechovky a baterie a roztřídit tyto předměty do košů v rozích hřiště. Naše herní strategie se opírala zejména o sběr plechovek a láhví. Sbírání a třídění baterií podle barvy je sice lépe bodově hodnoceno, ale odpovídá to samozřejmě vyšším nárokům na technologie spojené s jejich manipulací. Baterie jsme se rozhodli nesbírat.

Zásadním poznatkem pro sběr láhví a plechovek je jejich stejný průměr (60 mm). Několik

možných variant sběracích mechanismů jsme zkoušeli na provizorně postavených modelech. Jako nejlépe funkční se jevil svislý pásový dopravník s napnutými gumovými oky mezi dvěma rotujícími válci. Mým úkolem bylo tedy opět zvolený mechanismus realizovat, postavit a vyřešit jeho připevnění na kostru robota.

Pravidla soutěže také hovoří o dvojích maximálních rozměrech robota. Jeden z nich je maximální obvod robota v době startu zápasu (max. 120 cm) a druhý je maximální obvod robota v průběhu zápasu (max. 140 cm). Využil jsem této skutečnosti a volil sběrací mechanismus jako vyklápěcí. Spodní rotační válec je poháněn stejnosměrným modelářským motorem pomocí ozubeného řemínku. Pohybem zarážek je možné volit výšku spodního sběracího válce nad povrchem hřiště. K vyklopení sběrače po startu zápasu stačí gravitační síla, odjištění je řešeno malým modelářským servomotorem. Automaticky dojde k vyklopení plechových bočnic, vymezujcích obdélníkový kanál dopravníkové části sběrače.

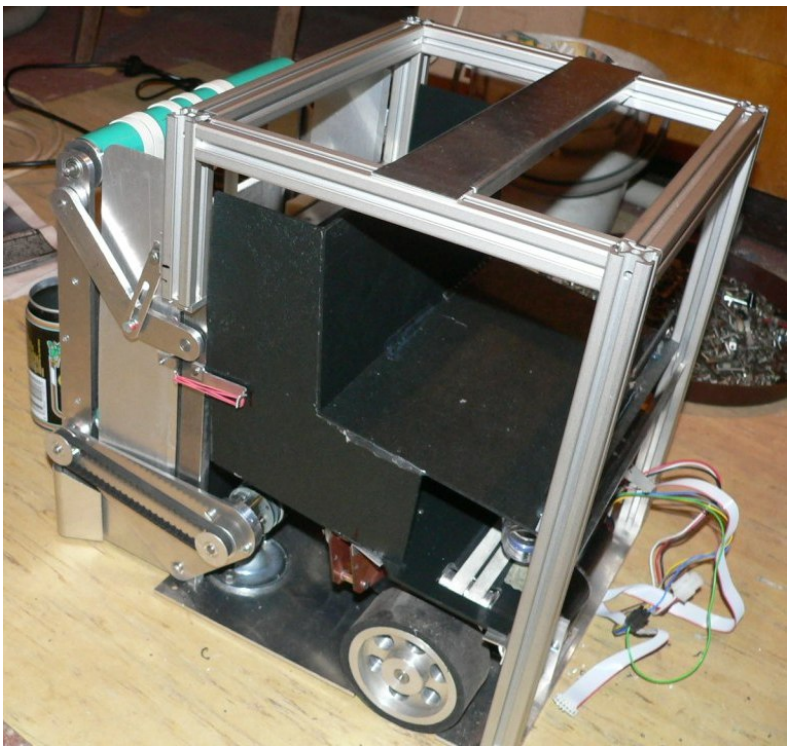
Funkce sběracího mechanismu je taková, že robot popojíždí pomalu vpřed a pomocí senzorů hledá sbíraný předmět. Pokud dojde ke kontaktu láhve nebo plechovky se spodním rotujícím válcem, je tento předmět vtažen na jazyk sběrače. Ve chvíli, kdy senzory natažení předmětu rozpoznají, jazyk sběrače se nakloní pomocí servomotoru. Jazyk je zhotovený z tenkého plechu a vynese sebraný předmět do prostoru, kde jej zachytí pás dopravníku a vyveze jej do horní části robota, kde je ústí zásobníku. Pokud se stane, že se plechovka, láhev nebo více předmětů najednou ve sběrači vzpříčí, nelze zvedout jazyk. Řešení této situace je ve změně směru rotace sběracích válců. Dojde tak k vyhození všech předmětů ze sběrače a po pootočení nebo popojetí robota do mírně odlišné pozice se opakuje sběrací manévr znovu. Veškeré mechanismy jsou ovládány z počítače MPC5200 (popsán v sekci 3.2), kde je spuštěna řídící aplikace.

Na obrázku B.3 je nákres sběrače v otevřeném a uzavřeném stavu, na obrázku 2.3 je fotografie sběrače přímo na robotu.

2.4 Zásobník

Po úspěšném sběru láhve nebo plechovky putuje tento předmět do zásobníku. Ten má profil zahnutého kanálu s obdélníkovým průřezem. Z bočního pohledu je zásobník zahnut téměř do pravého úhlu. Umožňuje to předmětu padajícímu z horní části sběrače srovnat se tak, aby se v zásobníku nevzpříčil. Předměty jsou v zásobníku v pořadí, ve kterém byly sebrány. Zásobník pojme až pět předmětů. Potom je nutné dojet ke košům a sebrané předměty roztřídit do košů.

Součástí zásobníku je v zadní části robota vyhazovací mechanismus. Vzhledem k tomu,

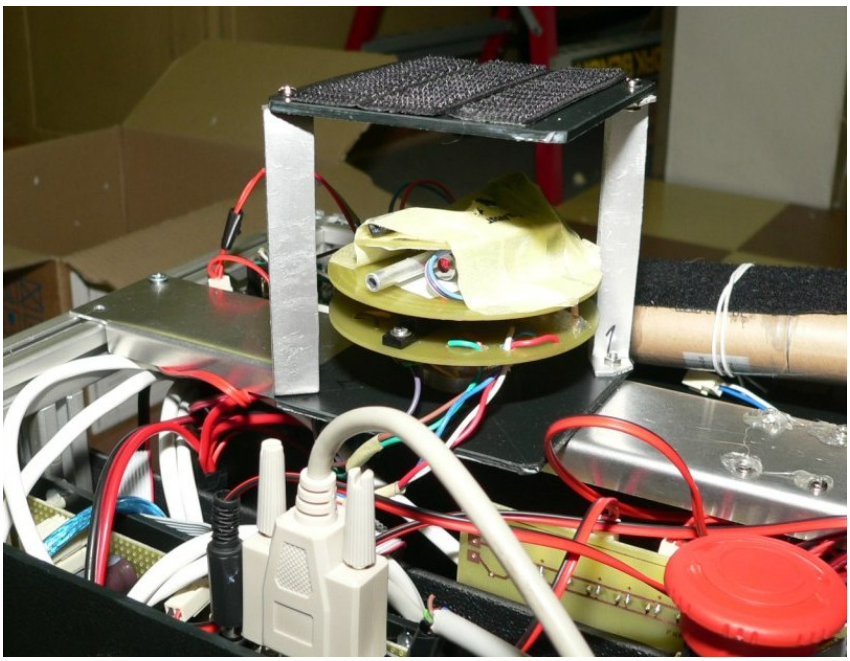


Obrázek 2.4: Zásobník připevněný ke kostře robota

že koše na odpad jsou ve dvou rozích hřiště, zvolili jsme strategii hry tak, že robot do rohu s koši zajede a bez nutnosti otáčení bude moci vyhazovat zároveň plechovky dozadu a láhve do obou boků robota. Pro vyhození plechovky jsou na konci zásobníku ze zadní části robota dvířka. Jejich otevřením pomocí servomotoru se může plechovka volně vykutálet do koše. Pro vyhození láhve do obou stran robota slouží gumový pás, poháněný přes převodovku stejnosměrným motorem a umístěný na dně zadní části zásobníku. Aby při vyhazování láhve nebo plechovky nedošlo k vypadnutí i všech ostatních předmětů uložených v zásobníku, je tento vybaven přepážkou ovládanou servomotorem. Sekvence ovládacích příkazů všech mechanismů zásobníku je také řízena z hlavní aplikace. Boční pohled na řez zásobníkem je na obrázku B.3.

2.5 Lokalizační maják

Pravidla soutěže Eurobot umožňují umístění třech reflexních odrazových válců po obvodu obdélníkového hřiště. S výhodou je tedy možné použít nějakou z absolutních optických metod určení polohy. Lokalizace samostatnou odometrií vede na časem se zvětšující neurčitost polohy



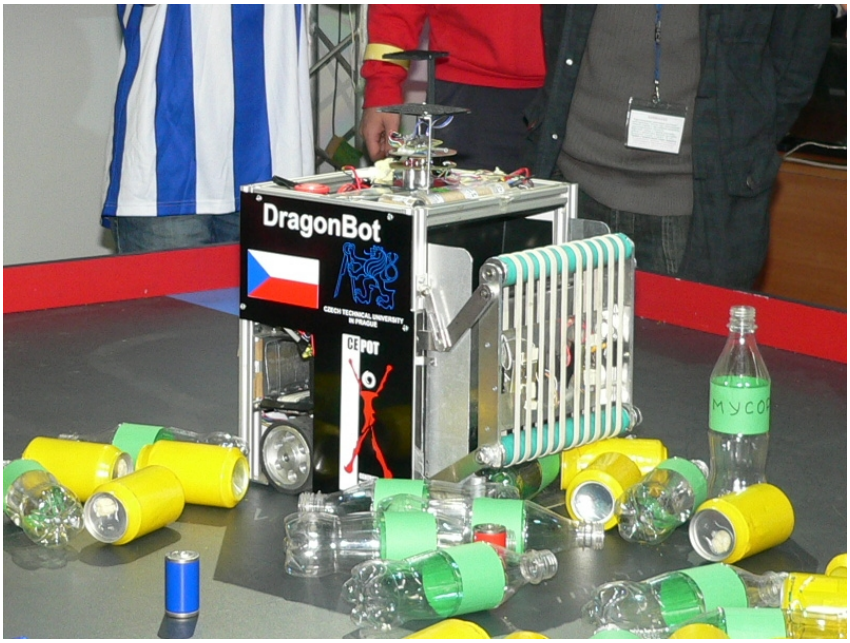
Obrázek 2.5: Lokalizační maják namontovaný v horní části robota

a zcela se ztrácí ve chvíli prokluzu kola po podložce. Algoritmus MCL (*Monte Carlo Lokalization*) je pro kombinaci absolutní a relativní metody určení polohy často používaný. Z důvodů velmi specifického zadání, omezení na rozměry a kompatibilitu s dalšími technologiemi robota, nebylo možné použít žádné z dostupných zařízení na trhu. Technická omezení zařízení jsou popsána v pravidlech soutěže [18].

Modul majáku je vybaven sběrnicí CAN, kterou je připojen k řídícímu počítači MPC5200, kde je prováděna numerická filtrace a výpočet polohy robota.

Vlastní laserový maják se skládá z pevné a otáčející se části. Na pevné části je umístěn pohonný komutátorový motor vybavený inkrementálním senzorem polohy. Dále je zde instalována infračervená optická závora poskytují synchronizační impuls jednou za každou otáčku destičky s optikou. K přenosu energie a signálu od laserového přijímače slouží tři sběrací kartáče ze studiových potenciometrů. Základem rotační části je cuprextitová deska s vyfrézovanými sběracími kroužky. Na ní je umístěn laserový vysílač s fokusační optikou. V těsné blízkosti je rovnoběžně umístěn optický přijímač s fototranzistorem, taktéž s ostřící optikou. Rotační část je precizně uložena ve dvou kuličkových ložiskách, tedy se zanedbatelnou vůlí. Nákres je na obrázku B.11, na obrázku 2.5 je fotografie majáku osazeného na robotu. Výrobní výkresy lokalizačního majáku nejsou v této práci uvedeny, neboť ještě plánujeme úpravy jeho mechaniky pro letošní ročník soutěže.

V průběhu testování tohoto laserového majáku se ukázalo jedno úskalí použitého technického řešení. Měděné sběrací kroužky postupem času zoxidovaly a na kontaktu uhlíku kartáče a mědi vznikal nezanedbatelný přechodový odpor. Tento problém byl krátkodobě řešen očištěním měděné vrstvy. Později byla přidána ještě druhá trojice kartáčů, která zmenšila počet rušivých impulzů v přenášeném signálu vlivem proměnného přechodového odporu po obvodu kroužků. Použité řešení bylo rychlé (výroba celého majáku trvala méně než dva dny), ale s krátkou životností. V současné době navrhujeme úpravu pro přenos signálu z přijímače jinou cestou, např. optickou. Uvažovaný je také přesun části elektroniky z rotační části na pevnou a použití rotujícího zrcadla.



Obrázek 2.6: Náš robot v konečné podobě na soutěži v Petrohradu

Kapitola 3

Komunikace po sběrnici CAN

V této kapitole budou nejprve ve stručnosti přiblíženy dva typy mikrokontrolérů, které jsou v robotu použité pro řízení. Jejich společnou vlastností je jejich výbava rozhraním sběrnice CAN (*Controller Area Network*), která je v robotu ke komunikaci použita. Dále je zde popsán protokol CANOpen, definující aplikační vrstvu komunikačního ISO/OSI modelu sběrnice CAN. Detailněji je pak přiblížen specializovaný profil protokolu CANOpen pro řízení pohonů a polohovacích mechanismů.

3.1 Mikroprocesor H8S/2638

H8S/2638 je 16 bitový mikroprocesor, zaměřený na použití v aplikacích vyžadující širokou výbavu periferiemi. Jeho výrobcem je firma Renesas. Použití tohoto mikroprocesoru má na katedře řídicí techniky velkou tradici. Existuje pro něj kompilátor jazyka C na platformě Linux s licencí GNU a řada již připravených vývojových nástrojů [2]. Na některých deskách plošných spojů s tímto mikroprocesorem je osazena externí paměť programu typu RAM. Při ladění řídícího programu je výhodnější nahrávat program do této RAM paměti, neboť jednou z mála nevýhod tohoto mikroprocesoru je malý počet garantovaných přepsání paměti FLASH vlivem starší použité technologie čipu paměti.

Ve zkratce uveďme několik základních parametrů a funkcí.

- Operační frekvence do 20MHz
- Napájecí napětí v rozmezí 4.5V – 5.5V
- Velikost paměti 256kB FLASH a 16kB RAM

- Adresní prostor o velikosti do 16MB
- Zabudované násobení dvou 16 bitových registrů
- 2 $\times$ řadič sběrnice CAN, splňující specifikaci CAN 2.0B
- Šest 16 bitových čítačů/časovačů
- 16 kanálů s výstupem PWM pro řízení motorů
- Tři sériové kanály
- 10 bitový A/D převodník s 12 kanály
- Osmi bitový D/A převodník, 3 kanály
- 55 interních a 7 externích zdrojů přerušení
- Řadič rychlého přenosu dat (DTC), 85 kanálů
- Watchdog časovač, 2 kanály
- Sériové komunikační rozhraní SCI, 3 kanály
- I^2C komunikační rozhraní, 2 kanály
- 72 I/O pinů, 12 pouze jako vstupních

Vzhledem k tomu, že jsem na tomto mikroprocesoru vyvíjel nadstavbu pro komunikaci CANOpen, přiblížím více pouze funkci CAN řadiče.

Mikroprocesor je vybaven dvěma nezávislými řadiči CAN sběrnice. Každý z nich využívá svojí řadu konfiguračních registrů a 16 schránek pro příjem či odeslání zprávy. Oba kanály jsou shodné, liší se pouze polohou registrů v paměti a výstupními piny. Každý kanál lze odděleně zapnout a je možné využít i módu snížené spotřeby.

Již na hardwarové úrovni v řadiči sběrnice je možné provádět filtrování příchozích zpráv. Při příjmu zprávy se nejdříve vyhodnotí její bezchybný přenos pomocí CRC součtu. Poté se řadič snaží zprávu uložit v některé ze schránek, které byly při inicializaci nastaveny pro příchozí zprávy. Řadič porovnává identifikátor příchozí zprávy s identifikátorem uloženém v registru schránky. V případě shody identifikátorů se příchozí zpráva do schránky uloží, v opačném případě se proces porovnání identifikátorů provádí u další schránky. Hledání volné schránky probíhá od té s číslem 15 až po schránku s číslem 0, která je nastavena pouze pro příjem.

Pokud je ve schránce nevyzvednutá příchozí zpráva a je přepsána další příchozí zprávou, inkrementuje je speciální registr řadiče značící počet nepřečtených přepsaných zpráv. Pokud se stane že zpráva je odmítnuta všemi vstupními schránkami, dojde se až ke schránce 0. Zde se provádí porovnávání identifikátorů nikoliv prostou shodou, ale podle bitové masky. Ve speciálním registru lze nastavit libovolnou masku a dovolit tak příjem zpráv s určitým rozsahem identifikátorů. Metoda filtrování zpráv bitovou maskou se nazývá *Local Acceptance Filtering*.

Kompletní technická specifikace mikrokontroléru H8S/2638 je k dispozici na internetových stránkách výrobce [20].

3.2 MPC5200

MPC5200 je malý, velice výkonný průmyslový mikrokontrolér založený na architektuře procesorového jádra PowerPC™. Na základní desce BOA5200, která tvoří konektorové rozhraní pro modul MPC5200, jej používáme jako hlavní řídící počítač v autonomním robotu pro soutěž Eurobot. Operačním systémem je Linux s jádrem 2.6. Pomocí sběrnice CAN se čtou senzory, řídí pohony a vnitřní mechanismy robota. Pomocí rozhraní LAN je k počítači připojeno zařízení pro bezdrátovou komunikaci pomocí WiFi, kterou je možné použít pro dálkové řízení.

Ke komunikaci pomocí sběrnice CAN je použito programové rozhraní Socket_CAN [6]. CanFestival, popsaný v sekci 4.2, umí s tímto rozhraním pracovat.

Uveďme zde některé základní vlastnosti počítače MPC5200.

- Jádro MPC603e ze série G2_LE core
  - výkon 760 MIPS při frekvenci 400MHz
- SDRAM / DDR rozhraní pamětí
  - až do frekvence sběrnice 133MHz
  - 256MB adresní prostor
  - 32 bitová datová sběrnice
- Multifunkční rozhraní sběrnice
  - podpora ROM/Flash/SRAM pamětí
  - 8/16/32 bitový přístup při 26 bitovém adresování

- Ovladač sběrnice PCI (*Peripherial Component Interconnect*)
  - 32 bitová adresová a datová sběrnice
  - 33MHz a 66MHz frekvence
- Ovladač sběrnice ATA – možnost připojení IDE disku
- Best Comm DMA subsystém s ovladačem pro přímý přístup do paměti
- 6 programovatelných sériových rozhraní PSC (*Programable Serial Controller*)
- Rozhraní ethernetu pro 10/100Mbit/s síť
- USB1.1 rozhraní
- I^2C (*Inter-Integrated Circuit Interface*), SPI (*Serial Peripherial Interface*)
- Dvě rozhraní CAN2.0 A/B kompatibilní

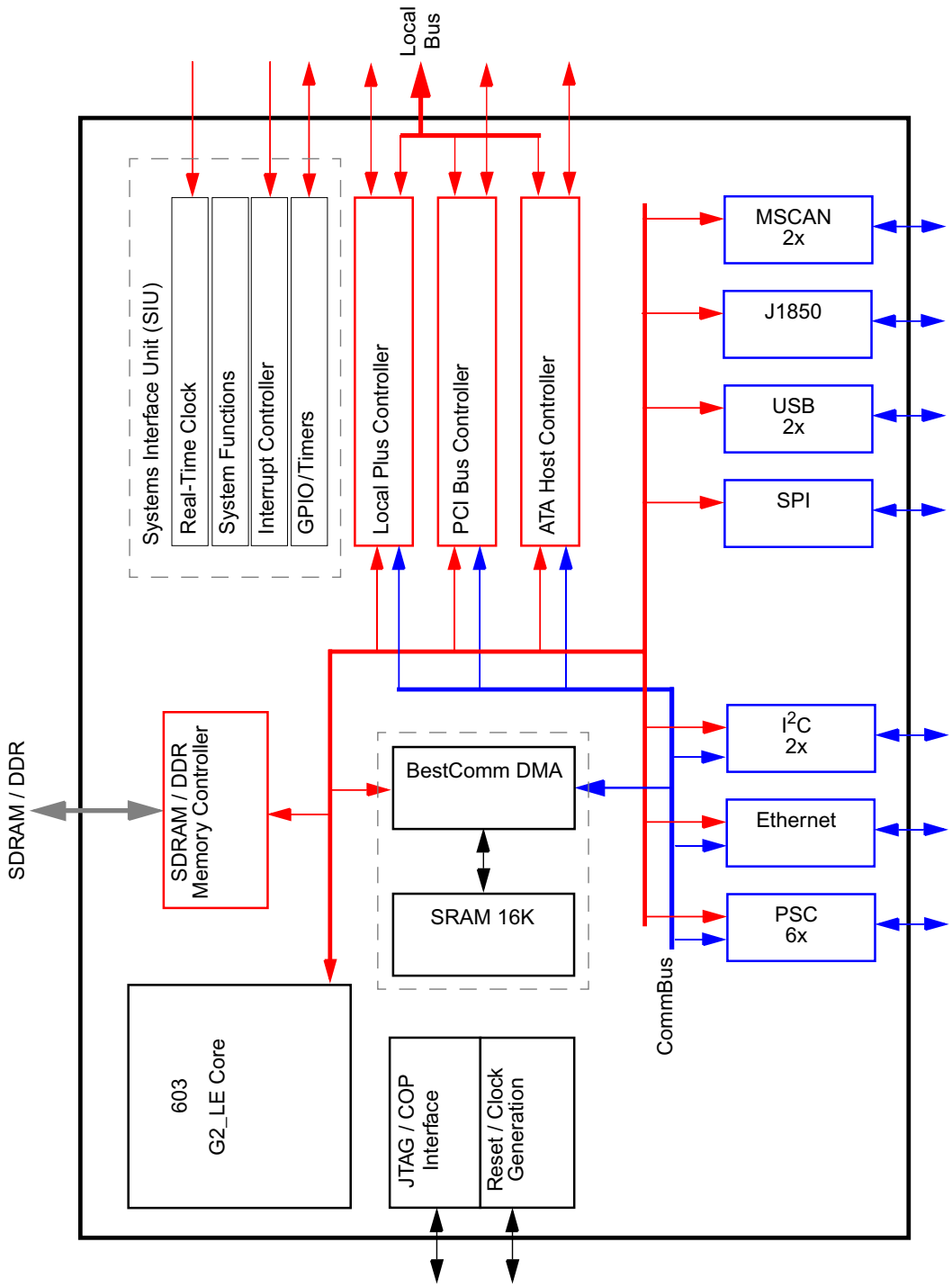
Podrobný technický popis mikrokontroléru MPC5200 je v [14] odkud je převzato blokové schéma na obrázku 3.1.

3.3 Sběrnice CAN

CAN (Controller Area Network) je sériová sběrnice, vyvinutá firmou BOSCH pro automobilový průmysl na začátku 90. let. V roce 1993 byl vydán standard ISO 11898, který pod označením CAN 2.0A popisuje fyzickou a linkovou vrstvu ISO/OSI modelu. Později byla specifikace linkové vrstvy rozšířena o CAN 2.0B. Od této chvíle se rozlišuje mezi datovým rámcem se standardním identifikátorem o délce 11 bitů a rozšířeným identifikátorem o délce 29 bitů.

