VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

DIGITALIZACE DIAGNOSTICKÝCH DAT Z FREKVENČNÍCH MĚNIČŮ S VYUŽITÍM TECHNOLOGIE B&R APROL EDGE CONTROLLER

DIGITALIZACE DIAGNOSTICKÝCH DAT Z FREKVENČNÍCH MĚNIČŮ S VYUŽITÍM TECHNOLOGIE B&R APROL EDGE CONTROLLER

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Tomáš Mičulka

AUTHOR

VEDOUCÍ PRÁCE — Ing. et Ing. Stanislav Lang, Ph.D.

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Tomáš Mičulka**
Studijní program:	Aplikovaná informatika a řízení
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. et Ing. Stanislav Lang, Ph.D.**
Akademický rok:	2021/22

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Digitalizace diagnostických dat z frekvenčních měničů s využitím technologie B&R Aprol Edge Controller

Stručná charakteristika problematiky úkolu:

Práce je věnována problematice sběru diagnostických dat z frekvenčních měničů, jejich zpracování a vyhodnocení (zejména vhodné vizualizace) v Edge zařízení (IIoT). Praktickou část práce bude student realizovat s využitím nástroje B&R Aprol Edge Controller, kde bude realizovat komunikaci s frekvenčními měniči (pomocí několika komunikačních protokolů), data vyhodnocovat, vhodně vizualizovat a případně i poskytovat do cloudu.
Práce bude realizována ve spolupráci s firmou B+R Automatizace s r.o., která pro realizaci práce poskytne potřebné technické prostředky.

Cíle diplomové práce:

Rešerše v oblasti frekvenčních měničů.
Rešerše v oblasti edge zařízení (IIoT) s důrazem na využití nástroje B&R Aprol Edge Controller.
Průzkum možností datové komunikace mezi vybranými frekvenčními měniči a Edge zařízením.
Realizace připojení vybraných frekvenčních měničů do Edge zařízení prostřednictvím různých komunikačních protokolů.
Sběr, zpracování a vizualizace diagnostických dat z frekvenčních měničů.
Poskytnutí získaných dat do cloudu pomocí MQTT. (nepovinné)

Seznam doporučené literatury:

B&R Industrial Automation. Industrial IoT [online]. [cit. 2020-10-23]. Dostupné z: https://www.br-automation.com/cs/technologie/industrial-iot/

ABB: Low voltage AC drives [online]. [cit. 2020-10-23]. Dostupné z: https://new.abb.com/drives/low-voltage-ac

LIGHT, Roger A. Mosquitto. Server and client implementation of the MQTT protocol. Journal of Open Source Software, 2017, 2.13: 265.

KOZIOREK, Jiří. Distribuované systémy řízení: učební text. Ostrava: Vysoká škola báňská - Technická univerzita, 2011. ISBN 978-80-248-2599-1.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2021/22

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Diplomová práce zpracovává problematiku sběru diagnostických dat z frekvenčních měničů s důrazem na využití B&R EdgeControlleru. Práce se zabývá komunikací mezi měničem a zařízením, dále ukládáním dat pro pozdější analýzu a vizualizací. V realizaci projektu je vytvořena demonstrační verze projektu. V tomto demu je vytvořena komunikace s měničem, pomocí protokolu Powerlink, dále logika pro historizací a v neposlední řadě vizualizace ve formě procesní grafiky a webových reportů.

ABSTRACT

Master thesis deals with the collection of diagnostic data from frequency converters using the B&R EdgeController. Thesis deals with communication between the inverter and the device, as well as data storage for later analysis and visualization. A demonstration version of the project is created in the project implementation. In this demo, communication with the inverter is created, using the Powerlink protocol, as well as logic for historization and, last but not least, visualization in the form of process graphics and web reports.

KLÍČOVÁ SLOVA

frekvenční měnič, diagnostická data, sběr dat, Průmyslová komunikace, Powerlink, Modbus, IoT, vizualizace, edge zařízení, B&R, EdgeController, APROL

KEYWORDS

Frequency converter, dignostic data, data collection, Industry communication, Powerlink, Modbus, IoT, visualization, edge device, B&R, EdgeController, APROL

ÚSTAV AUTOMATIZACE A INFORMATIKY

2022

BIBLIOGRAFICKÁ CITACE

MIČULKA, Tomáš. *Digitalizace diagnostických dat z frekvenčních měničů s využitím technologie B&R Aprol Edge Controller* [online]. Brno, 2022 [cit. 2022-05-16]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/137094. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce Stanislav Lang.

.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu práce Ing. et Ing. Stanislavu Langovi za cenné rady a připomínky, jeho ochotu, čas a odborné vedení diplomové práce. Také bych chtěl poděkovat panu Ing. Maroši Macejovi z firmy B&R a panu Ing. Vladimíru Krajánkovi z firmy ABB za poskytnutí cenných odborných rad při technickém řešení práce. V neposlední řadě patří obrovské poděkování mé rodině, zejména Barboře Pechové, bez které bych tuto práci nemohl dokončit.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., včetně možných trestně právních důsledků.

V Brně dne 20. 5. 2022 .....

Tomáš Mičulka

OBSAH

1 ÚVOD

V dnešní době se málokterá oblast našeho života obejde bez elektrických pohonů. V průmyslu a výrobních podnicích pracuje asi 30 milionů elektromotorů, z toho velký podíl zaujímají asynchronní motory, díky jejich konstrukční jednoduchosti, vysoké spolehlivosti. V případech, kde je vyžadována změna otáček motorů, se obecně používá měnič frekvence. Nasazení měniče umožnuje plynule, nebo skokově měnit rychlost otáčení, případně přímo řídit výstupní moment. Zároveň při použití měniče v rozběžích a doběžích se výrazně snižují velikosti proudových a momentových rázů.

S příchodem digitalizace získaly frekvenční měniče další funkce a tím ještě více na důležitosti. Přes rozhraní lze nastavit parametry, nebo zobrazovat data. Po připojení přes sběrnici (Ethernet, sériová linka, Profibus, Powerlink, atd.), je nastavení parametrů rychlejší a snadnější než pomocí ovládacího panelu. Připojení ke sběrnici nabízí i další možnosti než jen nastavení. Umožňuje monitorování dat, která lze vizualizovat, zpracovat a archivovat. Současně digitalizace umožnuje zefektivnění výroby, prodloužení životnosti, zkrácení doby výpadku při poruše. Všechny tyto vlastnosti umožnují celkové snížení nákladů výroby.

1.1 Vizualizace

Grafické zobrazení dat a procesů je v dnešní době nedílnou součástí většiny aplikací v různých odvětvích průmyslu a IT. Nabízí rychlé a snadné monitorování procesů a poskytování zpětné vazby uživateli, který může reagovat na vzniklé situace. Grafické rozhraní může být řešeno různě. Například textovými výpisy, alarmy, nebo grafickými prvky (trendy, dashboardy, webovými stránkami). Vizualizace a ukládání dat je důležité nejen v krátkodobém horizontu, ale i z pohledu delší doby. Starší data nám slouží při zpětné analýze, jako je například zjištění příčiny vzniku poruchy, porovnání kvality regulace při různých nastaveních nebo statistického vyhodnocení aj.

1.2 Popis diplomové práce

V rámci této diplomové práce by měla být vytvořena aplikace běžící na zařízení EdgeController od firmy B&R. Diplomová práce se zabývá komunikací mezi měničem a zařízením, dále ukládáním dat pro pozdější analýzu technikem a v neposlední řadě navrhuje možné grafické prvky pro usnadnění práce technika. Hlavním cílem je navrhnout řešení pro zařízení sběru a diagnostiky dat z měniče ve výrobě, tak aby servisní technik získal potřebná data pro analýzu stavu měniče a řízené technologie.

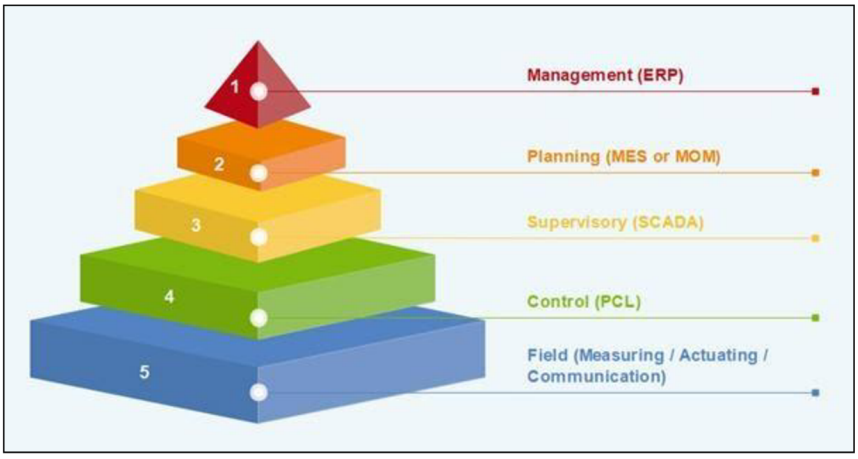
Diplomová práce je rozdělena do několika kapitol, které postupně popisují jednotlivé kroky práce. V úvodu je popsána hierarchie řízení v automatizaci, na kterou navazuje rešerše o Edge zařízeních včetně vysvětlení proč se čím dál častěji používají v průmyslu. Kapitola 4 popisuje řídící systém zajištující sběr dat, historizaci a následné zobrazení operátorovi. Dále se práce zabývá hlavními komunikačními protokoly zajištující komunikaci mezi počítačem a měničem. Následující kapitola 6 zmiňuje možnosti ukládání dat v řídícím systému. Kapitola 7

popisuje princip měniče frekvence a jeho metody řízení motoru. Závěrečná 9 kapitola obsahuje samotné řešení aplikace, práci s daty a jejich zobrazení operátorovi. Závěr shrnuje výsledek práce a směr možného navazujícího vývoje.

2 HIERARCHIE ARCHITEKTURY SYSTÉMŮ ŘÍZENÍ

V moderních sítích průmyslové automatizace se obvykle používá pětivrstvý model hierarchie komunikace. Popisuje požadované vybavení, architekturu sítě, režimy komunikace mezi zařízeními, povahu toku informací a jeho řízení.

Zobrazení těchto vrstev v grafické podobě nazýváme automatizační pyramida. [1]



Obr. 1 Automatizační pyramida. [1]

2.1 ÚROVEŇ I/O (Field Level)

Hlavním účelem úrovně I/O je propojit řídící proces s řídícím systémem pomocí senzorů a aktuátorů připojených na vstupy a výstupy. Tato úroveň obsahuje měřící přístroje jako jsou průtokoměry, senzory vzdálenosti, bezdotykové spínače. Dále zde najdeme elektromotory, hydraulické a pneumatické pumpy atd. [1] [2]

2.2 Úroveň řízení (Direct control)

Řídící, nebo také PLC vrstva ovládá všechny výrobní procesy. Pokud jsou ale procesy velmi složité, PLC nemusí být dostatečně výkonné a je nahrazeno distribuovaným řídícím systémem (DCS). Zařízení na úrovni řízení zpracovává vstupní signály ze senzorů a spolu s požadovanými parametry jsou generovaný příkazy k akčním členům. Nacházejí se zde PID regulátory nebo PLC zařízení. [1] [2]

2.3 Úroveň monitorování a dohledu (Plant Supervisory)

Na třetí úrovni najdeme systémy dohledu a získávání dat (SCADA), stejně jako prostředí člověk-stroj HMI (Human Machine Interface). V této vrstvě jsou procesní data sledována prostřednictvím uživatelských rozhraní a ukládána do databází.

SCADA (Supervisory Control And Data Acquisition) je v podstatě kombinací předchozích úrovní používaných k přístupu k datům a řídicím systémům z jednoho místa. Obvykle přidává grafické uživatelské rozhraní – HMI, které umožňuje vzdálené ovládání funkcí, výrobních procesů, plánování údržby.

SCADA může monitorovat a ovládat více systémů z jednoho místa. Není omezen na jediný stroj. [2] [3]

2.4 Úroveň plánování

Čtvrtá úroveň automatizační pyramidy se nazývá plánovací úroveň. Tato úroveň využívá počítačový systém pro správu výroby známý jako MES (Manufacturing Execution System). MES sleduje celý výrobní proces v závodě nebo továrně od surovin až po hotový výrobek.

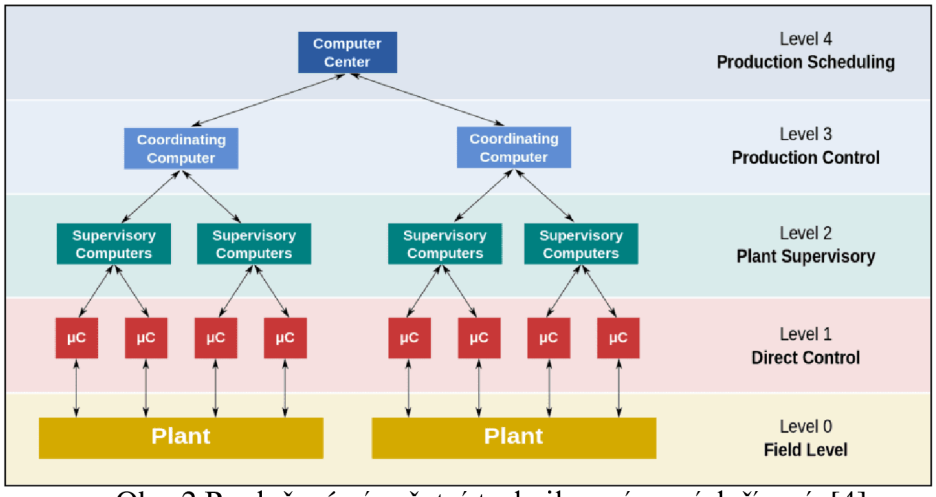
To umožňuje rozhodujícím pracovníkům ve výrobě vidět, co se děje, a umožňuje jim rozhodovat se na základě těchto informací. Mohou upravit objednávky surovin nebo plány dodávek na základě skutečných dat přijatých ze systémů. [2] [3]

2.5 Podniková úroveň

Nejvyšší část pyramidy je takzvaná úroveň řízení. Tato úroveň využívá integrovaný systém řízení společnosti, který je známý jako ERP (Enterprise Resource Planning) nebo plánování podnikových zdrojů.

V tomto systému může nejvyšší vedení společnosti vidět a kontrolovat provoz podniku. ERP je obvykle sada různých počítačových aplikací, které vidí vše, co se děje uvnitř společnosti. K dosažení této úrovně integrace využívá veškerou technologii předchozích úrovní a další software.

To umožňuje podnikům za pomoci jednoho počítače sledovat a řídit všechny úrovně podnikání od výroby, přes prodej, nákup, financování a mzdy a mnoho dalšího. [1] [3]



Obr. 2 Rozložení výpočetní techniky v úrovních řízení. [4]

3 DCS

Jak název napovídá, DCS (Distributed Control System) je systém senzorů, řadičů a přidružených počítačů, které jsou rozmístěny po celém závodě. Každý z těchto prvků slouží k jedinečnému účelu, jako je sběr dat, řízení procesů, ukládání dat a grafické zobrazení. Tyto jednotlivé prvky komunikují s centralizovaným počítačem prostřednictvím místní počítačové sítě závodu – často označované jako řídicí síť.

Oproti centralizovanému systému, ve kterém se nachází jeden řídící prvek (PLC) zajišťující provoz jedné jednotky, je distribuovaný řídící systém sestaven z několika dílčích částí. Spolu s řídícím systémem dohlíží nad složitými výrobními procesy nacházejícími se ve výrobních závodech. Instrukce z DCS jsou rozesílány po celém závodě k jednotlivým kontrolérům. [4] [5]

3.1 Výhody a nevýhody DCS

Výhody:

- Vyšší výkonnostní možnosti
- Možnost tvorby složitých systémů.
- Úspora kabeláže pro přivedení technologických signálů.
- Jednodušší odlaďování nové aplikace (jelikož aplikace jako celek je rozdělena mezi několik PLC a může být laděna po částech).
- Řídicí systém je možné postupně rozšiřovat přidáváním dalších zařízení na komunikační sběrnici.
- V případě poruchy nehavaruje celý systém

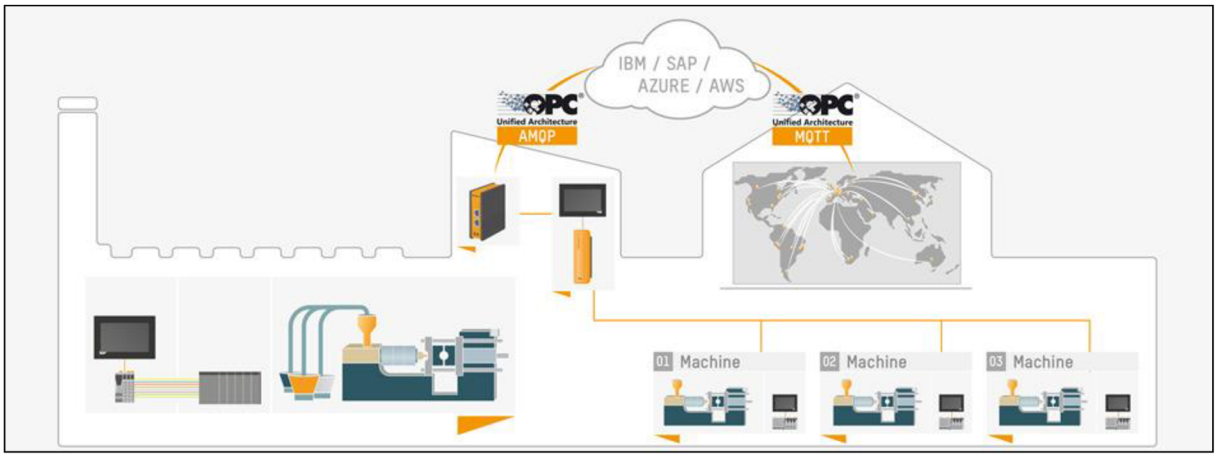
Nevýhody:

- Nepříznivý vliv prostředí – řídicí prvky jsou umísťovány přímo tam, kde je potřeba něco řídit a kde jsou potřebné signály.
- Komunikace mezi zařízeními různých výrobců.
- Nutná vyšší kvalifikace operátorů pro obsluhu systému
- Vyšší cena

4 INTERNET VĚCÍ

Internet věcí (IoT) odkazuje na miliardy fyzických zařízení po celém světě, která jsou nyní připojena k internetu, všechna shromažďují a sdílejí data. Díky příchodu super-levných počítačových čipů a všudypřítomnosti bezdrátových sítí je možné prakticky z čehokoli udělat IoT, od běžných domácích spotřebičů až po sofistikované průmyslové zdroje. Připojení všech těchto objektů a přidání senzorů k nim přidává zařízením úroveň digitální inteligence, což jim umožňuje komunikovat data v reálném čase bez zapojení člověka. Díky IoT se struktura technologií a průmyslu mění, spojuje digitální a fyzický svět. [6] [7]

Nyní máme vzdálený přístup k celozávodním datům, které pak díky systematické analýze můžeme využit ke zlepšení efektivity výroby, snížení spotřeby enrgii, generování předpovědí pro údržbu. Zároveň nám IoT dovoluje připojit továrny a podniky ke cloudovým službám a tím ještě více rozšířit možnosti procesu výroby. [6] [7]

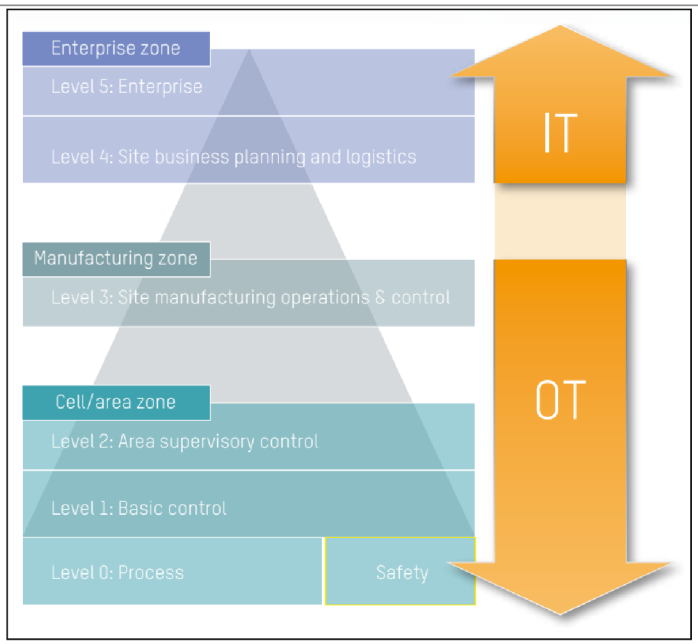


Obr. 3 IoT v průmyslu. [33]

4.1 Edge zařízení

Průmyslové IoT umožnuje získávat informace ze svých strojů a vybavení, které sahají daleko za jednoduchá upozornění a alarmy. Dosud ale studie ukázaly, že průměrná továrna využívá asi jedno procento dat, které generuje. Toto množství lze navýšit skrz použití edge computingu. [8]

Edge computing je metoda pro sběr velkého objemu dat, jejich analýzu a ukládání, což je náročné na datový tok. Vše probíhá blízko zdroje, který data vytváří. Mezi klientem a serverem probíhá jen nejnutnější komunikace. Vyhodnocovaní na serveru pak probíhá podstatně rychleji, protože ten není zatížen zpracováním nepotřebných dat. Edge zařízení funguje jako most mezi systémy reálného času fungujícími na úrovni strojů (OT)a procesů a světem IT. [8]



Obr. 4 Umístění edge computingu mezi IT a OT. [36]

4.1.1 Výhody edgecomputingu

Edge computing využívá výhod cloudových služeb a zároveň za pomoci edge zařízení (aplikačních serverů) maximalizuje výkon sítě potřebný k udržení dobré konektivity. To má za následek snížení doby odezvy na požadavek, zrychlení procesu, aktuálnější výsledky díky analýze dat přímo u strojů.

Decentralizace výpočetních sil zvyšuje nejen efektivitu, ale snižuje i dopady případných útoků. Jakékoli narušení ovlivní chod pouze jednoho bodu v síti. V krizových situacích jsou vyřazena z provozu jen místní zařízení a na zbylých částech infrastruktury se omezení dostupnosti neprojeví. [8]

4.1.2 Nevýhody edgecomputingu

Zabezpečení IoT ještě není zajištěno ve všech směrech a se zvyšující se popularitou roste i počet útoků na infrastruktury tohoto typu. Z jednoho pohledu je tedy edge computing bezpečnější, jelikož nabízí možnost izolovat už napadené body od zbytku infrastruktury, z toho druhého je ale zase jednodušší jej skrze IoT zařízení vůbec napadnout.

