VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

DISPEČERSKÉ ŘÍZENÍ MĚNÍRNY MHD

DISPATCHING CONTROL OF A PUBLIC TRANSPORT SUBSTATION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. Adam Madzia

VEDOUCÍ PRÁCE

SUPERVISOR

doc. Ing. Zdeněk Bradáč, Ph.D.

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Adam Madzia ***ID:*** 195383

Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Dispečerské řízení měnírny MHD

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte dispečerské tablo pro měnírny MHD. Vytvořte jedno klientské pracoviště, které by ovládalo několik měníren. Toto pracoviště napojte na primární a záložní databázový server pomocí IEC 104, který by uchovával všechny informace o všech měnírnách, ovládání by probíhalo přes tyto servery. Navrhněte strukturu SW a vytvořte programové vybavení pro PLC tak, aby simulovalo měnírnu. Navrhněte a realizujte vizualizaci pro HMI panel, který by představoval lokální řízení měnírny. Dále navrhněte a vytvořte SCADA systém pro dispečink. Funkcionalitu demonstrujte, otestujte a kompletně zdokumentujte.

1. Proveďte internetový průzkum a literární rešerši.
2. Analyzujte a navrhněte strukturu celkového systému řízení.
3. Navrhněte a definujte strukturu komunikačních rozhraní pro centrální dispečink MHD a měnírny.
4. Navrhněte a zrealizujte programové vybavení řídicího systému měnírny, včetně vizualizace.
5. Navrhněte a zrealizujte systém sběru a ukládání dat formou paralelně běžících databázových serverů.
6. Navrhněte strukturu dispečerského SCADA systému a zrealizujte klientské pracoviště (řízení, vizualizace).
7. Zhodnoťte a zdokumentujte dosažené výsledky, demonstrujte funkčnost navrženého systému.

DOPORUČENÁ LITERATURA:

Dennis A. Snow: Plant Engineer's Reference Book, Butterworth-Heinemann Ltd; 2nd Revised edition edition (17 Dec. 2001), ISBN: 978-0750644525

Dle pokynů vedoucího práce.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 18.5.2022

Vedoucí práce: doc. Ing. Zdeněk Bradáč, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato práce se zabývá dispečerským řízením měníren MHD. Je zde proveden průzkum firem, které vytváří dispečerské řízení měníren pro MHD a popis technologie, kterou k tomu využívají. Následně je navržena struktura celkového systému řízení, popis jednotlivých použitých komponentů a návrh struktury komunikačních rozhraní s definicí jednotlivých využitých komunikačních protokolů. V další části je vytvořena simulace měnírny na PLC a vizualizace pro HMI panel představující její lokální řízení. V předposlední části byl proveden návrh a realizace systému ukládání dat pomocí databázového nástroje Firebird a SCADA softwaru Promotic. Závěr práce obsahuje realizaci SCADA aplikace pro dispečerské pracoviště.

KLÍČOVÁ SLOVA

Měnírna MHD, PLC Tecomat, HMI Weintek, IEC 104, dispečerské řízení, databázový server, Promotic, SCADA

ABSTRACT

This thesis deals with the dispatching control of public transport substations. It contains a research about companies that are creating dispatching control of substations for public transport and a description of the technology they use. Following chapters contain the designed structure of the whole control system, a description of the individual components which were used in this thesis and the design of the structure of communication interfaces with the definition of individual communication protocols which were used. In the next section there is a simulation of public transport substation created on the PLC and a visualization for HMI panel which represents its local control system. In the following part, the design and implementation of the data storage system was created by using the Firebird database tool and Promotic SCADA software. The last chapter of the thesis contains creation of the SCADA application for the dispatching control.