CAN protokol byl navržen tak, aby umožňoval distribuované řízení systémů v reálném čase, s přenosovou rychlostí do 1Mb/s a vysokým stupněm zabezpečení. Protokol je typu multi-master, což umožňuje iniciovat přenos dat více uzlům.

Vzhledem k snadnému nasazení a vysoké spolehlivosti se protokol CAN rozšířil i do jiných průmyslových odvětví.



Obrázek 3.1: Blokové schéma počítače MPC5200

Tabulka 3.1: Maximální délka sběrnice v závislosti na požadované přenosové rychlosti

Přenosová rychlost	Délka
1Mbit/s	30 m
800kbit/s	50 m
500kbit/s	100 m
250kbit/s	250 m
125kbit/s	500 m
62.5kbit/s	1000 m
20kbit/s	2500 m
10kbit/s	5000 m (s opakovačem)

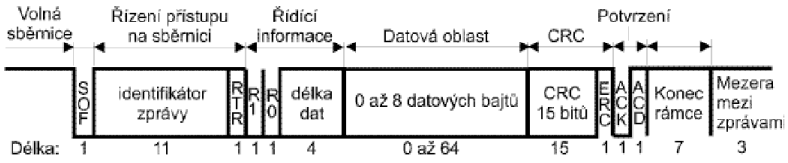
3.3.1 Fyzická vrstva

Základní vlastností fyzického přenosového média je realizace logického součinu. Protokol CAN definuje dvě odlišné úrovně na sběrnici *dominant* a *recessive*. Stav na sběrnici je výsledkem logického součinu úrovní vysílaných všemi uzly. Vysílají-li všechny uzly úroveň *recessive*, je na sběrnici úroveň *recessive*. Vysílá-li alespoň jeden uzel úroveň *dominant*, je na sběrnici úroveň *dominant*.

Vedení sběrnice obsahuje vodiče s označením CAN-H (*High*), CAN-L (*Low*), zemní vodič GND, stínění a volitelně také napájení 5V. Zmíněná recesivní úroveň je definována tak, že napětí na vodiči CAN-H není vyšší než napětí na vodiči CAN-L + 0.5V. Dominantní úroveň je pak určena napětím v rozsahu 3.5V až 5V na vodiči CAN-H a zároveň napětím menším než 1.5V na vodiči CAN-L, obojí proti zemi GND. Vedení sběrnice je na obou stranách zakončeno terminátory 120 Ω. Diferenční uspořádání sběrnice zvyšuje odolnost proti souhlasnému napěťovému rušení. Maximální přenosovou rychlost pro různé délky fyzického vedení popisuje tabulka 3.1. Omezením je rychlost šíření elektrického signálu ve vedení dána indukčností a kapacitou mezi vodiči. Více v [11].

3.3.2 Linková vrstva

Linkovou vrstvu lze rozdělit na dvě podvrstvy – MAC a LLC. MAC (*Medium Acces Control*) vrstva má na starosti kódování dat, vkládání doplňkových bitů do komunikace (*Bit Stuffing*), řízení přístupu k médiu s uvážením priority zprávy, detekci a hlášení chyb a potvrzování



Obrázek 3.2: Formát datové zprávy sběrnice CAN

správně přijatých zpráv. Vrstva LLC (*Logical Link Control*) filtruje přijaté zprávy (*Acceptance Filtering*) a provádí hlášení o přetížení sítě (*Overload Notification*).

3.3.3 Datová zpráva

Datová zpráva je základním prvkem komunikace uzlů po sběrnici. Umožňuje přenést 0 až 8 bytů dat. Protokol CAN definuje dva typy datových zpráv. První z nich je definován specifikací CAN 2.0A a je označován jako standardní formát zprávy s délkou identifikátoru 11 bitů. Specifikace CAN 2.0B definuje navíc tzv. rozšířený formát zprávy, lišící se od standardního pouze 29 bitovým identifikátorem. Oba typy zpráv mohou být na sběrnici používány současně. Řadič standardu 2.0A nemůže rozšířené zprávy přijímat, ale neoznačuje je za chybné. Části zprávy s identifikátorem a RTR bitem se říká arbitrážní pole (*Arbitration field*).

Vyslání datové zprávy je možné pouze tehdy, pokud je sběrnice volná. Pokud je na sběrnici více uzlů, které chtějí odeslat datovou zprávu a detekují volnou sběrnici ve stejném okamžiku, probíhá rozhodování podle priority zpráv určené identifikátorem. Hodnota identifikátorů zpráv vysílaných více uzly je porovnávána bit po bitu logickým součinem, jenž poskytuje přímo fyzická vrstva. Vyšší prioritu na konkrétním bitu tedy má zpráva s dominantní logickou úrovní a uzel který detekuje opačnou úroveň než sám vysílá, ukončuje další vysílání zprávy. Nejvyšší dosažitelnou prioritu udává identifikátor 0_h. Na obrázku 3.2 popsán formát datové zprávy.

3.3.4 Ostatní typy zpráv

Žádost o data Prostřednictvím této zprávy žádá zařízení vzdálený uzel o zaslání dat. Od klasické datové zprávy se liší nastaveným bitem RTR na úroveň *recessive* a neobsahuje datovou část. Vzdálený uzel odešle zpět datovou zprávu o stejném identifikátoru s jakým žádost o data přijal. Datová zpráva má oproti žádosti o data se stejným identifikátorem vyšší prioritu z důvodu přítomnosti dominantní úrovně na pozici RTR bitu.

Chybová zpráva Tuto zprávu[1] generuje kterýkoliv uzel, který detekuje chybu v přenášené zprávě (např. chyba bitu, chyba CRC, chyba vkládání bitů, chyba rámce). Na sběrnici vyšle šestici recesivních nebo dominantních bitů. Poruší se tak pravidlo o vkládání bitů a vyvolá se odeslání chybové zprávy i u ostatních uzlů. Po odeslání šestice bitů chybové zprávy každý uzel vysílá sedm recesivních bitů a dává tak prostor k odeznění indikace chyby od všech uzlů. Poté se některý z uzlů opět pokusí o standardní vyslání datové zprávy.

Zpráva o přetížení Zpráva o přetížení slouží k oddálení vysílání další datové zprávy nebo žádosti o data. Zpravidla tento způsob využívají zařízení, která nejsou schopna kvůli svému vytížení přijímat a zpracovávat další data. Struktura této zprávy je podobná zprávě o chybě, ale její vysílání může být zahájeno až po konci zprávy (*End of Frame*), konci oddělovače chyb nebo předcházejícího oddělovače zpráv přetížení.

3.4 CANOpen standard

S fyzickou a linkovou vrstvou ISO/OSI síťového modelu sběrnice CAN jsme se seznámili v kapitole 3.3. Aby byla umožněna interoperabilita více zařízení od různých výrobců i na programové úrovni, byl v devadesátých letech vydán standard CANOpen, definující aplikační vrstvu ISO/OSI modelu. Norma CANOpen je definována mezinárodní organizací CiA *(CAN in Automation)*, která sdružuje výrobce a uživatele působící v automatizačním průmyslu.

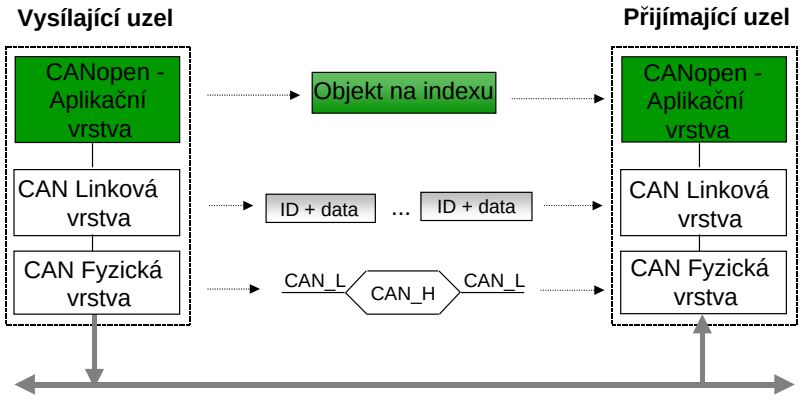
Obrázek 3.3 ukazuje spojení dvou CANOpen zařízení s naznačením komunikačních prostředků na různých úrovních zjednodušeného ISO/OSI modelu.

3.4.1 Model CANOpen zařízení

Základní strukturu CANOpen zařízení lze rozdělit do tří bloků, jak je uvedeno na obrázku 3.4.

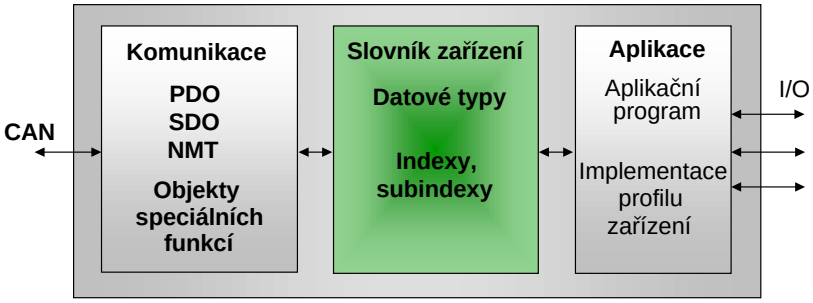
- **Komunikace** – poskytuje prostředky pro přenos dat po síti. Využívá k tomu procesní datové objekty (PDO), servisní datové objekty (SDO), servisní objekty správy sítě (NMT) a objekty se speciálních funkcí.

[1] Nejedná se o zprávu v pravém slova smyslu, jde pouze o sekvenci bitů stejné úrovně, nepřenášející žádná data ani identifikátor.



Obrázek 3.3: ISO/OSI model podle CANOpen standardu

- **Slovník zařízení** – (*Object Dictionary*) je globální datová struktura, přístupná prostřednictvím sítě. Jsou zde uloženy definice datových typů a parametry veškerých komunikačních objektů.
- **Aplikace** – implementuje funkce pro práci se slovníkem zařízení a komunikačními objekty, definuje také funkce specifické pro použitý profil zařízení.



Obrázek 3.4: Blokové schéma CANOpen zařízení

3.4.2 Slovník zařízení

Nejdůležitější částí každého CANOpen zařízení je jeho tzv. slovník zařízení. Jak již bylo naznačeno výše, sdružuje veškeré objekty přístupné přes síť, definuje jejich typ a počáteční nastavení. Přístup do slovníku je pomocí 16 bitových indexů a osmibitových subindexů. Rozložení

Tabulka 3.2: Skupiny indexů ve slovníku zařízení

Indexy	Objekty
0000_h	nevyužit
0001_h – $001F_h$	statické datové typy
0020_h – $003F_h$	komplexní datové typy
0040_h – $005F_h$	komplexní datové typy specifikovatelné výrobcem
0060_h – $007F_h$	statické datové typy specifikovatelné profilem zařízení
0080_h – $009F_h$	komplexní datové typy specifikovatelné profilem zařízení
$00A0_h$ – $0FFF_h$	rezervováno pro budoucí využití
1000_h – $1FFF_h$	parametry komunikačního profilu
2000_h – $5FFF_h$	parametry komunikačního profilu specifikovatelné výrobcem
6000_h – $9FFF_h$	parametry standardizovaného profilu zařízení
$A000_h$ – $BFFF_h$	parametry standardizovaného rozhraní
$C000_h$ – $FFFF_h$	rezervováno pro budoucí využití

informací na indexech ukazuje tabulka 3.2. Indexy v rozsahu 0000_h až $0FFF_h$ jsou určeny pro definice typů a jsou přístupné pouze lokálně.

3.4.3 Komunikační objekty

CANOpen používá pro zprávu sítě, zasílání dat a řízení komunikace následující komunikační objekty.

- Procesní datové objekty (PDO) – slouží pro přenos procesních dat v reálném čase
- Servisní datové objekty (SDO) – umožňují zápis a čtení položek slovníku zařízení
- Objekty pro zprávu sítě (NMT)
  - NMT Message Object – slouží k ovládání komunikačních stavů zařízení
  - Boot-up Object – zasílá zprávu o zapnutí zařízení
  - Error Control Object – posílání chybových zpráv
- Objekty speciálních funkcí
  - Synchronization (SYNC) – zajišťují synchronizaci zařízení na síti

- Time Stamp – umožňuje zasílání časových značek
- Emergency (EMCY) – posílání nouzových zpráv

3.4.4 PDO zprávy

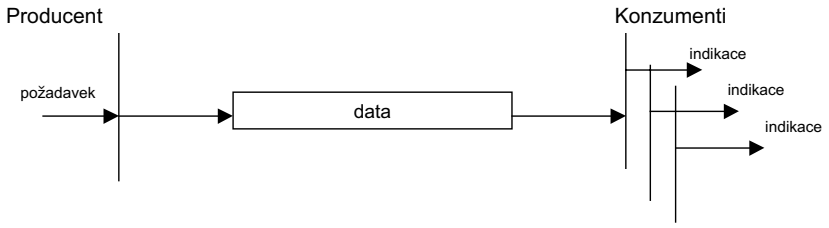
Přenos procesních dat v reálném čase je uskutečňován prostřednictvím PDO zpráv. Tento způsob komunikace je popisován modelem producent/konzument. Jedna PDO zpráva může nést maximálně 8 bytů dat a její formát je shodný s datovou zprávou sběrnice CAN. Z tohoto důvodu zde nejsou žádné nároky na další režii přenosu a tak je přenos dat touto metodou velice efektivní.

Ke každé PDO zprávě je ve slovníku zařízení uloženo několik parametrů. V první řadě je to tzv. mapování, určující které položky slovníku zařízení budou touto zprávou přenášeny. Dalšími parametry jsou identifikátory zprávy, registrace události iniciující její vyslání, doba platnosti a další. PDO zpráva jedním uzlem vysílaná (TPDO) musí být nastavena jako přijímaná (RPDO) na jednom nebo více ostatních uzlech. I mapování je zpravidla nastaveno tak, že položka slovníku zařízení na jednom uzlu je po přenesení sítí uložena na stejnou položku uzlu jiného. Není to ale nutnou podmínkou.

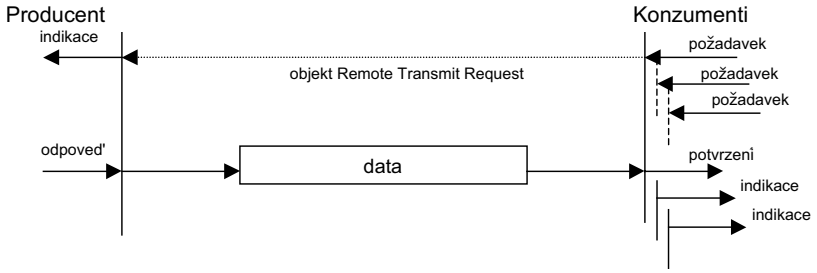
Odeslání PDO zprávy může být způsobeno synchronní nebo asynchronní událostí. Synchronní a zpravidla periodické odesílání PDO zpráv probíhá v závislosti na přijetí synchronizačního objektu (SYNC), který může vysílat buď to uzel sám nebo jej přijímá od vzdáleného uzlu pomocí sítě. Asynchronní odeslání PDO zprávy může být způsobeno vnitřní událostí z programu příslušného uzlu, jako je přerušení nebo zapsání aktuální hodnoty do mapované proměnné. Možné je také odeslání PDO zprávy po vzdáleném požadavku na data (RTR – *Remote Transmittion Request*).

Předávání PDO zpráv je na úrovni aplikační vrstvy považováno za nepotvrzovaný způsob komunikace. Může se stát, že žádný vzdálený uzel vyslanou zprávu nepřijme, protože nemá nastaveno mapování a komunikační parametry stejně jako u odesílajícího uzlu. Na úrovni linkové vrstvy komunikačního modelu samozřejmě k potvrzení příjmu vzdálenými uzly dojde po vyhodnocení CRC kódu zprávy. Aplikační vrstva se toto dovídá pouze v případě chyby prostřednictvím chybového hlášení. Přijetí PDO zprávy je indikováno pouze na straně příjemce. Odesílání PDO zprávy je ve specifikaci DS-301 [12] označováno jako *Write PDO Protocol* a situaci znázorňuje obrázek 3.5.

Druhý způsob přenosu PDO zprávy je na vzdálenou žádost (RTR – *Remote Transmittion Request*), označovaný ve specifikaci jako *Read PDO Protocol*. V tomto případě jeden nebo více



Obrázek 3.5: Nepotvrzovaný přenos PDO zprávy (*Write PDO protocol*)

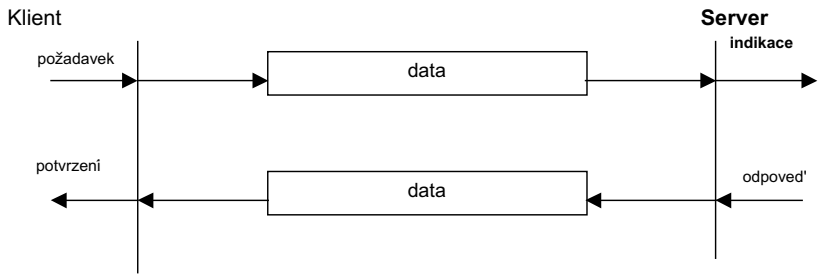


Obrázek 3.6: Potvrzovaný přenos PDO zprávy (*Read PDO protocol*)

příjemců PDO zpráv odešle do celé sítě požadavek na data. RTR objektem adresovaný uzel na žádost odpoví odesláním PDO zprávy. Na straně žadatele o data je přijetí zprávy indikováno a tak je tento způsob považován za potvrzovaný přijetím žádaných dat. Průběh přenosu PDO zprávy tímto způsobem znázorňuje obrázek 3.6

3.4.5 SDO zprávy

Pomocí SDO zpráv je umožněn přístup k libovolné položce slovníku zařízení, která je přes síť dostupná. SDO zprávy se využívají hlavně ke konfiguraci vzdálených uzlů. Komunikační model přenosu těchto zpráv je klient/server, kde klient iniciuje přenos a server je uzel vlastnící slovník zařízení na který se přistupuje. SDO klientem je zpravidla ten uzel, který konfiguruje jiné uzly a čte nebo nahrává do jejich slovníků zařízení konfigurační parametry. SDO serverem je pak uzel konfigurovaný. Jeden uzel může být zároveň klientem i serverem pro různé SDO zprávy. Důležitou vlastností tohoto typu zpráv je možnost přenosu většího množství dat v blocích, skládajících se z několika segmentů. Z toho důvodu přenos vyžaduje velkou režii, kdy se oba zúčastněné uzly navzájem předem informují o počtu, délce a formátu přenášených dat. Obrázek 3.7 naznačuje přenos SDO zprávy.



Obrázek 3.7: Přenos SDO zprávy

Pro přenos jednoho bloku dat je situace o něco jednodušší a služby nutné pro jeho uskutečnění lze rozdělit na následující.

- SDO Download
  - zahájení SDO downloadu
  - download SDO segmentu
- SDO Upload
  - zahájení SDO uploadu
  - upload SDO segmentu
- Ukončení SDO přenosu

Pro větší množství dat je nutné použít přenos bloku po více segmentech. Služby přenosu lze rozdělit na níže uvedené.

- SDO blokový download
  - zahájení downloadu bloku
  - download bloku
  - ukončení downloadu bloku
- SDO blokový upload
  - zahájení uploadu bloku
  - upload bloku

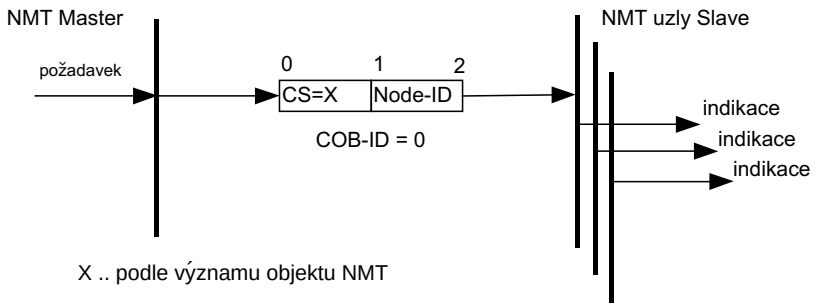
– konec uploadu bloku

Přenos SDO zpráv je dosti komplikovaný a jeho další popis je nad rámec této kapitoly. Plnohodnotnou specifikaci lze nalézt například v [12].

3.4.6 NMT zprávy

NMT zprávy slouží k řízení komunikačních stavů podřízených uzlů. Model komunikace je zde master/slave a podobně jako samotná sběrnice CAN je typu multimaster, tak i CANOpen umožňuje přítomnost více masterů v síti. NMT zprávy mají nulovou délku datové části a jejich význam je určen pouze identifikátorem. Přenos libovolné NMT zprávy je naznačen na obrázku 3.8. Uzel NMT master má k dispozici následující služby.

- Start Remote Node Protocol – uzel NMT master uvede vybrané uzly NMT slave do komunikačního stavu OPERATIONAL.
- Stop Remote Node Protocol – uzel NMT master uvede vybrané uzly NMT slave do komunikačního stavu STOPPED.
- Enter Pre-Operational Protocol – uzel NMT master uvede vybrané uzly NMT slave do komunikačního stavu PRE-OPERATIONAL.
- Reset Node Protocol – NMT master resetuje aplikaci uzlu NMT slave.
- Reset Communication Protocol – NMT master resetuje komunikaci uzlu NMT slave.



Obrázek 3.8: Přenos NMT zprávy

3.4.7 Standardizované profily v CANOpen

CANOpen poskytuje mnoho standardizovaných profilů a profilů rozhraní zvaných aplikační profily zařízení. Jejich specifikace jsou značeny DS-4XX. Standardy pro specificky zaměřená zařízení umožňují různým výrobcům použít jednotné metody komunikace a dovolit tak vzájemnou kompatibilitu svých výrobků. Ve slovníku zařízení je standardizovaným profilům vyhrazen rozsah indexů 6000_h – $9FFF_h$, standardizovaným rozhraním $A000_h$ – $BFFF_h$. Přibližme ve stručnosti některé profily:

- DS-401 – pro obecné vstupně/výstupní moduly např. vzdálené I/O moduly PLC,
- DS-402 – pro pohony a polohovací mechanismy, popsán v kapitole 3.5,
- DS-403 – pro HMI (*Human-Machine Interface*) moduly,
- DS-408 – pro řízení pneumatických a hydraulických systémů,
- DS-412 – pro zdravotnická zařízení,

a další.