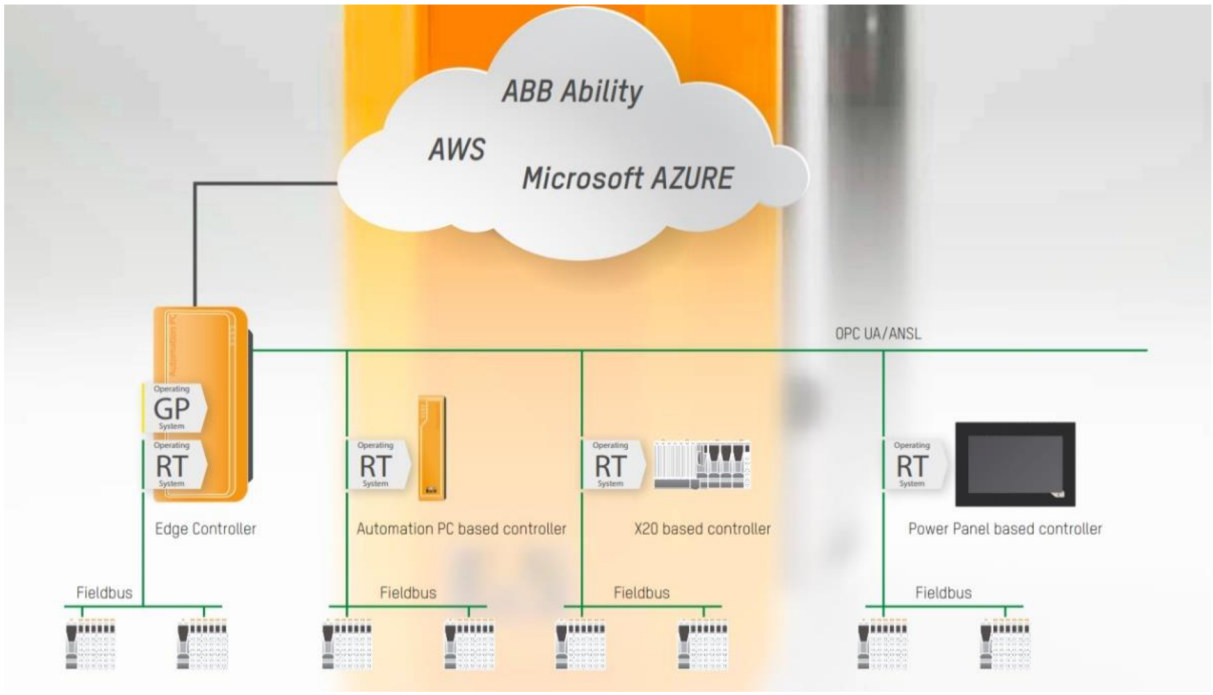
Architektura založena na výpočetní kapacitě na okraji sítě zvyšuje nároky na výkon hardwaru a tím i náklady. Cena lepších hardwarových jednotek ale neustále klesá, takže hardware se postupně stává dostupnějším. [8]

4.2 EdgeController

B&R Edge Controller je založen na řadě průmyslových počítačů – Automation PC, jehož nejvýkonnější řada APC 910 je vybavena procesorem Intel Core i7, schopném plnit i ty nejnáročnější úkoly.

Díky dobrému výkonu je APC schopné provozovat dva nezávislé systémy. Technologie „hypervisor“ umožnuje paralelně provozovat operační systém reálného času (AutomationRuntime – AR) a univerzální operační systém (GeneralPurposeOperatingSystem - GPOS) který je přizpůsoben pro sběr a předzpracování dat. Tento OS je podnikovou distribucí Linuxu, což zaručuje dostatečně robustný systém s dlouhodobou podporou. Zároveň hypervizor umožnuje jednoznačně přiřadit veškerý hardware jednomu OS, tím se zabrání, aby se systémy navzájem rušily.

EdgeController je plnohodnotným průmyslovým zařízením s délkou cyklu v řádu milisekund. Programy pro práci v reálném čase je možné vytvářet v jakémkoliv jazyce definovaném v normě IEC 61131. Také si zachovává vysokou modularitu díky velkému portfoliu modulů B&R schopných propojit se přes Powerlink, Ethernet, nebo OPC UA



Obr. 5 Ukázka použití EdgeControlleru. [36]

5 APROL

APROL (Active PROcess controL system) je systém určený pro řízení procesní výroby od společnosti B&R. Umožnuje propojit veškeré výrobní prostředky v závodě do kompletního hierarchického systému se strukturou umožňující konfiguraci dle potřeb výroby. V systému jsou využívány standardní výrobky značky B&R, jako např. programovatelné automaty, systémy I/O a průmyslové počítače standardu PC, zároveň je však možná komunikace se zařízeními třetích stran.

Na vrcholu pyramidy se nachází řídící systém, sledující veškerá data. Tato data ukládá k dalšímu zpracovaní např. pro generování reportů a grafů atd. Současně je využívá k ovládání automatizovaných.

Základem systému je operační systém Linux. Celý systém lze rozdělit do třech základních celků.

Engeneering server se používá pro návrh a konfiguraci systému řízení procesu. Celá distribuce zajištuje konfiguraci I/O, komunikaci, návrh grafického prostředí, systém sběru dat atd. Dále také zajištuje nahrávání do runtime, operátorských systému a jednotlivých PLC

Na základě dat z *Engeneering* systému tvoří *Rutime* systém hlavní centrum řízení procesu. Tento systém monitoruje, shromažďuje a distribuuje procesní data nakonfigurovaná v inženýrském systému pro další zpracování.

Systém operátora je určen hlavněpro komunikaci stroj-člověk (HMI). Pomocí různých programů umožnuje sledovat výrobní proces na jednom nebo více operátorských stanicích.

Engeneering, Runtime a Operator sytémy mohou být nainstalovány a fungovat samostatně, nebo paralelně na jenom počítači. [9] [10]

5.1 Vývojové prostředí

Hlavním programem v engeneering prostředí je CaeManager. Lze jej použít provedení téměř všech inženýrských a konfiguračních úloh nezbytných k vytvoření řídícího systému. Jedná se o strukturovaný systém umožňující souběžný vývoj v týmu, ale i vývoj offline. Vývoj může probíhat v jakémkoliv jazyce dle normy IEC 1131-3.

5.1.1 Základní informace o CaeManager

Po spuštění CaeManageru a přihlášení se otevře základní pracovní plocha. Tato pracovní oblast se používá ke generování a úpravám objektů nezbytných pro vytvoření aplikace řízení procesů.Patří mezi ně funkční diagramy, procesní grafika, PLC, řídicí počítače atd..

Dále zde můžeme vybrat oblast kde bude probíhat úprava. V rámci projektu to zahrnuje logickou a fyzickou strukturu. Logická struktura je podobná průzkumníkovi se složkami. Programy jsou zde rozděleny, jak už název napovídá, podle logické uspořádání. Ve fyzické struktuře jsou programy rozděleny tak jak se nacházejí hw konfiguraci (Runtime – Tásková třída – task – program). Lze zde i vytvářet uživatelské knihovny, které jsou umístěny pod záložkou *Libraries.*

5.2 Hypervisor

Hypervisor umožňuje čistě rozdělit hardwarové prostředky systému a přiřadit jeden nebo více logických procesorů, paměťových oblastí a periferních zařízení k různým operačním systémům běžícím na cílovém systému. Za běhu je každý operační systém logicky oddělen od ostatních a nemůže být poškozen nebo dokonce ovlivněn provozem druhého operačního systému.

B&R Hypervisor dovoluje paralelně provozovat Automation Runtime s Linuxem na jednom průmyslovém PC. To umožňuje uživateli těžit z chování a podpory hardwarových modulů B&R v reálném čase, které poskytuje Automation Runtime, a také z výhod Linuxu. Jedním z příkladů je správa velkého množství dat například pomocí databáze, trendování dat, nebo grafické prostředí pro uživatele, které AR normálně neposkytuje. Bez hypervizoru by k tomu musel uživatel používat různé kusy hardwaru.

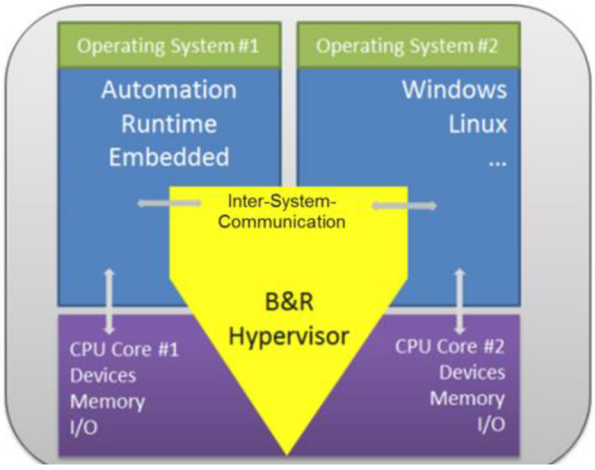
Pro zachování deterministického a real - time chování pro ARemb, má B&R Hypervisor režim zvaný „Privileged Mode“. V privilegovaném režimu si operační systémy zachovají plný přístup k hardwaru a využívají paravirtualizační rozhraní poskytované hypervisorem. To umožnuje ARemb běžet nativní rychlostí bez jakýkoliv latencí přidaných hypervisorem

Aby bylo možné používat tuto funkcionalitu je potřeba implementovat aplikační program pod původně nainstalovaný operační systém – hypervisor „Typ 1“

Obr. 6 B&R Hypervisor. [34]

Hypervizor poskytuje virtuální síťové připojení, které aplikacím umožňuje výměnu dat mezi operačními systémy. Stejně jako u běžného Ethernetového rozhraní se tak děje pomocí standardních síťových protokolů. Místo kabelu je vyhrazená paměťová oblast, která není přiřazena ani jednomu z operačních systémů.

Obrázek ukazuje příklad toho, jak B&R Hypervisor zahrnuje více operačních systémů do svého běhového prostředí. Hypervizor se stará o spouštění a oddělení dvou operačních systémů (AR a Windows nebo AR a Linux) současně. Provoz dvou GPOS (např. Linux a Windows) není možný. Hypervizor také poskytuje definované kanály pro mezi systémovou komunikaci (virtuální rozhraní Ethernet, sdílená paměť, události).



Obr. 7 Schéma komunikace mezi systémy B&R Hypervisor. [35]

Tab. 1 HW a SW požadavky pro B&R Hypervisor.

Minimální hw požadavky	SW požadavky
x86 architektura s BIOS	BIOS s Legacy boot (min. ver.: a1.12)
Intel Virtualization Technology for Direct I/O	GPOS: APROL min. R4.2.-05
Min. jedno logické CPU pro každý operační systém	RTOS:AREmbedded s AutomationRuntime min B4.45
Min. 2GB RAM, doporučeno 4 GB	
B&R Hypervisor licence – technology guard	

Postup instalace Hypervisoru:

1. Instalace GPOS[1] - APROL
   a. Při instalaci ponechat volnou partition (min. 2 GB)
   b. Naistalovat hypervisor drivery (AutoYast DVD)
2. Nastavení BIOS a Boot
   a. Boot type – Only Legacy
   b. Povolení Intel Virtualization technology
   c. Povolení real time prostředí – OEM Feature
3. Vytvoření Instalačního USB
   a. AS hypervisor projekt – AR Embeded
   b. Povolení hypervisoru v HW konfiguraci APC
   c. Přidelení Eth a USB portu mezi RTOS a GPOS. (pozn.: aspoň 1 eth a jeden usb pro AR)
   d. Nastavení Powerlink karty pro připojení vzdálených IO
   e. Vytvoření „USB hypervisor bootable stick“
4. Instalace hypervisoru s RTOS[2] – AutomationRuntime
   a. V Bootmanageru vybereme instalační USB
   b. Po spuštění instalce se provede kontrola volného diskového oddílů
   c. Během instalace proběhne několik restartů oddělující důležité operace
      - Po úspěšném rozdělení disků na oddíly(partition) se hardware restartuje a Hypervisor je nainstalovaný.
      - Během druhého restartu je nainstalován AutomationRuntime. Instalace je dokončena během následujícího restartu
      - Po třetím je nainstalován projekt.
      - Během čtvrtého restartu je nainstalován bootloader GRUB2.
      - V posledním restartu se kompletují instalace AR, Hypervisoru a projekt.

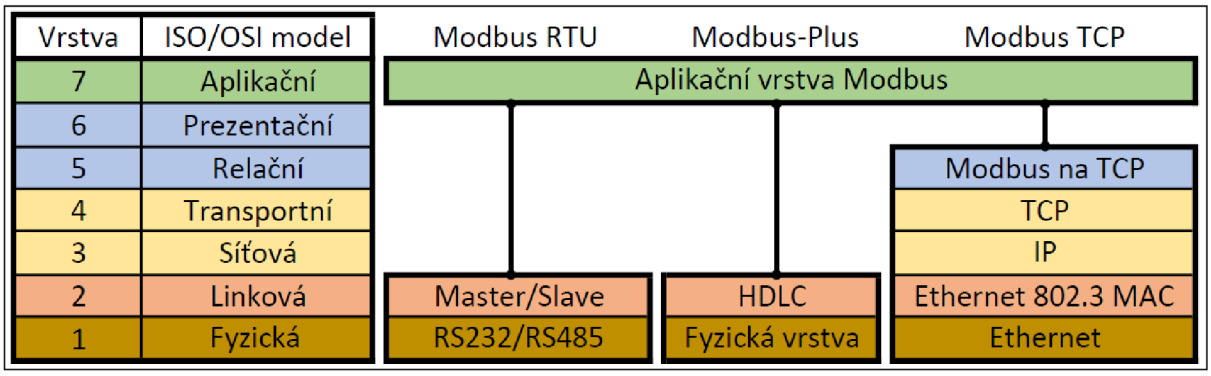
[1] GPOS – general-purpose operating system (Operační systém všeobecného použití)

[2] RTOS – real-time operating systém (Operační systém reálného času)

6 PROTOKOLY PRŮMYSLOVÉ KOMUNIKACE

6.1 Modbus

Protokol byl poprvé představen v roce 1979 firmou Medicon. Byl to jeden z prvních komunikačních protokolů pro výměnu dat mezi zařízeními v distribuovaných systémech a stal se velice rozšířeným protokolem.

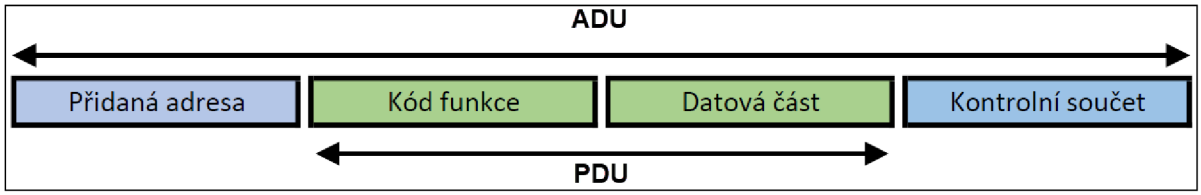


Obr. 8 Porovnání ISO/OSI modelu a Modbus.

Modbus je komunikační protokol nezávislý na fyzické vrstvě s definovaným formátem síťového prostředí. Funguje na třetí až páté vrstvě ISO/OSI modelu, umožňuje komunikaci na principu klient – server. Komunikace probíhá metodou žádost / odpověď (request / reply) a poskytuje přenos služeb specifikovaných tzv. kódem funkce (function code), který je součástí požadavku. Díky transparentnosti, otevřené architektuře, jednoduchosti a snadné implementaci umožnuje přenos po moha typech sítí, jako je sériová linka RS-232 a RS-485, optické a radiové sítě. S příchodem Ethernetu a protokolu IP byl transformován do relační vrstvy pod označením Modbus/TCP.

6.1.1 Aplikační protokol

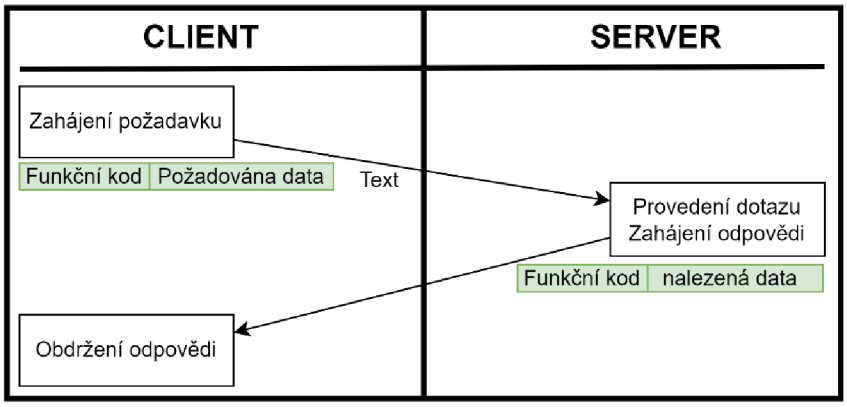
Na úrovni aplikační vrstvy je struktura protokolu Modbus definovaná pomocí PDU (Protocol Data Unit), je tedy nezávislá na typu komunikace. Naopak v závislosti na typu sítě a sběrnice zapouzdřuje rámec ADU (Application Data Unit).



Obr. 9 Formát aplikačního protokolu Modbus.

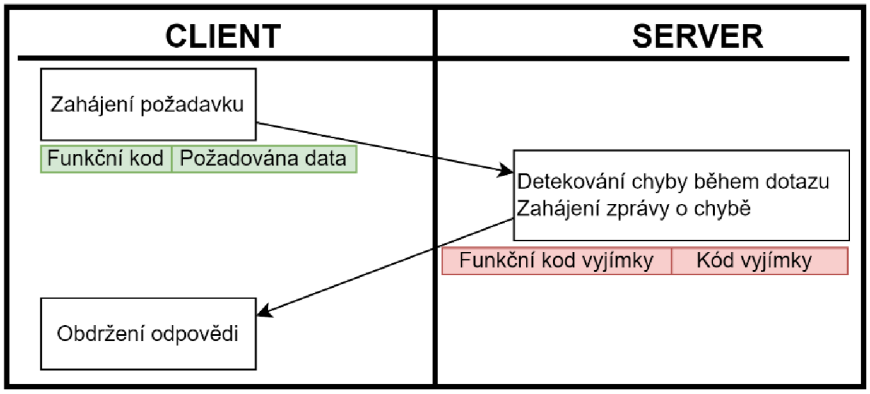
ADU je vytvořeno klientem, který inicializuje datovou komunikaci. Funkce indikuje serveru požadavek, jaký druh akce má provést.

Délka kódu funkce je jeden bajt. Platné kódy jsou v rozsahu 1-255. Když je zpráva poslána z klienta na server, funkční kód řekne serveru, jaký typ akce má provést. Obsahem mohou být položky jako adresa registru, počet zpracovaných položek, počet skutečných datových bajtů v poli. Délka celého ADU je limitována maximální délkou dat pro přenos na fyzické síti. Například 256 bajtů pro komunikaci přes RS-485. [11]Komunikace probíhá na základě žádosti (request) od Klienta na Server, při které se v PDU přenáší již zmíněný kód funkce a potřebná data. Pokud při provádění požadované operace nedojde k chybě, server po provedené akci odpoví zprávou obsahující kód funkce (požadavku) s požadovanými daty. [12]



Obr. 10 Modbus komunikace, bez chyby.

V případě chyby, je serverem vrácen kód funkce, který je ekvivalentem původnímu kódu požadavku, ale jeho nejvyšší bit je nastaven na logickou „1“ indikující neúspěch (exception response). V datové části je vrácen chybový kód (exception code) upřesňující důvod neúspěchu. [12] [12]



Obr. 11 Modbus komunikace, s chybou.

Tab. 2 Definice funkčních kódů Modbus. [12]

Typ funkce			**Název funkce**	**Kód funkce**
Přístup k datům	Bitový přístup	Fyzické diskrétní vstupy	Čti diskrétní vstupy	2
		Interní bity, nebo fyzické cívky	čti cívky	1
			Zapiš cívku	5
			Zapiš cívky	15
	16-bit access	Fyzické vstupní registry	Čti vstupní registr	4
		Interní registry, fyzické výstupní registry	Čti uchovávající registry	3
			Zapiš uchovávající registr	6
			Zapiš uchovávající registry	16
			Čtí / zapiš registry	23
			Registr masky zápisu	22
			Čti FIFO frontu	24
	Přístup k záznamům v souborech		Čti záznam ze souboru	20
			Zapiš záznam do souboru	21
Diagnostika			Read Exception Status	7
			Diagnostika	8
			Čti čítač komunikačních událostí	11
			Čti záznam komnunikačních událostí	12
			Sděl identifikaci klieta	17
			Čti identifikaci zařízení	43
Ostatní			Zapouzdřený přenos	43

6.1.2 Datový model

Datový model protokolu Modbus je založen na sadě tabulek, s charakteristickým významem. Definovány jsou čtyři základní tabulky:

Datový typ	Velikost	Přístup	Adresa	Komentář
Coils	1 - bit	čtení / zápis	00001 - 09999	Může být upraveno Mastrem
Input Registers	1 - bit	čtení	10000 - 19999	poskytnuta IO zařízením (Slave)
Input Discrete	16 - bits	čtení	30000 - 39999	poskytnuta IO zařízením (Slave)
Holding Registers	16 - bits	čtení / zápis	40000 - 49999	Může být upraveno Mastrem

Obr. 12 Datový model Modbus.

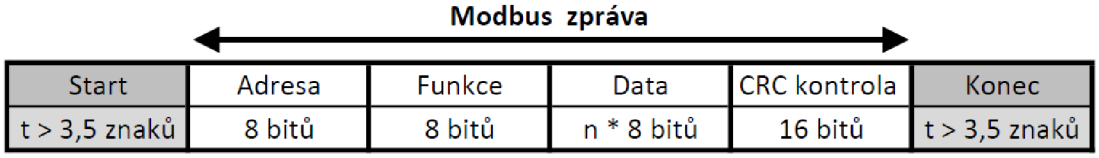
Všechna data zpracovaná přes Modbus (bity, registry) musí být umístěna v paměťové aplikaci zařízení. Na fyzické adrese paměti nezáleží, protože se pracuje s relativními odkazy. Jediným požadavkem je propojení datového odkazu s fyzickou adresou.

6.1.3 Přenosové režimy

Komunikace po sériové lince vždy probíhá metodou Master/Slave. Na lince muže být připojen pouze jeden Master a až 247 Slave. Komunikaci vždy zahajuje master, slave jen odpovídá. Zároveň je možná komunikace peer-to-peer mezi slave, ale bez účasti master.
Master posílá požadavky slave jednotkám ve dvou režimech:

- *unicast režim* – master adresuje požadavek jedné konkrétní slave jednotce a ta pošle odpověď
- *broadcast režim* – master posílá požadavek všem jednotkám, žádná jednotka neodpoví.

V *režimu RTU* (Remote Terminal Unit) obsahuje každý 8-bitový byte zprávy dva 4-bitové hexadecimální znaky. Vysílání zprávy musí být souvislé, mezery mezi znaky nesmějí být delší než 1.5 znaku. Začátek a konec zprávy je identifikován podle pomlky na sběrnici delší než 3.5 znaku.



Obr. 13 Formát Modbus zprávy v režimu RTU.

Pokud je komunikace po Modbus síti nastavena na *režim ASCII*, každý 8-bitový byte zprávy je poslán jako dva ASCII znaky. Hlavní výhodou tohoto režimu je, že umožňuje, aby se mezi znaky objevovaly časové intervaly až jedné sekundy, aniž by došlo k chybě.

Zpráva začíná dvojtečkou „:“ (ASCII - 3A hex) a je ukončena dvojicí „CRLF“ (ASCII - 0D, 0A hex). Síťová zařízení nepřetržitě monitorují síťovou sběrnici, zda neobsahují dvojtečku. Když je přijato, každé zařízení dekóduje další pole (pole adresy), aby zjistilo, zda se jedná o adresované zařízení. Mezi znaky ve zprávě mohou uplynout intervaly až jedné sekundy. Pokud dojde k delšímu intervalu, přijímající zařízení předpokládá, že došlo k chybě. [13]

Start	Adresa	Funkce	Data	LRC kontrola	Konec
:	2 znaky	2 zanky	N * znaků	2 znaky	CR LF

Obr. 14 Formát Modbus zprávy v režimu ASCII.

6.2 Industrial Ethernet

Během posledních 20 let vzniklo mnoho průmyslových sběrnic, které byly vyvinuty pro automatizaci zejména v průmyslu a řízení procesů výroby. Přesto stále existují různá omezení, která brání jejich výkonu.