KEYWORDS

Public transport substation, PLC Tecomat, HMI Weintek, IEC 104, Dispatching control, Database server, Promotic, SCADA

MADZIA, Adam. *Dispečerské řízení měnírny MHD*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2022, 117 s. Diplomová práce. Vedoucí práce: doc. Ing. Zdeněk Bradáč, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Adam Madzia
VUT ID autora:	195383
Typ práce:	Diplomová práce
Akademický rok:	2021/22
Téma závěrečné práce:	Dispečerské řízení měnírny MHD

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

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu mé diplomové práce panu doc. Ing. Zdeňkovi Bradáčovi, Ph.D. a firmě OHLA ŽS za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

V dnešní době jsou měnírny MHD řízeny individuálně a pro každou z nich existuje vlastní ovládací SCADA aplikace. Vzhledem k tomu, že aktuálně existuje velké množství měníren MHD a z důvodu modernizace a rozšiřování městské hromadné dopravy tento počet postupně narůstá, tak zároveň s tím narůstá také počet jednotlivých SCADA aplikací pro jejich ovládání, přičemž v rámci jednoho města se jednotlivé měnírny po softwarové stránce od sebe téměř neliší. Většinou jsou rozdíly pouze v počtu jednotlivých traťových úseků, takže vytvořené SCADA aplikace jsou si velmi podobné. Z tohoto důvodu vznikla tato diplomová práce, která se zabývá dispečerským řízením měníren MHD. Úkolem této práce je navrhnout centralizované řízení (dispečerské tablo) jednotlivých měníren a vytvořit jednu aplikaci, která by dokázala ovládat více měníren zároveň, aby se omezilo zbytečné vytváření dalších SCADA aplikací pro jednotlivé měnírny a klesla tak zároveň celková režie. Tato práce je vytvářena ve spolupráci s firmou OHLA ŽS, takže pro realizaci dispečerského řízení měnírny MHD budu převážně používat jejich komponenty a software, se kterým pracují, a budu se snažit dodržet jejich technologické postupy pro jednotlivé části.

V rámci této práce bude proveden průzkum firem, které se zabývají dispečerským řízením a popis technologie, kterou k tomu používají. Následně analyzuji a navrhnu strukturu celkového systému řízení. V této práci rovněž provedu návrh struktury komunikačních rozhraní pro centrální dispečink MHD a měnírny a popíši jednotlivé použité komunikační rozhraní a protokoly. Zároveň také bude potřeba vymyslet simulaci daných měníren, jelikož aktuálně není možné testovat navržené řízení na reálných měnírnách a k tomu vytvořím vizualizaci, která bude představovat jejich lokální řízení. Poté bude mým úkolem navrhnout a realizovat systém sběru a ukládání dat formou paralelně běžících databázových serverů, které budou uchovávat všechny informace o všech měnírnách. Na tyto databázové servery se bude následně připojovat klientské pracoviště (dispečink), které bude jednotlivé měnírny skrze tyto databázové servery ovládat. Pro toto klientské pracoviště vytvořím dispečerskou SCADA aplikaci a vymyslím způsob, jakým bude jednotlivé měnírny řídit. Nakonec všechny vytvořené části této práce spojím do jednoho celku a demonstruji jejich funkčnost.

1 Průzkum dispečerských tabel

Dispečerské tabla slouží k centrálnímu operativnímu řízení určitého technologického celku. V tomto případě se jedná o dispečerské řízení měníren MHD. Dispečerské tablo zobrazuje současně přehled několika měníren dohromady (například pro celé město) na rozdíl od standardu, kdy se vytváří řídicí pracoviště pro každou měnírnu zvlášť. Funkce měnírny je detailněji popsána v kapitole 1.1.

V dnešní době vytváří pro dopravní podniky dispečerská tabla několik firem. Každá firma používá rozdílné technologie a způsoby provedení. Níže se nachází jejich základní přehled.

1.1 Měnírny MHD

Měnírny MHD jsou elektrické stanice, které transformují vysoké střídavé napětí na nižší stejnosměrné napětí, které poté slouží k napájení tramvají a trolejbusů. Zpravidla dochází k transformaci 22 kV AC na 600 V DC. Stejnosměrné napětí se poté rozvádí do několika trakčních úseků.