3.4.8 Komunikační stavy CANOpen uzlu

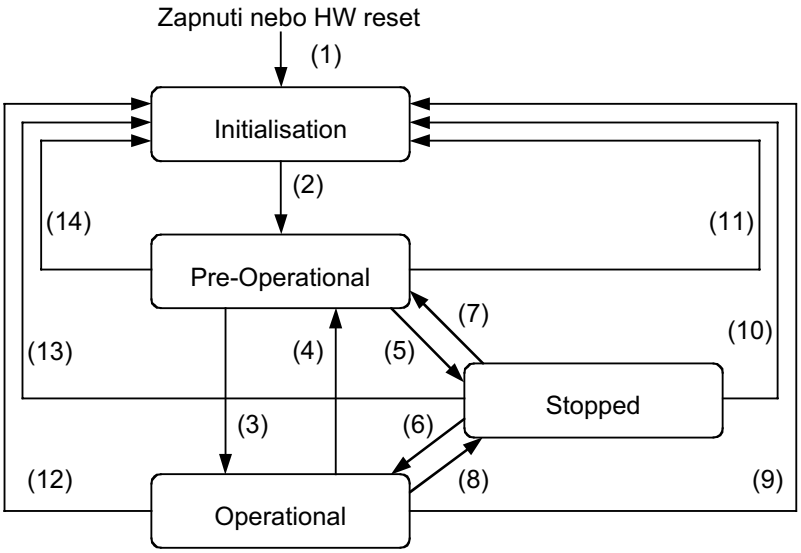
Komunikační stavy uzlu ukazuje obrázek 3.9. Vlastnosti stavů a podmínky jejich přechodů

Tabulka 3.3: Přechody mezi stavy v komunikačním stavovém diagramu

Číslo přechodu	Význam
(1)	Automaticky po zapnutí napájení
(2)	Automaticky po ukončení inicializace
(3),(6)	Přijata NMT zpráva *Start Remote Node*
(4),(7)	Přijata NMT zpráva *Enter Pre-Operational*
(5),(8)	Přijata NMT zpráva *Stop Remote Node*
(9),(10),(11)	Přijata NMT zpráva *Reset Node*
(12),(13),(14)	Přijata NMT zpráva *Reset Communication*

definuje DS-301 [12]. Přechody mezi stavy popisuje tabulka 3.3 a většina z nich je řízena z uzlu Master pomocí NMT zpráv. Důležité je to, že pouze ve stavu OPERATIONAL je možné komunikovat pomocí PDO zpráv. Stav PRE-OPERATIONAL slouží ke vzdálené konfiguraci

jiným uzlem pomocí zpráv SDO, které lze v tomto stavu vysílat i přijímat. Ve stavu STOPPED je zařízení dočasně pozastavena komunikace na síti.

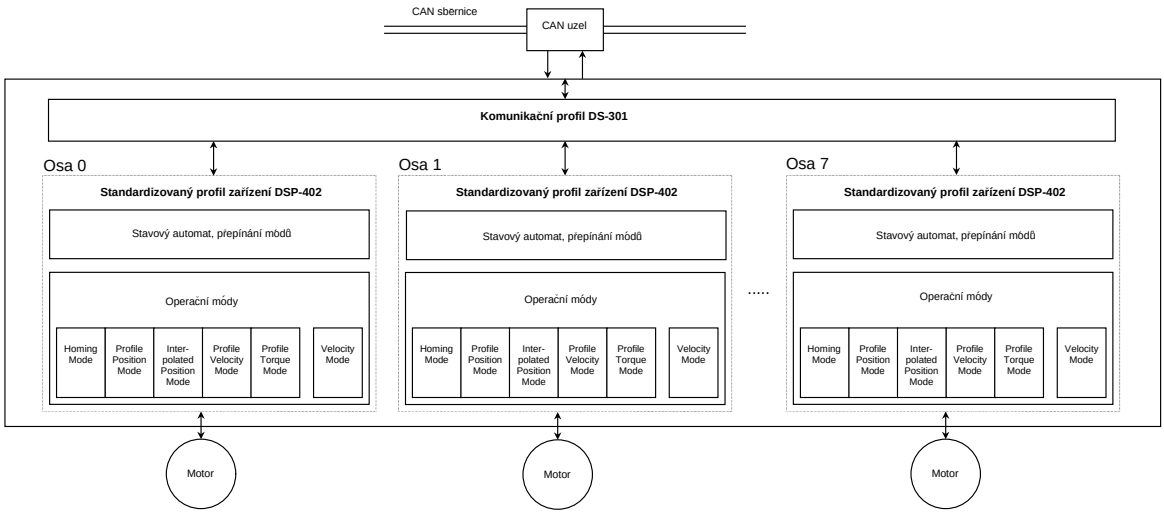


Obrázek 3.9: Komunikační stavový diagram CANOpen zařízení

3.5 Motion Control Profile

V této sekci je popsán standardizovaný profil protokolu CANOpen, určený pro řízení pohonů a polohovacích mechanismů (MCP – *Motion Control Profile*). Jedná se o specializující se nadstavbu obecného komunikačního standardu CANOpen, popsaného v normě DS-301 [12]. Při studiu specifikace jsem vycházel z dokumentu DSP-402 [10], který je ale pouze návrhem standardu DS-402. Aktuální norma je dostupná pouze za poplatek cca 250 euro a tu jsem k dispozici neměl. Možné odchylky mezi návrhem a aktuálním standardem jsem porovnával pomocí aktuální dokumentace k vyráběným modulům pro řízení pohonů od firmy ELMO [13].

Ve slovníku je standardizovaným profilům vyhrazen rozsah indexů od 6000_h do $9FFF_h$. Každý CANOpen uzel využívající MCP může obsahovat až 8 nezávisle řízených os pohybu. Veškerá konfigurační i procesní data všech os jsou v oblasti vyhrazené profilu. Většina předávaných parametrů má znaménkový nebo neznaménkový celočíselný formát. Tabulka 3.4 ukazuje členění indexů slovníku pro jednotlivé osy.



Obrázek 3.10: Blokové schéma se standardizovaným profilem pro řízení pohonů DS-402

V dalším popisu jsou proměnné a parametry uváděny s indexy v rozsahu 6000_h do $67FF_h$, tedy pro osu 0. Indexy stejných parametrů dalších os lze jednoduše získat podle vztahu

$$idx_n = idx_0 + n \cdot 800_h,$$

kde idx_n je index n-té osy, idx_0 je index osy 0, jak je uveden ve specifikaci a n je pořadové číslo osy od 0 do 7.

Tabulka 3.4: Rozsahy indexů ve slovníku pro jednotlivé osy

indexy	osa
6000_h – $67FF_h$	osa 0
6800_h – $6FFF_h$	osa 1
7000_h – $77FF_h$	osa 2
7800_h – $7FFF_h$	osa 3
8000_h – $87FF_h$	osa 4
8800_h – $8FFF_h$	osa 5
9000_h – $97FF_h$	osa 6
9800_h – $9FFF_h$	osa 7

Na obrázku 3.10 je blokové schéma uzlu CANOpen s profilem pro řízení pohonů. Je zde

Tabulka 3.5: Mapování přijímaných PDO zpráv podle DSP402

Číslo PDO	Mapování na indexu	Jméno mapovaného objektu	M/O	Komentář: M (mandatory) povinný O (optional) volitelný
1	6040_h	controlword	M	Řídící slovo stavu
2	6040_h 6060_h	controlword modes_of_operation	O	Přepínání módů
3	6040_h $607A_h$	controlword target_position	O	Mód profilovaného řízení pozice (pp)
4	6040_h $60FF_h$	controlword target_velocity	O	Mód profilovaného řízení rychlosti (pv)
5	6040_h 6071_h	controlword target_torque	O	Mód profilovaného řízení momentu (tq)
6	6040_h 6042_h	controlword vl_target_velocity	O	Mód řízení rychlosti (vl)
7	6040_h $60FE_h$	controlword digital_outputs	O	Digitální výstupy
8	6040_h 6060_h	controlword modes_of_operation	O	Přepínání módů (Broadcast PDO)
9 - 20			O	Rezervováno
21 - 64			O	Specifikovatelné výrobcem

naznačena vazba mezi společnými objekty slovníku zařízení a samostatnými objekty každého řízeného motoru.

3.5.1 Mapování PDO zpráv

Pro každou z osmi os je definováno umístění objektů pro nastavení mapování několika základních odesílaných a přijímaných PDO zpráv. Podobně jako je interval indexů 6000_h až $9FFF_h$ rozdělen na osm stejných úseků po 800_h pro jednotlivé osy, jsou i intervaly parametrů a mapování TPDO a RPDO zpráv rozděleny na osm úseků s offsetem[2] 40_h. Například mapování první RPDO (přijímané) zprávy pro osu 0 je ve slovníku na indexu 1400_h, mapování první RPDO zprávy osy 1 je na indexu 1440_h, osy 2 na indexu 1480_h atd. Mapování prvních osmi PDO zpráv každé osy je ve specifikaci určeno. Použití zprávy s číslem 1 je povinné, použití zpráv s čísly 2 až 7 je volitelné podle podporovaných módů řízení každé osy. Tabulky 3.5 a 3.6 ukazují určené mapování RPDO a TPDO zpráv.

[2]Softwarový projekt CanFestival, implementující protokol CANOpen, zde popsaný způsob mapování PDO zpráv neumožňuje. Odchylka od standardu je popsána v sekci 4.2

Tabulka 3.6: Mapování odesílaných PDO zpráv podle DS-402

Číslo PDO	**Mapování na indexu**	**Jméno mapovaného objektu**	M/O	**Komentář: M (mandatory) povinný O (optional) volitelný**
1	6041_h	statusword	M	Řídící slovo stavu
2	6041_h 6061_h	statusword modes_of_operation_display	O	Aktuální mód
3	6041_h 6064_h	statusword position_actual_value	O	Mód profilovaného řízení pozice (pp)
4	6041_h $606C_h$	statusword velocity_actual_value	O	Mód profilovaného řízení rychlosti (pv)
5	6041_h 6077_h	statusword torque_actual_value	O	Mód profilovaného řízení momentu (tq)
6	6041_h 6044_h	statusword vl_control_effort	O	Mód řízení rychlosti (vl)
7	6041_h $60FD_h$	statusword digital_inputs	O	Digitální výstupy
8 – 20			O	Rezervováno
21 - 64			O	Specifikovatelné výrobcem

3.5.2 Obecné informace o pohonu

Obecné informace o vlastnostech pohonného motoru jsou uvedeny na indexech 6402_h až $64FF_h$. Je zde například katalogové číslo motoru, adresa internetové stránky s katalogovým listem, perioda doporučených servisních prohlídek a základní technické údaje, jako jmenovité a mezní parametry. Podobně jsou na indexech mezi 6500_h a $65FF_h$ uvedeny parametry celého pohonu, obsahuje-li například převodovku. Celá tato část obecných informací není pro výrobce povinná, pouze doporučená.

3.5.3 Řízení stavu pohonu

Pohon se může nacházet v různých stavech. Pro přechody mezi stavy je zde pro každou z os 0 až 7 implementován stavový automat s devíti stavy a šestnácti přechody 3.5.3. Přesná definice stavů a jejich přechodů je poměrně rozsáhlá. Uvedena je v [10]. Přiblížím pouze, že pro skupinu stavů je vypnuto napájení výkonové části elektroniky pohonu (*Power Disabled*). V určitých stavech je napájecí napětí výkonové elektroniky přivedeno (*Power Enabled*).

Přechody mezi stavy jsou řízeny buď to vnitřními událostmi pohonu, nebo řídícím slovem (*controlword*). Vnitřní událost je například vznik chyby nebo změna parametrů pomocí panelu lokálního ovládání, je-li jím pohon vybaven. Daleko častěji však dochází k přechodu stavů řídícím slovem (*controlword*), které je na indexu 6040_h a je mapováno z příchozí PDO zprávy s číslem 1. Podobně zde existuje stavové slovo automatu (*statusword*) na indexu 6041_h. Je

Tabulka 3.7: Významy jednotlivých bitů řídícího slova

Bit	Význam	Bit	Význam
0	Switch On	8	Halt
1	Disable Voltage	9	rezervován
2	Quick Stop	10	rezervován
3	Enable Operation	11	specifikovatelný výrobcem
4	specifikován op. módem	12	specifikovatelný výrobcem
5	specifikován op. módem	13	specifikovatelný výrobcem
6	specifikován op. módem	14	specifikovatelný výrobcem
7	Reset Fault	15	specifikovatelný výrobcem

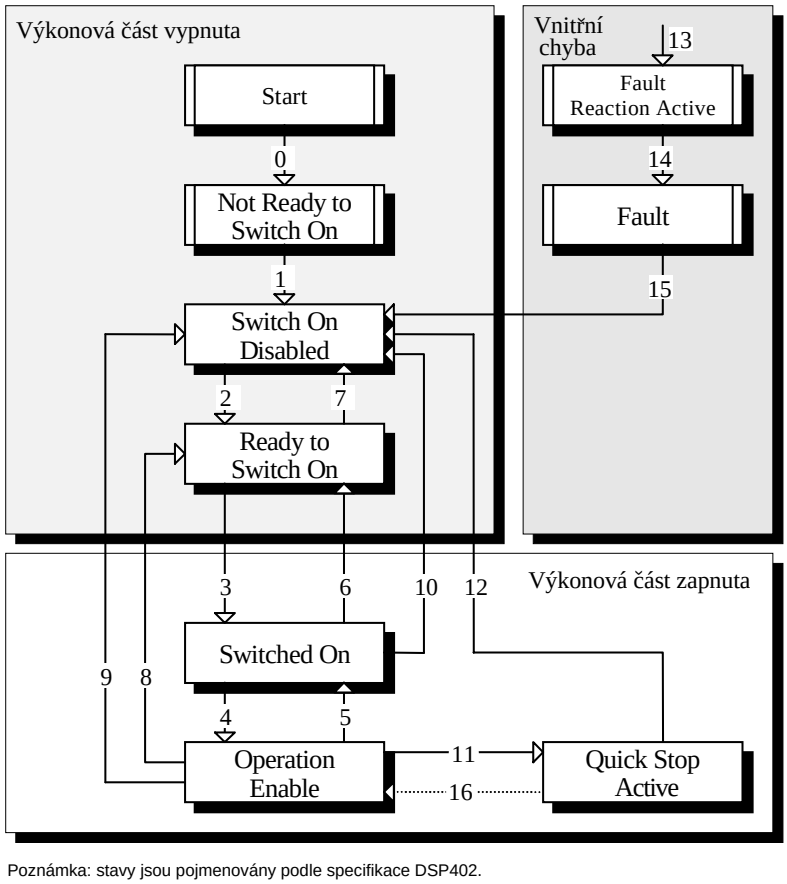
Tabulka 3.8: Významy jednotlivých bitů stavového slova

Bit	Význam	Bit	Význam
0	Ready to Switch On	8	specifikovatelný výrobcem
1	Switched On	9	Remote
2	Operation Enabled	10	Target Reached
3	Fault	11	Internal Limit Active
4	Voltage Disabled	12	specifikován op. módem
5	Quick Stop	13	specifikován op. módem
6	Switch On Disabled	14	specifikovatelný výrobcem
7	Warning	15	specifikovatelný výrobcem

mapováno do odchozí PDO zprávy s číslem 1. Stavové slovo dále obsahuje bity informující například o dosažení zadané cílové polohy pohonu. Významy bitů řídícího slova popisuje tabulka 3.7 a významy bitů stavového slova popisuje tabulka 3.8.

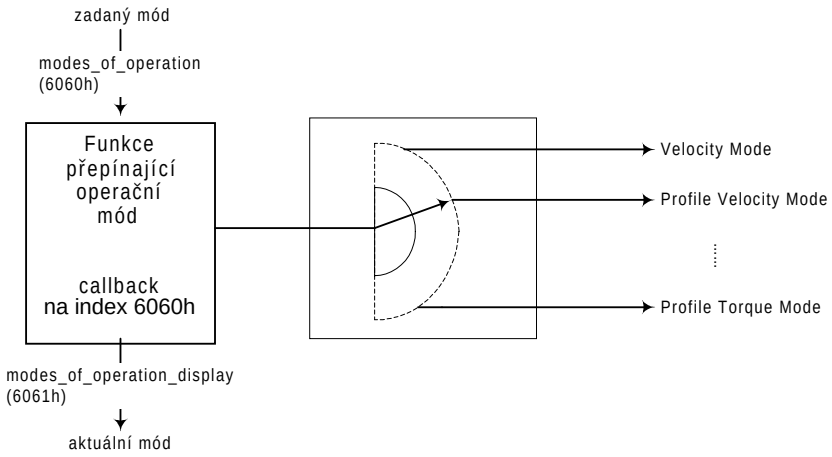
Stavový diagram struktury shodné pro každou osu pohonu je na obrázku 3.11. Přechod do stavu Fault Reaction Active (č.13) proběhne automaticky z libovolného stavu v okamžiku, kdy je registrována chybová událost. Další přechod (č.14) do stavu Fault je uskutečněn automaticky po provedení nezbytných bezpečnostních opatření. Může to být např. okamžité zastavení pohonu, nebo odpojení napájecího napětí silové části. Přechody č. 0 a 1 se provádějí automaticky v průběhu inicializace pohonu. Ostatní přechody jsou řízeny řídícím slovem.

Položkou slovníku *modes_of_operation* jsou řízeny operační módy. Pohon nemůže pracovat současně ve více módech a možnost přepínání není striktně určena. Je tedy na uživateli,



Poznámka: stavy jsou pojmenovány podle specifikace DSP402.

Obrázek 3.11: Stavový diagram pohonu každé osy na CANOpen s profilem DSP402. Přechod do stavu Fault Reaction Active (č.13) proběhne automaticky z libovolného stavu v okamžiku registrace chybové události.



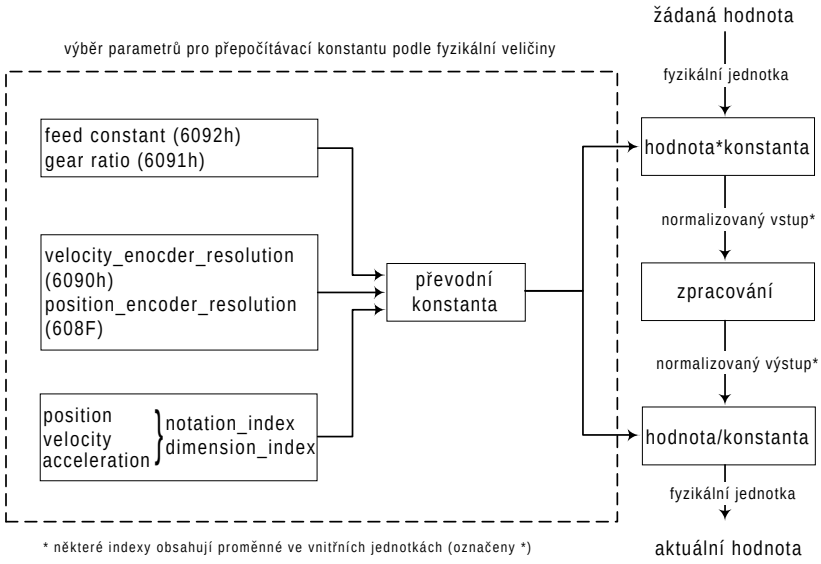
Obrázek 3.12: Přepínání operačních módů pohonu osy podle DS-402

pokud povolí přepnutí mezi módy například pouze při stojícím pohonu nebo pouze při inicializaci. Operační mód je možno vzdáleně ovládat pomocí položky slovníku na indexu 6060_h (*modes_of_operation*) a číst na indexu 6061_h (*modes_of_operation_display*). Situaci blokově zobrazuje obrázek 3.12.

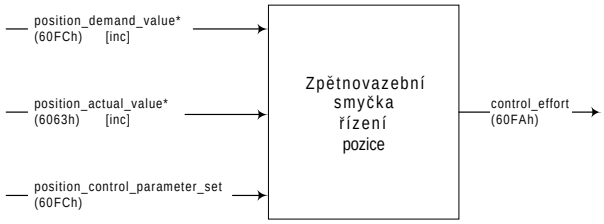
3.5.4 Převod fyzikálních jednotek

Cílovou polohu, rychlost, moment a jejich omezující parametry lze zadávat ve velkém množství fyzikálních jednotek. Řídící funkce pohonu zpravidla pracují v nějakých interních celočíselných jednotkách, jako jsou například inkrementy snímače polohy. Mezi těmito dvěma skupinami veličin je pro každou osu pohonu možno definovat přepočítávací faktory. Na indexech v rozsahu $607E_h$ až 6097_h je možné definovat rozlišení snímače polohy a rychlosti, převodový poměr, konstantu převodu rotačního pohybu na posuvný, znaménko směru a další. Pro veličiny polohy, rychlosti a zrychlení jsou definovány tzv. indexy veličiny (*notation index*) a indexy rozměru této veličiny (*dimension index*). Pozici je tak možno zadávat například v délkových nebo úhlových jednotkách (volbou indexu veličiny) a v rozsazích např. pro délku od mikrometru až po kilometr (volbou indexu rozměru).

Podrobnými definicemi se zabývá specifikace profilu [10]. Obrázek 3.13 ukazuje způsob převodu fyzikálních veličin.



Obrázek 3.13: Převod mezi síťovými a vnitřními fyzikálními jednotkami podle DSP402

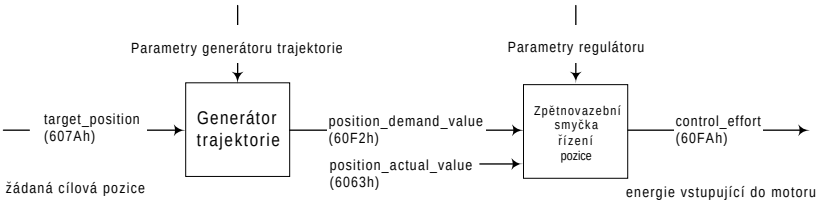


Obrázek 3.14: Blok zpětnovazební smyčky řízení pozice

3.5.5 Zpětnovazební smyčka řízení pozice

K řízení pozice polohovacího pohonu je v této sekci slovníku zařízení k dispozici několik parametrů definujících funkci zpětnovazebního regulátoru (*Motion Control Function*). Vstupy jsou požadovaná pozice a aktuální pozice, obě v interních jednotkách. Výstupem je pak akční zásah, neboli výkon vstupující do pohonného motoru.

Jediným povinným parametrem je proměnná nesoucí hodnotu aktuální pozice. Ostatní parametry jsou volitelné. Tato regulační smyčka je společná pro všechny operační módy, vyjma módu řízení rychlosti (*Velocity mode*), kde je aplikováno přímovazební . Obrázek 3.14 ukazuje vstupní a výstupní proměnné bloku pro zpětnovazební řízení pozice.



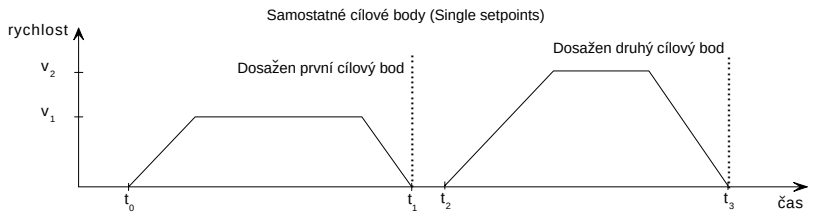
Obrázek 3.15: Blokové schéma módu profilovaného řízení pozice

3.5.6 Operační módy

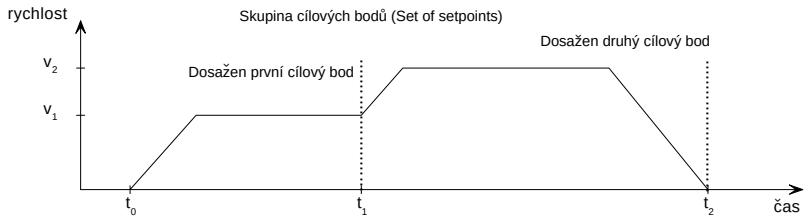
3.5.6.1 Mód profilovaného řízení pozice

V tomto módu se předpokládá najíždění pohonu na cílovou pozici. Profil rychlosti je přitom dán počátečním průběhem zrychlování, poté jízdou konstantní rychlostí a podobným způsobem zpomalování k cílové pozici. Povinně musí být uvedeny pro tento mód cílová pozice (*target_position*), přejezdová rychlost (*profile_velocity*), zrychlení na začátku a zpomalení na konci pohybu (*profile_acceleration, profile_deceleration*) a profil zrychlení (*motion_profile_type*). Obrázek 3.15 ukazuje, jak žádaná poloha působí na energii vstupující do motoru.