Vznikají stále větší požadavky na spolehlivý komunikační systém, který by nabízel vysokou flexibilitu a všeobecnou kompatibilitu. Zároveň se očekává podpora vylepšování systému a zavádění nových aktualizaci. Tyto vlastnosti splňoval Ethernet, který byl osvědčenou technologií, bez patentů a velice standardizován. Kromě toho měl velký potenciál sloužit jako konzistentní a integrované komunikační řešení, tj. umožňoval vzájemné propojení úrovní řízení, procesu a výroby. [14]

Standardní Ethernet v kombinaci s internetovým protokolem, jako je TCP / IP, není schopný přenosu dat v reálném čase. Příčinou je metoda náhodného přístupu CSMA / CD (Vícenásobný přístup / detekce kolizí), která sice zabezpečí že data budou přenesena, ale z hlediska determinismu je závažným handicapem. [15]

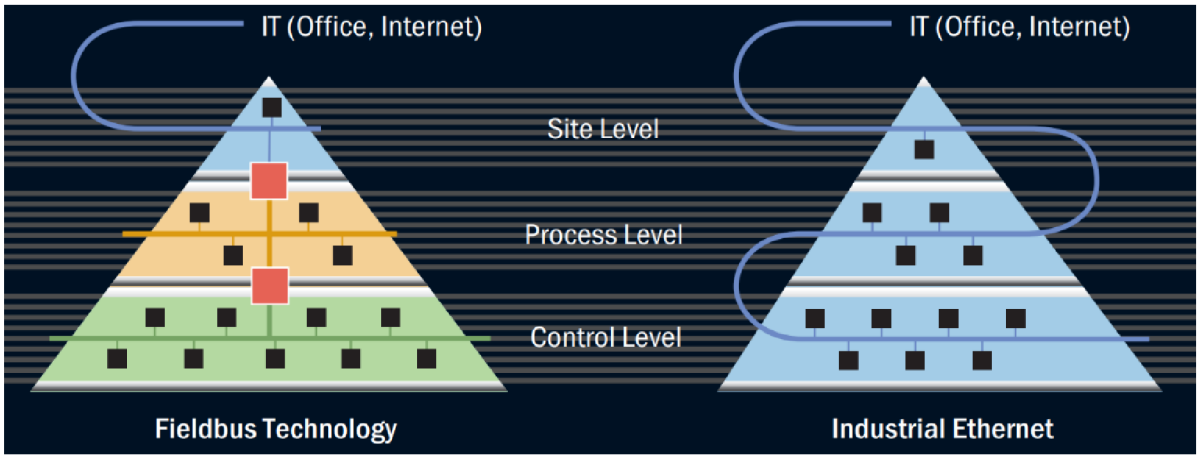
Většina řešení komunikace v síti Ethernet je založena na TCP/IP jako nadřazeném protokolu. Data jsou zabalena a následně rozbalena v přijímacím zařízení. Obě procedury realizuje zásobník TCP/IP (TCP/IP stack). Díky tomuto principu proběhne každý datový rámec zásobníkem dvakrát, což má za následek prodloužení komunikace v řádu stovek µs.

Další zvláštností protokolu TCP/IP je komunikace na pozadí, nezávisle na uživateli. Kdy TCP/IP kontroluje stav zařízení v síti. Tato komunikace může opět vést ke kolizím. [16]

Možné způsoby řešení problémů TCP/IP při použití v průmyslu:

- prioritizace paketů s daty podle IEEE 802.1Q/802.1p
- spolehnutí se na neexistenci (malou pravděpodobnost vzniku) kolizí
- segmentace pomocí přepínačů do tzv. kolizních domén
- použití principu master-slave,
- Slot Communication Network Management.

Řešení eliminující zpoždění a vytvoření komunikace na bázi Ethernetu v reálnem času označujeme jako technologie průmyslového ethernetu (Industrial Ethernet) [16]



Obr. 15 Průmyslový vs standardní ethernet. [14]

6.3 Ethernet Powerlink

První verze standardu Ethernet Powerlink (dále také EPL) byla uvedena na trh v závěru roku 2001 rakouskou firmou Bernecker & Rainer Industrie Elektronik GmbH.

Cílem bylo vybudovat na platformě Fast Ethernet velmi rychlou sběrnici s deterministickými odezvami a s minimálním rozptylem časování telegramů. Kromě rychlé cyklické výměny dat musela přitom existovat možnost acyklické komunikace, která však cyklickou nesměla nijak ovlivnit. Další podmínkou bylo umožnit synchronizaci vstupních/výstupních dat a parametrů pohonů navzájem mezi sebou i s řídicím systémem.

O dva roky později vyšla druhá verze Powerlink V2, která rozšiřuje V1 o aplikační vrstvu podobě standardizovaného aplikačního rozhraní založeného na mechanismech definovaných ve standardu komunikačního protokolu CANopen. Vzhledem k tom že systém vychází ze standardu Ethernet, nevyžaduje žádný speciální hardware. Lze proto využít všechny čipy pro Ethernet. Vlastností pro použití v reálném čase dosahuje čistě softwarovým řešením. [15]

6.3.1 Základní principy fungovaní systému

Z mechanismů umožňujících vytvořit z Ethernetu TCP/IP sériový deterministický komunikační systém, EPL využívá především důsledné časové rozdělení přenosového cyklu na část izochronní (cyklický přenos časově kritických dat v reálném čase) a asynchronní (přenos časově nekritických dat protokolem IP). Jako další mechanismus směřující k determinismu používá EPL modifikovaný způsob tvorby bezkolizních domén při použití rozbočovačů (*hub*) s opakováním zpráv namísto jinak standardně používaných přepínačů (*switch*), které by zanášely do izochronní rychlé části cyklu nežádoucí zpoždění, a dále důsledné oddělení segmentů sítě pracujících v reálném čase (*Real-Time* – RT) od těch, které práci v reálném čase nepodporují (*Non Real-Time* – NRT). [15]

Standard Ethernet Powerlink využívá komunikační model producent/konzument, což opět vede k většímu výkonu a propustnosti sítě. Pro deterministickou synchronizaci izochronních přenosů dále zavádí mechanismy synchronizace prostřednictvím distribuovaných hodin reálného času podle standardu IEEE 1588.

Díky základu ve standardu Ethernet umožnuje Powerlink libovolnou topologii, kterou lze kdykoliv upravovat. Spolu s možností přímé křížové komunikace bez použití Mastera dodává dostatečnou flexibilitu pro použití v decentralizovaných systémech.

6.3.2 Protokoly dolních vrstev

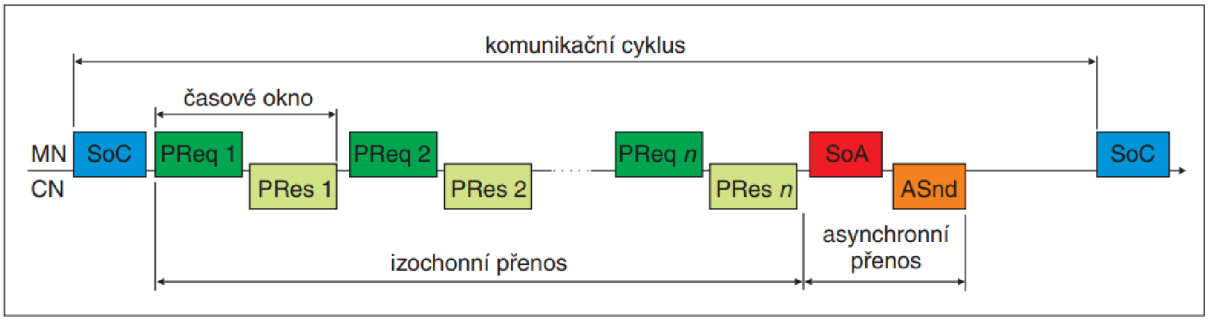
První i druhá vrstva komunikačního modelu EPL jsou plně kompatibilní s odpovídajícími vrstvami průmyslového provedení Ethernetu. Co se týče spojové vrstvy, každé zařízení v síti EPL podporuje dva operační módy: základní mód Ethernet TCP/IP (tzv. mód *Ethernet*) a mód *Powerlink*.

Základní mód každého EPL zařízení umožnuje zavést základní software a konfigurační soubory potřebné k režimu reálného času přímo ze sítě Ethernet. Každé zařízení v základním módu může být připojeno do libovolné sítě Ethernet, bez ohledu zda-li daná sít pracuje v režimu reálného času nebo ne a chová se jako zařízení nepodporující tento režim.

Režim Powerlink umožnuje stanici v síti pracovat v reálném čase, tj. komunikovat s deterministickým přidělením časových oken (time slot).

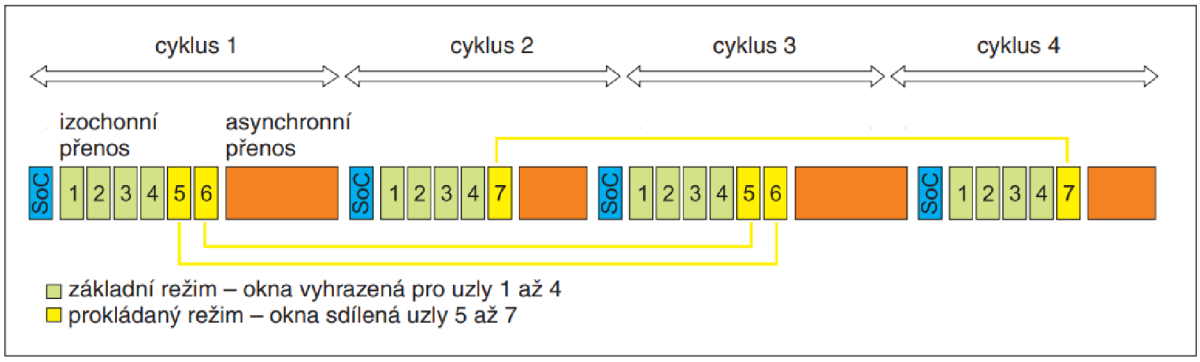
V tomto chodu je použita metoda řízení master-slave známá z jiných komunikačních sítí. Je určen jeden arbitr (Managing Node – MN), který synchronizuje všechny další účastníky (Controlled Nodes – CN) a přiřazuje jim časová okna v přenosovém cyklu.

Cyklus začíná startovacím příznakem (*Start of Cycle* – SoC) zasílaným mastrem sítě všem řízeným uzlům. Na základě SoC si všechny řízené uzly synchronizují své hodiny reálného času, aby správně přijímaly a vysílaly data i inicializovaly jejich zpracování. Následuje fáze izochronního přenosu časově kritických dat. Arbitr sítě posílá jednotlivé žádosti o přenos každému řízenému uzlu (*Poll Request* – PReq) podle časového plánu v daných naplánovaných časových oknech. Vyzvaný řízený uzel vyšle v přesně definovaném časovém úseku bezprostředně po výzvě od řídicího uzlu MN do sítě data, která mohou všechny řízené uzly okamžitě přijímat (*Poll Response* – PRes).



Obr. 16 Schéma komunikace Powerlink.

Po izochronní fázi, rozhodující pro činnost sítě EPL jako komunikačního prostředku s vlastnostmi reálného času, následuje v každém cyklu asynchronní fáze, ve které arbitr (MN) umožní jednotlivým stanicím poslat data, jejichž přenos není časově kritický. Asynchronní fázi opět spouští arbitr sítě (*Start of Asynchronous* – SoA). V asynchronní fázi se typicky přenášejí parametry zařízení, diagnostická data apod. Protokoly typu IP se realizují přímo bez tunelování a konverze dat. Aby se co nejlépe využilo přenosové pásmo, lze v módu Powerlink realizovat také tzv. prokládaný režim. V něm jsou údaje nevyžadující přenos dat v každém cyklu přenášeny v rámci sdílených časových oken, která nejsou vyhrazena pro jeden konkrétní řízený uzel, ale jsou sdílena několika uzly. Tyto uzly odesílají svá data postupně v rámci sdílených časových oken. [15]



Obr. 17 Ukázka prokládaného režimu pro Powerlink.

Základní vlastnosti Powerlink [14]:

- Maximální délka vedení - 2 km
- Délka segmentu mezi uzly - max. 100 m
- Až 240 uzlů v jedné doméně
- Možnost nastavení unikátních adres
- Šířka pásma – 100Mbit/s
- Nejistota synchronizace (jitter) - 0.1 μs
- Nejkratší doba cyklu - 100 μs
- vysoká propustnost dat i pro krátké cykly
- Přímá (peer-to-peer) komunikace všech uzlů navzájem
- možnost připojovat i odpojovat zařízení v síti za chodu (hot-plugging) bez negativního ovlivnění systému

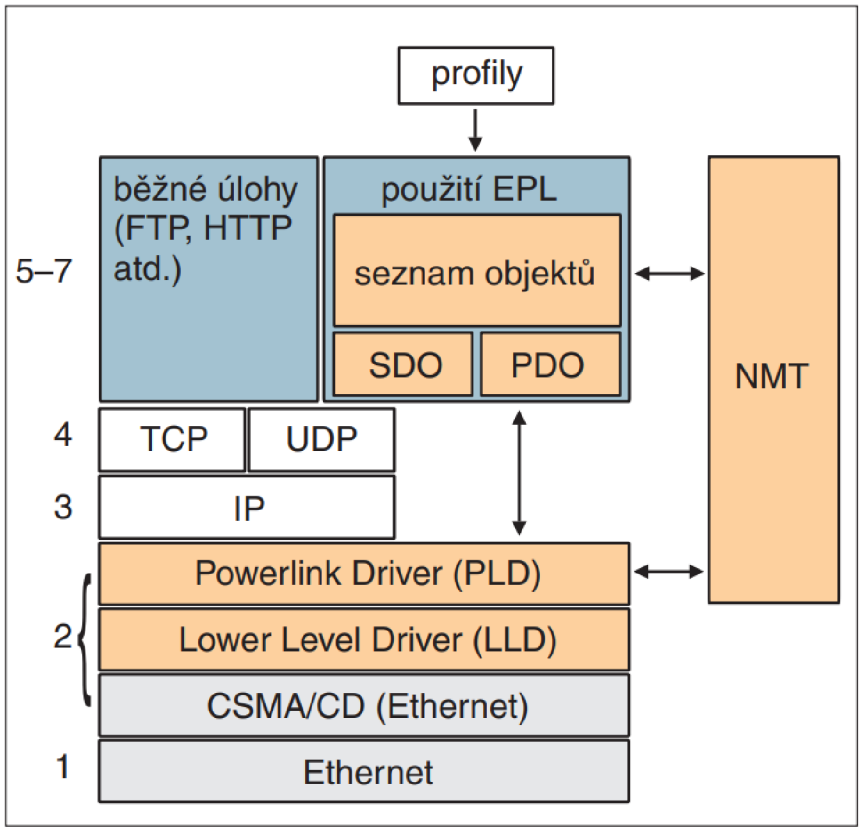
6.4 CANopen

CANopen je jedním z nejrozšířenějších aplikačních protokolů. Jednou z hlavních výhod tohoto protokolu jsou standardizované popisné soubory, které zpřístupňují informace o stavu a vlastnostech zařízení, konfigurační parametry a další relevantní data. Proto je aplikační vrstva Powelinku definovaná jako nosič všech CANopen mechanismů.

CANopen a Powerlink používají stejné slovníky objektů, popisné soubory zařízení i stejné komunikační mechanismy. Proto mohou být všechny CANopen aplikace a profily bez problému implementovány do prostředí Powerlink. Díky tomu můžeme Powerlink označit také jako CANopen over Ethernet. [17]

CANopen systémy, profily a objekty použité v Powerlink:

- *systém řízení sítě* (*Network Management* – NMT), což je mechanismus řízení a sledování konzistence sítě v etapách jejího zavádění (*boot-up*) i provozu (*run time*),
- *seznam objektů a modely zařízení* (*Object Dictionary*) reprezentující jednotnou metodu pro přístup k datům, parametrům a funkcím konkrétních zařízení nebo typů zařízení komunikujících v síti, [15]
- *signalizaci chyb* jako obecnou metodu pro signalizaci chyb přenosu a indikaci chybových stavů nezávisle na typu nebo na výrobci zařízení,
- *objekty provozních dat* (*Process Data Object* – PDO) jako obecný mechanismus umožňující uživateli specifikovat data vyměňovaná mezi zařízeními od různých výrobců, [15]
- *objekty servisních dat* (*Service Data Object* – SDO), opět obecný mechanismus pro přenos velkého objemu např. konfiguračních dat apod.,
- *profily zařízení*, což jsou standardizované definice dat, parametrů a funkcí určitých typů zařízení, jako jsou pohony, moduly I/O, snímače polohy, programovatelné automaty atd. [15]



Obr. 18 CANopen systémy, profily a objekty.

Časově kritická výměna dat se uskutečňuje prostřednictvím objektů typu PDO přenášených v izochronní fázi přenosového cyklu EPL s využitím komunikačního modelu producent/konzument, který dovoluje přenášet data z jednoho uzlu do většího počtu uzlů současně. Obsahy PDO lze konfigurovat již v etapě uvádění sítě do chodu, což umožňuje optimalizovat výměnu dat v reálném čase, která takto není zatížena žádnou konfigurační režií.

Parametry, funkce a časově nekritická data jsou přenášeny prostřednictvím objektů typu SDO při použití pomalejšího komunikačního modelu klient/server, kde každý jednotlivý účastník může, jak dosáhnout na libovolný uzel v síti, tak z něj být dosažen. Délka přenášených zpráv není omezena a vzhledem k tomu, že přenos SDO probíhá v asynchronní fázi přenosového cyklu EPL a s využitím protokolů UDP/IP v transportní vrstvě, může být zařízení připojené k síti EPL dosaženo pro účely přenosu SDO i z internetového segmentu připojeného přes směrovač k síti EPL. [15]

7 HISTORIZACE

Aby technik, nebo operátor mohl správě vyhodnotit stav měniče, musí mít možnost zobrazit parametry v průběhu času. Proto musíme hodnoty parametrů ukládat do databáze a následně být schopni tyto hodnoty srozumitelně zobrazit přes HMI. EdgeInverter a Aprol využívá dvě databáze Chronolog a MariaDB. Chronolog, založený na Berkeley databázi, je určen pro rychlý zápis, ale má jen omezené možnosti, jak pracovat s daty. Naopak MariaDB dovoluje složité SQL dotazy, které jsou použity při generovaná webových reportů pomocí služby JasperServer.

7.1 Oracle Berkeley DB

Berkeley DB je open source databázová knihovna, která je navržena tak aby umožňovala rychlé, snadné a spolehlivé ukládání dat. Databáze běží ve stejném adresním prostoru jako aplikace. V důsledku toho není pro databázové operace vyžadována žádná mezi-procesová komunikace, ať už přes síť, nebo mezi procesy na stejném stroji. Všechny databázové operace probíhají uvnitř knihovny. Proto může databázi používat současně několik procesů, nebo vícero vláken v jednom procesu.

Berkeley DB není relační databáze, proto nemá představu o schématech ani datových typech. Relační databázové systémy jsou sémanticky bohaté a nabízejí přístup k databázi na vysoké úrovni. Ve srovnání s takovými systémy je Berkeley DB vysoce výkonná knihovna pro ukládání záznamů. Na Berkeley DB je možné vybudovat relační systém. Ve skutečnosti populární relační systém MySQL používá Berkeley DB pro správu tabulek v chráněném režimu a stará se o veškeré parsování a provádění SQL. [18] [19]

Pro přístup k datům používá relativně jednoduché služby:

- Vložit záznam do tabulky.
- Odstranit záznam z tabulky.
- Nalézt záznam v tabulce vyhledáním jeho klíče.
- Aktualizovat záznam, který již byl nalezen.

V rámci Berkeley DB jsou data uložena do tzv. kontejnerů. Kontejnery jsou ve skutečnosti kolekce XML dokumentů a informací o těchto dokumentech. Kontejner plní podobnou funkci jako tabulka v relačních databázích. Kontejnery jsou pojmenované a jsou to soubory, které obsahují řadu informací, jako jsou záznamy, indexy a indexové statistiky, datový slovník a další systémová metadata. Jednotlivé záznamy jsou v kontejnerech uložené jako páry (klíč, hodnota). [20]

7.2 MariaDB

Maria DB je relační databází založená v roce 2009 jako odnož MySQL. Hlavními iniciátory byli původní vývojáři MySQL, kteří se obávali o další směřovaní tohoto softwaru po odkoupení společností Oracle. Současně si ale stále udržuje vysokou kompatibilitu s MySQL, s binární paritou knihovny a s hodnými SQL příkazy a API. [21] [22]

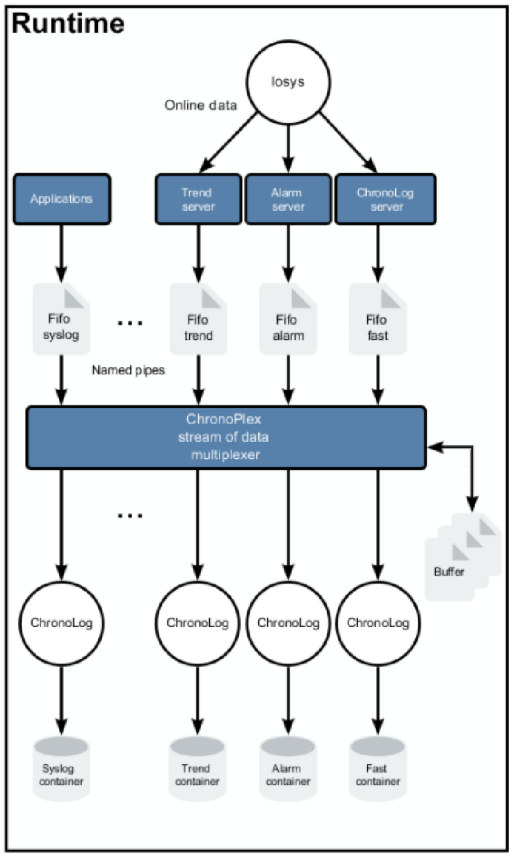
MariaDB je součástí linuxové distribuce OpenSuse, na které je založený i operační systém Aprol. Proto se tato databáze nachází v tomto OS.

Tato databáze se v Aprolu převážně používá pro komplexnější dotazy, které zahrnují provázanost tabulek, filtrování, seřazovaní anebo seskupování podle hodnot.

7.3 ChronoPlex

ChronoPlex je hlavní správce pro zaznamenávaní všech typů dat v Aprolu, který je založený na Berkeley databázi. Čte pojmenované datové toky a spouští podřízené procesy Chronologu, aby je uložil do kontejneru na logovacím serveru.

Samotný server nevyžaduje žádnou konfiguraci projektovým manažerem, automaticky rozpozná všechny instance bloků, ze kterých zaznamenává časové komprimovaná data do vyrovnávací pamětí (FiFo), které slouží pro přenos k následnému zpracování (Chronoplex).

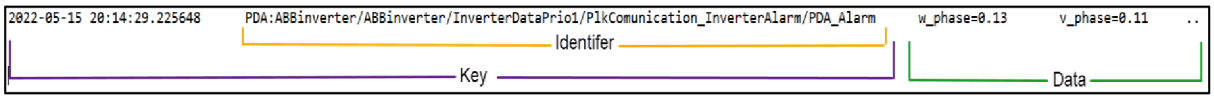


Obr. 19 Struktura ChronoPlexu. [30]

7.4 Chronolog

Program „ChronoLog“ slouží jako podřízený proces „ChronoPlex“ ke sběru dat pro všechny části systému APROL.
Obecné informace o formátování souborů:

- Formát souborů kontejnerů ChronoLog je definován databázovou knihovnou Berkeley. Kontejnery lze zapisovat a číst pouze aplikací ChronoLog.
- ChronoLog vždy ukládá kompletní datovou sadu skládající se z klíče a seznamu párů „název=hodnota“:
  <Key> <Field name>=<Value> [<Field name>=<Value> …]
  Klíč obsahuje datum, čas a identifikátor. Identifikátor je jedno pole složené z předpony, názvu projektu a cesty k instanci v rámci projektu



Obr. 20 Struktura záznamu v ChronoLogu.

```
2022-03-14 15:45:18.705596
PDA:ABBinverter/ABBinverter/InverterDataStatus/Save2Chrono_hm_InsertInto_Chrono_hm_Insert_Stat_InverterDataSt/PDA_default
warning2=0      warning1=0      main_status=0   inverter_stat=2   fault2=0        fault1=0        drive_status3=0
drive_status2=0 drive_status1=0 dio_status=0    di_status=0     beak_status=0   act_mode=0
```

Obr. 21 reálný záznam v ChronoLogu.