1.2 Technologie firmy ZAT

Firma ZAT provádí návrh, vývoj a výrobu řídicích systémů, které následně využívá v nejrůznějších odvětvích. V rámci řídicích systémů používá vlastní systém SandRA (Safe and Reliable Automation) nebo pracuje se systémy ostatních výrobců (např. Allen Bradley, SAIA, Siemens, Schneider atd.). Pro tvorbu centrálních dispečinků firma používá zahraniční platformu Wonderware, Cimplicity, českou platformu Reliance, Control Web nebo již dříve zmíněný jejich vlastní systém SandRA. [1, 2, 9]

1.2.1 Systém SandRA

Jedná se o moderní řídicí systém, patřící do kategorie distribuovaných řídicích systémů (DCS - Distributed Control System). Díky různým druhům komponent a jejich možnému uspořádání, lze tento systém použít pro nespočet možných aplikací - od řízení malých technologií, jako jsou například regulační stanice, tak až po řízení rozsáhlých technologických celků, jako jsou jaderné elektrárny. [3]

Procesní stanice

Systém SandRA nabízí několik typů procesních stanic. Řada Z200 je zastoupena převážně v energetice v systémech kontroly a řízení. Slouží k provádění algoritmů

přímého řízení, jako jsou například sekvenční úlohy, regulační smyčky, logické funkce atd. Komunikace mezi ostatními zařízeními a nadřazenou operátorskou úrovní probíhá přes firemní protokol PERNET pomocí Ethernetu nebo pomocí jiného standardizovaného komunikačního protokolu (například IEC 60870-5-104). Tato řada je navržena jako modulární, díky čemuž lze u těchto stanic do určité míry měnit počet vstupů a výstupů, jejich výkon, komunikační kanály atd. V případě, kdy je potřeba řešit náročnější úlohy, lze použít multiprocesorový provoz. [4]

Řada Z210 se vyrábí v kompaktním provedení. Díky svým malým rozměrům se umisťuje do rozvaděčů a na DIN lištu. Má omezený počet vstupů a výstupů řádově do 100 I/O, kvůli čemuž je tato řada vhodná spíše pro řízení menších technologií. Slouží například k řízení malých vodních elektráren a výměníkových stanic, monitoringu produktovodů atd. Pro připojení technologických signálů používá stanici vzdálených vstupů a výstupů X20, která je připojena pomocí Ethernetu. Stanice pracují na operačním systému Linux. [5]

Řada Z100 slouží k realizaci speciálních bezpečnostních funkcí, patřících do nejvyšších bezpečnostních tříd A a B podle ČSN EN 61226. Jsou zkomponovány jako modulární systém. Používají se s řídicím systémem SandRA k řízení jaderných elektráren. [6]

Stanice vzdálených vstupů a výstupů

Slouží k připojení vstupů z různých typů senzorů a výstupních signálů, určených pro ovládání pomocí protokolů typu fieldbus, jako je například Modbus TCP, Profibus DP, Powerlink atd. Pomocí těchto stanic lze připojit také stanice jiných výrobců tak, aby se kompletně integrovaly do řídicího systému. Jsou také podporovány v nástroji Pertinax. [7]

Speciální přístroje pro energetiku

Mezi tyto přístroje se řadí kompaktní regulátory buzení synchronního generátoru pro různé budicí systémy, číslicové a synchronizační zařízení s funkcí fázovače, funkcí kontrolního obvodu a s automatickým připojením generátoru k síti a jeho regulací. Patří zde také řídicí systém natáčeného zařízení se synchronizací, speciálně vyvinutý pro společnost Doosan Škoda Power. [8]

Vizualizace a dispečerská pracoviště

Poskytují dohled nad celou technologií. Lze pomocí nich řídit technologické celky podle daných postupů, nastavovat různé parametry a povely a zobrazovat stav dané technologie tak, aby byla přehledná pro daného operačního pracovníka. Mohou se skládat z několika dispečerských pracovišť, datových úložišť a serveru pro zajišťování

optimálního chodu podniku, na kterém běží systémy MES a MRP nebo ERP. Systém SandRA poskytuje tři základní koncepty pro dispečink. [9]

První koncept je vhodný pro menší lokální systémy. Vše se vykonává přímo na operátorské stanici, včetně komunikace, zpracování dat, řízení a grafické vizualizace technologie. [9]

Druhý koncept je vhodný pro místa, kde je potřeba mít několik dispečinků a pro místa s obsáhlejšími systémy. K poskytování dat z řídicích stanic jednotlivým dispečinkům se zde používají speciální servery. Na dispečerských stanicích se poté vykonává již jen vlastní zpracování dat, jednotlivé řídicí úlohy a grafická vizualizace technologie. [9]