Mód profilovaného řízení pozice umožňuje buď to zadávat cílové body jednotlivě až po dosažení předchozího cílového bodu (*single setpoint*), nebo zadávat skupinu cílových bodů (*set of setpoints*). V druhém případě se pohon okamžitě po dosažení jednoho bodu rozjíždí k novému cílovému bodu aniž by zastavoval. Časové průběhy rychlosti pro oba zmíněné případy jsou na obrázcích 3.16 a 3.17.



Obrázek 3.16: Rychlostní profil pohybu při zadávání samostatných cílových bodů



Obrázek 3.17: Rychlostní profil pohybu při zadávání skupiny cílových bodů

3.5.6.2 Mód hledání referenční pozice

V tomto módu polohovací zařízení hledá svoji referenční pozici (*home position*). Existuje zde mnoho funkcí pro hledání referenčního spínače v pracovní dráze pohonu nebo na jejích krajích. K hledání referenční pozice je také možno využít koncové spínače dráhy, pokud jsou přítomny, a definovat referenci kdekoliv mezi nimi. Předdefinovaných metod hledání referenčního bodu je 35, ale uživatel nebo výrobce má širokou možnost definovat si další metody sám.

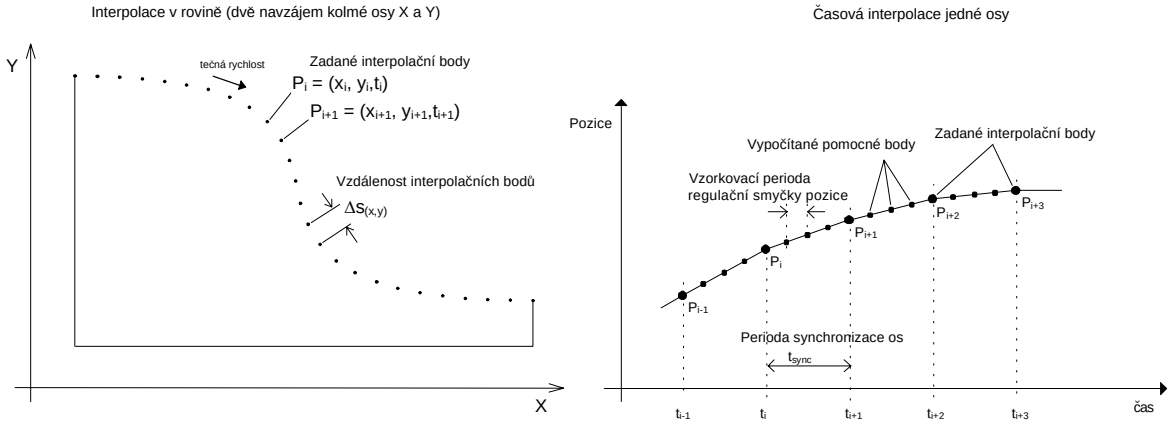
Povinnými vstupními parametry tohoto módu je volba metody pomocí jejího kódu (*homing_method*) a skupina předepsaných rychlostí pro hledání referenční pozice (*homing_speeds*).

3.5.6.3 Mód interpolovaného řízení pozice

Tento mód je určen pro řízení více koordinovaných os nebo pro osy s potřebou časové interpolace mezi skupinou bodů. Pro více os je zde nutnost jejich časové synchronizace, například pomocí synchronizačních objektů (SYNC).

Pro použití interpolované pozice není povinně definován žádný parametr. Volitelně lze zadat například volbu více implementovaných interpolačních algoritmů (*interpolation_submode_select*) a k těmto algoritmům potřebná vstupní data, jako například koeficienty interpolačního polynomu. Tento strukturovaný parametr se nazývá *interpolation_data_record*. Dále je možno definovat periodu a vlastnosti synchronizace více os (*interpolation_time_period*). Prostorovou a časovou interpolaci popisuje obrázek 3.18.

Vstupními daty pro interpolátor jsou zadané interpolační body. Časové rozmezí, ve kterém jsou tyto body k dispozici je dáno periodou synchronizace os. Pro každý okamžik vzorkování regulační smyčky se s využitím parametrů interpolátoru interně vypočítají pomocné záchytné body. Účel interpolace je je tedy zajistit dostatečně plynulý pohyb při přejezdech pohonu mezi interpolačními body. V případě prostorové interpolace více os je časová interpolace prováděna



Obrázek 3.18: Prostorová a časová interpolace pohybu v módu interpolovaného řízení pohybu

na každé této ose.

3.5.6.4 Mód profilovaného řízení rychlosti

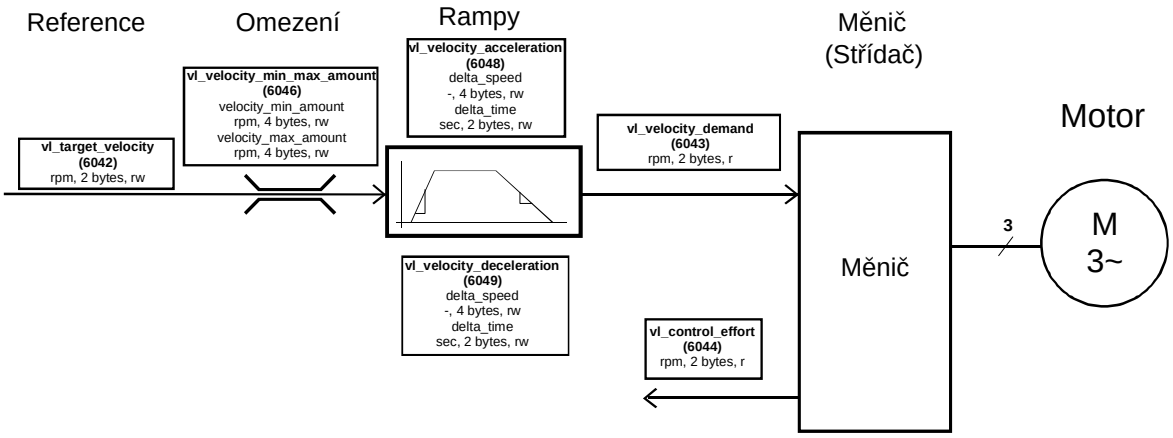
Zde je podobně jako u profilovaného řízení pozice řízena rychlost na požadovanou zadanou hodnotu. Vstupní hodnotu rychlosti je cílová rychlost (*target_velocity*). Aktuální hodnota rychlosti je k dispozici v proměnné (*velocity_actual_value*) a může být získána ze senzoru aktuální rychlosti nebo derivací aktuální polohy ze senzoru polohy. Do řízení rychlosti zasahují omezení jako maximální zrychlení nebo maximální dosažitelná rychlost. S podobným významem jako maximální sledovací chyba při řízení pozice zde figuruje parametr udávající maximální rychlostní skluz (*max_slippage*), tedy dovolený rozdíl aktuální a žádané rychlosti.

3.5.6.5 Mód profilovaného řízení momentu

Interpolované řízení momentu je svojí podstatou podobné interpolovanému řízení pozice nebo rychlosti. I zde je vstupní proměnnou žádaný kroutící moment (*target_torque*) a aktuální hodnota momentu je uložena v proměnné (*torque_actual_value*). Názvy všech proměnných vycházejí z toho, že pohonnou jednotkou je rotační motor. Například u lineárních motorů je kroutícímu momentu ekvivalentní působící síla, která je ovšem ve slovníku zařízení uložena stále pod názvem kroutícího momentu. Dalšími zadanými parametry je například profil změny momentu (*torque_profile_type*), maximální moment (*max_torque*) nebo jemu odpovídající maximální proud motorem (*max_current*).

3.5.6.6 Mód řízení rychlosti

Tento mód je určen především pro řízení rychlosti pohonů s elektronickým střídačem. Pouze právě zde se nepředpokládá žádná polohová ani rychlostní zpětná vazba. Informaci o aktuální rychlosti může poskytovat například frekvenční měnič, ale nejedná se o měření přímo na hřídeli motoru. Blokové schéma jakési minimální konfigurace je na obrázku 3.19. Předpokládá se, že eventuální regulační smyčka je vedena a počítána přímo v měniči motoru.



Obrázek 3.19: Blokové schéma využití módu řízení rychlosti

Kapitola 4

Implementace řízení pohonů prostřednictvím CANOpen

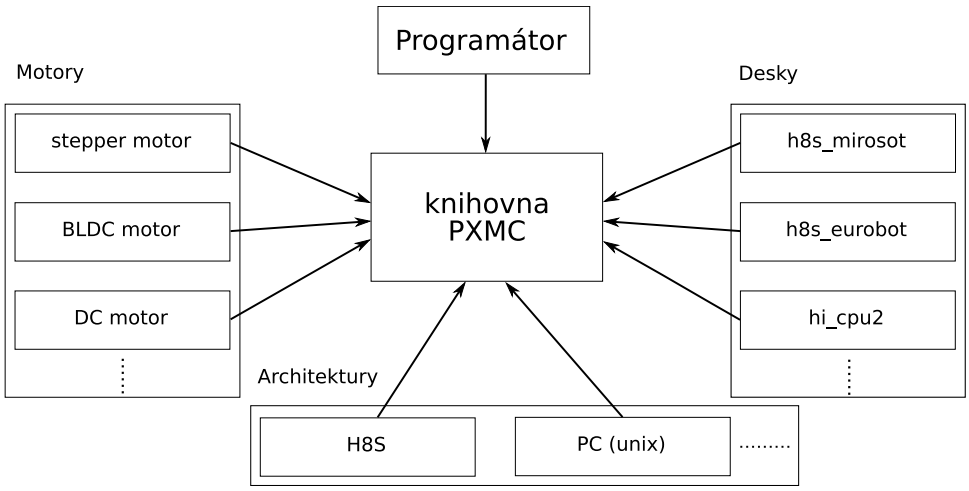
V této kapitole jsou popsány existující softwarové projekty a nástroje, použité dále při implementaci profilu pro řízení pohonů protokolu CANOpen. V dalších sekcích jsou představeny ukázkové aplikace využívající Motion Control Profile. Je zde popsán způsob ovládání a uvedení do provozu.

4.1 PXMC

PXMC (*Pikron eXtensible Motion Control*) je univerzální softwarová knihovna pro řízení motorů. Je vyvinutá firmou PiKRON spol. s r.o. [5] a hlavní zásluhu na ni nese Ing. Pavel Píša, který také působí na Katedře řídicí techniky. Tato knihovna umožňuje velice propracovaným způsobem řízení mnoha různých typů motorů, pomocí různých typů řídících desek osazených různými mikrokontroléry. Hlavní přednost je v rozdělení zdrojových kódů na část závislou na použité architektuře kontroléru, na část závislou na typu řídící desky a na část hardwarově i platformě nezávislou, viz obrázek 4.1.

Podle použité architektury lze definovat například makra pro jednotlivé registry, využít specifika přerušovacího systému, nebo definovat speciální funkce ovladačů pro libovolné I/O rozhraní apd. Na OS Linux lze knihovnu PXMC v současné době provozovat pouze jako simulátor.

Platformě nezávislým kódem je pak například implementace datových struktur a parametrů. Nepochybně sem patří i funkce implementovaného PID regulátoru, generátoru trajektorie a ostatních logických funkcí.



Obrázek 4.1: Grafické znázornění modularity PXMC knihovny

Značná modularita a univerzálnost je ještě posílena použitým kroskompilačním systémem OMK (*Ocera Make System*) [4], který je produktem Katedry řídicí techniky. Překlad zdrojových kódů pro cílovou desku a procesor je pak možné nastavovat pouze změnou několika globálních proměnných make-systému a vytvořením různých kompilačních stromů. Tyto kompilační stromy jsou vytvořeny pomocí symbolických odkazů na adresáře se zdrojovými kódy projektu.

Poměrně rozsáhlý popis funkce knihovny PXMC je v [21]. Já se zde omezím pouze na přiblížení obsahu globální datové struktury `pxmc_main_list`, ve které jsou uložena stavová slova a veškeré procesní parametry všech os řízených z jednoho programu. Příklad zdrojového kódu datové struktury definované pro tři osy `mcs_axle0` až `mcs_axle2`.

```
pxmc_state_t *pxmc_main_arr[] = {&mcs_axle0, &mcs_axle1, &mcs_axle2};

pxmc_state_list_t pxmc_main_list = {
  pxml_arr:pxmc_main_arr,
  pxml_cnt:sizeof(pxmc_main_arr) / sizeof(pxmc_main_arr[0])
};
```

Pole `pxmc_main_arr[]` obsahuje tolik instancí datové struktury `pxmc_state_t`, kolik os je řízeno. Každá z os, zde označených jako `mcs_axleX`, nese vlastní stavové, konfigurační a chybové slovo. Samostatné jsou i proměnné jako aktuální pozice, aktuální rychlost, konstanty PID regulátoru, proměnné třífázového generátoru pro BLDC motory, ukazatele na některé funkce knihovny PXMC a mnoho dalších. Proměnná `pxmc_main_list` pak obsahuje ukazatel na pole dat všech os `pxml_arr` a počet těchto os `pxml_cnt`.

Vlastní řízení pohybu motoru se provádí pomocí volání funkcí knihovny. Například pohyb na zadanou cílovou pozici se provádí funkcí `pxmc_go`, jejíž parametry jsou cílová pozice ve vnitřních jednotkách, osa na které se má pohyb provést a výběr zadání v relativních nebo absolutních parametrech cílové pozice. Funkce při rozjezdu motoru nastaví příslušné bity stavového slova informující o prováděném pohybu. Po dosažení cíle tyto stavové bity vrací do původních hodnot. Podobnými funkcemi lze řídit rychlost a moment motoru.

Uveďme zde některé proměnné ze struktury `pxmc_state_t`, např. ke kterým se v implementaci přistupuje při čtení nebo zápisu slovníku zařízení. Uvedené parametry jsou vstupními veličinami zpětnovazebního PID regulátoru.

`pxms_flg` – stavové slovo pohonu.

`pxms_ap` – aktuální pozice ve vnitřních jednotkách.

`pxms_as` – aktuální rychlost ve vnitřních jednotkách za vzorkovací periodu snímače polohy.

`pxms_rp` – žádaná cílová pozice ve vnitřních jednotkách.

`pxms_rs` – žádaná rychlost pohybu.

`pxms_md` – maximální dovolená sledovací odchylka.

`pxms_ms` – maximální dovolená rychlost, parametr regulátoru.

`pxms_ma` – maximální dovolené zrychlení, parametr regulátoru.

`pxms_p` – proporcionální konstanta PID regulátoru.

`pxms_i` – integrační konstanta PID regulátoru.

`pxms_d` – derivační konstanta PID regulátoru.

4.2 CanFestival

CanFestival je otevřený softwarový projekt, který implementuje specifikaci CANOpen. Pod licencí GPL dává k dispozici programový balík pro komunikaci více aplikací prostřednictvím standardu DS-301 i mnoha rozšiřujících profilů DS-4XX. CanFestival je psán v jazyce ANSI C a podobně jako PXMC je rozdělen na část závislou na použitém rozhraní CAN sběrnice, závislou na cílové platformě a na nezávislou část.

Zdrojové kódy projektu, tak jak je lze stáhnout z repositáře projektu, jsou rozděleny do následujících adresářů podle významu.

- `./doc` – dokumentace k projektu ve formátu *.pdf včetně zdrojových kódů programu LaTeX, dokumentace formátu HTML generovaná programem Doxygen.

- `./drivers` – zdrojové kódy ovladačů pro platformy unix, unix real-time (Xenomai), win32, mikrokontrolér HCS12. Dále jsou zde ovladače různých CAN rozhraní jako LinCan, SocketCan, Peak_Can (I/O karta) nebo virtuální rozhraní pomocí soketů.

- `./examples` – příklady několika CANOpen aplikací.

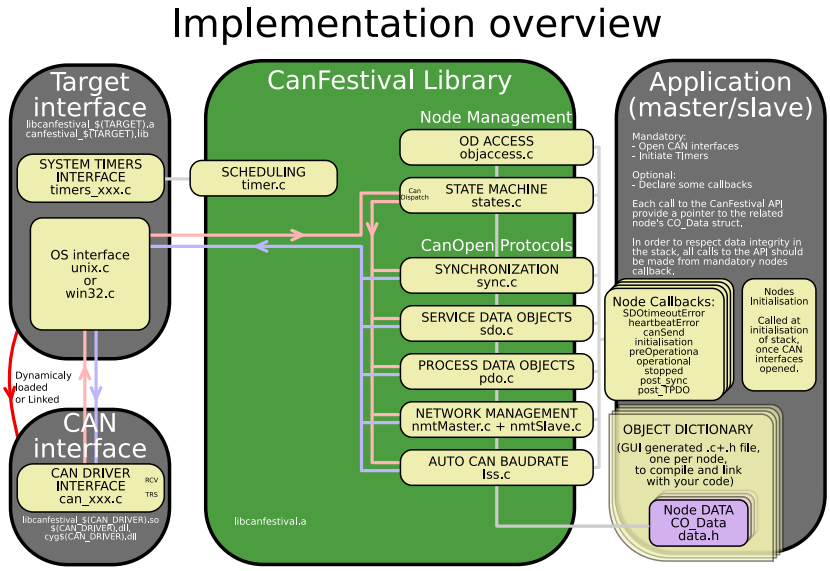
- `./include` – hlavičkové soubory ovladačů.

- `./objdictgen` – grafická aplikace pro interaktivní generování a editování slovníku zařízení. Jsou zde definovány konfigurační soubory pro různé profily zařízení podle norem DS-4XX.

- `./src` – zdrojové kódy implementující funkcionalitu podle DS-301, jako manipulaci s objekty PDO, SDO, SYNC, NMT, EMCY atd.

Poznámka: Během práce s CanFestivalem jsem rozšířil adresáře `./drivers` a `./include` o podadresář `./h8s` s podporou pro mikrokontrolér H8S a adresář `./objdictgen/config` o soubor konfigurující editor slovníku zařízení podle návrhu normy DSP-402 (*Motion Control Profile*). Ovladač pro mikroprocesor H8S/2638 a CanFestival je prací Stanislava Marka, kterou zdokumentoval ve své diplomové práci [17]. S mírnými úpravami jsem tento ovladač přenesl do aktuální verze CanFestivalu a použil ve své práci. □

Blokovou strukturu aplikace naznačuje obrázek 4.2, který jsem převzal z dokumentace ke CanFestivalu [22]. Blok *CAN Interface* představuje rozhraní sběrnice CAN a je dáno použitým hardwarem (ovladače jsou v adresáři `./drivers`). V případě nasazení na mikrokontroléru tento blok představuje přímo registry. Blok *Target interface* obsahuje funkce pro přístup k časovačům operačního systému a dalším funkcím jádra OS. Knihovna *CanFestival library* obsahuje funkce pro práci s objekty CANOpen jako jsou PDO, SDO a NMT zprávy. V bloku *Aplication* uživatel specifikuje vlastní aplikaci a definuje callback funkce pro přístup ke slovníku zařízení vygenerovaném aplikací Objdictedit.



Obrázek 4.2: Blokové schéma aplikace a jejích programových jednotek CanFestivalu

4.2.1 Vytvoření slovníku zařízení

Jak je ukázáno na obrázku 4.2, k sestavení uživatelské aplikace je nutné vytvořit slovník zařízení. Ten má každá aplikace vlastní a jeho obsah je specifický podle úkolu a budoucí funkce aplikace. K vytvoření slovníku je v projektu k dispozici grafická aplikace Objdictedit, napsaná v jazyce Python. Pomocí ní lze interaktivně sestavovat proměnné slovníku a definovat jejich počáteční hodnoty. Výstupem jsou vygenerovaný zdrojový soubor `.c` a hlavičkový soubor `.h`, které deklarují a definují položky slovníku zařízení jako proměnné v prostředí jazyka C a které se překládají spolu s výslednou aplikací.

Konfigurace programu Objdictedit pro různé profily zařízení se provádí pomocí souboru `*.prf` v adresáři `./objdictgen/config`. Pro profil řízení pohonů zde konfigurační soubor nebyl, musel jsem ho vytvořit. Struktura těchto konfiguračních souborů ovšem v dovolovala vytvoření konfigurace pouze pro řízení jedné osy. Studiem zdrojového kódu jsem našel definované, ale téměř nikde nepoužité proměnné `plurivar`, `plurirec` a `pluriarray`, které použití více os s určitým omezením dovolí. Bez dokumentace jsem musel zdlouhavě testovat, zda je použití zmíněných proměnných možné v zamýšleném rozsahu.

V konfiguračním souboru je popsána každá položka (index a subindexy) pomocí skupiny parametrů. Jejich význam není popsán v žádné dokumentaci, proto se ho pokusím nastínit. Na položku se pak v programu odkazuje pomocí jejího indexu. Strukturu jedné položky v souboru

`*.prf` ukazuje následující příklad:

```
0x6000 : {"položka1" : "parametr1", "položka2" : "parametr2", "values" :
          [{"položka01" : "parametr01", "položka02" : "parametr02"},
           {"položka11" : "parametr11", "položka12" : "parametr12"}]},
```

Tento příklad ukazuje konfiguraci pro index 6000_h. V uvozovkách jsou uvedena jména položek popisující příslušný index `položka1, položka2`, atd. a jejich parametry `parametr1, parametr2`, atd. Subindex s číslem 0 je pak popsán na dalším řádku pomocí `položka01, položka02`, atd. a k nim příslušných parametrů `parametr01, parametr02`, atd. Podobně je popsán subindex 1 položkami a parametry označenými ”11, 12, atd.”. Význam nejpoužívanějších parametrů je shrnut v následujícím seznamu.

`"name"` – jméno položky indexu nebo subindexu, parametrem je řetězec znaků.

`"struct"` – značí typ proměnné na indexu, parametry jsou:

- `var` – proměnná jednoduchého typu, obsahuje pouze subindex 0, na kterém je uložena vlastní hodnota,
- `rec` – strukturovaný typ s dynamickým počtem položek stejného typu[1], na subindexu 0 je počet položek a na dalších subindexech jsou jednotlivé proměnné,
- `array` – strukturovaný typ s pevným počtem položek různého jednoduchého typu, opět na subindexu 0 je počet položek a na dalších jsou jednotlivé proměnné,
- `plurivar, plurirec, pluriarray` – proměnné typů var, rec a array ovšem s možností definování více indexů jednoho konkrétního typu. Toto je využito u konfigurace pro víceosé zařízení, kde je shodná struktura indexu první osy (např. 6040_h-*controlword*), ale i dalších os (6840_h, 7040_h, ..).

`"incr"` – inkrement, se kterým jsou do slovníku přidávány indexy pro více os, v souboru `DSP402-pxmc.prf` je použita hodnota 800_h, což je offset mezi stejnými indexy jednotlivých os.

`"nbmax"` – maximální možný počet indexů stejné struktury, použita hodnota 8 – max. počet os ve slovníku zařízení.

[1] V této definici se zdrojové kódy programu Objdictedit odchylují od standardního chápání pojmů `rec` a `array`.

"callback" – dovoluje či zakazuje registraci callback[2] funkce k příslušnému indexu, parametry jsou `True/False`.

"need" – udává, zda je přítomnost indexu ve slovníku povinná nebo volitelná, parametry jsou `True/False`.

"values" – parametrem je strukturovaná definice subindexů, viz příklad výše.

"type " – udává typ proměnné na subindexu, parametry jsou kódy v rozsahu 0_h až $25F_h$, jejichž definice je uvedena ve specifikaci DS-301 [12].

"acces" – povoluje přístup k proměnné, parametry jsou `'ro'`=*read only*, `'wo'`=*write only*, `'rw'`=*read/write*.

"default" – nastavuje počáteční hodnotu proměnné.

"pdo" – povoluje mapování do PDO zpráv, parametry jsou `True/False`.