7.4.1 Přehled Aprol kontejnerů

Do stejného kontejneru může současně zapisovat více procesů chronologu, ale proces Chronologu může zapisovat jen do jednoho kontejneru.

Tab. 3 Dostupné kontejnery v APROLu.

Název	Popis
alarm.clc	APROL alarm system records
audittrail.clc	Logging of **AuditTrail** (actions made by operators)
changecontrol.clc	Logging of **ChangeControl** (changes in the CAE)
ctrl-logbook.clc	
fast.clc	
opcua.clc	Recording of changes via OPC UA
parameter.clc	Logging in the scope of parameter management (process parameter up/download)
pda.clc	Process Data Acquisition, quick
pda-slowrate-customer.clc	Process Data Acquisition, slow (customer)
pda-slowrate-system.clc	Process Data Acquisition, slow (system)
sfclog.clc	Logging of the course of action of the SFC and the control of SFCs
shiftlog.clc	Recording of the APROL shift logbook
slow.clc	
sysconfig.clc	Recording of APROL system configurations that have been made
syslog.clc	APROL system messages
syslog-fastrate.clc	APROL system messages, very quick (no forwarding)
systemstate.clc	Recording of the APROL start management (change of system status, change of APROL runlevel, start/stop of applications)
targetcontrol.clc	Logging of executed downloads to controllers or control computers
triggered.clc	User-defined logging via block ChronoLog

7.5 ChronoTrend

Záznam trendů také využívá službu ChronoLog. Z důvodů výkonu se implementace liší od zbytku systému ChronoLog.

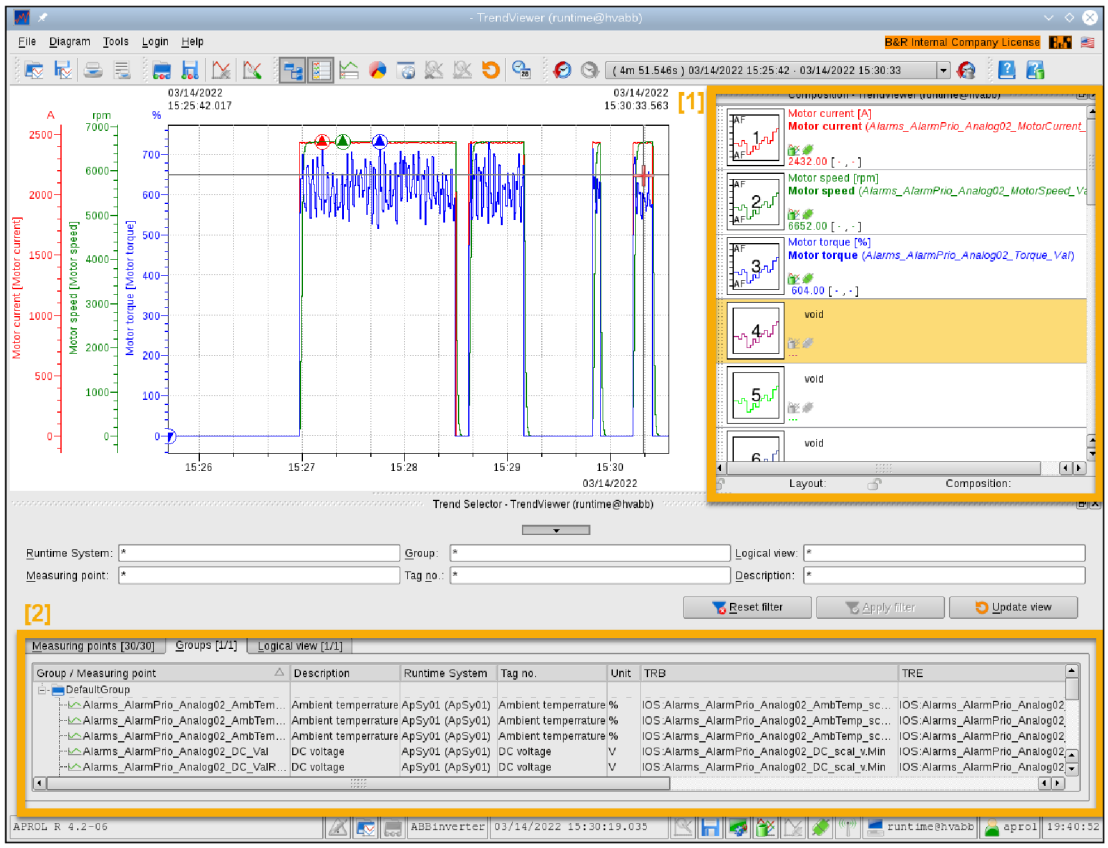
Záznam trendu je vázán na runtime systémy APROL a je prováděn lokálně */home/aprolsys/APROL_DATA/chronotrend/<název projektu>/<účet CC>/TRD*. Není zde kontejner s příponou souboru ".clc", ale mnoho souborů obsahujících názvy všech trendových proměnných nakonfigurovaných v projektu. Údaje se zaznamenávají zvlášť pro každou procesní proměnou (Process Variable - PV) do „jeho“ souboru. Také vnitřní formát těchto souborů neodpovídá obvyklým kontejnerům. Soubory trendů se skládají pouze z chronologicky seřazených hodnotových párů časové značky a okamžité hodnoty procesní proměnné. TrendServer ukládá takový záznam pro každou změnu hodnoty.

Při vyhodnocování dat lze na celý adresář nahlížet jako na kontejner, ke kterému přistupují aplikace jako ChronoTrend, nebo TrendViewer.

7.6 TrendViewer

TrendViewer je analytický nástroj, který se používá ke grafickému znázornění procesů. Časový průběh procesních dat se zobrazuje ve formě křivek a diagramů.

Program umožnuje spravovat až 100 000 trendu v 5000 skupinách v různých kompozicích. Kvůli přehlednosti je omezen počet trendů v jednom okně na 20.



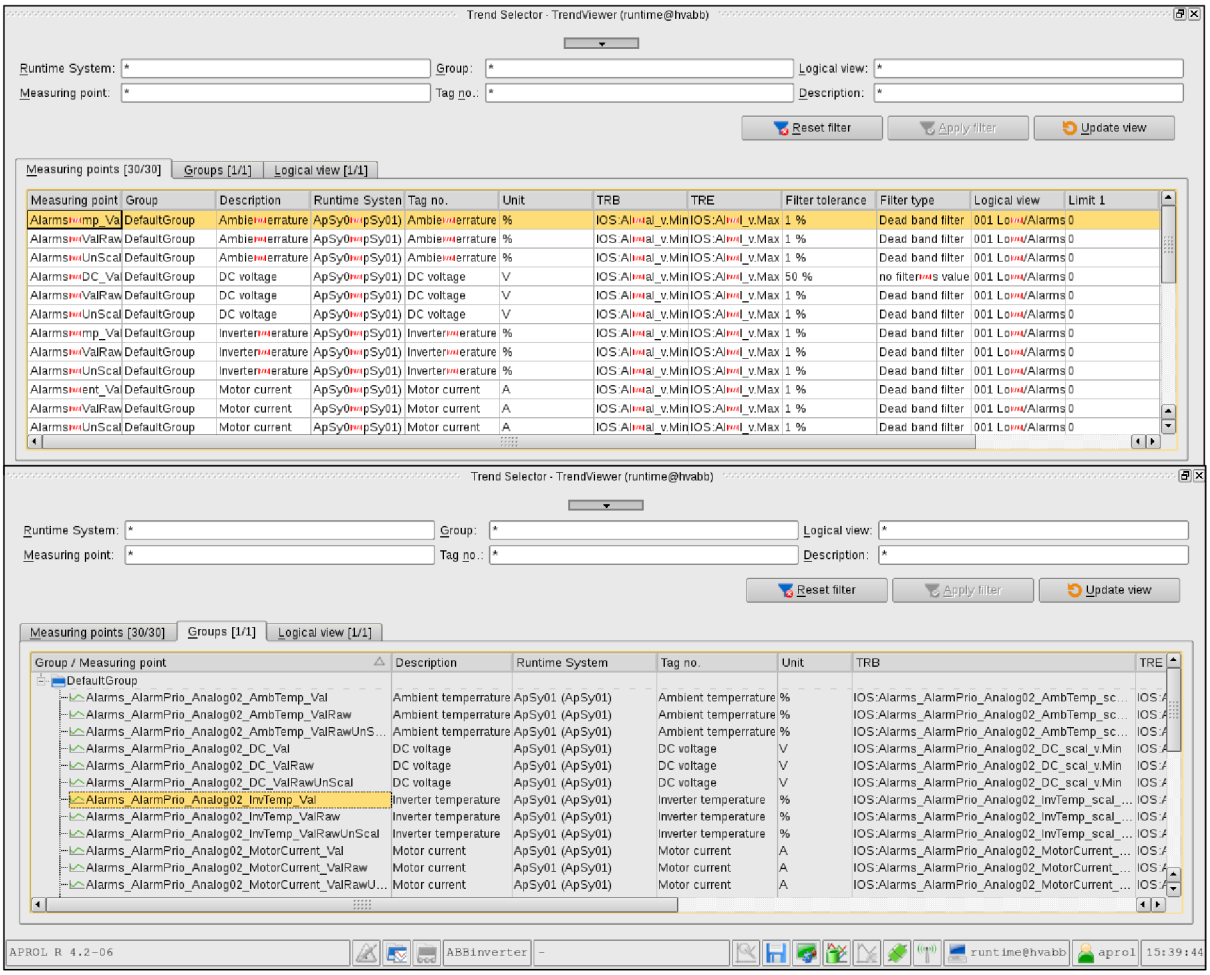
Obr. 22 TrendViewer uživatelské prostředí 1- Composition, 2-Trend selection.

7.6.1 Obecný popis aplikace

Kromě hlavního okna s grafy, můžeme zobrazit i „Trend Selector“ a „Composer“ pro konfiguraci složení trendů a vytvoření požadovaného rozvržení.

7.6.2 Trend Selection

V okně „Trend Selection “ jsou uvedeny všechny trendy, které byly nastaveny v aktivních CFC (Continuous Function Chart) během polední kompilace Ty jsou seřazeny podle měřicích bodů (název procesních proměnných na bloku trendů v tabulkové formě), skupin a logického pohledu. V záložkách *Skupiny* a *Logický pohled* lze vybranou větev nebo celý adresářový strom otevřít a zavřít pomocí místní nabídky. Stromová struktura v záložce *Logický pohled* odpovídá adresářové struktuře projektu CAE. Toto pohodlné zobrazení umožňuje rychlou navigaci k požadovaným trendům za předpokladu, že projekt CAE má vhodnou strukturu.



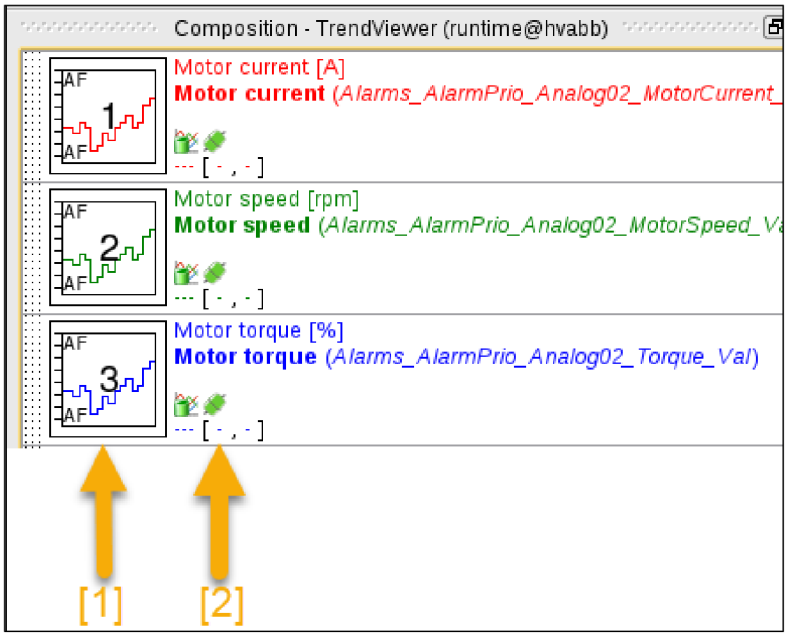
Obr. 23 TrendSelection a) Všechny meřící body b) Skupiny.

7.6.3 Composition

Composition (kompozice) se používá k individuální konfiguraci trendů pro zobrazení v diagramu. Umožnuje definovat různá rozložení a přiřadit je k trendům samostatně. Aby nebyla narušena jasná reprezentace v diagramu, je maximální počet trendů/rozvržení omezen na 20.

Pro přidání požadovaných trendů z TrendSelection (výběrů trendu) do slotů můžeme použít tzv.: Drag & Drop

Každý slot v kompozici se skládá ze dvou polí: pole rozložení a pole dat. *Pole rozložení* představuje vizuální konfiguraci trendu (barva čáry, šířka čáry atd.). *Datové pole* zobrazuje popis, číslo tagu, měřicí bod a jednotku přiřazeného trendu. Barva přiřazená trendu se použije pro text datového pole. Popisek vysvětluje text v datovém poli.



Obr. 24 Kompozice v TrendViewer 1. Pole rozložení 2. Datové pole.

7.6.4 Další funkce TrendVieweru

- Pravítko – chceme-li vyhodnotit související data, máme možnost přidat do diagramu až dvě pravítka měření. Tato funkce také umožňuje analyzovat označené body v čase ve statistikách a integrovat je do doplňkových informací.
- Časová navigace – uživatelsky přívětivá časová navigace aplikace TrendViewer umožňuje rychle a pohodlně vybrat požadované časové rozsahy pro zobrazení v diagramu. K dispozici jsou předdefinované časové rozsahy a možnosti individuálního výběru, stejně jako intuitivní ovládání pomocí myši a klávesnice.
- Zoom a Combi -zoom - zoom umožňuje rychle a pohodlně analyzovat detaily zobrazeného trendového obrázku a poté se vrátit k původnímu zobrazení. Pro použití výběru rozsahu lze jednoduše nakreslit myší v diagramu.
- Online mode – v online režimu je možné průběžně aktualizované zobrazení trendů za běhu. TrendViewer je přímo napojen na systém, a proto jsou hodnoty okamžitě aktualizovány, zároveň se plynule posouvá i časový rozsah.
- Komentáře – operátor zároveň může zaznamenat komentář ke konkrétní hodnotě a tím předat kontext hodnoty ostatním operátorům. Komentáře jsou trvale uloženy v kontejneru ChronoLogu „slow.clc“.

8 MĚNIČ FREKVENCE

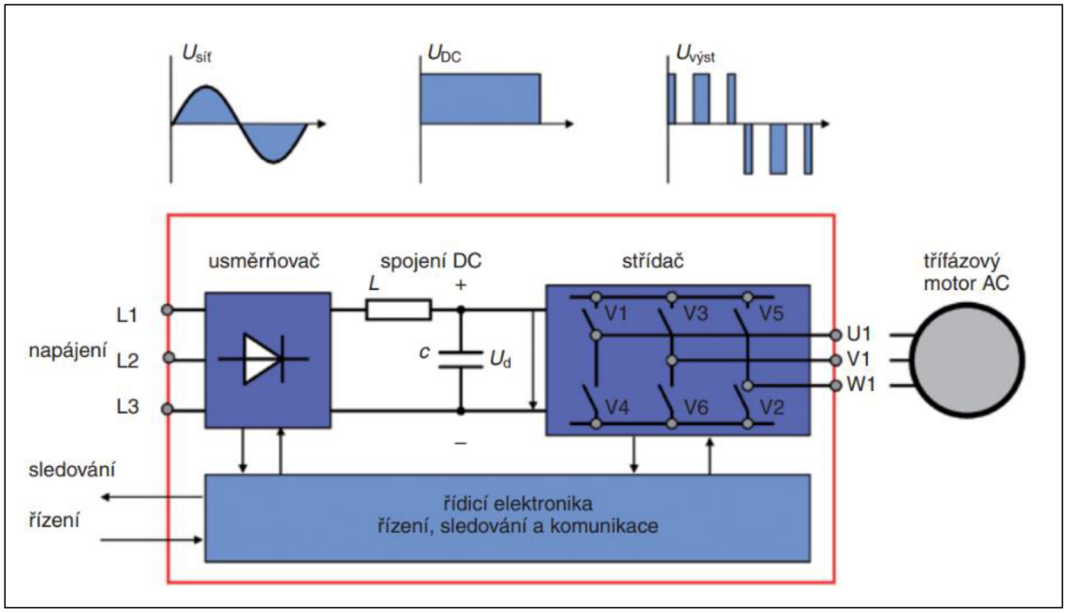
Měnič frekvence (VFD – Variable frequency drive) je elektronický přístroj, který umožňuje měnit frekvenci sítě na požadovanou frekvenci. Měniče frekvence jsou určeny pro nejrůznější použití, avšak v běžném technickém slovníku se jimi rozumějí měniče pro asynchronní motory. Tímto spojením získávají střídavé točivé stroje schopnost hospodárně regulovat otáčky v širokém rozsahu.

Pro úplnost je třeba se zmínit, že se pro stejné zařízení často používají též méně správné názvy frekvenční měniče nebo střídače.

Základní rozdělení měničů je na přímé (maticové měniče a cyklo konvektory) a nepřímé měniče frekvence (nepřímé měniče frekvence s napěťovým a proudovým meziobvodem). Měniče frekvence se skládají z výkonové části, zajišťující přeměnu parametrů napájecí sítě, a z řídicí elektroniky, která ovládá výkonovou část a umožňuje komunikaci s okolím. Řídicí elektronika moderních měničů často zvládne mnoho úloh, které by jinak musely být zahrnuty v nadřazeném řídicím systému.

8.1 Nepřímé měniče kmitočtu s napěťovým meziobvodem

Fungování měniče frekvence lze nejlépe pochopit z blokového schématu na Obr.25 Síťové napětí projde odrušovacím filtrem a usměrní se (nejčastěji usměrňovacím můstkem). Ve stejnosměrném meziobvodu se napětí filtruje tlumivkou a kondenzátory a toto stejnosměrné napětí se přivede na vstup střídače, který opět vytvoří střídavou třífázovou síť, nyní však s proměnným napětím a frekvencí. Na výstup měniče frekvence je připojen asynchronní motor, jehož otáčky jsou přímo úměrné frekvenci. Vlastní měnič je v současné době osazován téměř výhradně spínacími tranzistory (IGBT – *Integrated Gate Bipolar Transistor*). [23]

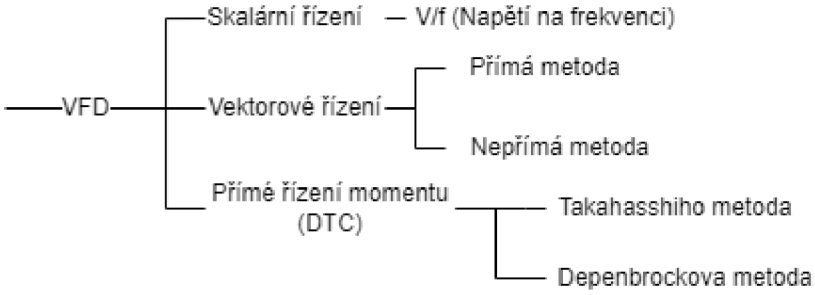


Obr. 25 Blokové schéma měniče frekvence.

Regulační obvody zajišťují vlastní činnost měniče, optimalizují práci motoru a mají rovněž dohlížecí funkci. V případě výrazné odchylky některých parametrů (např. napětí, proudu, teploty aj.) od běžných provozních hodnot vydají varování a při dalším nebezpečném vývoji provozního stavu oznámí poruchu a měnič odstaví. Celé řízení je v současné době digitální a velmi spolehlivé. Měnič frekvence lze ovládat z panelu, kde je obvykle k dispozici několikařádkový alfanumerický displej a několik tlačítek. Běžně se však ovládání z panelu používá velmi zřídka a spíše se využívají buď analogové a digitální (tlačítkové) vstupy, nebo u rozsáhlejších projektů ovládání z nadřazeného řídicího systému po sběrnici (Profibus, Modbus apod.). [23]

8.2 Metody řízení pohonů

Z hlediska vnitřního řízení měniče frekvence se vyskytuje několik systémů, o kterých je třeba se zmínit. Nejjednodušší je skalární řízení, které je založeno na řízení poměru U/f a v podstatě vytváří síť proměnného napětí a frekvence nezávisle na motoru. Je proto dynamicky nejpomalejší, avšak pro jednoduché úlohy plně vyhovuje. Naopak velmi přesné a dynamické řízení je vektorové, které však obvykle vyžaduje otáčkovou zpětnou vazbu (např. tachogenerátor). [23]



Obr. 26 Schéma rozdělení metod řízení motorů.

8.2.1 Skalární řízení

Skalární řízení frekvenčních měničů vychází z rovnic pro ustálený stav asynchronního stroje. Model motoru nerespektuje elektromagnetické jevy uvnitř stroje a z tohoto důvodu neumožňuje řízení okamžité hodnoty momentu, což má za následek zhoršenou dynamiku regulace rychlosti. Skalární řízení bylo pro svou jednoduchost využíváno v pohonech starší generace a dále je využíváno v levných pohonech s nízkými nároky na dynamiku pohonu, například v pohonech čerpadel nebo ventilátorů.

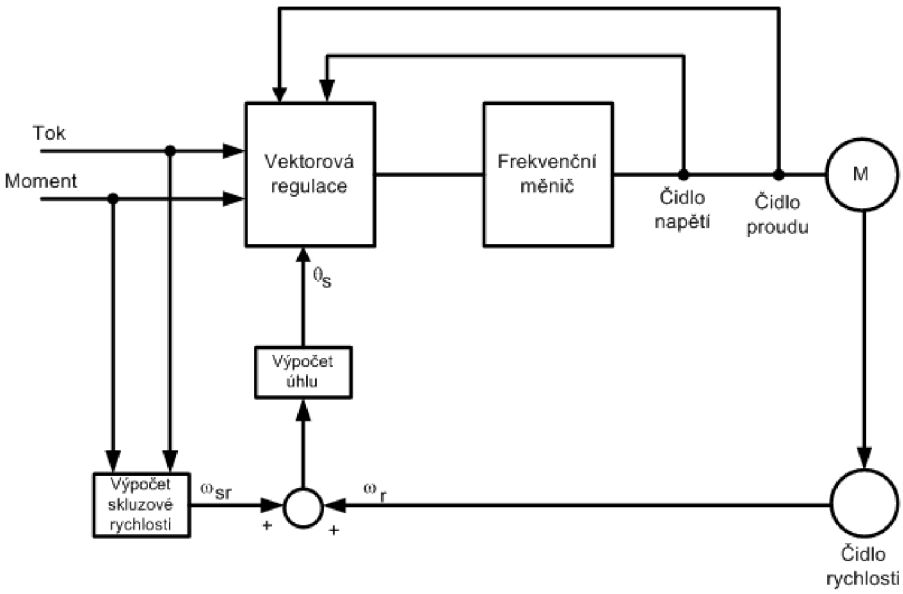
Pro dosažení maximálního možného momentu pomocí statorového proudu musí být velikost magnetického toku strojem konstantní a blízká jeho jmenovité hodnotě. K řízení na konstantní magnetický tok je nutné řídit napájecí napětí U a napájecí frekvenci f v konstantním poměru. [24]

8.2.2 Vektorově orientované řízení

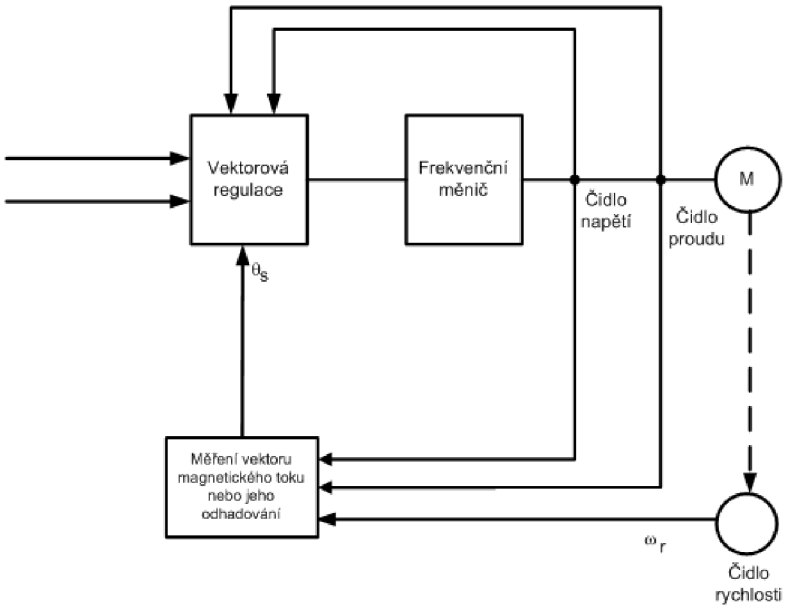
Vektorové řízení asynchronních motorů vychází z modelu popisujícího jak elektromagnetické, tak elektromechanické jevy ve stroji. Díky tomuto modelu lze efektivně řídit okamžité hodnoty toků a proudů ve stroji a v důsledku rovněž okamžitou hodnotu momentu stroje. Prostorové vektory veličin v modelu stroje jsou dále transformovánu pomocí Clarkovy a Parkovy transformace do/z souřadného systému svázaného s prostorovým vektorem spřaženého rotorového nebo statorového magnetického toku. V řídicí struktuře se v ustáleném stavu jeví zadávané hodnoty jako stejnosměrné. Na asynchronní pohon s vektorovým řízením pak lze pohlížet jako na stejnosměrný pohon s cizím buzením, kde lze řídit nezávisle tok motorem a jeho moment. Základní schéma nepřímého a přímého řízení jsou zobrazena na Obr.27 a Obr.28.