Třetí koncept je určen pro ty nejrozsáhlejší systémy. Veškeré operace s daty a grafikou probíhají na redundantních aplikačních a výpočetních serverech. Dispečerské stanice poté slouží jen k zobrazení výsledných dat. [9]

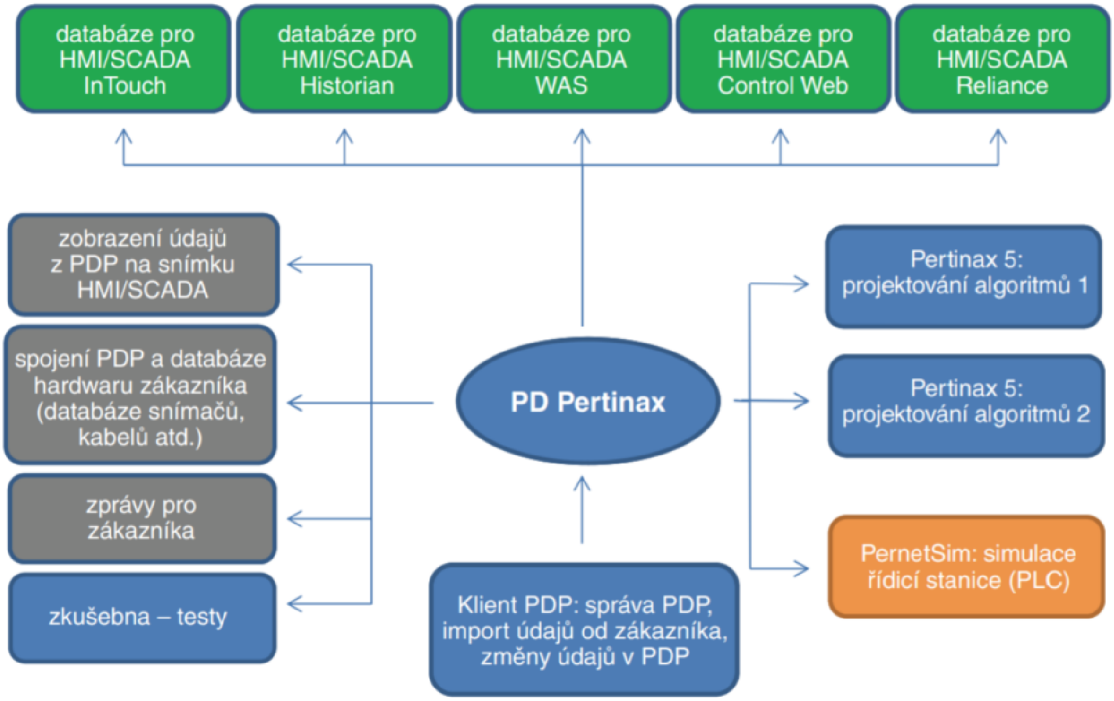
Projektová databáze Pertinax

Tato databáze byla vytvořena na Microsoft SQL 2008 R2 serveru. Slouží pro správu dat v celém řídicím systému a zároveň jsou data z této databáze použita pro archivaci a ve vizualizaci na platformě Wonderware nebo Pertinax. Díky využití této databáze nedochází k neúplnosti a nekonzistenci dat a zároveň je zaručena dostupnost těchto dat. Systém správy dat v celém řídicím systému je jednotný a provádí se pouze z programového prostředku Klient PDP. Při použití této databáze musí všechny vstupy a výstupy jít skrze ni, jinak na ně nebude brán zřetel. Pro využití například databáze snímačů od jiné firmy je zde implementována možnost propojení s externí databází založené na SQL nebo Microsoft Access. Na obrázku 1.1 je zobrazen způsob implementace databáze Pertinax do řídicího systému. [10]