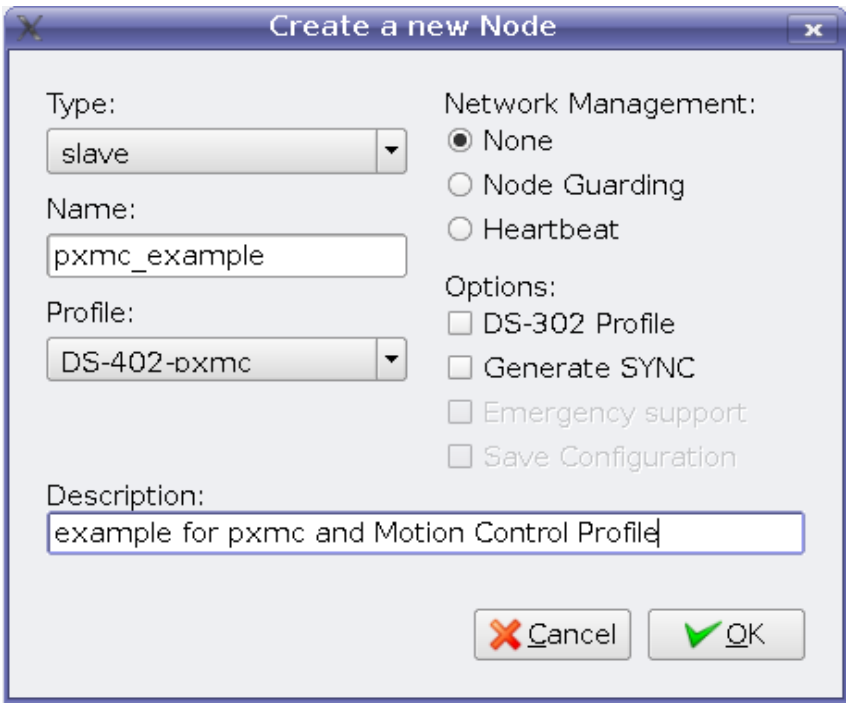
Konfigurační soubor obsahuje ještě položku se jménem `AddMenuEntries`, pomocí které se dá editovat do záložky *Add* na nástrojové liště aplikace Objdictedit libovolná položka. Na kliknutí se přidají předem definované skupiny indexů ze souboru `*.prf`. Následující příklad ukazuje syntaxi, jak ji lze uvést do konfiguračního souboru.

```
AddMenuEntries = [("Položka1",[0x6000, 0x6001, 0x6002]),
                  ("Položka2",[0x6100, 0x6101, 0x6102, 0x6103, 0x6104])]
```

Tenoto příklad přidává do menu *Add* dvě položky s názvy `Položka1` a `Položka2`. Při kliknutí na Položku1 se přidají do slovníku indexy 6000_h až 6002_h. Při opětovném kliknutí se do slovníku přidají indexy další, tentokrát v rozsahu 6800_h až 6802_h, ale jenom pokud jsou indexy 6000_h až 6002_h definovány typu `plurivar`, `pluriarray` nebo `plurirec` s parametrem `"incr":0x800`. Pro položku2 to platí podobně. Tímto způsobem je docíleno přidávání indexů slovníku zařízení dalších řízených os až do počtu 8. Indexy jsou uvedeny pouze jako příklad, tyto uvedené nemusí být vůbec normou definovány.

Poznámka: Soubor `DSP402-pxmc.prf` neobsahuje všechny indexy definované návrhem normy DSP-402, ale jenom ty, které jsou použity při řízení pomocí knihovny PXMC. Důvod je ten, že se při přidání osy do slovníku pomocí *Add→Axle in PXMC* přidají všechny indexy definované v položce souboru `AddMenuEntries` a většina indexů by pak v programu nebyla

[2]Callback je souhrnný název pro funkce volané po zápisu hodnoty na určitý index slovníku za. K většině indexů ve slovníku lze registrovat nějakou callback funkci.



Obrázek 4.3: Vytvoření nového slovníku zařízení v programu Objdictedit

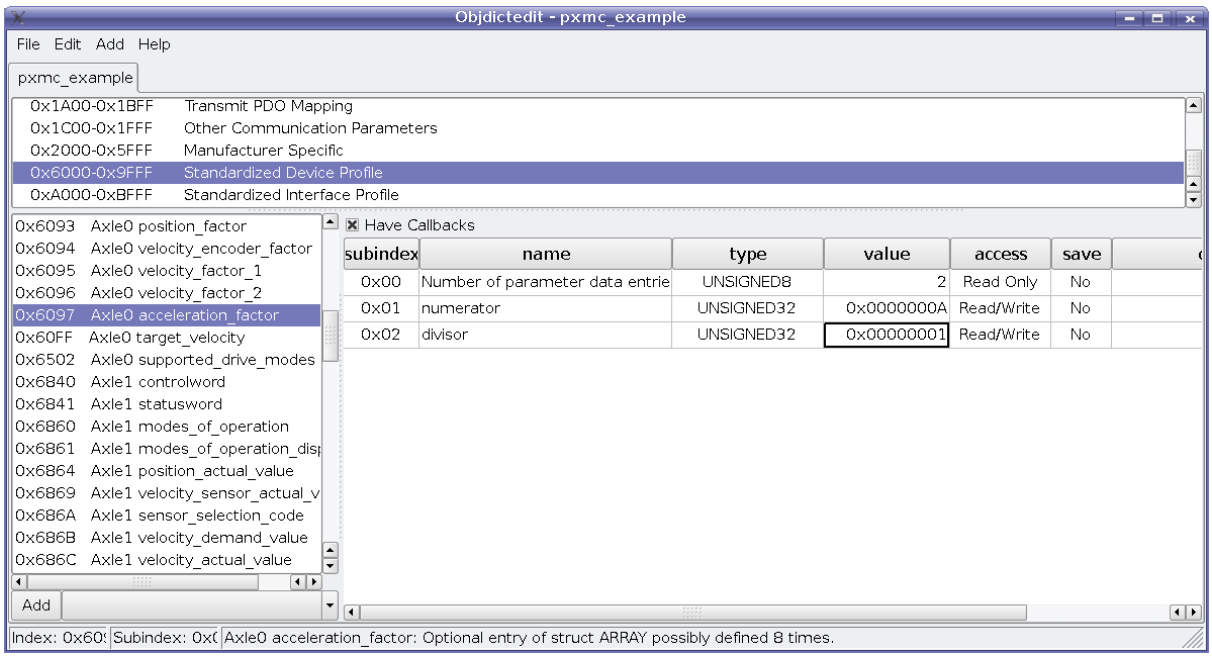
využita. S ohledem na použití v mikrokontrolérech se tak šetří paměťový prostor. Pokud by někdo potřeboval určitý index do slovníku pro práci se zmiňovaným profilem přidat, může jej připsat do konfiguračního souboru podle výše zmíněného návodu.

V adresáři `./objdictgen/config` je ještě soubor `DSP-402.prf`, ve kterém jsem definoval všechny indexy podle DSP-402. Struktura tohoto konfiguračního souboru se ovšem drží filosofie CanFestivalu a není zde možné definovat více os. □

Ukažme si ještě postup, jakým lze vytvořit slovník zařízení pro PXMC, například se třemi osami. Popis funkcí programu Objdictedit je v [22], já ze detailněji popíši pouze kroky specifické pro práci s konfigurací pomocí souboru `DSP402-pxmc.prf`.

1. Vytvoříme slovník nového uzlu. Nastavíme jeho úlohu v síti jako NMT Master nebo NMT Slave. Zadáme jméno, podle kterého se budeme na vygenerovaný slovník odkazovat ze zdrojového kódu aplikace. Vybereme profil DSP402-pxmc. Lze nastavit ještě další parametry jako generování SYNC zpráv nebo management sítě. Klikneme OK. Okno aplikace je na obrázku 4.3.

2. Pro vytvoření třech skupin indexů pro tři osy klikneme v menu *Add* třikrát na položku *Axle in PXMC* a potom třikrát na položku *Multiaxle PDOs*. V levém sloupci okna apli-



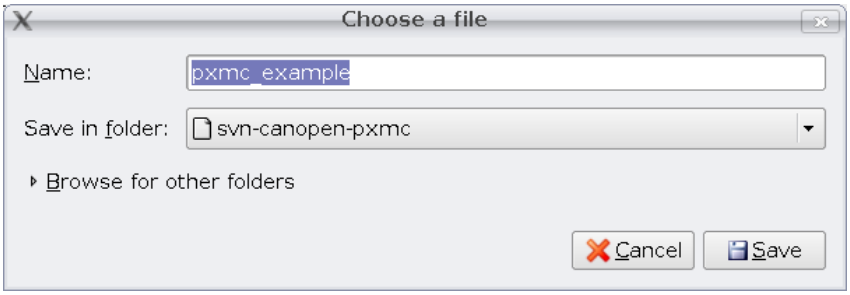
Obrázek 4.4: Nastavení počátečních hodnot nového slovníku zařízení v programu Objdictedit

kace se pří výběru sekce *Standardized Device Profile* objeví jména indexů začínající Axle0, Axle1, Axle2. Stejně tak se v rozsahu indexů 1400_h až $1BFF_h$ určujících vlastnosti PDO objektů objeví proměnné s názvy začínajícími Axle0, Axle1, Axle2.

3. Pomocí aplikace lze všechny indexy a subindexy procházet a nastavit například jejich počáteční hodnoty. Ukazuje to obrázek 4.4.

4. Vygenerování `*.c` a `*.h` souboru provedeme v menu *File→Build dictionary* nebo klávesami Ctrl+B. Dialogové okno aplikace je na obrázku 4.5.

Poznámka: V průběhu implementace jsem narazil na problémy s názvy proměnných ve vygenerovaných zdrojových souborech. Původně jsem totiž nepředpokládal, že nebudu muset zasahovat do zdrojových skriptů programu Objdictedit. Při rozšíření možnosti konfigurace pro víceosé zařízení vznikl problém s vícenásobnou deklarací proměnné se stejným jménem. Např. *controlword* osy 1 bylo pojmenováno shodně jako *controlword* osy 0 a vznikl konflikt pro překladač. Jediným možným řešením byl zásah do zdrojového kódu programu Objdicedit, přesněji do souboru `gen_cfile.py`. Pro zachování výše popsané funkcionality pro víceosá zařízení je bohužel nutné použít program přiložený na CD nebo z repositáře tohoto projektu na serveru [2]. O začlenění úpravy do projektu CanFestival budu jednat s týmem vývojářů. □



Obrázek 4.5: Vygenerování zdrojových souborů nového slovníku zařízení v programu Objdictedit

4.3 Popis funkce ukázkových aplikací

Programy a funkce popsané v této kapitole umožňují předávání dat pro řízení pohonů prostřednictvím CANOpen. K vlastní komunikaci a práci se slovníkem zařízení je použit projekt CanFestival, popsaný v sekci 4.2. Doplněny pak byly funkce pro řízení pohonů, pracující podle specifikace DS-402. Byla vytvořena aplikace pro uzel master, který konfiguruje podřízené uzly slave a který přímo komunikuje s uživatelem. Vlastní zpětnovazební řízení každé osy ze soustavy pohonů se provádí prostřednictvím PXMC v uzlech slave, kde je implementováno taktéž rozhraní CANOpen s profilem pro řízení pohonů.

Celá filosofie řízení je založena na modelu master/slave. Uzel master, jak se o něm hovoří v této kapitole, je v programu reprezentován ukázkovou aplikací `cf_mcp_master_demo`. Ta je určena pro operační systém Linux a je jí možno spustit např. na PC nebo na řídícím počítači MPC5200. Uzel master uvede po spuštění všechny uzly slave do komunikačního stavu OPERATIONAL a poté v terminálu komunikuje s operátorem pomocí příkazů popsaných v sekci 4.5 a chybových či informačních hlášení. Jednotlivé uzly slave reprezentuje aplikace `cf_mcp_pxmc_demo`, určená zejména pro mikrokontrolér H8S/2638, ale kterou lze díky možnostem OMK přeložit třeba pro účely ladění i pro jinou platformu, např. OS Linux. Aplikace uzlu slave obsahuje funkce z knihovny PXMC a přímo provádí řízení motorů podle žádaných hodnot přicházejících z uzlu master, tedy pomocí CANOpen komunikace a jejího standardizovaného profilu. Pro zjištění případné poruchy komunikace uzlu slave je použit tzv. Heartbeat Protocol. Uzel master nastaví ve fázi inicializace každému uzlu slave periodu, se kterou odesílá do sítě NMT zprávu informující o aktuálním komunikačním stavu. Na uzlu master je perioda příchozích NMT zpráv od každého uzlu slave kontrolována a při překročení časového limitu se generuje chybové hlášení.

Každý uzel v síti má svůj slovník zařízení. Jak je popsáno v [10], slovník zařízení může

v oblasti vyhrazené pro standardizovaný profil nést informace o maximálně osmi řízených osách. Tato skutečnost trochu omezuje variabilitu sítě, neboť jeden uzel master může řídit maximálně 8 os, rozdělených libovolně na několika uzlech slave. Jeden mezní případ může být takový, že je řízeno všech 8 os z jednoho uzlu master prostřednictvím jednoho uzlu slave. PXMC umožňuje řízení teoreticky neomezeného počtu motorů, ale ve skutečnosti jsme omezeni možnostmi hardwaru a počtem výkonových budičů pro motory. Přirozeně, že použitý mikrokontrolér má omezený výpočetní výkon. Druhý mezní případ konfigurace sítě je existence osmi uzlů slave, kde každý řídí jednu osu.

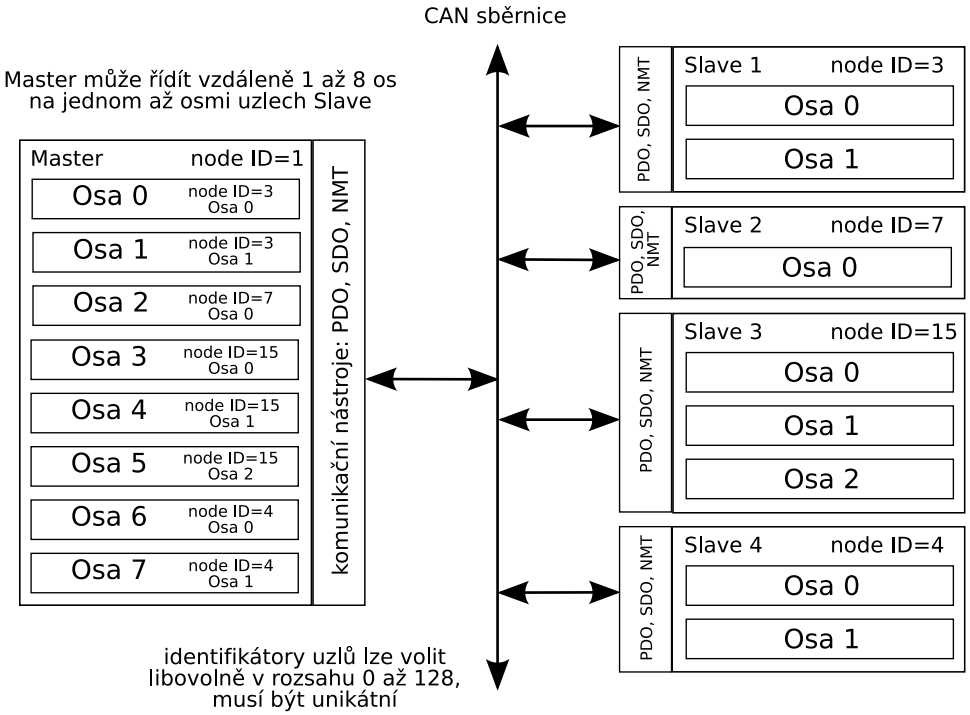
Obecný příklad konfigurace sítě je na obrázku 4.6. Je zde naznačeno, že proměnné slovníku zařízení každé osy na uzlu slave jsou aktualizovány pomocí PDO zpráv do kopie části slovníku na uzlu master. Tímto způsobem je zajištěno, že operátor nebo nadřazená aplikace má k dispozici aktuální data o každém z řízených pohonů. Perioda aktualizace je rovna periodě posílání PDO zpráv, jejichž odesílání je řízeno z uzlu master pomocí synchronizačních objektů (SYNC). Perioda odesílání SYNC objektů je na uzlu master nastavena na indexu 1006_h (*communication cycle period*) a zadává se v mikrosekundách. Reálně dosažitelná perioda je v rozsahu mezi desítkami a stovkami milisekund. Záleží to na množství operací, které se každou periodu musí vykonat.

Konfigurace sítě se provádí před spuštěním komunikace staticky pomocí sestavení a naplnění příslušné datové struktury na uzlu master. Ten jediný má pak informace o počtu a nastavení uzlů slave, které komunikují pouze se svým uzlem master, nikoliv mezi sebou. Statické datové pole pro příklad sítě uvedené na obrázku 4.6 je ve zdrojovém kódu definováno následovně.

```
mcp_node_t mcp_nodes[]= {
        {.node_ID=3,.number_of_axles=2},
        {.node_ID=7,.number_of_axles=1},
        {.node_ID=15,.number_of_axles=3},
        {.node_ID=4,.number_of_axles=2}};

mcp_node_arr_t mcp_node_arr = {
        mcp_nodes:mcp_nodes,
        length:FIELD_ELEMS(mcp_nodes)};
```

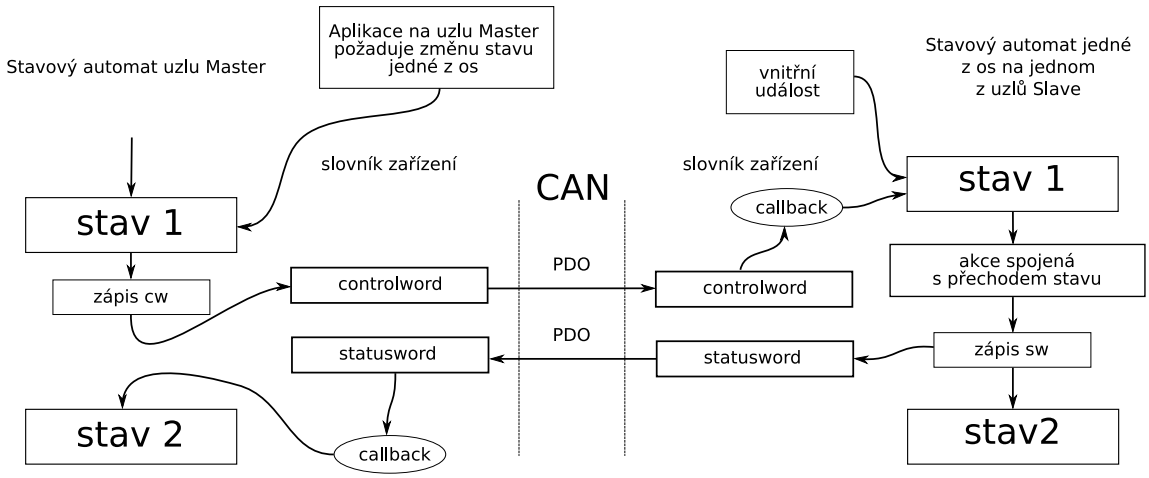
Volba identifikátorů jednotlivých uzlů je libovolná, každý musí být ovšem unikátní. V popisovaných aplikacích se identifikátor zatím nastavuje staticky již v době překladu aplikace. Konfigurace sítě je tak plně v rukou programátora. Pokud by byly uzly vybaveny externími přepínači na portu mikroprocesoru, dalo by se nastavení jejich identifikátorů provést i bez nut-



Obrázek 4.6: Data řízených os ve slovníku zařízení uzlu Master a v uzlech Slave

nosti nového překladu aplikace, jak tomu bývá u průmyslových modulů se sběrnicí CAN.

Uzel slave musí podle specifikace pro každou řízenou osu implementovat stavový automat. Jeho stavy a přechody stavů jsou popsány v 3.5.3 a na obrázku 3.11. Uzel master uchovává informace o aktuálním stavu každé osy pomocí podobného stavového automatu, se stejným počtem stavů i přechodů, ovšem s jinými vazbami na vnitřní události. Pokud si například operátor přeje změnu aktuálního stavu libovolné osy pohonu, zadá požadavek na přepnutí stavu například pomocí příkazového rozhraní. Prostřednictvím řídícího slova (*controlword*) se v PDO zprávě přenese požadavek na příslušný uzel slave. Automaticky se vyvolá callback na zápis nového řídícího slova do slovníku zařízení a pokud to vnitřní podmínky v pohonu dovolují, provede se přechod do nového stavu a s ním spojené úkony. Změna stavu vyvolá i zápis příslušných bitů ve stavovém slově. To je prostřednictvím PDO zpráv odesíláno na uzel master, kde se podobně vyvolá callback na zápis nového stavového slova do slovníku zařízení. Zavolaná funkce provede změnu aktuálního stavu automatu podle informace uložené ve stavovém slově. Viz obrázek 4.7. Pokud nový aktuální stav stále ještě není tím cílovým, vyvolá se automaticky znovu požadavek na přechod stavu prostřednictvím nového řídícího slova. V rámci jednoho zde popsaného cyklu se provede pouze jeden přechod. Tento cyklus se opakuje do té doby, kdy je dosaženo žádaného cílového stavu.



Obrázek 4.7: Vázané stavové automaty pohonu os na uzlu Master a Slave

Může se však stát, že nějaká podmínka přechodu není na uzlu slave splněna a cílový stav tak nemůže být dosažen. Z toho důvodu je při zadání příkazu v příkazovém rozhraní založeno vlákno, které zadaný požadavek předá do nižších vrstev komunikace a na určitou dobu se uspí. Po vypršení časového intervalu odvozeného od periody posílání PDO zpráv se kontroluje dosažení žádaného stavu. Není-li tento stav dosažen, je vygenerováno chybové hlášení. V opačném případě vlákno obsluhující vykonání žádaného příkazu zaniká. I ostatní příkazy, které budou popsány v sekci 4.5 obsluhují časovaná vlákna. Správné provedení příkazu je po krátké době kontrolováno a v případě neúspěchu je v terminálu aplikace indikována chyba.

Poznámka: Způsob zadávání řídících příkazů a kontrolování jejich úspěšného provedení lze implementovat jako čistě událostmi řízený systém. Norma DS-402 nepředepisuje způsob, jakým řešit chybové situace. Dává však k dispozici předdefinované objekty EMCY, sloužící k hlášení chybových událostí pomocí číselných kódů chyb, které nastaly. Z důvodu poměrně rozsáhlého způsobu řešení chybových situací, jsem se již k implementaci těchto metod z časových důvodů nedostal. Řešení v podobě použití časových intervalů pro kontrolu správného provedení úkonu nenahrazuje plně definovanou funkcionalitu a neumožňuje jednoduše zjistit příčinu chyby. □

Přechod stavu pohonu může být vyvolán i bez příkazu z řídícího slova nějakou samostatnou vnitřní událostí. I v tomto případě se provede automaticky zápis nového stavového slova a PDO zprávou se dostane informace na uzel master, kde je příslušný stav také aktualizován. Výše popsaná funkce je graficky zobrazena na obrázku 4.7.

V současné době je implementován periodický přenos čtyř vysílaných a čtyř přijímaných PDO zpráv, v tabulkách 3.5 a 3.6 značených čísly 1 až 4. Perioda vysílání je řízena periodou objektů SYNC. V CanFestivalu je použití libovolné proměnné na příslušném indexu slovníku zařízení nutné deklarovat již při jeho sestavování, např. pomocí programu Objdictedit. Za běhu programu je možné položky slovníku měnit včetně jejich parametrů, není již však možno jednoduše nové položky přidávat. Počet zmíněných PDO zpráv je tedy určen již v době překladu programu a není ho možné za běhu měnit. Zvolil jsem proto použití osmi zmíněných PDO zpráv pro dva implementované módy řízení pozice a rychlosti.