Metoda přímého vektorového řízení závisí na generaci jednotkového vektoru ze statorových veličin anebo z magnetického toku ve vzduchové mezeře. Magnetický tok ve vzduchové mezeře se obdrží buď přímý měřením anebo se odhaduje ze statorového proudu a napětí. V případě přímého vektorového řízení není potřeba použít čidlo polohy pro získání informace o poloze rotoru.

V případě nepřímého vektorového řízení se poloha rotoru získá integrací součtu skluzové rychlosti a rotorové rychlosti. Pro odhad skluzové rychlosti je použit model, jehož vstupy jsou žádané veličiny vektorového řízení. [24]



Obr. 27 Blokové schéma přímého vektorového řízení. [24]



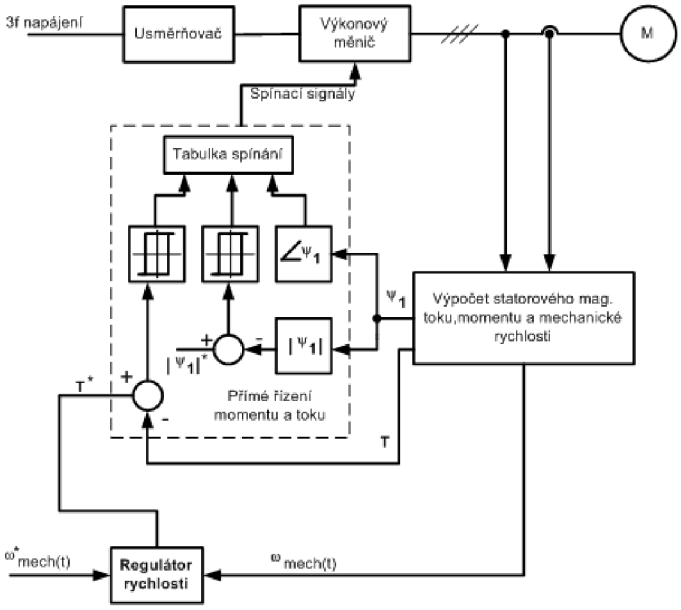
Obr. 28 Blokové schéma nepřímého vektorového řízení. [24]

8.2.3 Přímé vektorové řízení momentu a statorového magnetického toku

V současné době asi nejdokonalejší řízení je tzv. přímé řízení momentu (DTC – *Direct Torque Control*). Základní myšlenka DTC je naznačena na *obr. 10*. Jádrem systému jsou hysterezní regulátory momentu a magnetického toku, které využívají optimalizovanou spínací logiku, čímž odpadá prvek modulátoru. Velmi důležitou částí řízení je přesný model motoru. V něm se vypočítává skutečný moment, statorový magnetický tok a otáčky hřídele z proudu měřeného ve dvou fázích motoru a ze stejnosměrného napětí v meziobvodu. Tyto výpočty jsou během jedné sekundy uskutečněny 40 000krát, takže regulátor DTC přesně ví, jak se chová hřídel motoru. Přesnost modelu motoru závisí na tzv. identifikačním běhu, který proběhne při uvádění pohonu do provozu. Hlavními parametry modelu motoru jsou indukčnosti a odpor statoru. Bere se v úvahu rovněž vliv magnetické indukce na velikost indukčností. [23]

Referenční hodnoty momentu a toku jsou porovnávány se skutečnými hodnotami a řídicí signály jsou generovány dvouúrovňovou hysterezní logikou. V DTC není samostatný šířkově pulzní modulátor (PWM – *Pulse Width Modulator*), který by řídil napětí a frekvenci. Řízení DTC je popisováno jako spínání *just in time*, každé sepnutí je potřebné a využité. U klasického řízení s PWM bývá 30 % sepnutí nevyužitých. Díky uvedeným vlastnostem umožňuje DTC mimořádně rychlou momentovou odezvu (pod 2 ms) a velmi rychlou reverzaci. Moment vykazuje značnou linearitu v celém rozsahu otáček, včetně nulových. Přesnost regulace otáček je velmi dobrá v celém otáčkovém rozsahu, a to i bez nutnosti použít zpětnovazební čidlo otáček. Navíc při použití čidla otáček je takovýto pohon z hlediska regulace úhlové rychlosti roven stejnosměrnému pohonu (statická chyba otáček je 0,01 %), a splňuje tak nejpřísnější požadavky jak na dynamiku, tak na přesnost.

Dalšími přednostmi řízení DTC jsou možnosti překlenutí krátkodobých výpadků napájecího napětí, letmý start, potlačení momentových rázů, snížení hlučnosti, optimalizace magnetického toku motoru, brzdění tokem a velký moment i v nulových otáčkách. [23]



Obr. 29 Struktura přímého řízení momentu. [24]

9 AKTUÁLNÍ ŘEŠENÍ

9.1 ASC880 – měnič pro testování

Průmyslové měniče ACS880 jsou navrženy na běžné architektuře měničů ABB. Jsou přizpůsobeny do průmyslových odvětví. Pohon se dodává v devíti různých velikostech rámu pro snadnou instalaci a uvedení do provozu.

Srdcem pohonu je přímé řízení točivého momentu (DTC), přední technologie řízení motoru ABB. Široká škála možností zahrnuje EMC filtry, rozhraní enkodérů a resolverů, dv/dt filtry, sinusové filtry, tlumivky a brzdný rezistor. Vestavěné bezpečnostní prvky snižují potřebu externích bezpečnostních komponent. Pro synchronizovanou komunikaci mezi jednotkami lze zapojit více jednotek do sériového zapojení. [25]

Důležité údaje

- Rozsah výkonu 0,55 až 250 kW
- Postaveno na společné architektuře měničů ABB
- Stupně krytí IP20, IP21 a IP55
- Přímá regulace momentu (DTC) ve standardní verzi
- Zabudované bezpečnostní funkce včetně STO (SafeTorqueOff) ve standardní verzi
- Intuitivní ovládací panel s konektorem USB
- Zabudovaný EMC filtr (Electromagnetic Compatibbility), tlumivka a brzdový střídač
- Podporuje širokou řadu sběrnicových protokolů, vstupů/výstupů a kodérů
- Flexibilní možnosti vstupů/výstupů a kodérů
- Design schválený pro námořní účely [25]

Obr. 30 měnič ASC880.

9.2 ABB Ability Digital Powertrain

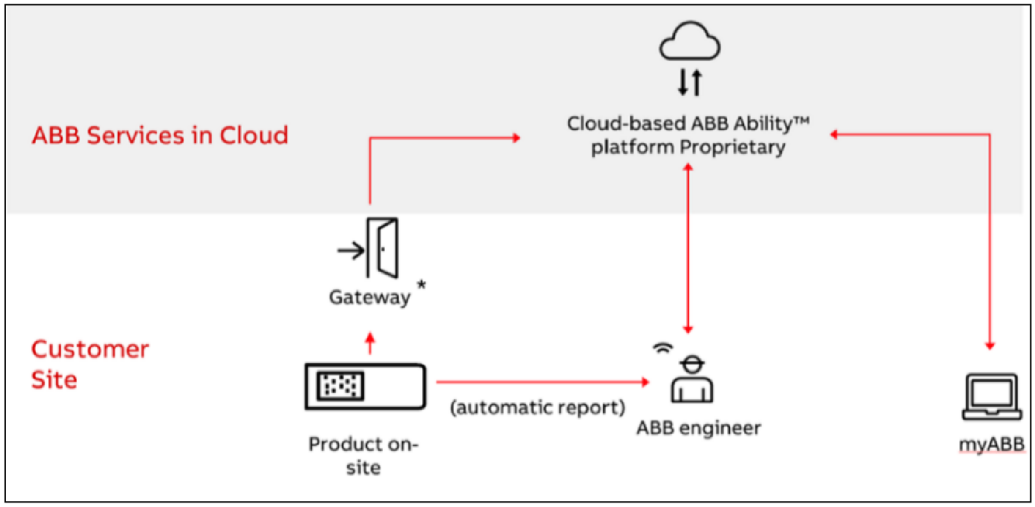
Digital Powertrain propojuje pohony, motory, čerpadla a ložiska, čímž posouvá dobu provozuschopnosti prostřednictvím *ABB Ability*™ [26]. Statistiky dat získané z hnacího ústrojí umožňují zákazníkům lépe se propojit se svými aktivy a činit ještě lepší rozhodnutí pro zajištění bezpečného, spolehlivého a efektivního provozu.

The ABB Ability™ *Digital Powertrain* [27] je vhodná pro aplikace v procesním, diskrétním a infrastrukturním průmyslu. Umožňuje uživatelům digitálně „vidět“ provozní proměnné a zdravotní indikátory prostřednictvím integrovaného portálu na jednom místě – včetně dostupnosti, podmínek prostředí a poruchových událostí [28]

Součástí této aplikace je i *ABB Ability Condition Monitoring for drives* [26], která monitoruje stav měničů s proměnnými otáčkami a přesně odhaduje zbývající životnost jejich klíčových součástí. Pomocí KPI a signálových dat napomáhá identifikovat příznaky možných poruch, než se promění ve vážně provozní problémy. Služba je zaměřena na pohony v kritických aplikacích v ropném a plynárenském průmyslu, kovoprůmyslu, vodárenství a odpadních vodách a celulózovém a papírenském průmyslu, kde je nezbytné vyhnout se nákladným neplánovaným odstávkám závodů. [29] [26]

Služba nepřetržitě monitoruje klíčové součásti měniče s proměnnou rychlostí (VSD) – ventilátory, polovodiče a kondenzátory. Tepelné, napěťové a výkonové senzory shromažďují data o okolní teplotě a změnách zatížení komponent a měří každodenní dopad na jejich životnost. Cloudové algoritmy a statistická analýza odhadují úroveň namáhání součástí a počítají jejich zbývající životnost.

Je-li porucha předpovídána před další plánovanou údržbou, management může učinit plně informované rozhodnutí o akci a zabránit neplánované odstávce. V některých případech může služba *Condition Monitoring* indikovat, že součásti měniče jsou méně namáhány než normálně a vydrží déle, než se očekává. Proto je možné prodloužit intervaly pravidelné preventivní údržby pro zvýšení produktivity a snížení nákladů. [29] [27]



Obr. 31 Princip fungovaní předávání dat v ABB Ability. [27]

9.2.1 Nedostatky

- Analýza dat se provádí pouze v cloudovém prostředí *ABB Ability* [26]. Data se mohou stáhnout offline a následně nahrát do cloudu, nebo musí být senzory připojeny k internetu
- Malá vzorkovací frekvence
- Nedostatečná analýza statusových slov
- Chybí otáčková reference
- Nedostatečná monitoring parametrů a vzájemných závislostí

9.2.2 Požadavky/Vize projektu

- Schopnost analyzovat data u zákazníka. Z důvodu bezpečnosti některé závody nedovolují nahrávat data mimo interní síť
- V případě poruchy uložit časové okno, s co největším vzorkováním
- Sledování závislosti teploty, DC napětí, a proudu
- Sledování rozběhu od startu do nastavených otáček
- Alarmový systém sledující statusová slova měniče a limity pro určité parametry
- Rozložení statusových slov do tabulky, aby byly lépe čitelné pro operátora
- Zachovat propojení na cloudový systém

10 REALIZACE PROJEKTU

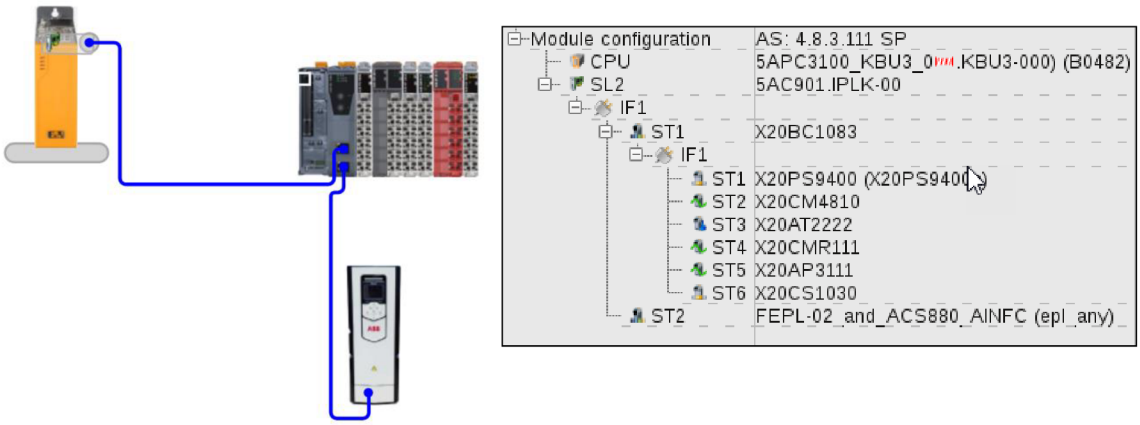
Projekt je navržen tak aby otestoval základní požadavky a funkce, ať po hw nebo sw stránce. Hlavním cílem je vytvořit základ, na který se dá navázat a pokračovat ve vývoji aplikace. V této první fázi by měla být aplikace schopna načíst data z měniče, vyhodnotit základní stavy a současně uložit data pro pozdější analýzu.

Projekt je realizován ve vývojovém prostředí CaeManeger. Kde je vytvořen jeden řídící systém (*ApSy01*). Pro tento systém jsou pak vytvořeny programy a procesní grafika sloužící k zobrazení informací. K vytvoření CFC programů jsou použity prvky, které jsou obecnou součásti Aprolu, ale i individuálně vytvořené bloky, nacházející se v knihovně *ABBinverter*. Knihovna je rozdělena do logických částí:

- Powerlink komunikace
- grafické bloky pro procesní grafiku
- PDA bloky pro ukládání dat
- python skripty
- hypermakra
- funkční bloky (C, ST).

10.1 HW konfigurace

V rámci projektu byla navržena testovací hw konfigurace, která má za účel otestovat jednotlivé části systému je jejich možnosti. Konfigurace se skládá z produktu B&R a ABB. Hlavním prvkem je průmyslové PC APC 3100, na kterém běží díky hypervisoru dva systémy. Real-time řídící systém (Automation runtime) a nadřízený systém Aprol. K této hlavní jednotce je připojen powerlinkový řadič sběrnice X20BC1083 a následně samotný měnič. Toto schéma bylo zvoleno z důvodu jednoduché rozšiřitelnosti. K řadiči lze snadno připojit další karty pro další I/O (např.: senzor teploty, vibrací, analogové stupy atd.). Zároveň je možné v budoucnu připojit více měničů v libovolné struktuře, tedy z jedno řadiče můžeme číst několik měničů, nebo ke každému měniči být přidělen jediný řadič s měřicím kartami.



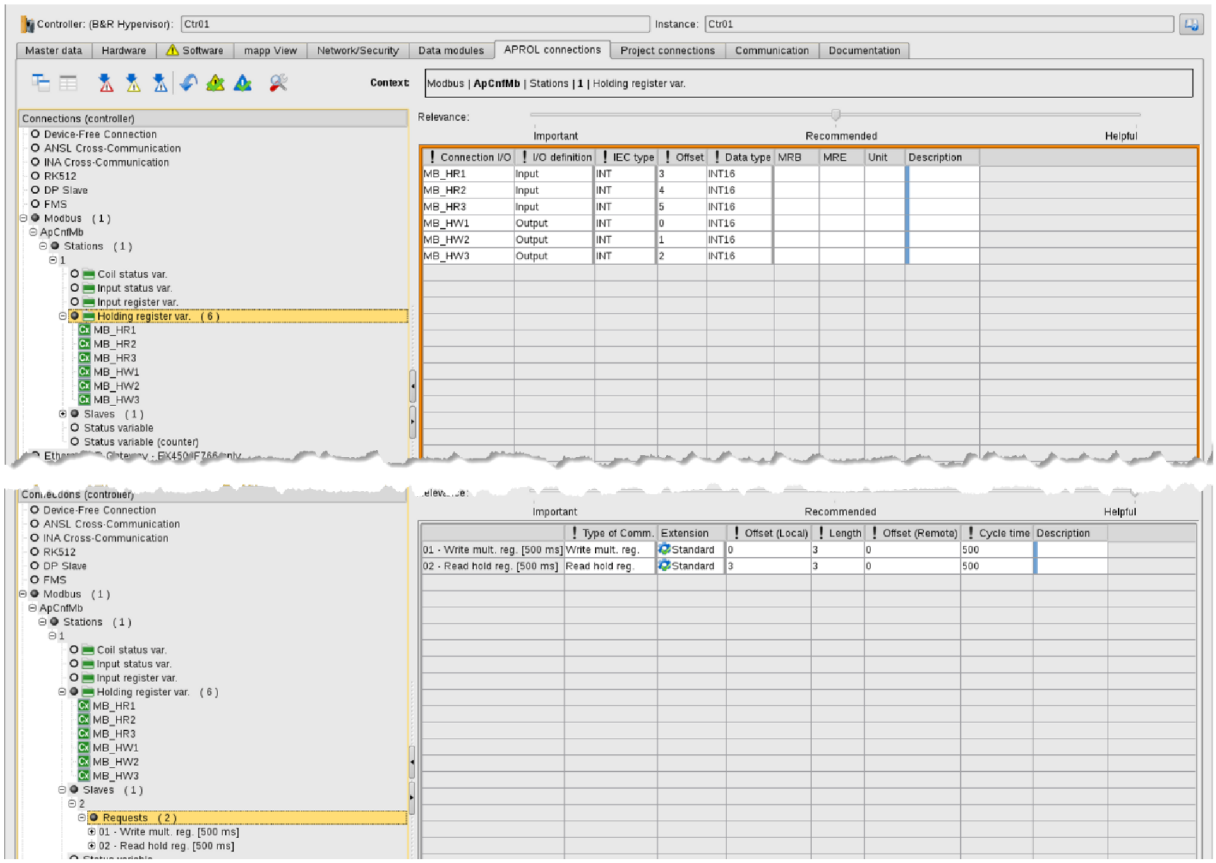
Obr. 32 Schéma HW konfigurace aplikace.

10.2 Komunikace měnič – APROL

10.2.1 Modbus

Pro komunikaci s externími zařízeními po sériové lince pomoci protokolu Modbus RTU/ASCII využívá APROL driver ApDrvMb, který se konfiguruje v CaeManageru, respektive v nastavení kontroleru, v záložce "APROL connections". Driver podporuje cyklickou komunikaci v režimu master a pasivní v režimu slave. Pokud v APROLu vytvoříme dotazy na data, chová se jako master.

Se základními znalostmi systému, datových typů (*Read coils, imput status, holding registers, input registers*) a funkcí (1-16) lze snadno a rychle nastavit komunikaci po sériové lince. K otestování funkčnosti byly vytvořeny dva dotazy. Jeden pro zápis a druhy pro čtení. Protože zprovoznění komunikace přes Modbus nevyžaduje vytvářet dodatečné funkce, není dále v této práci zmiňována.



Obr. 33 Nastavení komunikace přes Modbus.

10.2.2 Powerlink

Měnič dokáže poskytnout omezené množství cyklických dat, která jsou přenášena mezi PLC a měničem. V produktech ACSx80 je toto maximum 12 datových slov přenášených dovnitř a ven. Dále je možné data přenášet acyklicky

Standardně jsou parametry měniče, které jsou přenášeny přes Powerlink, mapovány jako 32bitové DINT i když některé parametry mají pouze 16bitovou hodnotu (např. 04.01 Chyba vypnutí). Mapování pouze 32bitových hodnot omezuje maximální počet parametrů na 6 vstupů/výstupů.

Pro úsporu místa v datových slovech lze parametry mapovat také jako 16bitové. Pro přemapování parametrů je třeba upravit XDD soubor, který se pak importuje do AutomationStudia.

Parametry mohou být unsigned, nebo signed integery. Při editaci souboru XDD by měla být upravena pouze délka dat (16-bit nebo 32-bit). Typ parametru signed/unsigned by měl zůstat nezměněný.

Kódy datových typů v XDD

Signed

- Signed INT (16-bit) - dataType="0003"
- Signed DINT (32-bit) - dataType="0004"

Unsigned

- Unsigned INT (16-bit) - dataType="0006"
- Unsigned DINT (32-bit) - dataType="0007"

```
<Object index="6076" name="Motor rated torque" objectType="7" PDOmapping="optional" accessType="rw" dataType="0007" />
<Object index="6077" name="Torque actual value" objectType="7" PDOmapping="optional" accessType="ro" dataType="0003" />
<Object index="6078" name="Current actual value" objectType="7" PDOmapping="optional" accessType="ro" dataType="0003" />
<Object index="607A" name="Target position" objectType="7" PDOmapping="optional" accessType="rw" dataType="0004" />
```

Obr. 34 Upravené parametry v XDD souboru měniče.

Pro naše účely jsme vybrali 10 parametrů, které se budou přenášet cyklicky. U těchto parametrů byla změněna délka na 16 bitů v XDD soboru a následně byly namapovány do Aprolu aby s nimi bylo možná dále pracovat v rámci projektu. Ostatní hodnoty se budou přenášet acyklicky a je třeba vytvořit funkční blok, který bude tyto data získávat z FM. Oba typy komunikace jsou zpracovávány v programu *PlkCommunication.*

I/O mapping of I/O module - CaeManager (engin@hvabb)

Instance: Ctr01 Module: epl_any Slot: SL2.IF1.ST2 Simulation switched off

	I/O	Channel	Signal Mode	I	E	R	Project variable	MSR No.	IEC type	Substitute value	Resulting substitute value	A	AOS	B	U	O	S	MRB
1		ModuleOk	1 Bit						BOOL		0							
2		_0101_Motor_speed_used_I4001_S01	16 Bit signed				I101		INT		0							-327
3		_0107_Motor_current_I4001_S07	16 Bit signed				I107		INT		0							-327
4		_0110_Motor_torque_I4001_S0A	16 Bit signed				I110		INT		0							-327
5		_0111_DC_voltage_I4001_S0B	16 Bit signed				I111		INT		0							-327
6		_0121_U_phase_current_I4001_S15	16 Bit signed				I121		INT		0							-327
7		_0122_V_phase_current_I4001_S16	16 Bit signed				I122		INT		0							-327
8		_0123_W_phase_current_I4001_S17	16 Bit signed				I123		INT		0							-327
9		_0131_Ambient_temperature_I4001_S1F	16 Bit signed				I131		INT		0							-327
10		_0511_Inverter_temperature_I4005_S0B	16 Bit signed				I511		INT		0							-327
11		_9001_Motor_speed_for_control_I405A_S01	16 Bit signed				I9001		INT		0							-327

Obr. 35 I/O mapping cyklických proměnných v APROLu.