Pertinax 6

Tento software je určen k programování a kompletní konfiguraci procesních stanic. Programování probíhá pomocí funkčních bloků a portů, které reprezentují vstupy a výstupy. Podporuje projektové databáze i s následnou vizualizací. Je v něm obsažena podpora redundance hardwaru. Umožňuje parametrizovat stanice a měnit vlastnosti programu na dálku během jejich běhu, se zachováním všech hodnot vnitřních stavů a výstupů, aniž by bylo potřeba restartovat stanici. Pro spuštění aplikace je vyžadován hardwarový HASP klíč. [12]

Kromě firemního protokolu Pernet dokáže komunikovat přes klasické standardy, jako je například Modbus RTU, protokoly používané v energetice typu M-bus, ČSN EN 60870-5-104 atd., a podporuje také komunikaci s chytrými čidly a akčními členy, například pomocí HART, Profibus-DP atd. [12]



Obr. 1.1: Způsob implementace PDP do řídicího systému [11]

Software pracuje ve třech režimech. V režimu Editor se přímo vytváří řídicí funkce pro procesní stanici. V režimu Simulátor lze zkoušet na počítači vytvořené funkce se simulovanými signály, aniž by byla připojena fyzická stanice. Poslední režim je Analyzátor. Přes tento režim již probíhá nahrávání kódu do stanice, čtení dat ze stanice, správa stanice a její diagnostika. [12]

PertinaxDiag

Jedná se o diagnostický nástroj, který umožňuje uživateli získat kompletní přehled o stavu řídicího systému. Zaznamenává různé stavy a události systému, se kterými lze poté skrze tento nástroj dále pracovat. Data z jednotlivých stanic vyčítá pomocí Ethernetu, aniž by musel do těchto stanic cokoliv zapisovat. [13]

1.2.2 Platforma Wonderware

Wonderware System Platform vznikla na technologické infrastruktuře ArchestrA. Infrastruktura ArchestrA umožňuje intuitivní vývoj modulárních průmyslových aplikací a dle potřeby i jejich případnou změnu. Platforma je určena pro návrh a provoz informačních a automatizačních řešení pro: [14]

- supervizní a vizualizační HMI aplikace;

- Geo-SCADA (geograficky rozlehlé aplikace) a SCADA aplikace;
- aplikace typu Production and Performance Management (řízení výroby a analýza její výkonnosti), resp. MES (výrobní informační systém).

Tato platforma se skládá z průmyslového aplikačního serveru, procesní historizační databáze, informačního portálu a I/O nebo DA serveru pro komunikaci s řídicími zařízeními (například PLC, I/O atd.). Všechny tyto součásti splňují požadavky na provoz v reálném čase. [14]

Pro dlouhodobou funkčnost procesů, zlepšujících provoz a výkonnost výroby, obsahuje Wonderware System Platform sadu klíčových průmyslových služeb, dále služeb pro propojení se zařízeními a dalšími aplikacemi, služeb pro vývoj aplikací, správu a řízení informací a služeb pro správu a rozšíření systému. Pomocí jednotlivých komponent průmyslových aplikačních služeb se vytvoří objekt s danou funkčností, ze kterého následně vznikne šablona a může se opakovaně nasazovat na potřebná místa. Programování se provádí objektově skrze Wonderware Development Studio. Tvorba vizualizace pro HMI probíhá přes Wonderware InTouch. [14]

Díky funkcím pro komunikaci v reálném čase, vícestupňovému zabezpečení, rozsáhlé správě alarmů a výkonné distribuované síťové architektuře nemá platforma žádné nároky na speciální servery nebo fault-tolerant hardware, ale zároveň přímo nabízí vícestupňové zálohování. Proto lze pomocí Wonderware System Platform implementovat do průmyslových aplikací i běžné počítače a operační systémy. [14]

Pro spojení řídicích prvků a dalších systémů, jako jsou například relační databáze, vizualizace, systémy ERP, řídicí systémy atd., lze použít různé komunikační a propojovací systémy. Díky tomu lze sjednotit systémy různých výrobců a typů do jednoho celku. [14]