Po startu zařízení, ještě před započetím komunikace po sběrnici, se ve všech uzlech provede inicializace parametrů PDO zpráv. Parametry PDO zpráv (ne již počet zpráv) je pak již možno měnit např. SDO zprávami nebo přímo z lokální aplikace.

Poznámka: V průběhu implementace jsem zjistil, že použitý software projektu CanFestival mapování PDO zpráv s offsetem 40_h neumožňuje. Byl proto zvolen v implementaci plovoucí offset o počtu mapovaných zpráv každé osy. Funkce CanFestivalu odesílající a přijímající zprávy začíná mapovat na indexu 1400_h resp. 1800_h, ale v cyklu inkrementuje indexy o jedničku až do počtu všech mapovaných zpráv. PDO zprávy mapované na indexech 1440_h resp. 1840_h a dále již nenachází. V tomto směru jsem se musel odchýlit od specifikace DS-402. Způsob mapování v mé aplikaci je možné měnit nastavením předdefinovaných konstant v hlavičkovém souboru. □

4.4 Popis zdrojových souborů

Zdrojové kódy funkcí a aplikací jsou rozděleny do tří adresářů s následujícími soubory. Soubory `Makefile` a `Makefile.omk` slouží ke konfiguraci kompilačního systému OMK. V souboru `Makefile.omk` je uvedeno, které zdrojové soubory a knihovny budou použity k překladu aplikace nebo modulu. V následujících několika sekcích je ve stručnosti přiblížen obsah zdrojových souborů ukázkových aplikací. Podrobnější komentáře jsou přímo v souborech.

```
./cf-common/
        Makefile
        Makefile.omk
        cf_mcp_common_DS301.h
        cf_mcp_common_DSP402.h
        cf_mcp_common_def.h
```

```
        cf_mcp_common_fcn.h
        cf_mcp_common_fcn.c
        cf_mcp_common_struct.h
./cf-master/
        Makefile
        Makefile.omk
        cf_mcp_master_cmd.c
        cf_mcp_master_def.h
        cf_mcp_master_demo.c
        cf_mcp_master_fcn.c
        cf_mcp_master_fsm.c
        cf_mcp_master_objdict.od
        cf_mcp_master_objdict.c
        cf_mcp_master_objdict.h
./cf-pxmc/
        Makefile
        Makefile.omk
        cf_mcp_pxmc_def.h
        cf_mcp_pxmc_demo.c
        cf_mcp_pxmc_fcn.c
        cf_mcp_pxmc_fsm.c
        cf_mcp_pxmc_objdict.od
        cf_mcp_pxmc_objdict.c
        cf_mcp_pxmc_objdict.h
```

4.4.1 Adresář `cf-common`

4.4.1.1 Soubor `cf_mcp_common_DS301.h`

Obsahuje definice maker pro práci s parametry a mapováním PDO zpráv.

4.4.1.2 Soubor `cf_mcp_common_DSP402.h`

Zde jsou uvedena makra počítající hodnotu indexu proměnné z jejího názvu uvedeného ve specifikaci DSP-402 a čísla osy od 0 do 7. Ve výpočtu se uvažuje offset stejnojmenných indexů dvou os 800_h a výsledkem je hodnota indexu v rozsahu určeném pro standardizovaný profil zařízení (*Standardized Device Profile*).

4.4.1.3 Soubor `cf_mcp_common_def.h`

Obsahuje definici maker pro práci s řídící a stavovou proměnnou přepínače operačních módů (*Profile Position Mode, Profile Velocity Mode, atd*). Dále jsou zde makra pro práci s řídícím slovem (*controlword*) a stavovým slovem zařízení (*Statusword*). Stavová a řídící slova jednotlivých os se čtou a nastavují pomocí bitových masek, které jsou zde definovány.

4.4.1.4 Soubor `cf_mcp_common_fcn.h`

Obsahuje hlavičky funkcí definovaných v souboru `cf_mcp_common_fcn.c`.

4.4.1.5 Soubor `cf_mcp_common_fcn.c`

Zde jsou definovány funkce použité jak v souborech pro uzel master, tak v souborech pro uzel slave (pojmenovány cf_mcp_pxmc_*).

4.4.1.6 Soubor `cf_mcp_common_struct.h`

Deklaruje stavy stavového automatu ve výčtovém typu `enum mcp_axle_state`. Dále deklaruje strukturu `struct mcp_axle` s ukazateli na často používané proměnné každé z os. Je zde také proměnná nesoucí aktuální stav pohonu.

4.4.2 Adresář `cf-master`

4.4.2.1 Soubor `cf_mcp_master_cmd.c`

Obsahuje funkce volané příkazovým rozhraním po zadání příkazu v terminálu programu `cf_mcp_master_demo`.

4.4.2.2 Soubor `cf_mcp_master_def.h`

Obsahuje hlavičky funkcí definovaných v souboru `cf_mcp_master_fcn.c`.

4.4.2.3 Soubor `cf_mcp_master_demo.c`

Obsahuje funkci main a definici globálních proměnných programu `cf_mcp_master_demo`.

4.4.2.4 Soubor `cf_mcp_master_fcn.c`

Definuje funkce použité v programu `cf_mcp_master_demo`. Jsou to například callback funkce některých položek slovníku zařízení volané automaticky CanFestivalem po zápisu na příslušnou položku.

4.4.2.5 Soubor `cf_mcp_master_fsm.c`

Zde je definována funkce obsluhující stavový automat na uzlu master. Ta přepíná stav stavového automatu osy podle aktuálního stavu a splněných podmínek přechodu.

4.4.2.6 Soubor `cf_mcp_master_objdict.od`

Obsahuje všechny položky slovníku zařízení pro uzel master. Data jsou uložena ve formátu XML a soubor je možné kdykoliv načíst programem Objdictedit a provést změny. Pro demonstraci programem `cf_mcp_master_demo` jsou ve slovníku definovány dvě osy včetně nastavení PDO zpráv.

4.4.2.7 Soubor `cf_mcp_master_objdict.c`

Definice proměnných slovníku zařízení po automatickém vygenerování programem Objdictedit.

4.4.2.8 Soubor `cf_mcp_master_objdict.h`

Deklarace proměnných slovníku zařízení po automatickém vygenerování programem Objdictedit.

4.4.3 Adresář `cf-pxmc`

4.4.3.1 Soubor `cf_mcp_pxmc_def.h`

Tento soubor obsahuje hlavičky funkcí definovaných v souborech `cf_mcp_pxmc_fsm.c` a `cf_mcp_pxmc_fcn.c`.

4.4.3.2 Soubor `cf_mcp_pxmc_fcn.c`

Obsahuje funkce volané z aplikace `cf_mcp_pxmc_demo`. Jedná se zejména o callback funkce položek slovníku zařízení, které mají vazbu na práci s PXMC knihovnou.

4.4.3.3 Soubor `cf_mcp_pxmc_demo.c`

Obsahuje funkci main demonstračního programu `cf_mcp_pxmc_demo`. Pomocí direktiv překladače lze volit překlad pro mikroprocesor H8S nebo pro Linux.

4.4.3.4 Soubor `cf_mcp_pxmc_fsm.c`

Obsahuje funkci obsluhující stavový automat každé osy na uzlu slave. Stavy a jejich přechody jsou vázány pomocí stavového a řídícího slova se stavy odpovídajícího stavového automatu na zařízení master.

4.4.3.5 Soubor `cf_mcp_pxmc_objdict.od`

Obsahuje položky slovníku zařízení uzlu slave ve formátu XML, podobně jako u uzlu master.

4.4.3.6 Soubor `cf_mcp_pxmc_objdict.c`

Definice proměnných slovníku zařízení po automatickém vygenerování programem Objdictedit.

4.4.3.7 Soubor `cf_mcp_pxmc_objdict.h`

Deklarace proměnných slovníku zařízení po automatickém vygenerování programem Objdictedit.

4.5 Příkazy terminálu demonstračního programu

Pro demonstraci funkce řízení pohonů byly vytvořeny nové příkazy do převzatého příkazového rozhraní. Část zdrojových kódů rozhraní, použitá filosofie určující formát příkazů a způsob jejich zpracování, byla převzata od firmy PiKRON, která tímto způsobem dovoluje ovládat svoje jednotky pro řízení malých robotů. Podrobně je formát příkazů popsán například v manuálu k jednotce MARS 2 [19].

Příkaz se skládá ze jména, operačního znaku a parametrů. Jednotlivé části mohou dělit mezery. Formát příkazů je následující.

Jméno je libovolná kombinace písmen a číslic začínající písmenem. Příkazy se vztahují k jednotlivým řízeným osám pomocí posledního písmena příkazu v rozsahu A (pro osu 0) až H (pro osu 7). Tato pozice je dále označována písmenem '*m*'.

Operační znak určuje, jedná-li se o nastavení hodnoty (':') nebo o čtení hodnoty ('?').

Parametr může být v současné době pouze celé číslo. Záporné číslo začíná znakem '-'.

Následující seznam obsahuje implementované příkazy. Znak '/' ukazuje na možnost použití obou typů operačních znaků (':','?').

- **HELP** – vypíše seznam definovaných příkazů s jejich krátkým popisem
- **ON***m:* – parametr není, uvede řízení zadané osy *m* do stavu *OperationEnable*
- **OFF***m:* – parametr není, uvede řízení zadané osy *m* do stavu *SwitchOnDisabled*
- **G***m:* – parametr -2147483647 až 2147483648, přepne osu *m* na operační mód profilovaného řízení pozice a najede na pozici zadanou v aktuálních jednotkách
- **AP***m?* – parametr není, vypíše aktuální pozici osy *m* v aktuálních jednotkách
- **MS***m:/?* – nastaví/vypíše se maximální přejezdovou rychlost v [jednotka pozice/s] osy *m*, parametr v rozsahu 0 až 4294967296
- **MA***m:/?* – nastaví/vypíše se maximální zrychlení v [jednotka pozice/s^2] osy *m*, parametr v rozsahu 0 až 4294967296
- **SPD***m:* – parametr -2147483647 až 2147483648, přepne osu *m* na operační mód profilovaného řízení rychlosti a uvede pohon do pohybu s rychlostí zadanou v aktuálních jednotkách
- **STOP***m:* – provede okamžité zastavení pohonu osy *m* v libovolném operačním módu
- **PF***m:/?* – vstupem jsou dva parametry v rozsahu 1 až 4294967296, parametry převodní konstanty mezi aktuálními jednotkami **pozice** a vnitřními jednotkami pohonu osy *m*, první parametr je čitatel druhý jmenovatel konstanty v podobě zlomku, s operačním znakem '?' se vypíše aktuální nastavení
- **VF***m:/?* – vstupem jsou dva parametry v rozsahu 1 až 4294967296, parametry převodní konstanty mezi aktuálními jednotkami **rychlosti** a vnitřními jednotkami pohonu osy *m*, první parametr je čitatel druhý jmenovatel konstanty v podobě zlomku, s operačním znakem '?' se vypíše aktuální nastavení

- **AF***m:/?* – vstupem jsou dva parametry v rozsahu 1 až 4294967296, parametry převodní konstanty mezi aktuálními jednotkami **zrychlení** a vnitřními jednotkami pohonu osy *m*, první parametr je čitatel druhý jmenovatel konstanty v podobě zlomku, s operačním znakem '?' se vypíše aktuální nastavení

Poznámka: Příkazy **PF**, **VF** a **AF** nastavují převodní konstanty popisované v sekci 3.5.4 a reprezentované ve formě racionálních čísel. Rozdělení převodních konstant na zlomek, umožňuje v prostředí celých čísel zmenšit chyby zaokrouhlování při převodu mezi jednotkami zadávaných veličin a vnitřními jednotkami pohonu (zde např. jednotkami používanými v PXMC). Nevhodnou volbou zadávaných jednotek a z toho vyplývající volbou převodní konstanty zde ale přesto k velkým zaokrouhlovacím chybám může dojít. Je vhodné volit jednotky zadávaných veličin tak, aby byl optimálně využit rozsah zadávaných celých čísel. Zde popisovaný přepočet veličin se provádí na mikroprocesoru H8S/2638 a převodní konstanty jsou uloženy ve slovníku zařízení jednotky slave. □

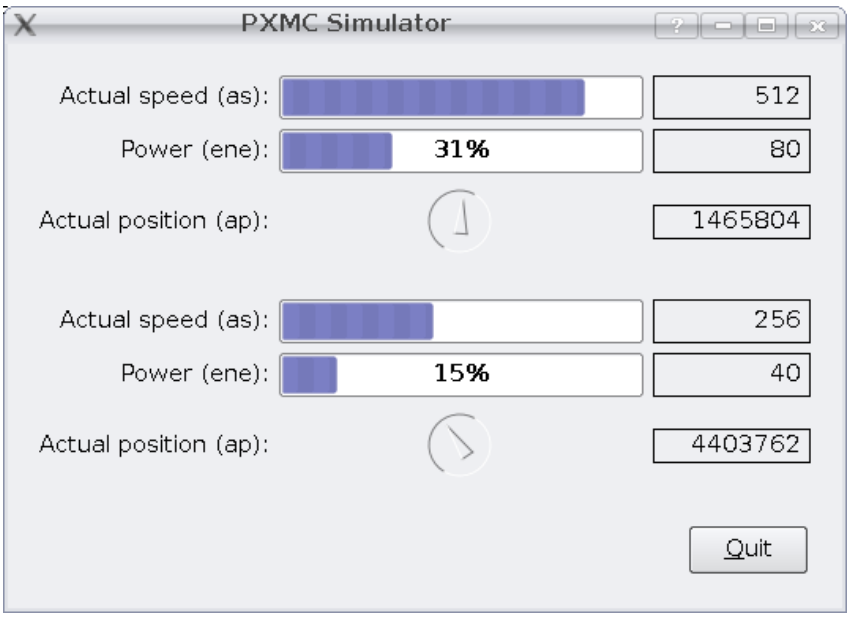
4.6 Spuštění testovací aplikace

V této sekci je popsán postup, jakým lze přeložit a spustit výše uvedené aplikace pro demonstraci CANOpen a Motion Control Profilu. Součástí PXMC je pro účely demonstrace a ladění k dispozici simulátor knihovny PXMC v prostředí OS Linux. Funkce simulátoru je z velké části shodná s funkcí aplikace spuštěné na mikrokontroléru (např. H8S/2638). Hardwarové časovače jsou nahrazeny časovači OS Linux a řídící zásahy regulátorů nepůsobí na fyzický motor, ale jsou prostřednictvím socketu předávány grafické aplikaci. Tímto způsobem je možné pro dva řízené zobrazit aktuální polohu (`pxms_ap`), aktuální rychlost (`pxms_as`), akční zásah regulátoru (`pxms_ene`) a indikovat otáčení rotoru pomocí rotující šipky. Dialogové okno aplikace simulátoru `pxmc_sim_gui` je na obrázku 4.8.

4.6.1 Položky slovníků zařízení

Slovníky zařízení pro aplikace `cf_mcp_master_demo` a `cf_mcp_pxmc_demo` jsou již v tomto projektu vytvořeny. Pro možnost experimentování pouze ve zkratce uvedu, které položky (indexy) jsou ve slovnících uvedeny. Uzel master zde řídí dvě osy na jednom uzlu slave.

Slovník `cf_mcp_master_objdict` obsahuje následující indexy, jejichž parametry je možné v programu Objdictedit měnit.



Obrázek 4.8: Dialogové okno aplikace simulátoru knihovny PXMC

1000_h, 1001_h, 1018_h – vloží se automaticky po vytvoření nového slovníku.

1005_h – identifikátor objektu SYNC použitého pro synchronizaci uzlů v síti.

1006_h – perioda vysílání objektů SYNC, zadaná v mikrosekundách.

1018_h – strukturovaný parametr obsahující definice délek časových intervalů příjemce NMT zpráv protokolu HeartBeat, zde je uveden pouze jeden parametr na subindexu 1 pro jeden uzel slave, který je odesílatelem NMT zpráv protokolu HeartBeat.

1280_h – struktura parametrů jednoho SDO klienta.

1400_h–1407_h, 1600_h–1607_h, 1800_h–1807_h, $1A00_h$–$1A07_h$ – parametry a mapování dvou čtveřic RPDO a dvou čtveřic TPDO zpráv. Poznamenejme, že osa používá 4 RPDO a 4 TPDO zprávy.

6000_h–7000_h – parametry Motion Control profilu dvou řízených os

Slovník `cf_mcp_pxmc_objdict` obsahuje následující indexy, jejichž parametry je možné v programu Objdictedit měnit.

1000_h, 1001_h, 1018_h – vloží se automaticky po vytvoření nového slovníku.

1005_h – identifikátor objektu SYNC použitého pro synchronizaci uzlů v síti.

1006_h – perioda vysílání objektů SYNC, zadaná v mikrosekundách.

1017_h – parametr obsahující definice délky časového intervalu producenta NMT zpráv protokolu HeartBeat.

1280_h – struktura parametrů jednoho SDO serveru.

1400_h–1407_h, 1600_h–1607_h, 1800_h–1807_h, $1A00_h$–$1A07_h$ – parametry a mapování dvou čtveřic RPDO a dvou čtveřic TPDO zpráv. Poznamenejme, že osa používá 4 RPDO a 4 TPDO zprávy.

6000_h–7000_h – parametry Motion Control profilu dvou řízených os

4.6.2 Použití virtuální sběrnice CAN na OS Linux

Pro simulaci komunikace uzlů na sběrnici CAN je zde využit driver LinCAN v konfiguraci jako virtuální zařízení. Překlad zdrojových souborů modulu jádra a jeho zavedení je krok po kroku popsáno v souboru `./can/lincan/README`. Přeložený modul se do jádra zavede následujícím příkazem.

```
insmod can.ko hw=virtual io=0
```

4.6.3 Překlad zdrojových kódů ukázkových aplikací

Překlad aplikací pro OS Linux je prostřednictvím OMK proveden z adresáře `./linux`. Podobně se překlad aplikací pro mikrokontrolér H8S provádí z adresáře `h8300-boot`. V adresáři `./linux` zadáme příkazy:

```
make distclean
make default-config
make
```

Pokud kompilace proběhla úspěšně, jsou v adresáři `./linux/_compiled/bin` spustitelné binární soubory `cf_mcp_master_demo` a `cf_mcp_pxmc_demo`.

4.6.4 Spuštění ukázkových aplikací

Nejprve je nutné spustit simulátor knihovny PXMC `pxmc_sim_gui`. To provedeme následujícími příkazy.

```
../pxmc/pxmcbsp/sim_posix/pxmc_sim_gui/pxmc_sim_gui
```

V terminálu spustíme aplikaci `cf_mcp_pxmc_demo`.

```
_compiled/bin/cf_mcp_pxmc_demo
```

Nakonec spustíme v jiném terminálu aplikaci `cf_mcp_master_demo`.

```
_compiled/bin/cf_mcp_master_demo
```

V terminálu, kde je spuštěna aplikace `cf_mcp_master_demo` je možné zadávat příkazy popsané v sekci 4.5. Abychom mohli vzdálený pohon osy uvést do provozu, je ho nutné nejprve zapnout příkazem `ONA:` pro osu 0 a `ONB:` pro osu 1. Pro jízdu osy 0 na cílovou pozici 1000 můžeme zadat příkaz `GA:1000`. Druhou osu můžeme uvést třeba do pohybu konstantní rychlostí 100. Provedeme to příkazem `SPDB:100`. V libovolný okamžik lze motor zastavit příkazem `STOPA:` nebo `STOPB:`. V dialogovém okně `pxmc_sim_gui` je vidět odezva motorů na zadané příkazy.

4.6.5 Překlad aplikace `cf_mcp_pxmc_demo` pro H8S/2638

Pro spuštění aplikace `cf_mcp_pxmc_demo` na mikroprocesoru H8S/2638 provedeme nejprve překlad zdrojových kódů. V adresáři `./h8300-boot` zadáme příkazy:

```
make distclean
make default-config
make
```

Desku mikroprocesoru spojíme s PC sériovou linkou RS232. V podadresáři se zdrojovými kódy aplikace `cf_mcp_pxmc_demo` spustíme nahrávání binárního souboru do mikroprocesoru.

```
make
make load run
```

Kapitola 5

Závěr

Jedním z cílů této práce bylo zkonstruovat a postavit robota pro soutěž Eurobot 2007. Toto zadání bylo splněno v celém rozsahu. Při návrhu mechanické konstrukce byl kladen na mechanickou odolnost a variabilitu kostry robota, kterou tvoří rám a základní deska s namontovaným pohonem. Jako použitý materiál byla zvolena lehká hliníková slitina, která poskytuje dostatečnou pevnost, nízkou hmotnost a dobrou obrobitelnost. Pro dosažení vysoké přesnosti byla k výrobě některých dílů použita CNC technologie. Rovněž byl proveden návrh pohonu včetně volby vhodných motorů.

Na univerzální kostru byly poté montovány mechanismy specifické pro aktuální zadání soutěže. Funkce sběracího mechanismu a zásobníku byla spolu s celkem testována a doladěna na vlastním hřišti, které jsme pro tyto účely postavili v prostorách katedry. S robotem jsme se zúčastnili republikového kola Eurobota a mezinárodní soutěže North Star Cup v Ruském Petrohradu, kde jsme obsadili páté místo a obdrželi cenu za nejlepší technickou prezentaci.

Dalším cílem práce bylo vytvořit programovou podporu pro CANOpen komunikaci a řízení pohonů. Jako implementaci protokolu CANOpen jsem zvolil projekt CanFestival, pro který již existovala podpora pro mikrokontrolér H8S/2638 a OS Linux. Profil pro řízení pohonů zde ale definovaný nebyl, a tak sem jej musel po nastudování specifikace DSP-402 vytvořit. Struktura konfiguračních souborů programu Objdictedit ovšem neumožňovala sestavení slovníku zařízení pro více řízených os pohonu. Po prostudování zdrojových kódů jsem vytvořil konfigurační soubor pro profil řízení pohonů tak, aby byl maximálně dodržen standard a byla zachována standardní funkce editoru Objdictedit. Vzhledem k tomu, že k tvorbě profilů v programu Objdictedit neexistuje žádná dokumentace, popsal jsem v této práci strukutu souboru *.prf a vysvětlil význam nejpoužívanějších symbolů.

Pro demonstraci řízení pohonů pomocí profilu jsem vytvořil ukázkové aplikace jednotek slave a master. Jednotku slave je v současné době možno spustit v operačním systému Li-

nux se simulátorem motoru nebo na mikrokontroléru H8S/2638. Pro CANOpen komunikaci je použit CanFestival a funkci regulátoru zde plní knihovna PXMC. Podle specifikace DSP-402 byly vytvořeny funkce pro ovládání stavu zařízení a implementován stavový automat pohonu. Vytvořena byla také skupina funkcí implementujících požadované chování uzlu podle profilu MCP a zprostředkovávající výměnu dat mezi PXMC a slovníkem zařízení v CanFestivalu. V současné době je možné uzlem slave přímo řídit až 8 pohonů, tedy maximální počet definovaný normou. Jednotka master je určena pouze pro operační systém Linux. Pomocí jednoduchých příkazů a výpisů terminálu umožňuje operátorovi ovládat pohony připojené k síti.