10.2.3 Acyklická komunikace

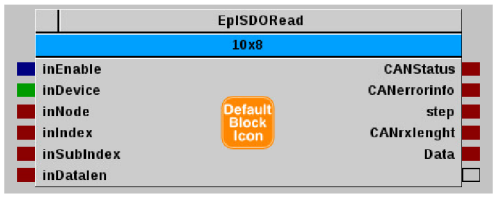
Pro acyklické získávaní hodnot byl vytvořen funkční blok využívající AS knihovny AsEPL, která umožnuje komunikaci se zařízeními třetích stran přes powerlink aniž by byl potřeba dodatečný driver.

Pro čtení dat je použita funkce EplSDORead, která přistupuje na konkrétní adrese parametru. Před spuštěním funkce je potřeba ji nastavit cestu k zařízení v topologii Powerlink (pDevice), dále node, index, subindex a maximální délku hodnoty kterou budeme číst.

Protože funkce dokáže v jednom momentu přistupovat k jedné hodnotě, bylo nutné vymyslet způsob čtení všech zbylých proměnných.

První možnost bylo vytvořit funkční blok se stavovým automatem, který by postupně přepínal mezi jednotlivými proměnnými a vyčítal je. Z důvodu snížení počtů vstupů byly veškeré parametry pro komunikaci nastaveny v rámci kódu. Při testování se ukázaly problémy při konfigurací spojení, obsluhy chyb (např.: ztráty komunikace), nebo vyčítání hodnot v cyklech menších než 500 ms.

Proto se opustilo od tohoto řešení a byl vytvořen funkční blok, který čte parametr samostatně. Tento blok obsahuje vstupy pro nastavení komunikace, tak výstupy pro detekci chyb. Následně byl tento blok použit v HyperMakru (HM) *ReadPlk*, kde pro každý parametr měniče byla použita nová instance funkčního bloku. Toto řešení umožnuje snadno a rychle upravovat parametry pro komunikaci, nebo přidávat / odebírat parametry, které chceme vyčítat.

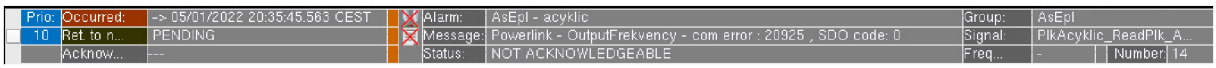


Obr. 36 Blok pro čtení acyklických dat.

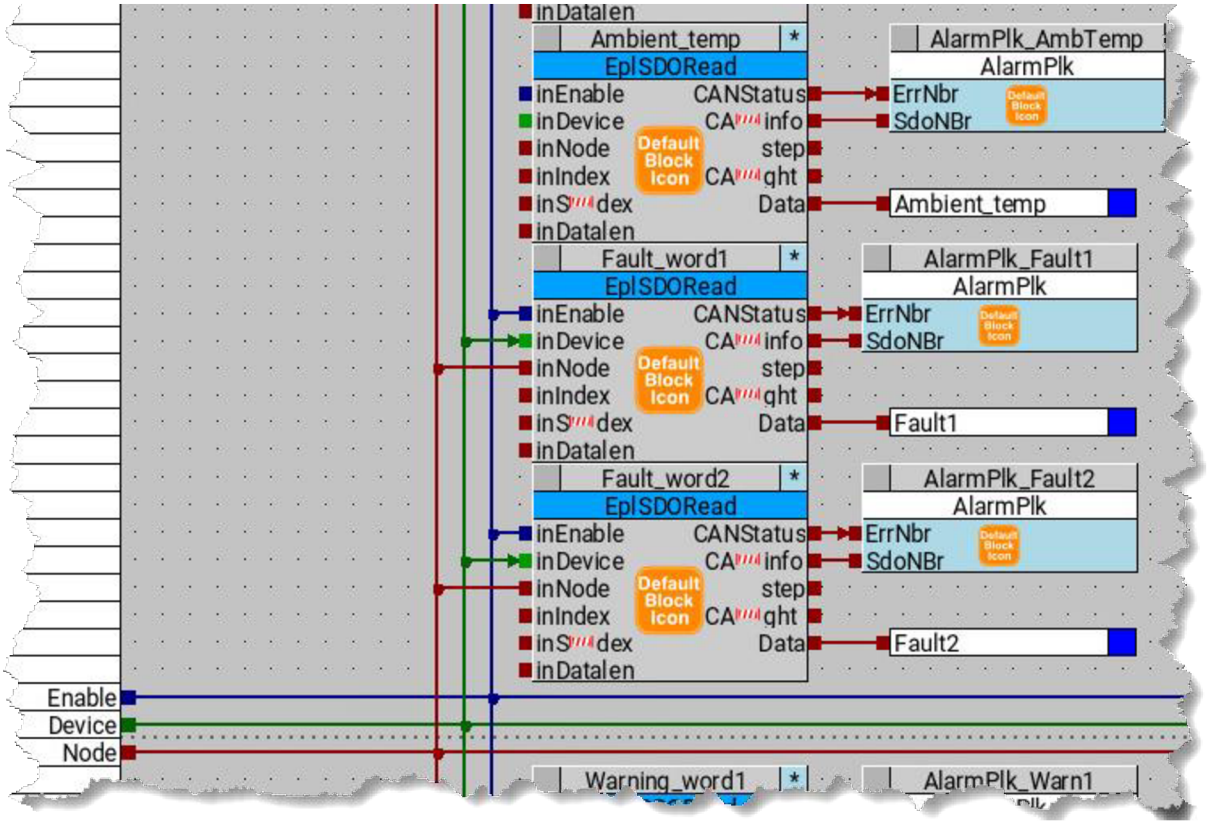
```
FUNCTION_BLOCK EplSDORead
(*. ATTENTION: Do not edit above this line. *)

IF (inEnable = 1) THEN
    CASE step OF
        0: //setup
            EplReadData_0.enable. := TRUE;
            EplReadData_0.pDevice = ADR ( inDevice);
            EplReadData_0.node := inNode;
            EplReadData_0.index := inIndex; (* Index.of. IO device*)
            EplReadData_0.subindex := inSub Index; → (* Subindex.of.IO *)
            EplReadData_0.pData. := ADR (Data);
            EplReadData_0.datalen. := inDatalen;

        step: =1;

        1: //wait for response
            IF Epl ReadData_0.status = 0 THEN
                Epl ReadData_0.enable = FALSE;
                CANStatus:= Epl ReadData_0.status;
                step := 0;
            ELSIF Epl ReadData_0.status = 65535 . THEN
                CANStatus:= Epl ReadData_0.status;
                (* busy *)
            ELSE
                (*error*)
                step = 10;
            END IF
        10://error handling
            CANStatus : = EplReadData_0.status;
            CANerrorinfo := EplReadData_0.errorinfo;
            step = 0;
            EplReadData_0.enable. := FALSE;
    END_CASE;
ELSE
    EplReadData_0.enable = FALSE;
    step := 0;
    CANStatus : = 0;
    CANerrorinto = 0;
END_IF

Epl ReadData_();

(*. ATTENTION: . Do not edit below this line.*)
END_FUNCTION_BLOCK
```

Obr. 37 Ukázka kódu pro načtení parametru přes Powerlink V2.

Součástí hypermakra je detekce chyb (*AlarmPlk*). Pokud blok nevrátí po zavolání funkce kód pro OK(0), nebo Busy (65555), je okamžitě tato událost zaznamenána v alarmovém systému a zobrazen alarm v HMI s informací a jaký parametr se jedná, číslo chyby s dodatečným SDO kódem, který upřesňuje příčinu chyby.



Obr. 38 Ukázka alarmu pro chybu při čtení acyklických hodnot



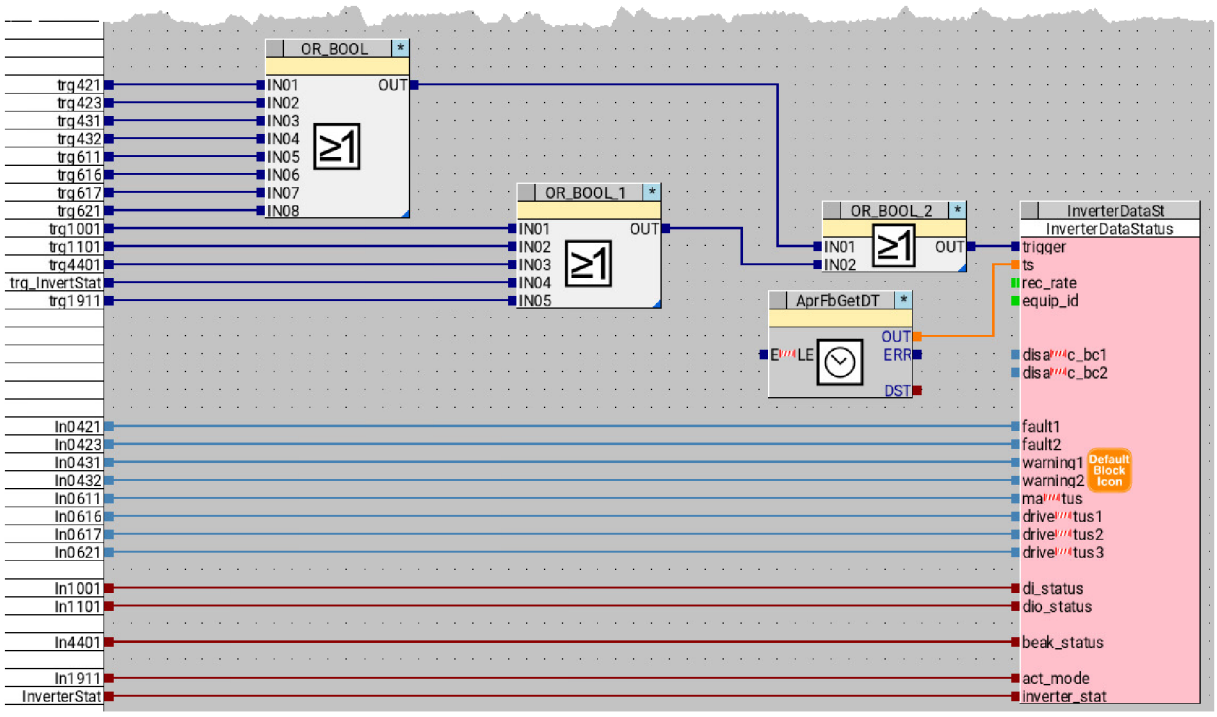
Obr. 39 Rozložení bloků pro acyklické čtení a alarmů.

10.3 Ukládání dat

Hodnoty parametrů se ukládají do chronologu pomocí bloků „Proces Data Acquisition (PDA)“. Při nástupné hraně na vstupu „trigger“ se zapíšou veškeré hodnoty, které se aktuálně nacházejí na vstupech bloku. U cyklického zápisu spouští ukládání časovač. Pro hodnoty, které se mění nepravidelně a neočekávají se rychle změny, jo uložení spuštěno v závislosti na změnu hodnoty.

PDA bloky je třeba nakonfigurovat. Při vytvoření bloku jsou vytvořeny základní vstupy(piny), další vstupy je možné si přidat dle potřeb aplikace.
Základní piny:

- *ts* pokud není převedena vlastní časová značka je použit systémový čas APROL-u
- *rec_rate* zase umožnuje rozdělit ukládaná data do dvou podčástí Chronologu, tyto podčásti jsou *Fast* a *Slow*
- *Equip_id* umožňuje přidání pomocného identifikátoru pro následnou filtraci dat. Většinou se používá jméno zařízení
- *Disable* proměnné jsou v tomto bloku pro potřebu ignorování nějaké námi nakonfigurované proměnné při daném cyklu zápisu (tato funkcionalita se v projektu nevyužívá)



Obr. 40 PDA blok a jeho použití.

Tab. 4 PDA bloky pro ukládaní hodnot

Název	Systém ukládání	Popis
Prio1	cyklicky	hodnoty, které mají nevětší význam při analýze stavu
Status	při změně některé z hodnot	statusová slova měniče
Scaling	při změně některé z hodnot	Proměnné pro škálování, podle kterých se upravují hodnoty některých parametrů; tak aby odpovídaly skutečné hodnotě
Other	cyklicky	ostatní parametry, které se získávají z měniče

Tyto bloky jsou opět použity v hepermakru *InsertInto_Chrono*, které spravuje kdy se má uložení spustit. Výchozí doba periody pro ukládání dat je nastavena na 15 s. Není potřeba mít kratší dobu vzorkování, protože informace jsou ukládány pro dlouhodobý přehled, měsíc a více. Jedinou výjimkou jen blok Prio1. Protože obsahuje data, které jsou pro technika nejdůležitější, jen tento blok použit ve dvou případech. V prvním se data ukládají stejně jako ostatní, pro obecný souhrn. Dále je použit pro zaznamenání důležitých událostí, jako je rozběh motoru, překročení horního limitu, chyba na měniči (statusová slova měniče Fault1, Fault2). Frekvence ukládání je podstatně kratší (200 ms). Tyto data se ukládají do samostatného kontejneru.

Zároveň je pro tyto důležité parametry měniče použito hypermakro (*Analog02* – viz Alarmové bloky), které ukládá hodnoty do chronotrendu a kontroluje překročení mezí. Pokud je mez překročena, je tato událost uložena a také je zobrazen alarm operátorovi.

10.4 Tabulky v relační databázi

Jak už bylo řečeno dříve (č. Kap 7.1) ChronoLog, respektive Berkeley databáze má jen omezené možnosti, jak pracovat s daty pomoci SQL dotazu. Největší výkonnosti dosahuje při chronologickém výběru dat s minimem použitých filtrů pro selekci, nebo spojováni dat. Protože byl systém navrhován tak, aby byl v budoucnu schopen informace kombinovat a tím co nejvíce ulehčil práci technikovi při analýze stavu měniče a řízeného motoru, byla zvolena relační databáze MariaDB. V rámci této databáze byla vytvořena struktura tabulek, která rozděluje parametry do logických celků. Kombinace relační databáze a rozdělení informaci v tabulkách, by měla v budoucnu značně zjednodušit a zrychlit SQL dotazy např.: při vytváření reportu.

Tab. 5 Tabulky v MariaDB

Motor	
Název	Dat. typ
ts	datetime
MotorSpeed_used	int
MotorSpeed_ctr	int
MotorCurrent	int
MotorTorque	int
DC_voltage	int
U_phase	int
V_phase	int
W-phase	int
Out_voltage	int
Out_power	int
Out_freq	int

Status	
Název	Dat. typ
ts	datetime
Main_status	int
Drive_status1	int
Drive_status2	int
Drive_status3	int
Fault1	int
Fault2	int
Warning1	int
Warning2	int
DI_status	int
DIO_status	int
BeakCtrl_status	int
ActOperation_mode	int

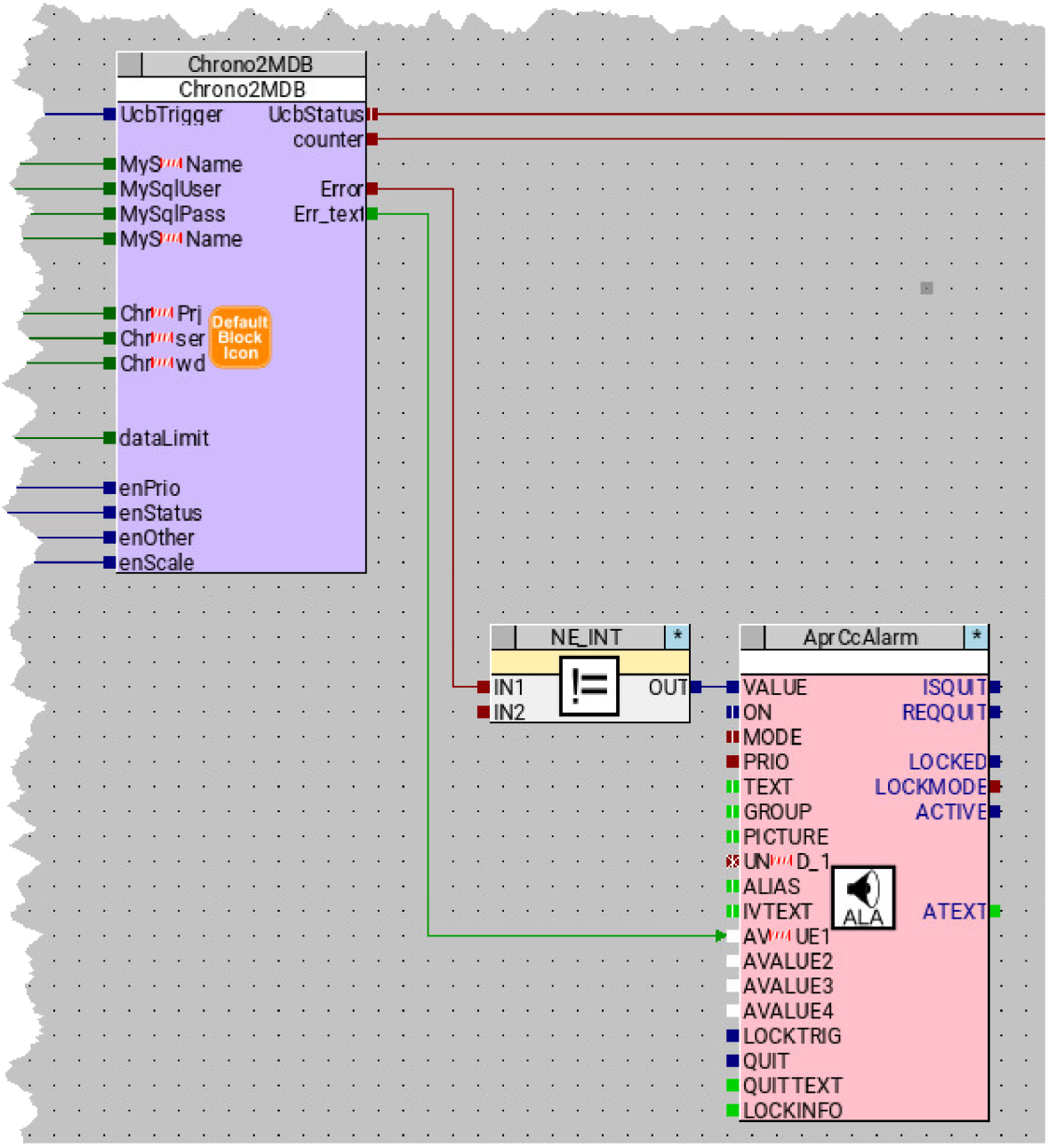
Temperature	
Název	Dat. typ
ts	datetime
Ambient	int
Inverter	int
MotorEst	int
Measured1	int
Measured2	int
BreakingResistor	int

Interface	
Název	Dat. typ
ts	datetime
AI_act1	int
AI_act2	int
A0_act1	int
A0_act2	int

Scaling	
Název	Dat. typ
ts	datetime
Speed	int
Frequency	int
Torque	int
Power	int
Current	int

10.4.1 Přenesení dat z Chronologu do MariaDB

Pro přenesení dat z chronologu do relační databáze se spouští python script. Ten nejdříve vyhledá datum naposledy uložených proměnných v mariaDB. Od této časové značky vybírá data z Chronologu k překopírování. Protože struktura chronologu a mariaDB není stejná, python script po překopírování spouští uloženou proceduru, která nově uložená data rozpracuje do příslušných tabulek. Standartně se script spouští každých 5 minut, ale cyklus kopírování není definován napevno, takže v rámci aplikace možné upravit. Pro všechny důležité funkce je vytvořena správa chyb, které jsou následně přeneseny Aprolu a AlarmServeru.



Obr. 41. Použití bloku UCB pro přenesení dat z ChronoLogu od mariaDB.

```
BEGIN
INSERT INTO BUR_PDA.Motor (
    ts,
    MotorSpeed_used, MotorSpeed_ctr,  MotorCurrent, MotorTorque,
    DC_Voltage,   U_phase, V_phase, W_phase,
    Out_voltage,   Out_power, Out_freq)
SELECT
  prio.ts,
  prio.motor_speed,
  prio.motor_speedctr,
  prio.motor_current,
  prio.motor_torque,
  prio.dc_voltage,
  prio.u_phase,
  prio.v_phase,
  prio.w_phase,
  other.output_voltage,
  other.output_power,
  other.output_freq
FROM
  BUR_TMP.tmpPrio AS prio
  INNER JOIN BUR_TMP.tmpOther AS other ON prio.ts = other.ts
WHERE
  prio.ts > (
    SELECT
       CASE WHEN max(ts) IS NULL THEN '2021-01-01 00:00:00' ELSE max(ts) END AS ts
    FROM BUR_PDA.Motor
  );


INSERT INTO BUR_PDA.Temperature (
     ts,
     Ambient, Inverter,
     MotorEst, Measured1, Measured2, BreakingResistor)
SELECT
  prio.ts,
  prio.ambient_temp,
  prio.inverter_temp,
  other.motor_temp,
  other.temeperature_1,
  other.temeperature_2,
  other.brak_temp
FROM
  BUR_TMP.tmpPrio AS prio
  INNER JOIN BUR_TMP.tmpOther AS other ON prio.ts = other.ts
WHERE
  prio.ts > (
    SELECT
      CASE WHEN max(ts) IS NULL THEN '2021-01-01 00:00:00' ELSE max(ts) END
     AS ts
    FROM BUR_PDA.Temperature
  );
END
```

Obr. 42. Ukázka procedury přerozdělující data v MariaDB.

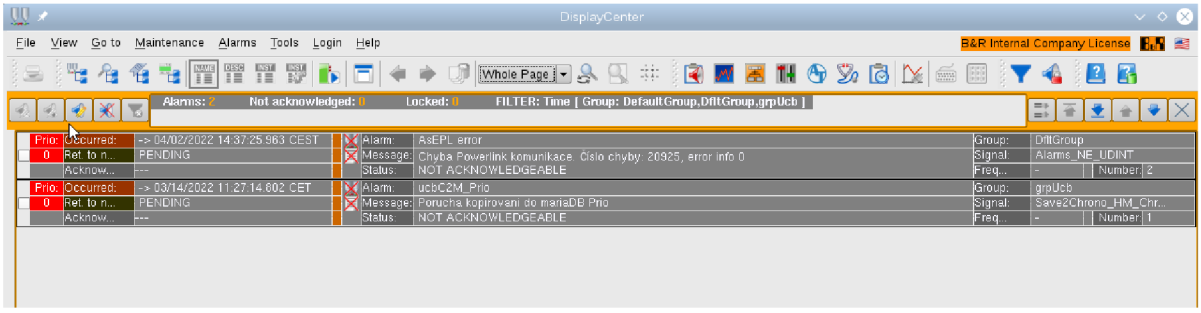
10.5 Vyhodnocování poruch

10.5.1 Alarmy

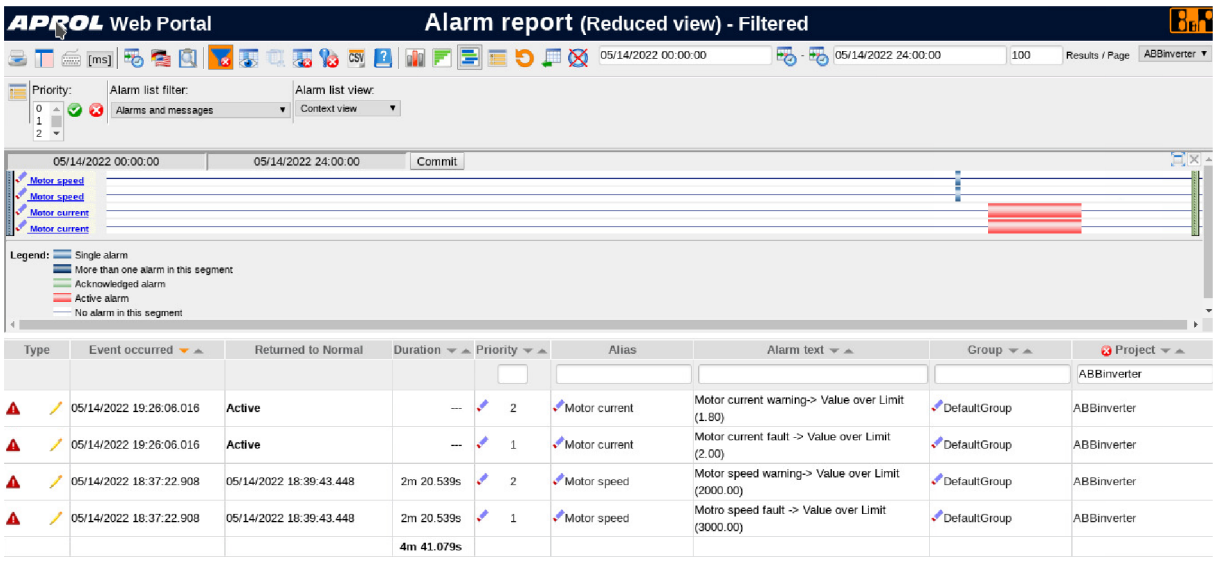
Alarm se aktivuje v případě poruchy, překročení meze a dalších podobných událostech. Pro tuto funkcionalitu je využíván alarm server a sním spojené bloky. Tento server, který je součástí Aprolu, má na starosti zaznamenaní události v historizační databází a předání informace do operátorských stanic.