V současné době je v aplikacích implementován mód profilovaného řízení pozice (single setpoint) a mód profilovaného řízení rychlosti. K výměně procesních dat mezi uzlem master a slave, jako jsou např. aktuální a žádaná pozice, slouží periodicky předávané PDO zprávy. K jednorázovému nastavení položek slovníku zařízení slave používá aplikace master SDO zprávy. Konfigurace sítě se provádí staticky na uzlu master a lze ji měnit i za běhu programu. Nejsou zde ovšem implementovány žádné diagnostické nástroje pro dynamickou úpravu konfigurace sítě, které mají spíše význam u rozsáhlejších sítí v průmyslu. Za běhu je správná funkce uzlů slave ověřována tzv. Heartbeat protokolem, který je součástí standardu CANOpen. V současné době není implementován přenos chybových hlášení pomocí objektů EMCY.

Testování aplikací bylo prováděno na dvou úrovních. Nejprve byla testována komunikace pomocí virtuálně vytvořené sběrnice CAN na osobním počítači a se simulátorem motoru. Poté byla aplikace slave přenesena na mikrokontrolér H8S/2638 osazený na desce Hi_CPU2 vyrobené firmou PiKRON. Aplikace master byla ponechána na osobním počítači a přístup na fyzickou sběrnici CAN byl řešen I/O kartou Kvaser PCIcan 4xHS do PCI slotu.

Literatura

[1] Controller area network.
http://en.wikipedia.org/wiki/Controller_Area_Network, 2008.

[2] Internal site of Real-Time Systems group. http://rtime.felk.cvut.cz, 2008.

[3] MAXON MOTOR. http://www.maxonmotor.com, 2008.

[4] OCERA Make System homepage.
http://rtime.felk.cvut.cz/omk, 2008.

[5] PiKRON spol. s r.o. http://www.pikron.com, 2008.

[6] Socket-can project homepage.
http://developer.berlios.de/projects/socketcan/, 2008.

[7] TYMA CZ spol. s r.o. http://www.tyma.cz, 2008.

[8] ULMER spol. s r.o. http://ulmer.invite.cz, 2008.

[9] UZIMEX spol. s r.o. http://www.uzimex.cz, 2008.

[10] CAN IN AUTOMATION (CIA). *CiA Draft Standard Proposal DSP-402: CANopen device profile for drives and motion control*, 1998. http://www.can-cia.org.

[11] CAN IN AUTOMATION (CIA). *CiA 102 DS V2.0 CAN physical layer for industrial applications*, 1999. http://www.can-cia.org.

[12] CAN IN AUTOMATION (CIA). *CiA 301 DS V4.0.2: CANopen application layer and communication profile*, 2002. http://www.can-cia.org.

[13] ELMO MOTION CONTROL. *CANopen DS 402 Implementation (Guide Ver. 1.2)*, 2006.
http://www.elmomc.com/support.

[14] FREESCALE SEMICONDUCTORS. *MPC5200 Technical Data Sheet*, 2006. http://www.freescale.com.

[15] HEROUT, P. *Učebnice jazyka C–1. díl, IV. přepracované vydání*. No. 80-7232-220-6. České Budějovice: Nakladatelství KOPP, 2004.

[16] HEROUT, P. *Učebnice jazyka C–2. díl, IV. přepracované vydání*. No. 80-7232-221-4. České Budějovice: Nakladatelství KOPP, 2004.

[17] MAREK, S. *Zařízení podporující komunikační protokol CANopen*. Diplomová práce, ČVUT–FEL, Katedra řídicí techniky, Praha, 2006.

[18] OBDRŽÁLEK, D. *Eurobot 2007 rules – Czech version*, 2006. http://www.eurobot.cz.

[19] PÍŠA, P. *Uživatelský manuál k jednotce MARS 2*. PiKRON spol. s r.o., 2004. http://www.pikron.com/pages/products/motion_control/mars_2.html.

[20] RENESAS TECHNOLOGY CORPORATION. *H8S/263X Hardware Manual*, 2007. http://eu.renesas.com/.

[21] SKUP, K. *Motion Control for Mobile Robots.* Diplomová práce, ČVUT–FEL, Katedra řídicí techniky, Praha, 2007.

[22] TISSERANT, E., DUPIN, F., AND BESSARD, L. *CanFestival v3.0 Manual*, 2007. http://www.canfestival.org.

Příloha A

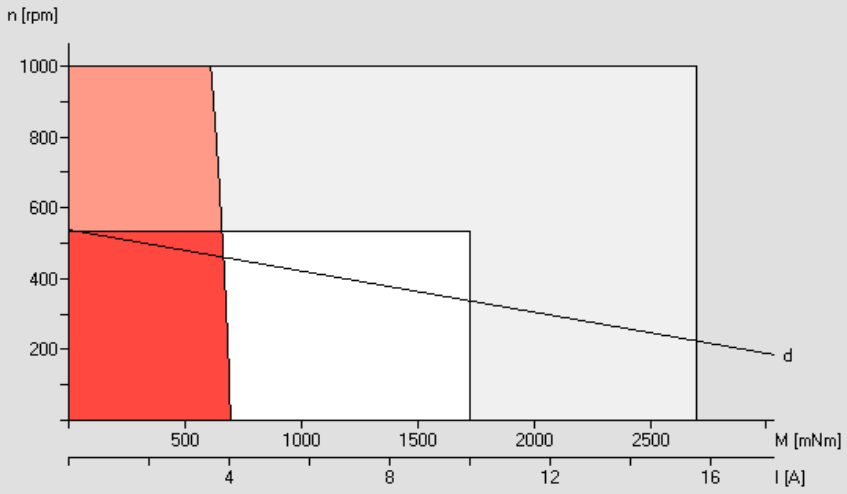
Protokol o výpočtu pohonu

maxon motor
driven by precision

maxon selection program 2005 (V1.0) 1 / 4

maxon drive unit			
Gear	203116	GP 42 C	Gear reduction ratio 15:1, Preloaded ball bearings
Motor	272762	EC-max 30	60W, brushless, 2 Shaft ends, Ball bearings
Control			

Key data of drive		
Available motor voltage (= Umot)	12	V
No-load speed (at Umot)	533	rpm
Load stall torque (theoretic, at Umot)	4630	mNm
Starting current (theoretic, at Umot)	26,8	A
Diameter	42	mm
Length	119,4	mm
Ambient temperature (=Tamb)	25	°C



[ ] Scale motor	[ ] Output power (e)
[x] Working points	[ ] Efficiency (f)
[ ] Max. contin. torque motor (a)	[ ] Winding temperature (g)
[ ] Contin. current (b)	[ ] Max. contin. torque gear (h)
[ ] Max. current (c)	[ ] Max. torque gear (i)
[x] Max. voltage (d)	[ ] Max. speed gear (k)

Power supply input		
Min. supply voltage	0	V
Max. supply voltage	12	V
Max. continuous current	10	A
Max. current (peak)	10	A
Max. duration of peak current	1	s

Drive parameter

maxon motor
driven by precision

Motor order number				272762
EC-max 30, 60W, brushless, 2 Shaft ends, Ball bearings				
Motor data	Min.	Nominal	Max.	
Assigned power rating		60		W
Nominal voltage		12		V
No-load speed	7351	7990	8629	rpm
Stall torque	324	381	438	mNm
Speed-torque gradient	17	21,2	25,4	rpm/mNm
No-load current		268	402	mA
Terminal resistance (phase to phase)	0,416	0,447	0,478	Ohm
Max. speed		15000		rpm
Max. continuous current		4,39		A
Max. continuous torque		55,6		mNm
Reference speed		5000		rpm
Max. efficiency		81		%
Torque constant	13,1	14,2	15,3	mNm/A
Speed constant	618	672	726	rpm/V
Mechanical time constant		4,9		ms
Rotor inertia	19,7	21,9	24,1	gcm²
Terminal inductance (phase to phase)	0,0392	0,049	0,0588	mH
Therm. resistance winding-housing		0,5		K/W
Therm. resistance housing-ambient		7,4		K/W
Therm. time constant winding		2,7		s
Therm. time constant stator		1000		s
Max. winding temperature		155		°C

Motor data at actual motor voltage and ambient temperature	Nominal	
Ambient temperature (=Tamb)	25	°C
Motor voltage (=Umot)	12	V
No-load speed (at Umot and Tamb)	7983	rpm
Stall torque (theoretic, at Umot and Tamb)	380	mNm
Starting current (theoretic, Umot and Tamb)	26,8	A
Max. continuous current (at Tamb)	4,39	A
Max. continuous torque (at Tamb)	62,3	mNm

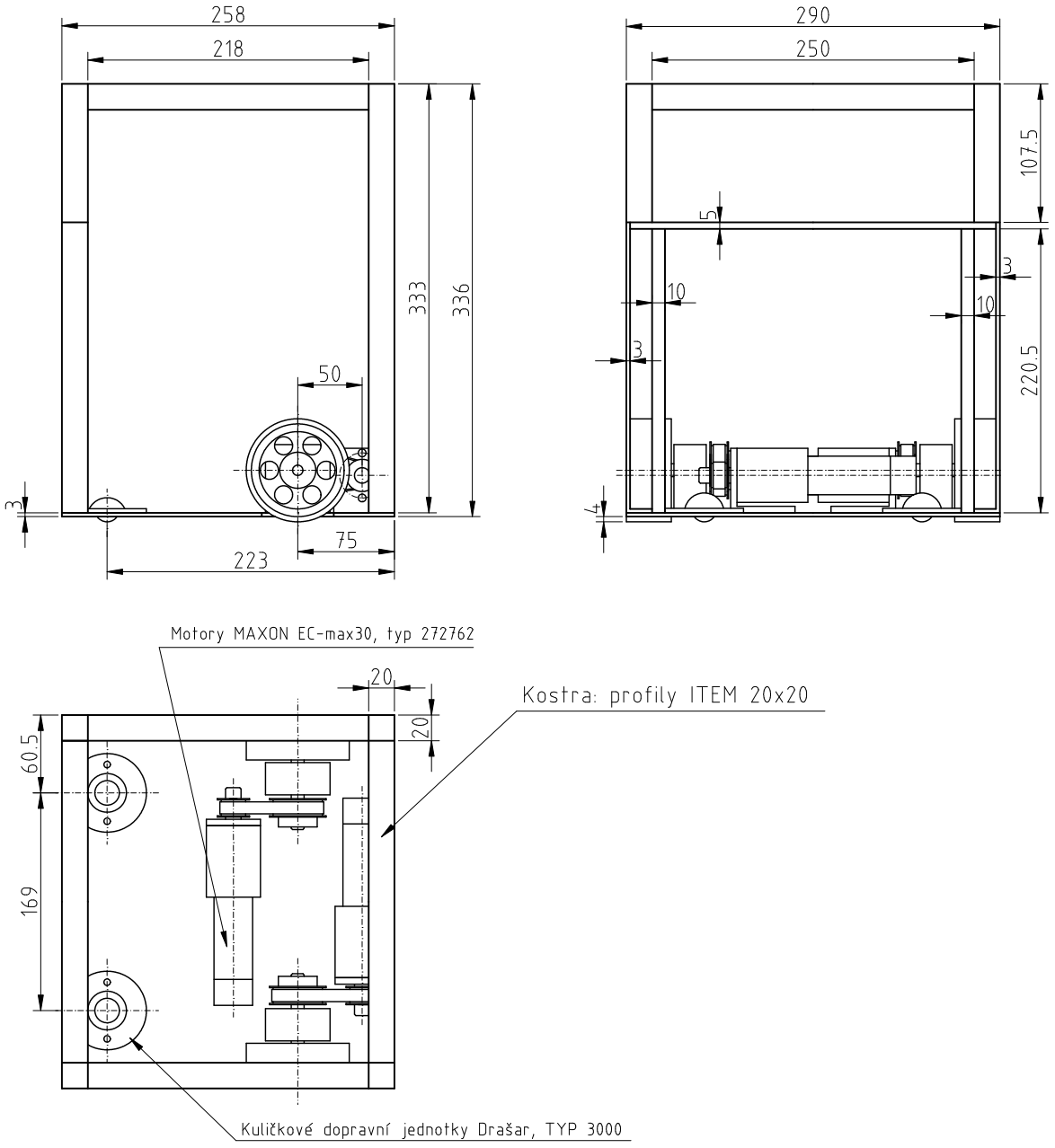
Gear order number		203116
GP 42 C, Gear reduction ratio 15:1, Preloaded ball bearings		
Gear data		
Gear reduction ratio	15	:1
Gear reduction absolute	0	:1
Gear efficiency	81	%
Max. continuous gear torque	7500	mNm
Short-term torque	11300	mNm
Max. gear input speed	8000	rpm
Gear mass inertia	15	gcm²
No of stages	2	-

Controller order number

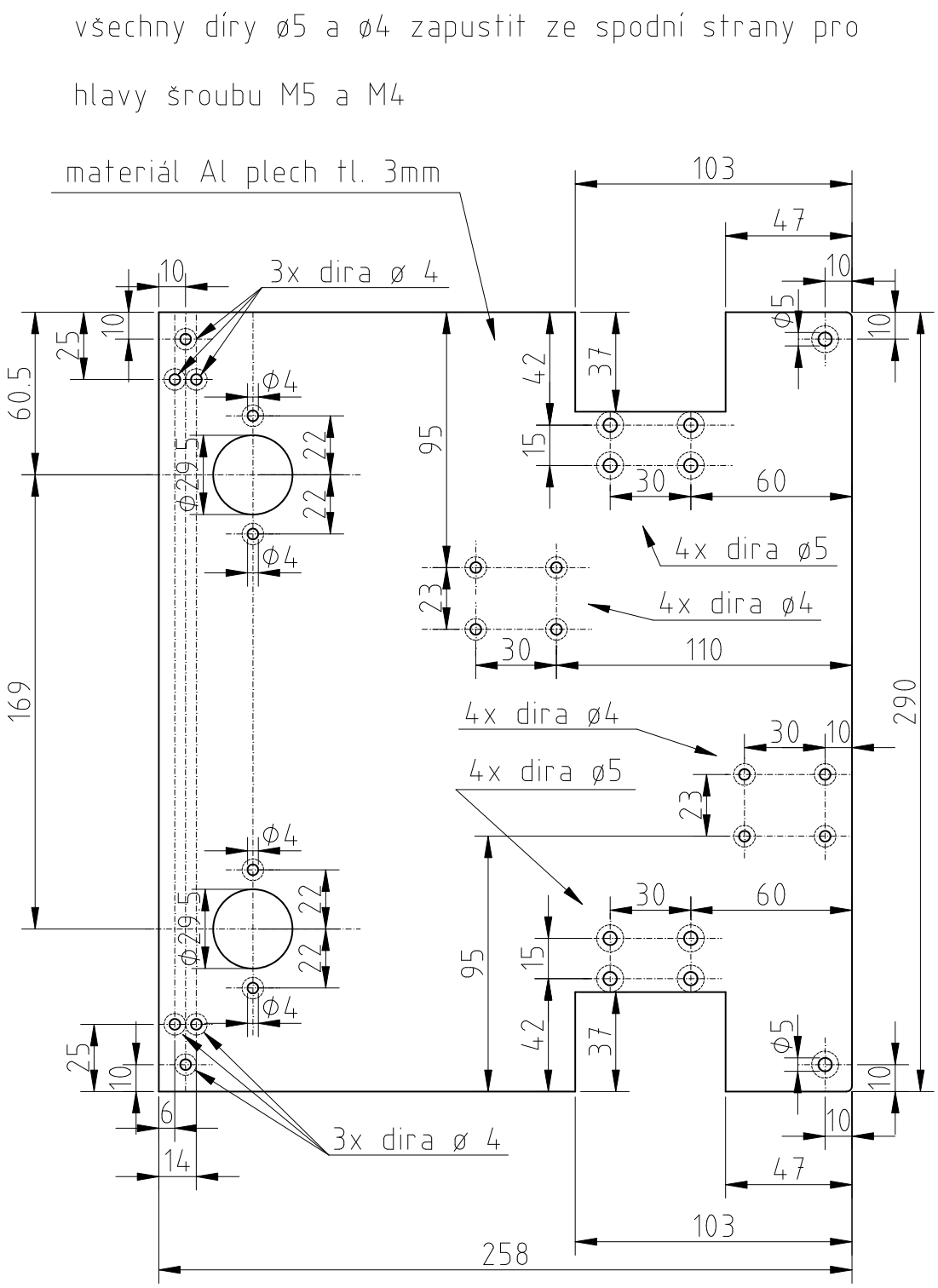
Sensors						
Sensor type	Sensor No.	Counts/turn	Channels	Line Driver	Max.freq.(kHz)	Brake
Encoder HEDL 5540	110514	500	3	Yes	100	No
Encoder	228452	500	3	Yes	200	No
Encoder	228456	1000	3	Yes	200	No