Alarmové zprávy se dají zobrazit na několika místech. Aktuálně probíhající, neukončené, nebo potvrzené alarmy se zobrazují v alarmové liště (AlarmMonitor) v DisplayCentru. Její rozložení a grafickou podobu lze nastavit v konfiguračním xml souboru. Nastavili jsme počet řádku v náhledovém režimu, uspořádání informací pro samostatné alarmy a zabarvení priority z důvodu lepší rozeznatelnosti.

Dále Alarm server poskytuje report o proběhlých alarmech, které se v systému vyskytly. V tomto reportu lze zobrazit kompletní historii, filtrovat podle data, typu události, původem. Dále v grafech můžeme zobrazit četnost alarmu v daném časovém úseku, nebo průběh alarmů v čase.



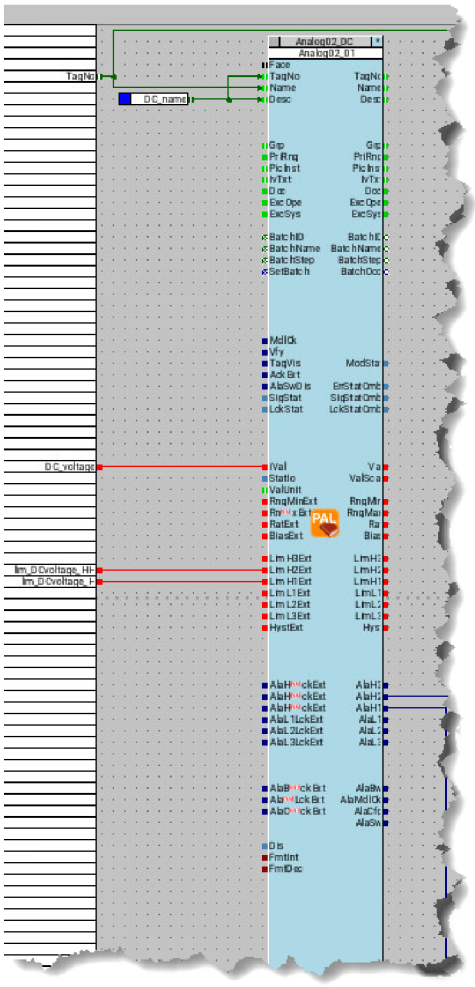
Obr. 43 AlarmMonitor v DisplayCentru.



Obr. 44 Alarm report ve webovém prostředí.

10.5.2 Alarmové bloky

Alarmové bloky se v projektu nachází na několika místech. Jsou součástí hypermakra *Analog02,* které obsahuje PAL (Process Automation Library) [30] knihovna. Toto makro umožnuje monitorování limit H3-H1, L1-L3, vstupních parametrů, hysterezi, škálování, trendování anebo vložení fiktivní hodnoty. Blok *Analog02* je použit pro všechny důležité parametry měniče. Na vstupech jsou připojeny názvy, hodnoty a meze parametrů. Dále je také do makra přivedeno označení měniče. Z výstupu využíváme příznak o překročení meze H1 (varovaní), a H2 (porucha), podle kterých se pak určuje celkový stav měniče.

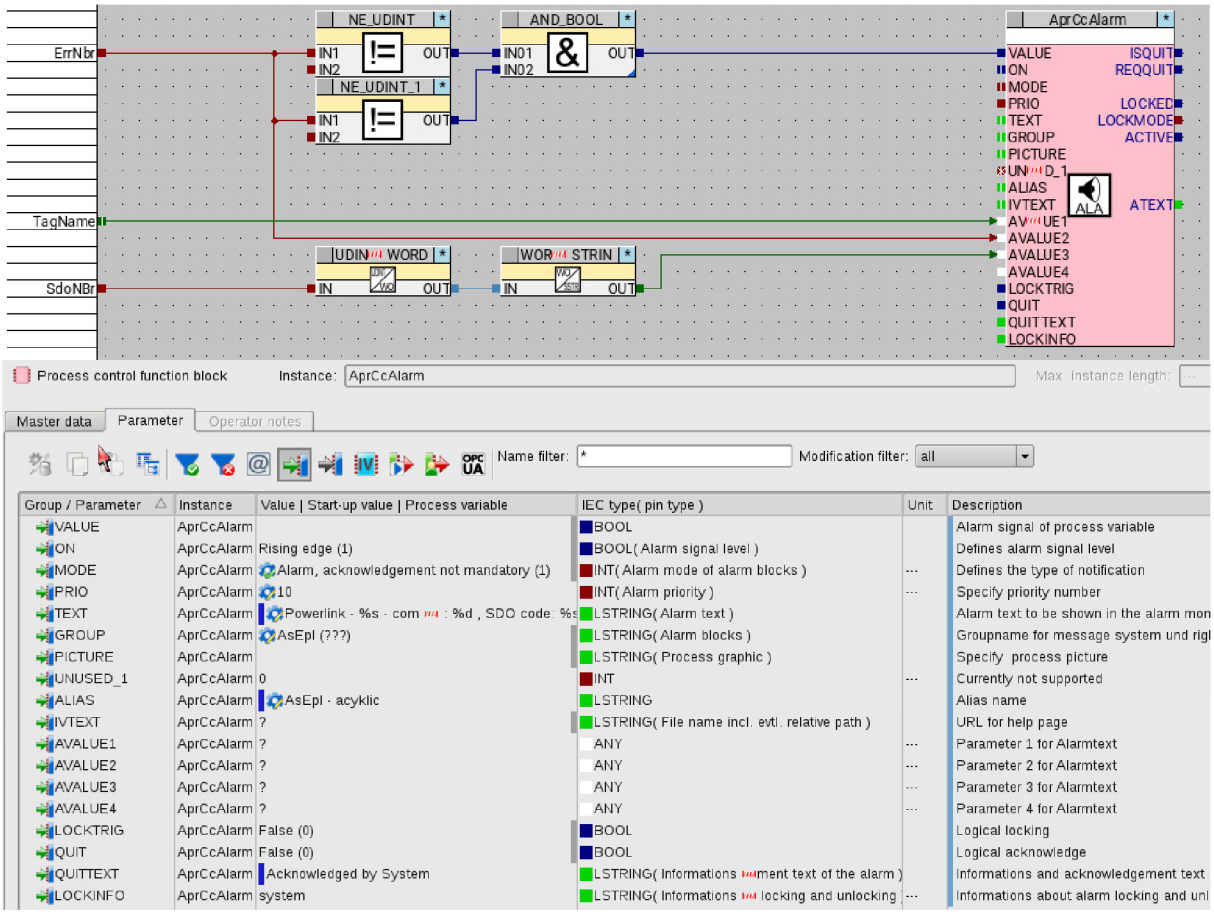


Obr. 45 Hypermakro obsahují Analog02.

V případě monitorování dalších funkcí, jako je Powerlink komunikace, kopírování dat chronolog – MariaDB, nebo kontrola statusových slov měniče jsou použity bloky *AprCCAlarm* a zjednodušena verze *AprCCAlarmBasic.* Před každým takovým blokem se nachází logika spuštění alarmu. Basic zobrazuje pouze předem definovaný text, naopak standartní verze umožňuje přidat do alarmové zprávy i hodnoty, jako hodnota meze, aktuální hodnota kontrolované veličiny atd.

Dále lze v bloku nastavit úroveň alarmu, prioritu a alarmovou skupinu:

- úroveň alarmu – definuje typ upozornění
  - message (zpráva) – informace nebude zobrazena v AlarmMonitoru
  - alarm – alarm bude zobrazen v AlarmMonitoru dokud příčina alarmu trvá
  - alarm vyžadující potvrzení – alarm je zobrazen v AlarmMonitoru dokud podmínka pro zobrazení přetrvává a alarm není potvrzen
  - alarm s potvrzovacím textem – alarm je zobrazen, dokud přičína vzniku alarmu stále trvá a alarm není potvrzen textem
- priorita – je použita jako kritérium k výběru a seřazení alarmů v DisplayCentru
  - 0–9: Porucha HW, odpojený modul, odpojené plc, …
  - 10–19: porucha při čtení dat z měniče přes Powerlink
  - 20–29: status měniče, překročení meze, statusové slovo
  - 80–90: chyba v python skriptu (UCB)
- alarmová skupina – se používá jako výběrové a řídící kritérium v online nástrojích a historizačních nástrojích (např.: AlarmReport). Dále se podle tohoto parametru určují, jaké alarmy může určitý operátor spravovat.



Obr. 46 Alarmový blok AprolCCAlarm a jeho konfigurace.

10.5.3 Stav měniče

Pro jednoduší přehled o stavu byl vytvořen blok *Inverter_status*, který spojuje informace o překročení mezí, rozběhu a stavových slovech měniče. Na základě těchto dat určuje obecný stav, který je uložen do databáze a současně i zobrazen v grafice pro operátora (viz grafické bloky).

a. offline / online – určuje, zdali je měnič z pohledu Aprolu dostupný přes powerlink
b. rozběh – ze statusového slova *Actual_mode* kontroluje bit 2 (startuje) a bit 8(na otáčkách).
c. Fault měnič – měnič detekoval chybu, statusové slovo měniče *Fault1*, *Fault2*
d. Warning měnič – měnič detekoval varování, statusové slovo Warning1, Warning2
e. Překročení meze H1 – hodnota parametru překročil první mez, varování
f. Překročení meze H2 – hodnota parametru překročil druhou mez, chyba
g. Chyba počet H1 – v případě že se vyskytuje více varování (více jak 3), měnič se překlápí do stavu chyby

V případě stavů b.-g. je uloženo speciální datové poruchové okno obsahující parametry v *Prio1*, které jsou zaznamenány s menší vzorkovací periodou. Časový rozsah okna je 5 min před událostí a 2 min po události, výjimkou je rozběh kdy se snímá celý rozběh a minuta po najetí na nastavené otáčky.

10.6 Zobrazení dat

Informace o měniči lze zobrazit na dvou místech. Na stanici v operátorském prostředí, nebo si lze načíst online report vygenerovaný JasperServerem.

Pro zobrazení na stanici jsou vytvořeny obrazovky, které se skládají prvků obsažené v hypermakrech, grafických bloků a widgetů které jsou součástí Aprolu.

10.6.1 Grafické bloky (GB)

Je interface mezi logikou a procesní grafikou. V grafické editoru lze z různých komponent poskládat nejrůznější tabulky, okna, statusové symboly. Dále lze použít dynamizaci jako je změna barvy, posunutí, nebo natočení. Kromě toho bloky podporují i použití python kódu, které je použito pro vypsání logu s posledními stavy měniče.

Popis jednotlivých použitých GB:

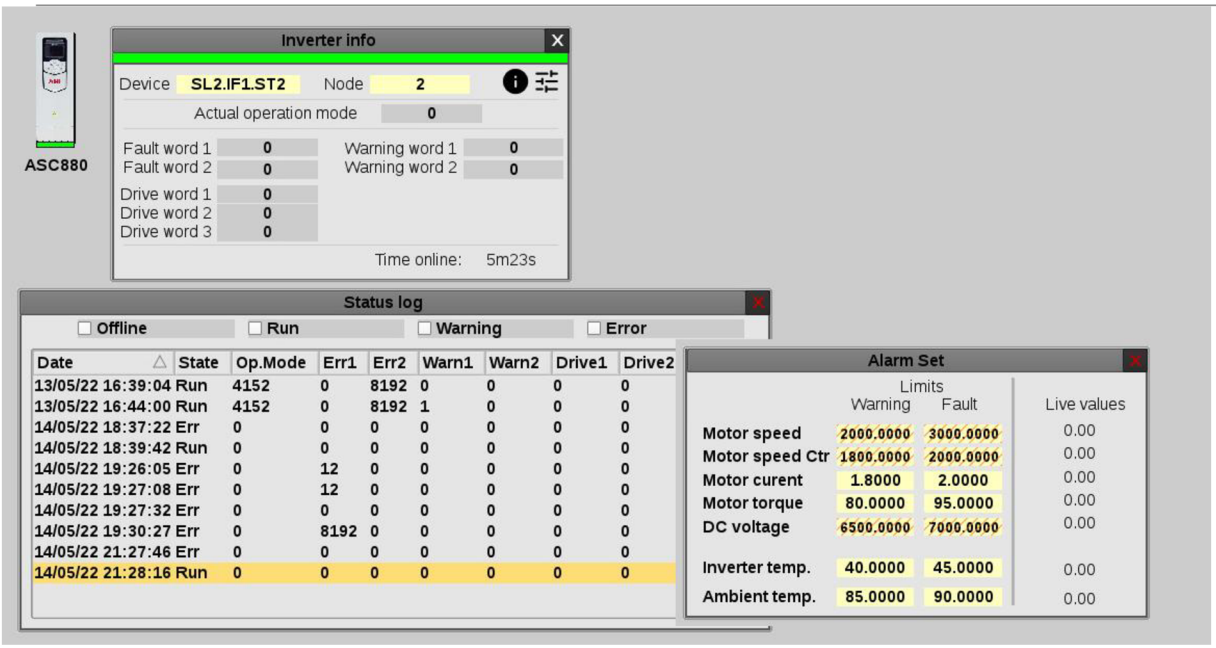
InverterStatus_L0 – Zobrazuje stav měniče v nejjednodušší formě. V závislosti na hodnotě vstupní proměnné *inStatus* je barevně zobrazeno, zdali se měnič nachází v stavu offline, rozběh, překročení meze. Pokud měnič vygeneruje chybové slovo vytvoří příznak na vstupy *en_fault*, *en_warn* a grafický blok zobrazí výstražný symbol.

InverterStatus_L1 – rozšiřuje informace, které jsou zobrazeny v L0. V oknu jsou zobrazeny jednotlivá statusová slova měniče, které se barevně podbarvují, pokud se nerovnají nule. V okně je možné nastavit logickou adresu a uzel (node) potřebné pro acyklickou komunikace přes Powerlink. Dalšími funkcemi jsou zobrazení oken pro nastavení mezí, nebo logu.

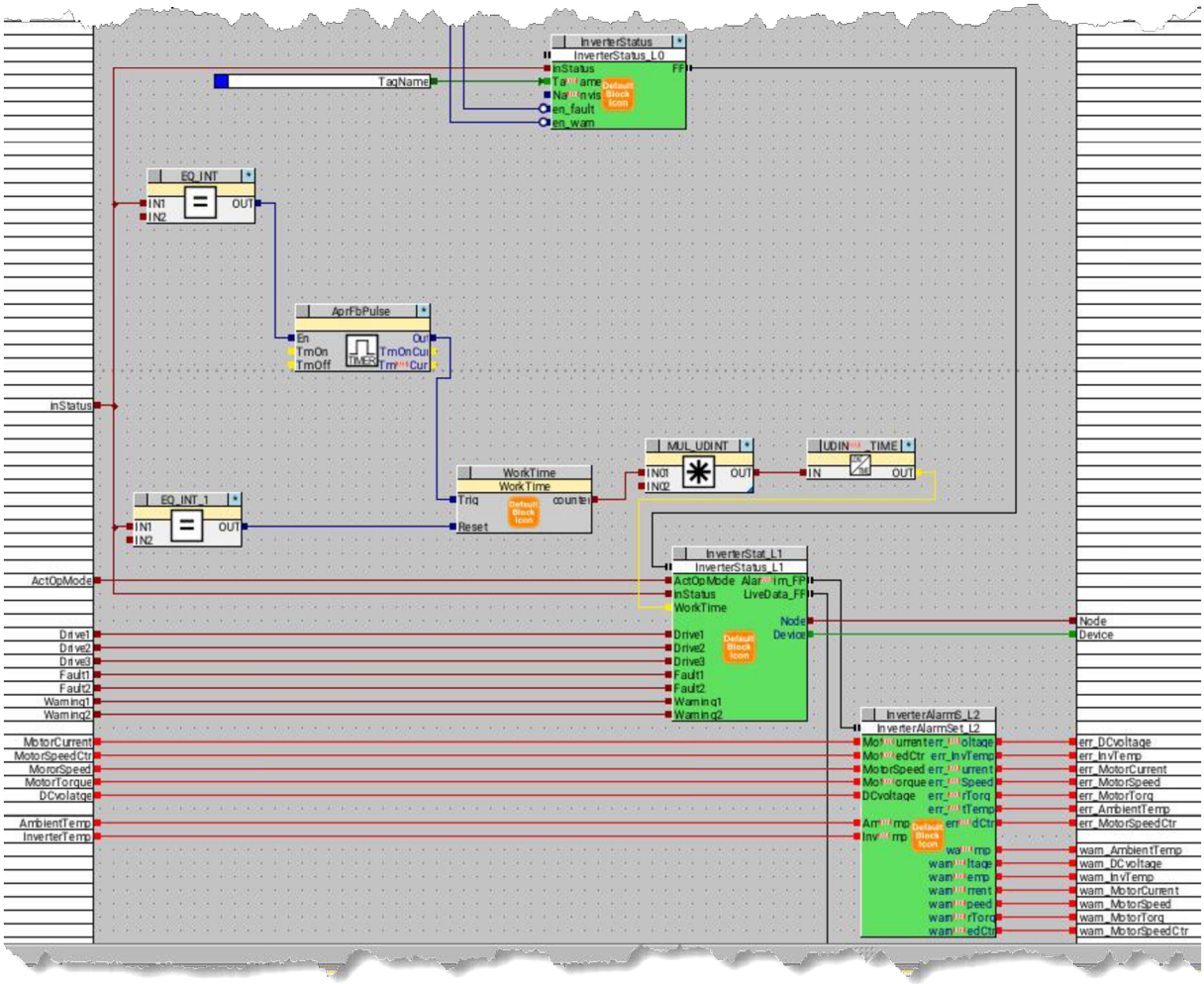
AlarmSet_L2 – otevře se po kliknutí na ikonu posuvných ovladačů v *InverterStatus_L1*. V nově otevřeném okně lze vidět aktuální hodnoty spolu s nastavením mezí H1, H2, které jsou pak použity v bloku *Analog02*.

StatusLog_L2 – otevře se po kliknutí na ikonu *i* v *InverterStatus_L1*. Pomocí python skriptu, načte z MariaDB posledních 10 záznamů o stavu měniče a zobrazí je v okně

Prio_trends – je samostatná skupina bloků, která umožnuje zobrazit trendy v aplikaci TrendViewer dle vlastního výběru. V okně vybereme trendy chceme chceme načíst a potvrdíme tlačítkem „Otevřít graf“. Python skript vygeneruje konfigurační soubor pro TrendViewer a následně spustí aplikaci s daným nastavením.



Obr. 47 Grafické bloky *InverterStatus* v procesní grafice.



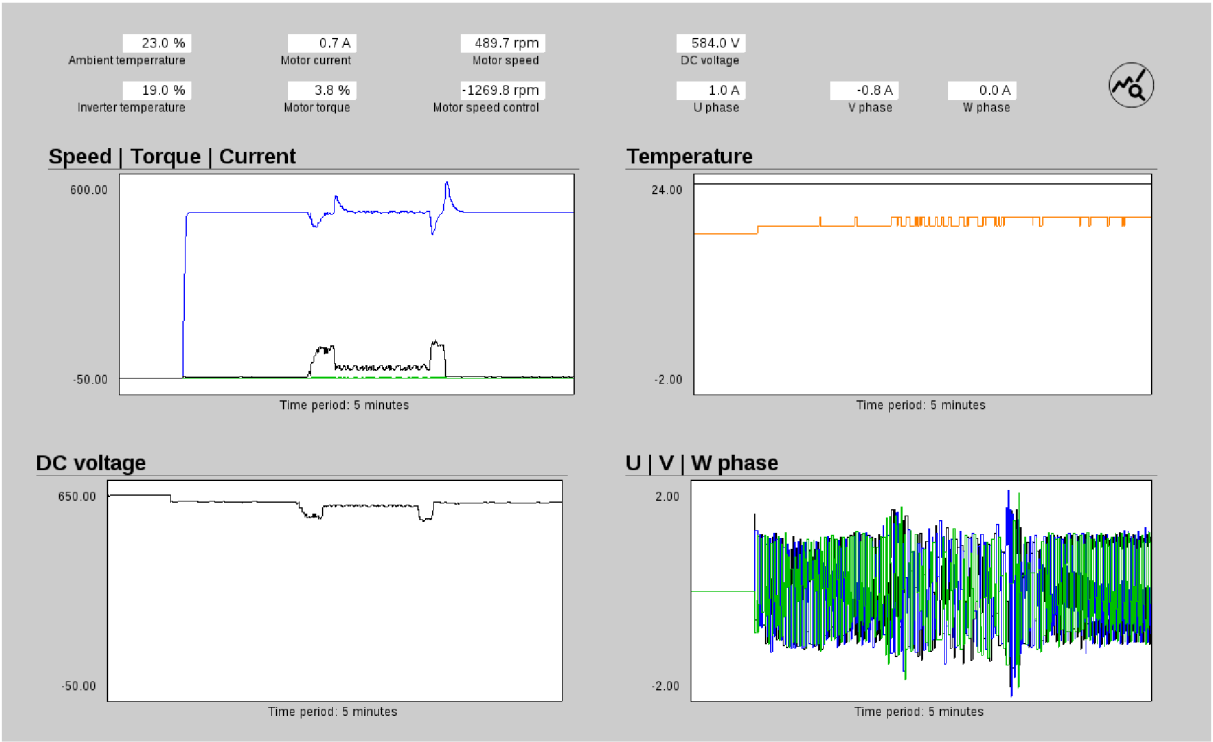
Obr. 48 Použití grafický bloků v logice.