Příloha B

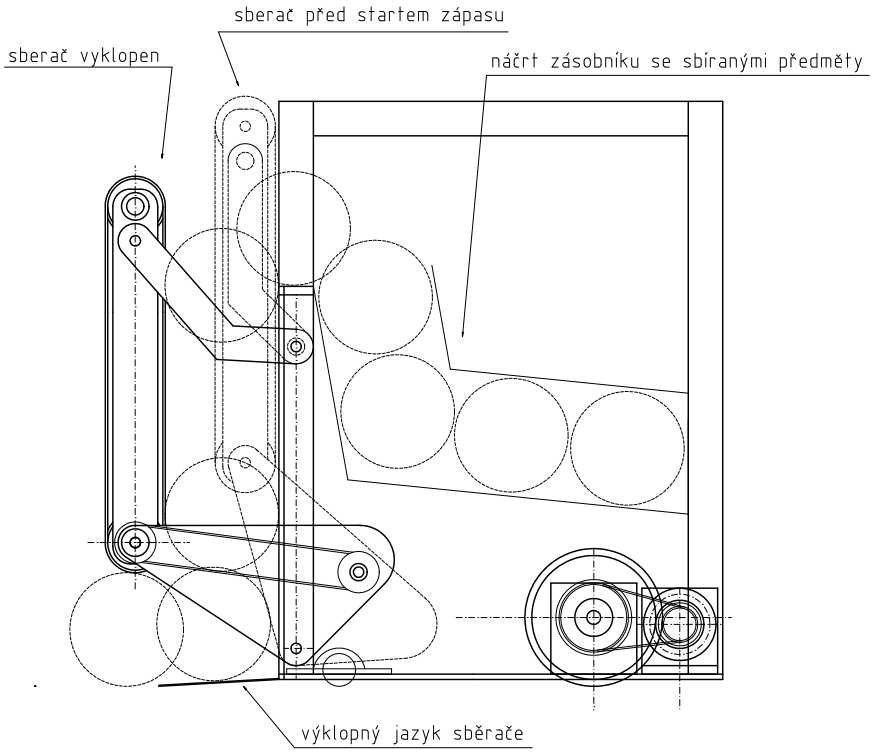
Výkresy některých mechanických součástí

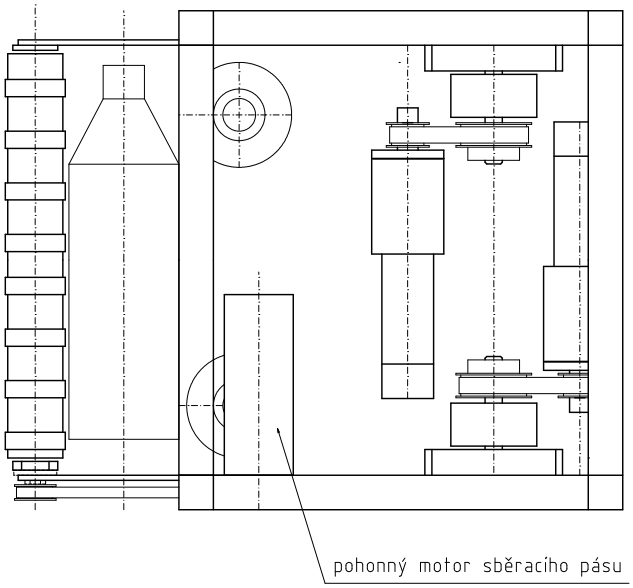


Obrázek B.1: Kostra robota s umístěným pohonem, sestava

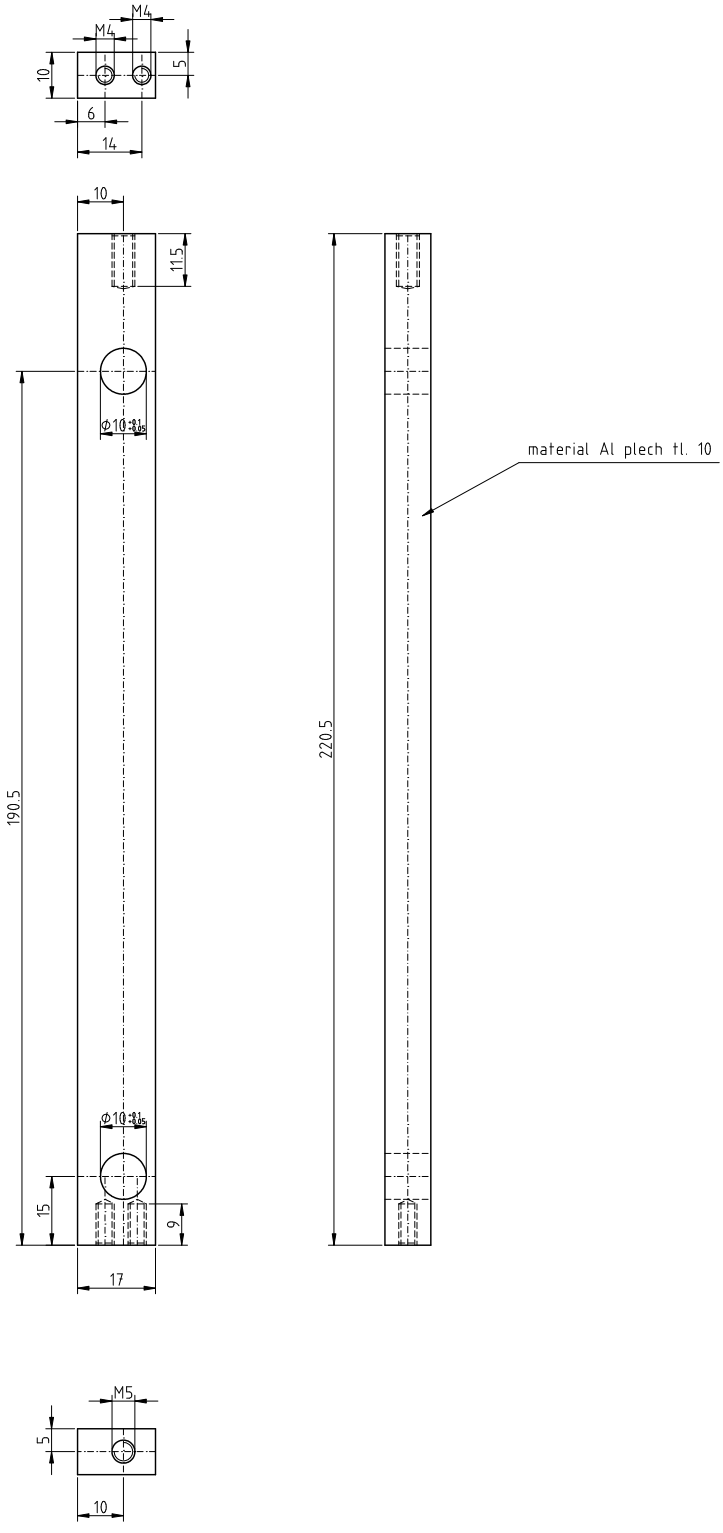


Obrázek B.2: Spodní deska kostry

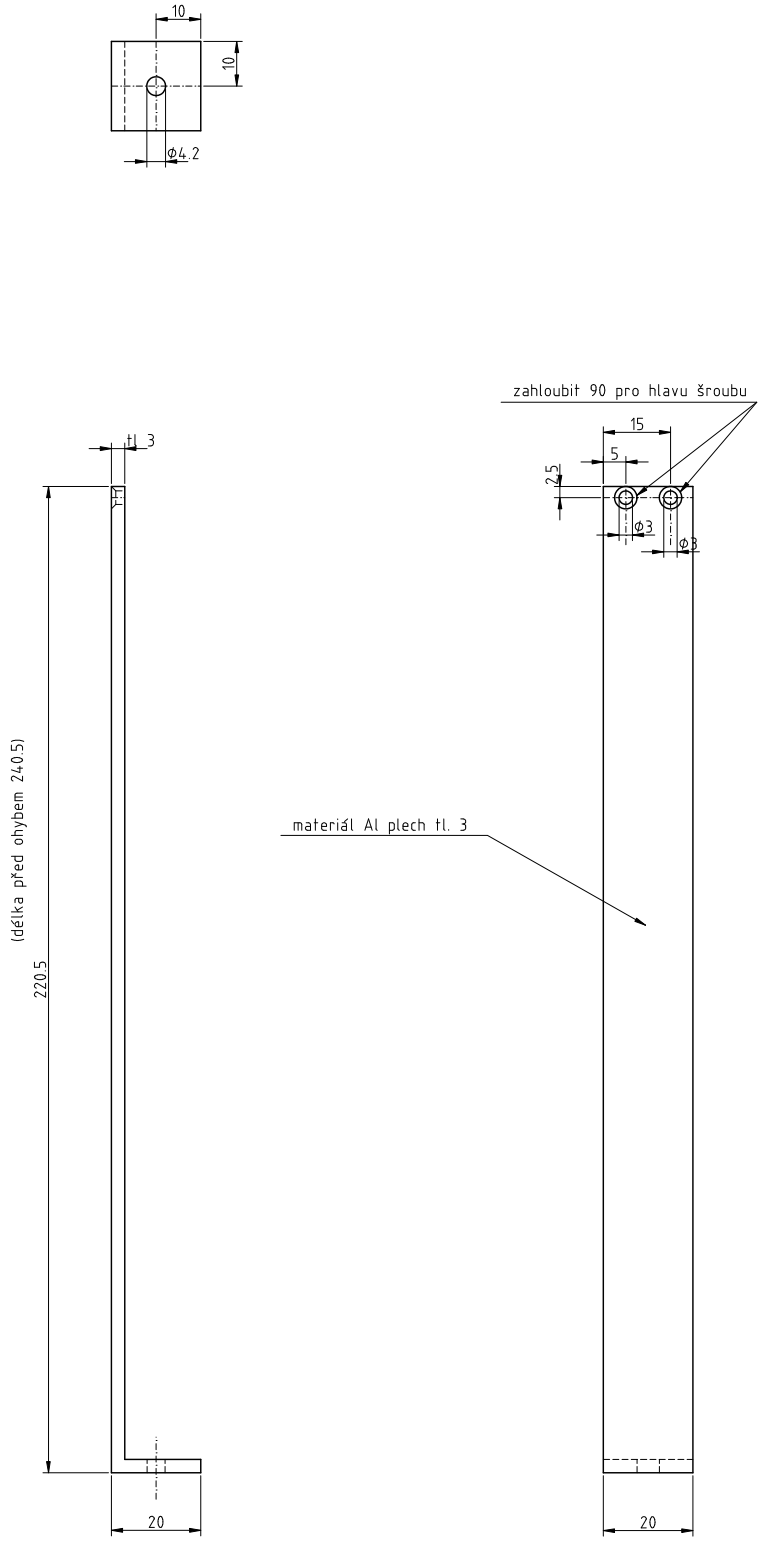




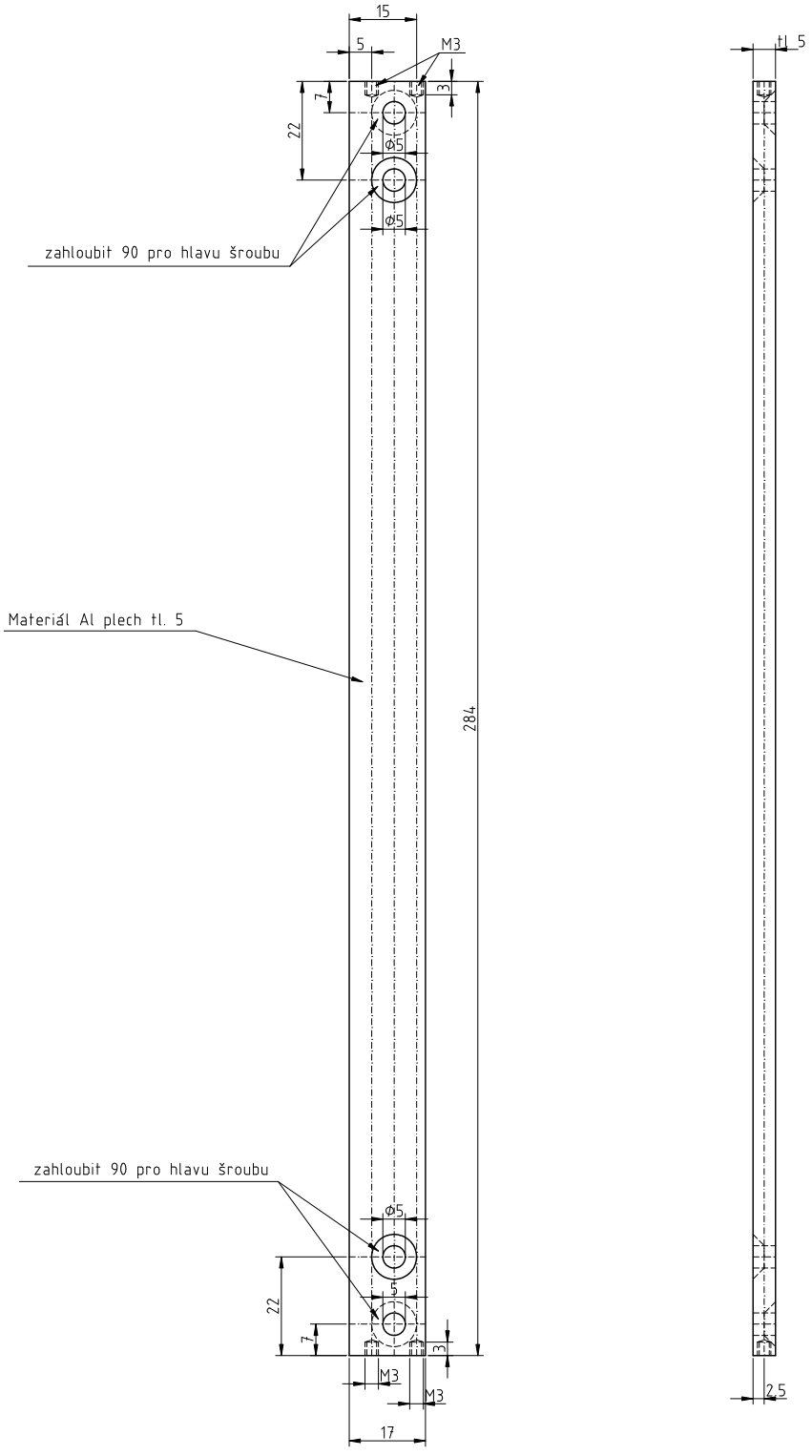
Obrázek B.3: Přední sběrací mechanismus, sestava



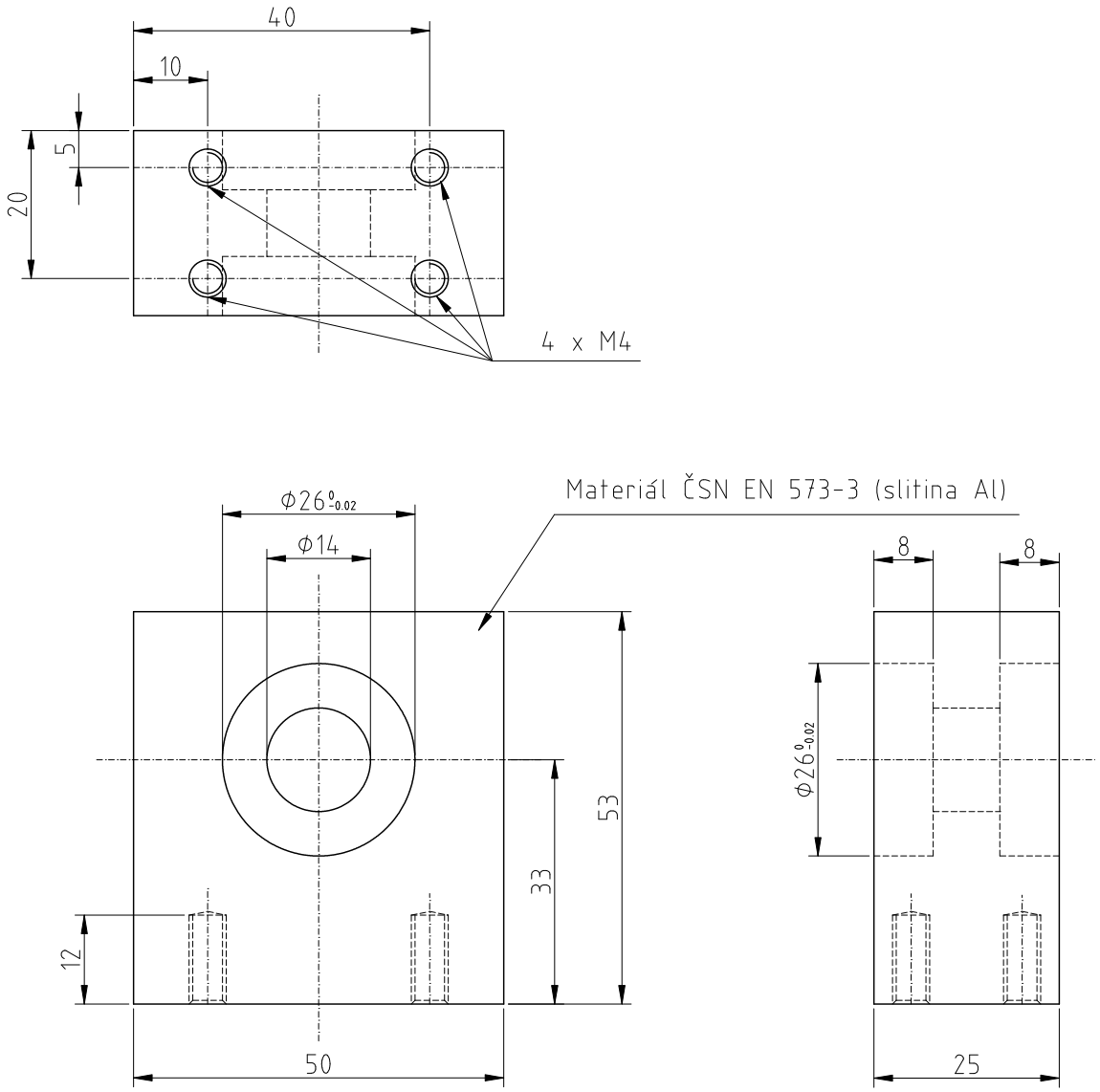
Obrázek B.4: Přední sloupek rámu kostry robota



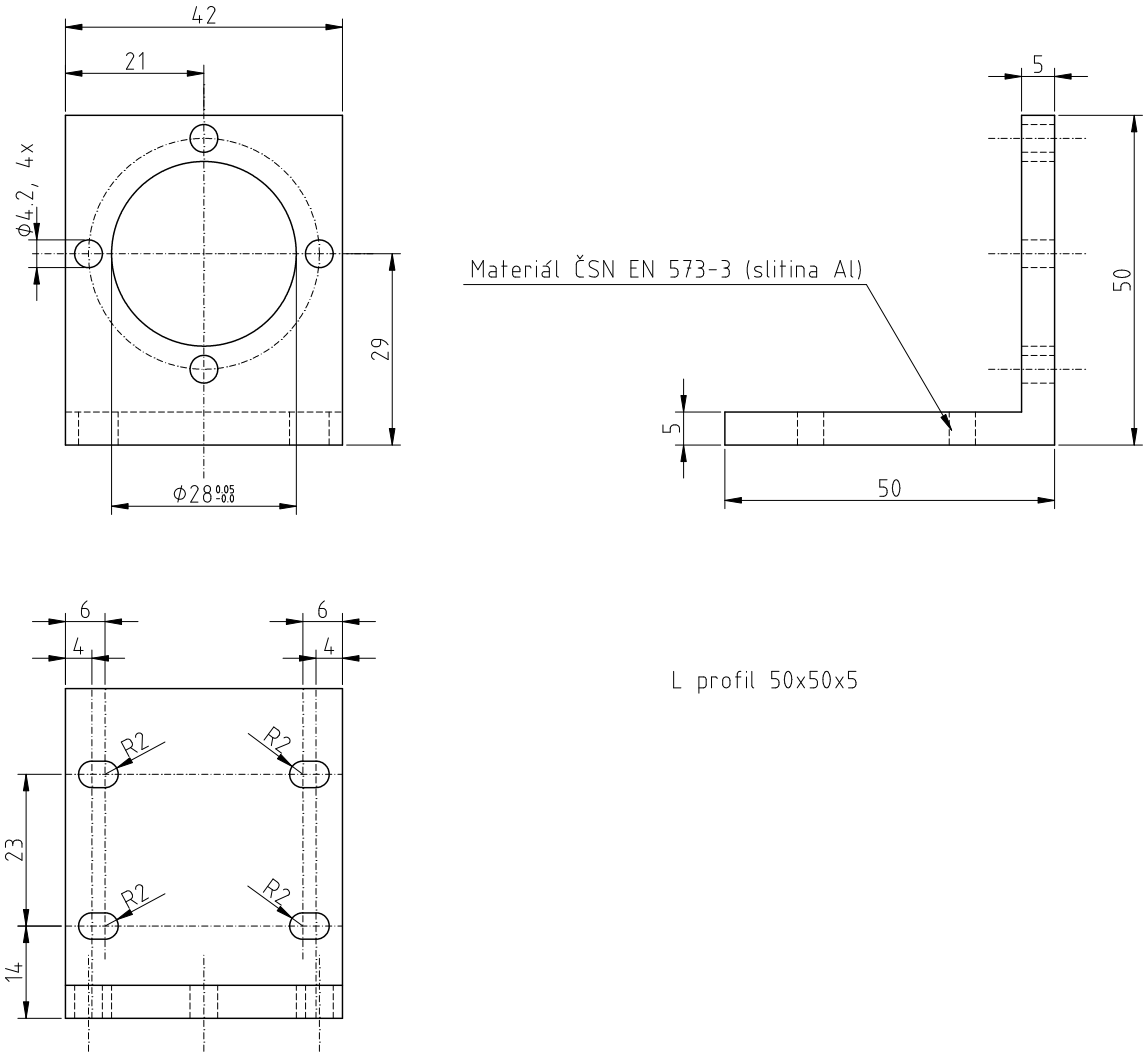
Obrázek B.5: Přední výztužný sloupek rámu kostry robota



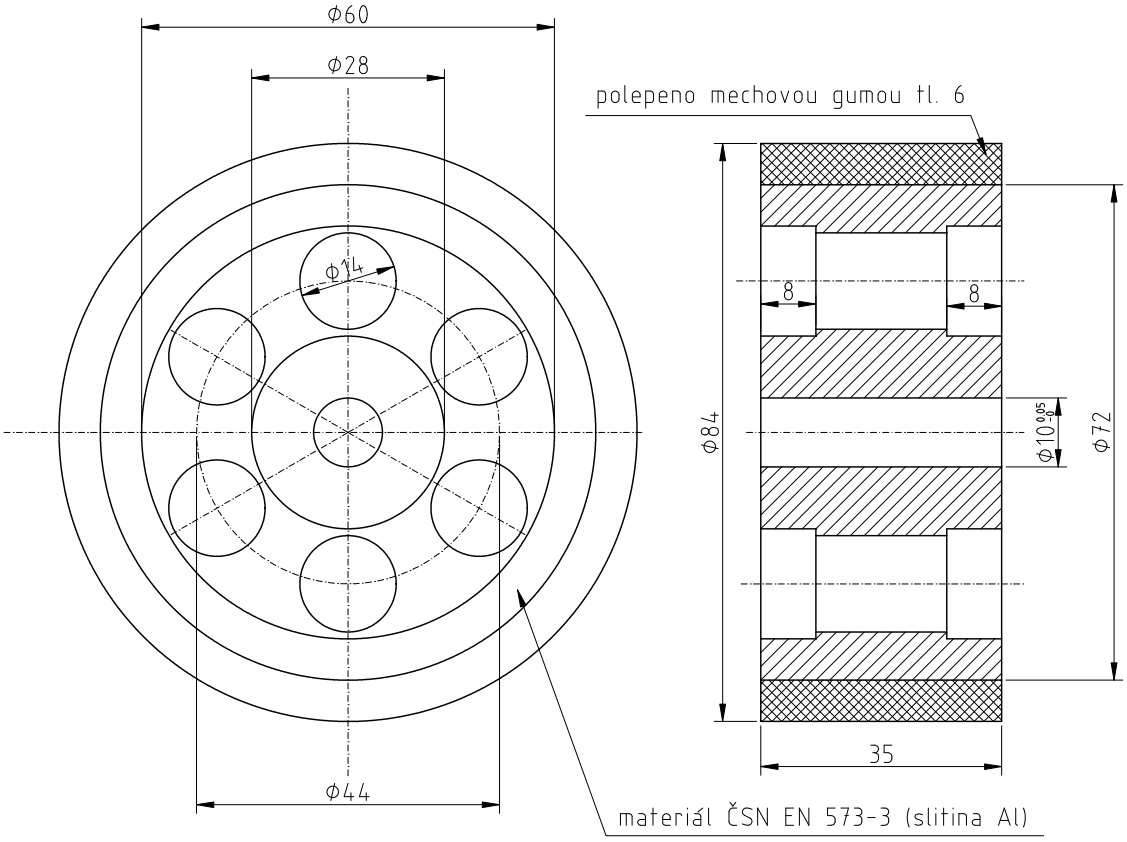
Obrázek B.6: Příčník umístěný v přední části rámu kostry



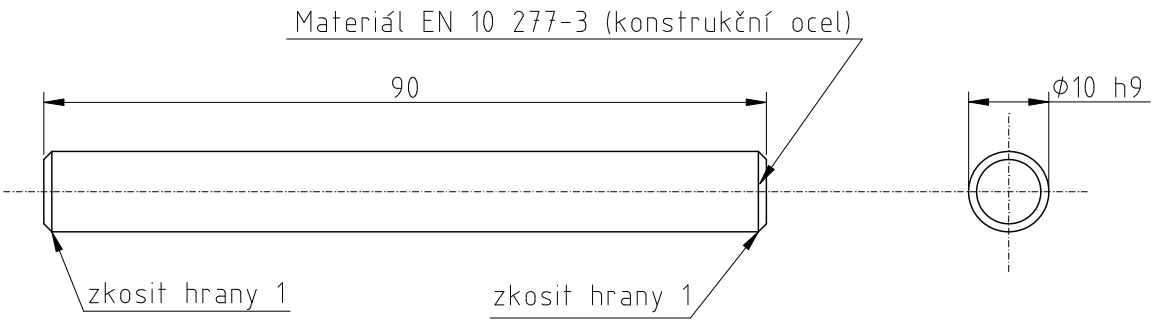
Obrázek B.7: Blok ložiska pohonného kola



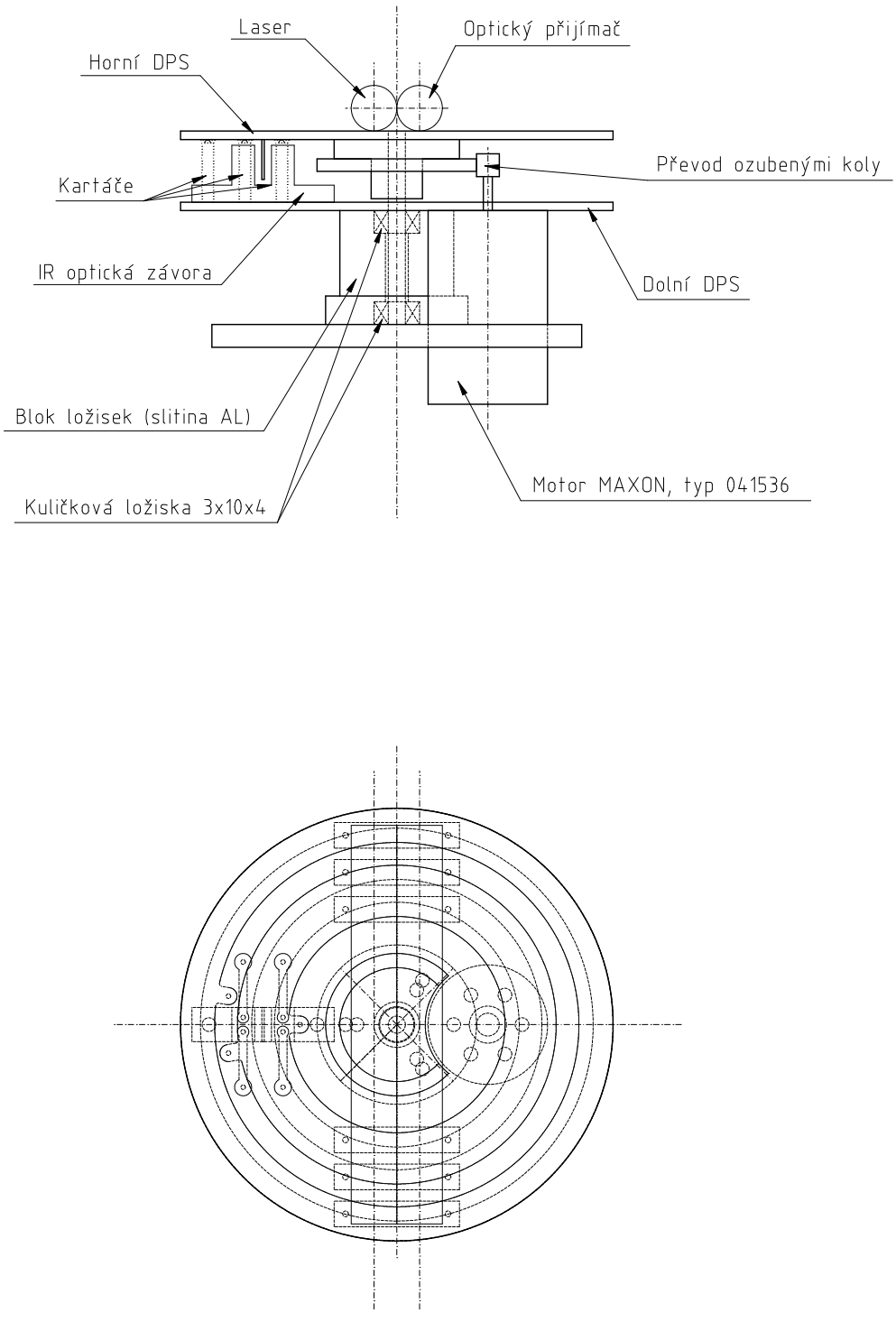
Obrázek B.8: Blok pro připevnění pohonného motoru



Obrázek B.9: Pohonné kolečko



Obrázek B.10: Hřídel pohonného kolečka



Obrázek B.11: Nákres lokalizačního majáku

Příloha C

Obsah přiloženého CD

K této práci je přiloženo CD, na kterém jsou uloženy zdrojové kódy. Adresářová struktura je následující.

`./DP` – tato diplomová práce ve formátu pdf.

`./src` – zdrojové kódy použitých projektů a vytvořených aplikací.