10.6.2 Procesní grafika

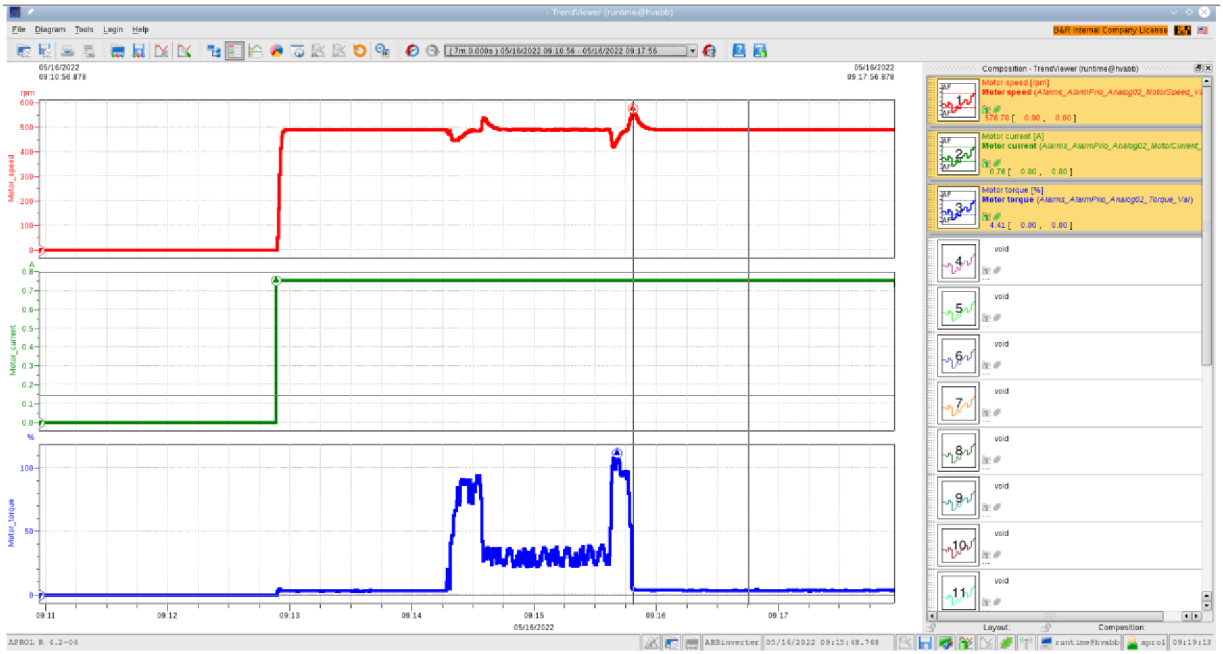
Procesní grafika seskupuje jednotlivé komponenty do jednoho prostřední, DisplayCenter. DisplayCenter je centrální aplikace pro řízení a monitorování systému. Lze zde pomocí dynamické procesní grafiky sledovat stavy, provádět cílené ovládání a potvrzovat alarmy. V tomto projektu byly vytvořeny dvě grafiky:

InverterInfo – ukazuje informace o konkrétním měniči. Z hypermakra *Analog01,* využívá možnosti zobrazit informace o aktuální hodnotě. Dále po rozkliknutí umožnuje zobrazit alarmy, meze a trend ke konkrétnímu parametru. Vedle aktuálních hodnot se nachází blok *Prio_trends* pro načtení parametrů v čase, kde si operátor může vybrat období v jakém se parametry mají zobrazit. Posledním prvkem jsou online trendy. V grafech se zobrazují hodnoty za posledních 5 minut.

InverterOverview – slouží k obecnému přehledu o měničích, nachází se zde grafické bloky *InverterStatus*, *AlarmSet*, *StatusLog*. V současné verzi se zde nachází jediná instance měniče, ale myšlenka tento obrazovky je že se zde budou nacházet všechny měniče, které jsou k Aprolu připojeny.



Obr. 49 Procesní grafika InverterInfo.



Obr. 50 Vygenerované trendy po použiti bloku Prio_trends.

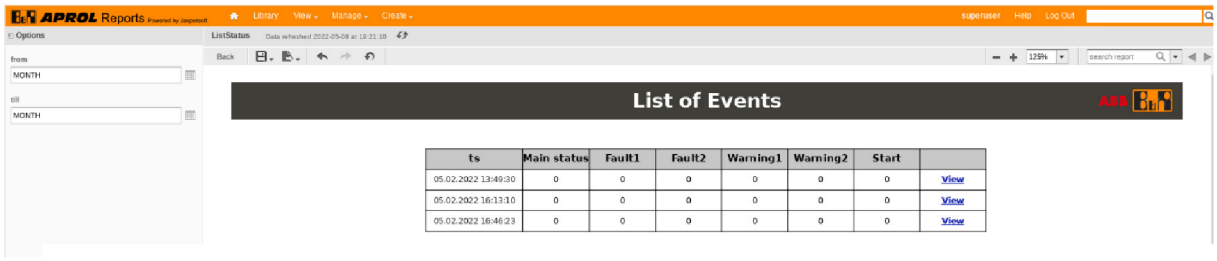
10.7 Tibco JasperSoft

TIBCO Jaspersoft [31] je platforma pro vytváření, správu a distribuci reportů a řídících panelů (dashboardů). Tento software nabízí kompletní sadu nástrojů, které poskytují veškerou funkcionalitu k vytvoření a poskytování reportovacích služeb.

TIBCO JasperReports Server je samostatný reportovací server. Poskytuje reporty a analýzy do webového prostředí. Také dokáže fungovat jako centrální informační centrum tím, že doručuje důležité informace v reálném čase, nebo podle plánu do prohlížeče, e-mailové schránky, sdílené uložiště v různých formátech souborů (pdf, xml, excel, ..). TIBCO Jaspersoft Studio je vývojové prostředí založené na Eclipse [32]. Kombinace těchto aplikací umožňuje vytvářet sofistikovaná rozvržení obsahující grafy, obrázky, sub-reporty, křížové tabulky a další.

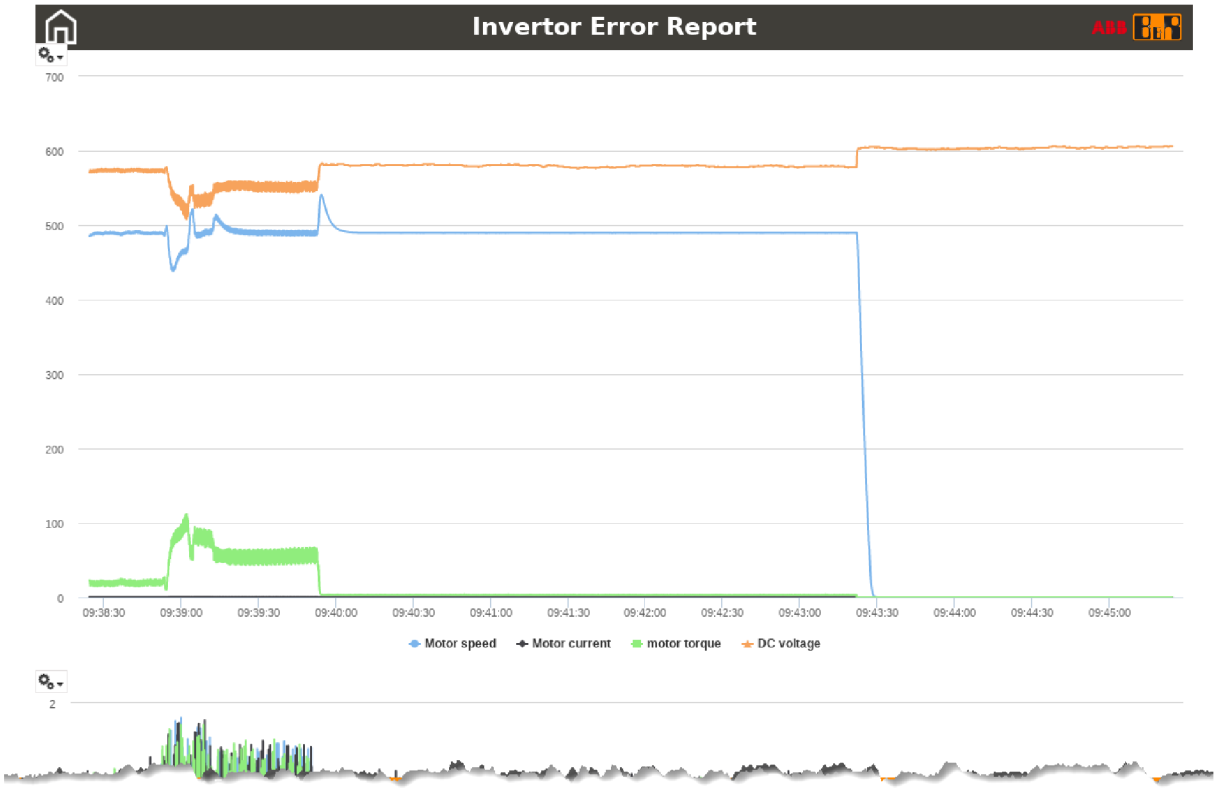
10.7.1 Reporty

V případě výskytu důležité události, jako je rozběh, překročení meze atd. je tato informace o výskytu zaznamenána do tabulky *Status.* Současně se z chronologu zkopírují data, důležitých parametrů (Prio1), v časovém okně okolo události. Pro obecný přehled o událostech je vytvořen report „*List of Events*“ se seznam událostí za poslední měsíc. Tento rozsah lze samozřejmě upravit parametry umístěnými na levé straně. Součástí tabulky je časová značka vzniku, statusová slova měniče a tlačítko „*View*“ pro načtení detailu události.



Obr. 51 Report *List of Events.*

Po kliknutí na tlačítko „*View*“, se načte v závislosti na typu události jiný typ reportu. V tabulce *ErrEventData* se najdou všechny hodnoty s danou časovou značkou události. Hodnoty jsou pak vykresleny do kombinace časových grafů. V grafech lze použít zoom, nebo vypnout zobrazení určitého parametru.



Obr. 52 Detailní report poruchy.

Dále je vytvořen dashboard „*Overview*“ s obecným přehledem, ve kterém s spojí informace z různých tabulek. Celkově se zde nacházejí 4 grafy. První ukazuje výstupní hodnoty měniče, dále se zobrazuje poměr napětí, proudu a momentu. V neposlední jsou zobrazeny teploty měniče a okolí. Časový rozsah všech grafů je defaultně nastaven na poslední kalendářní týden.

Obr. 53 *Overview* dashboard, týdenní přehled.

10.8 Možností rozšíření aplikace

Možnost rozšíření aplikace představuje hlavně přidání dalších funkcionalit:

- Komunikace přes Modbus RTU z důvodu podpory starších měničů
- Zajištění komunikace do cloudu pomoci MQTT
- Rozložení poruchových a alarmových slov do uživatelsky přívětivějšího zobrazení
- Predikce stavu a poruch
- Hlubší diagnostika, sledovaní blízké technologie pomocí vibro-diagnostiky a měření teplot
- Monitoring cirkulačních čerpadel a výměníků tepla

11 ZÁVĚR

Cílem této práce bylo realizovat sběr diagnostických dat z frekvenčního měniče, tyto data následně vyhodnotit a zobrazit operátorovi. V rámci rešerše byly zpracovány komunikační protokoly používané v průmyslu. Dále byl popsán základní princip měniče frekvence a hlavní metody řízení asynchronního motoru frekvenčním měničem. Práce dále popisuje důvody použití edge zařízení v automatizaci a řídící systém APROL běžící na tomto zařízení.

Po rešerši bylo realizováno řešení projektu. Z dostupných průmyslových počítačů poskytnutých firmu B&R bylo použito APC3100. Zařízení poskytuje dostatek výkonu v kompaktních rozměrech. Na tento hardware byl nainstalován hypervisor umožňující paralelní fungovaní dvou operačních systémů. Hlavním operačním systémem je APROL, zajištující historizaci, vyhodnocení dat a grafické rozhraní mezi operátorem a počítačem. Přes vnitřní virtuální ethernet port APROL komunikuje s druhým OS AutomationRuntime, který slouží jako komunikační most s měničem a nadřazeným systémem.

Pro otestování komunikace poskytla firma ABB testovací soupravu obsahující měnič ASC880 s motorem o výkonu 0,22kW. Pro komunikaci mezi měničem a počítačem byl zvolen protokol Powerlink. Protože měnič dokáže poskytnout v cyklické komunikaci jen omezené množství parametrů, musel být opraven konfigurační soubor XDD. Tato úprava umožnila přenášet v cyklické komunikaci nedůležitější parametry, zbylé hodnoty musí být přeneseny v acyklické části. Pro tuto komunikaci byl vytvořen funkční blok, který v pravidelných intervalech vyčítá hodnoty z definovaných adres.

Následně byla vytvořena logika, která zpracovává obdržené informace, dále vyhodnocuje poruchové stavy založené na překročení mezí a statusových slovech měniče. Pokud dojde k důležité události je o tom operátor informován pomocí grafického prostředí. Dále jsou data ukládaná do Berkeley databáze. Protože je Berkeley DB vhodná pouze pro rychlý zápis, jsou data následně kopírovány pomocí vytvořeného python skriptu do relační databáze mariaDB. Relační databázi následně využívá reportovací systém JasperSoft. V JasperServeru, který je součástí JasperSoftu a APROLU, jsou pak vytvořeny reporty, obsahující informace o vzniklých událostech. Nejprve si operátor může zobrazit chronologický přehled událostí, s výpisem statusových slov měniče a stavu měniče. Z toho přehledu se lze pak „prokliknout“ do detailu, ve kterém se nacházejí grafy s časovou oblastí, ve které událost vznikla. Dále je vytvořen přehled důležitých hodnot zobrazující data za poslední týden. Tento přehled je ve formě dashboardu.

Současně může operátor zobrazit informace i přímo na stanici v procesní grafice. V první vytvořené vizualizaci jsou zobrazeny online hodnoty. Parametry jsou zobrazeny ve dvou formách, číselně a v grafech. Grafy ukazují hodnoty za posledních 5 minut. V této vizualizaci si zároveň může operátor načíst trendy, do kterých se vybere vlastní kombinaci parametrů. Další vizualizace obsahuje obecný stav měniče. Stav je zobrazen pomocí barevného rozdělení a výstražných symbolů. Jednou z posledních funkcí v procesní grafice je poskytnutí aktuálního statusová slova měniče a nastavení mezí pro důležité parametry.

12 SEZNAM POUŽITÝCH ZDROJŮ

[1] MANTLE, Jackles. The 5 Layers of the Automation Pyramid and Manufacturing Operations Management. In: *Syspro.com* [online]. Tustin, Syspro [cit. 2022-05-07]. Dostupné z: https://www.syspro.com/blog/erp-for-manufacturing/the-5-layers-of-the-automation-pyramid-and-manufacturing-operations-management/

[2] Architektura sítě průmyslové automatizace. In: *Časopis ElektroPrůmysl.cz* [online]. Brno: Časopis ElektroPrůmysl.cz, Brno [cit. 2022-05-07]. Dostupné z: https://www.elektroprumysl.cz/automatizace/architektura-site-prumyslove-automatizace

[3] KUMAR, Ravi. What is the five layer automation pyramid?. In: *Medium.com* [online]. [cit. 2022-05-07]. Dostupné z: https://medium.com/world-of-iot/92-what-is-the-five-layer-automation-pyramid-d0ccc1b903c3

[4] What is a Distributed Control System?. In: *Control Station* [online]. [cit. 2022-05-07]. Dostupné z: https://controlstation.com/what-is-a-distributed-control-system/

[5] KOZIOREK, Jiří. *Distribuované systémy řízení*. 1. vyd. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2011. ISBN 978-80-248-2599-1.

[6] RANGER, Steve. What is the IoT? Everything you need to know about the Internet of Things right now: The Internet of Things explained. What the IoT is, and where it's going next. In: *ZDnet* [online]. Red Ventures, 2020 [cit. 2022-05-07]. Dostupné z: https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/

[7] What is IoT?. In: *Oracle* [online]. Redwood Shores: Oracle, c1995–2022 [cit. 2022-05-07]. Dostupné z: https://www.oracle.com/cz/internet-of-things/what-is-iot/

[8] JAKOUBKOVÁ, Veronika. Cesta k nižší latenci: co je edge computing?. In: *MasterDC* [online]. Brno: Master Internet, 2020 [cit. 2022-05-07]. Dostupné z: https://www.master.cz/blog/edge-computing-slibuje-nizsi-latenci/

[9] *Systém APROL zvyšuje výkonnost při zpracování plastů* [online]. 2013, [cit. 2022-05-07]. Dostupné z: http://automa.cz/Aton/FileRepository/pdf_articles/10877.pdf

[10] TM800 – APROL System Concept. In: *B&R: Perfection in Automation* [online]. Eggelsberg: B&R automation [cit. 2022-05-08]. Dostupné z: https://download.br-automation.com/BRP4440000000000000603355/TM800TRE.42-ENG_APROL%20System%20Concept.pdf?px-hash=6442b339d404975d0dd7373fee598ab6&px-time=1651960824

[11] *AUTOMA: Jak na Modbus?* [online]. 2009, [cit. 2021-01-11]. Dostupné z: https://automa.cz/Aton/FileRepository/pdf_articles/38594.pdf

[12] MODBUS APPLICATION PROTOCOL SPECIFICATION. In: *Modbus.org* [online]. Hopkinton: Modbus Organization, 2012, s. 50 [cit. 2021-01-11]. Dostupné z: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[13] Modbus tools: Protocol Description. In: *Modbus tools* [online]. Witte Software, 2022 [cit. 2022-05-09]. Dostupné z: https://www.modbustools.com/modbus.html

[14] *POWERLINK Basics* [online]. Berlin: POWERLINK OFFICE of the EPSG, 2008 [cit. 2021-01-11]. Dostupné z: https://www.ethernet-powerlink.org/uploads/ media/ POWERLINKBasics_brochure_e.pdf

[15] *AUTOMA: Průmyslový Ethernet VIII: Ethernet Powerlink, Profinet* [online]. 2008, [cit. 2021-01-11]. Dostupné z: https://www.automa.cz/Aton/FileRepository/pdf_articles/37288.pdf

[16] *AUTOMA: Ethernet Powerlink – komunikace v reálném čase* [online]. 2001, [cit. 2021-01-11]. Dostupné z: https://automa.cz/cz/casopis-clanky/ ethernet-powerlink-komunikace-v-realnem-case-2001_03_33500_2769/

[17] CANOPEN AND POWERLINK: POWERLINK = CANopen over Ethernet. In: *ETHERNET POWERLINK* [online]. Berlin: EPSG [cit. 2021-01-11]. Dostupné z: https://www.ethernet-powerlink.org/powerlink/technology/canopen-and-powerlink

[18] Berkeley DB Tutorial and Reference Guide, Version 4.1.24. In: *Stanford university* [online]. Stanford: Stanford university, 2002 [cit. 2022-05-08]. Dostupné z: https://web.stanford.edu/class/cs276a/projects/docs/berkeleydb/reftoc.html

[19] Berkeley DB. In: *Database of Databses* [online]. Carnegie Mellon Database Group, 2022 [cit. 2022-05-08]. Dostupné z: https://dbdb.io/db/berkeley-db

[20] Introduction to Berkeley DB XML. In: *Oracle* [online]. Austin: Oracle, 2022 [cit. 2022-05-08]. Dostupné z: https://docs.oracle.com/cd/E17276_01/html/programmer_reference_xml/intro.html

[21] *MariaDB* [online]. In: . Espoo, Finland: mariaDB, 2022 [cit. 2022-05-08]. Dostupné z: https://mariadb.com/

[22] MariaDB. In: *Database of Databses* [online]. Carnegie Mellon Database Group, 2022 [cit. 2022-05-08]. Dostupné z: https://dbdb.io/db/mariadb

[23] *Https://automa.cz/Aton/FileRepository/pdf_articles/41060.pdf* [online]. Automa – časopis pro automatizační techniku, [cit. 2022-05-08]. Dostupné z: https://automa.cz/Aton/FileRepository/pdf_articles/41060.pdf

[24] GAJDŮŠEK, Pavel. Přehled metod řizení asynchronních motorů. In: *Elektro revue, časopis pro elektroniku* [online]. International Science and Engineering Society, 2013 [cit. 2022-05-08]. Dostupné z: http://www.elektrorevue.cz/clanky/05020/index.html

[25] ABB industrial drives: ACS880, single drives 0.55 to 6000 kW. In: *ACS880, single drives 0.55 to 6000 kW* [online]. ABB, 2022 [cit. 2022-05-08]. Dostupné z: https://search.abb.com/library/Download.aspx? DocumentID=3AUA0000098111&LanguageCode=en&DocumentPartId=1&Action=Launch

[26] ABB Ability™ Condition Monitoring for drives. In: *ABB* [online]. ABB, 2020 [cit. 2022-05-08]. Dostupné z: https://library.e.abb.com/public/e88187a1008d4bbf8d6f1e7744cc0f79/ Condition_Monitoring_brochure_3AUA0000189113_RevF_EN_lowres.pdf

[27] ABB Ability™ Digital Powertrain - Condition Monitoring for drives. In: *ABB* [online]. ABB, 2022 [cit. 2022-05-08]. Dostupné z: https://new.abb.com/drives/services/advanced-services/condition-monitoring

[28] *ABB Ability™ Digital Powertrain for efficient, safe and reliable operations* [online]. In: . ABB, 2022 [cit. 2022-05-08]. Dostupné z: https://new.abb.com/news/detail/17846/abb-ability-digital-powertrain-for-efficient-safe-and-reliable-operations

[29] Condition-Based Maintenance service predicts when drive components need replacing. In: *ABB* [online]. Helsinki: ABB [cit. 2022-05-08]. Dostupné z: https://new.abb.com/news/detail/36666/condition-based-maintenance-service-predicts-when-drive-components-need-replacing

[30] *APROL documentation - B&R industrial automation*. 2022.

[31] TIBCO Product Documentation. In: *TIBCO JasperTeports* [online]. Palo Alto (California): TIBCO Software, 2022 [cit. 2022-05-08]. Dostupné z: https://docs.tibco.com/products/tibco-jasperreports-server-8-0-1

[32] *Eclipse Foundation* [online]. Darmstadt: Eclipse Foundation, 2022 [cit. 2022-06-06]. Dostupné z: https://www.eclipse.org/

[33] Industrial IoT. In: *B&R: Perfection in Automation* [online]. Eggelsberg: B&R automation, 2022 [cit. 2022-05-07]. Dostupné z: https://www.br-automation.com/fileadmin/1501609840124-de-html-1.0.jpg

[34] B&R hypervisor. In: *B&R: Perfection in Automation* [online]. Eggelsberg: B&R automation, 2022 [cit. 2022-05-14]. Dostupné z: https://www.br-automation.com/fileadmin/1500242124451-en-html-1.3.jpg

[35] *Automation Studio- B&R Help*. 2022.

[36] HAFNER, Ludwig. *Industrial IoT – Factory Automation Edge Controller*. Egelsberg, 2018.

13 SEZNAM OBRÁZKŮ

14 SEZNAM TABULEK

15 SEZNAM PŘÍLOH

Tabulka parametrů průmyslových PC B&R

Elektronické přílohy:
Technická data ASC 880
Tabulka vyčítaných diagnostických parametrů měniče
Export projektu pro APROL CaeManager
Export knihovny pro APROL CaeManager
Export Sql databáze
Export JasperSoft reportů

PŘÍLOHA 1

Parametry APC 910 / APC 3100 – porovnání

	5PC910.SX02-00	5APC3100.KBU3-000
CPU - Type	Intel Core i7-6820EQIntel i7-7600U	Intel Core i7-6820EQ
CPU - Frequency	2800 Mhz	2800 Mhz
Number of Cores	2	4
Architektura CPU	14nm	14nm
GPU	Intel HD Graphics 530	Intel HD Graphics 620
Memory		
Storage	Depending on CPU	max 256 GB
RAM	DDR4 - max 32GB	DDR4 - max 32GB
Cfast	2x SATA III	Sata III
Interfaces		
USB	4x USB 3.0, 1x USB 2.0 iternal TG	4xUSB3.0, 1xUSB2.0
Eth	2 x RJ45 shielded	2 x RJ45 shielded
Transfer rate	10/100/1000 Mbit/s	10/100/1000 Mbit/s
Max. baud rate	1 Gbit/s	1 Gbit/s
Monitor interface	1 x DisplayPort, 1 x DVI-I	1x DVI-D
Operation conditions		
EN 61131-2	Pollution degree 2	Pollution degree 2
EN 60529	IP20	IP20
Galvanic isolation	yes	yes
Dimension		
width	254.75 mm	248.3mm
height	270 mm	173.5 mm
depth	254.75 mm	248.3 mm
weight	2550 g	1380 g