1.2.3 Platforma CIMPLICITY

CIMPLICITY je HMI a SCADA platforma vyvíjená firmou GE Digital. Lze pomocí ní řešit vysoce výkonnou vizualizaci a řízení typu klient-server jak pro jednoduché stroje, tak i pro velké technologické celky. Od verze 11.1 je integrováno Proficy Historian a Proficy Operations Hub, což umožňuje centralizované, vzdálené a mobilní operace pro lepší přístup k informacím a řízení, které zvyšují účinnost a zkracují dobu potřebnou pro opravu při poruše. Pro snížení rizika kybernetického útoku jsou nabízeny pomocí CIMPLICITY Global Discovery serveru komunikační protokoly, založené na certifikátech. Pro větší spolehlivost není programování založeno na jazyku Java, ale je použit firmou vyvinutý programovací jazyk. Platforma také používá OPC UA servery, například pro zaznamenávání a zpracování alarmů. [15]

Pro rychlejší nasazení do provozu má CIMPLICITY vylepšenou databázi, jednoduchou opakovatelnost a rozšířené možnosti modelování. Programování je objektově

orientované. K urychlení tvorby této databáze lze používat třídy, objekty a také šablony, které lze poté distribuovat mezi ostatní klienty. [15]

1.2.4 Systém Reliance

Jedná se o český HMI a SCADA systém, sloužící pro monitoring a ovládání technologií v průmyslu a pro automatizaci budov. Nabízí vysokou robustnost, bezpečnost a škálovatelnost jak pro menší aplikace, tak i pro aplikace velkého rozsahu. Používá se například pro dispečinky, plynárenství, dopravní zabezpečovací systémy, vizualizace různých systémů a mnoho dalšího. [16]

Zajišťuje spolehlivost díky vestavěné redundanci datových toků. Minimalizuje výpadky pomocí včasného varování obsluhy a pokud k výpadku i přesto dojde, tak umožňuje následně daný výpadek analyzovat pomocí funkce Postmort. Tato funkce zaznamenává průběh sledovaného procesu, se kterým lze poté pracovat. K vizualizaci a datům lze přistupovat 24 hodin denně pomocí zabezpečeného přístupu, například přes mobil nebo tablet. Vývoj probíhá v prostředí RAD. Pro základní funkce není potřeba psát žádný zdrojový kód. Systém se skládá z několika částí popsaných níže. [16, 17]

Vývojové prostředí

Vytvoření a úprava aplikace (vizualizace) probíhá skrze vývojové prostředí Reliance 4 Design. Pro lokální aplikace s jedním počítačem slouží verze Desktop. Pro síťové aplikace s přístupem přes web, mobil nebo tablet je určena verze Enterprise. [18]

Ve verzi Desktop může pracovat pouze jeden počítač s libovolným počtem PLC, telemetrických stanic nebo jiným I/O hardwarem. Přes tuto verzi nelze vytvářet vizualizace pro tenké klienty a ani pro síťové aplikace. [18]

Verze Enterprise je verze Desktop, rozšířená o možnost vývoje síťových aplikací a práci s tenkými klienty. V této verzi lze také mít libovolný počet počítačů (dispečerských pracovišť), které pracují s libovolným počtem stanic (PLC, telemetrická stanice atd.). [18]

Reliance 4 Combi Package

Jedná se o balíček licencí pro vývojové prostředí Reliance Design a příslušné runtime moduly. Licence pro obojí je uložena na jednom klíči. Existují dvě verze balíčku - Desktop a Enterprise. Výhoda tohoto balíčku spočívá v tom, že licenční klíč je přímo u uživatele a vývojové prostředí lze spouštět přímo u uživatele pomocí vzdálené plochy odkudkoliv. Další výhoda je při jednorázové aplikaci, aby programátor nemusel

kupovat pro sebe zvlášť licenci, tak ji po vytvoření aplikace přenechá koncovému uživateli. [19]

Komunikační drivery

Pomocí komunikačních ovladačů vizualizace získává data z technologických stanic a odesílá povely zpět na stanice za pomoci stanicí požadovaného komunikačního protokolu. Pro tyto ovladače je potřeba zakoupit speciální licenci v závislosti na použitém zařízení, pro které je ovladač určen. [20]

Runtime moduly

Starají se o běh aplikace na počítači uživatele. Pomocí těchto modulů může uživatel sledovat a ovládat danou technologii. Runtime moduly a tencí klienti získávají data ze serverových modulů. Runtime moduly používají čtyři programy. [21]

Pro ovládání daného procesu skrze vizualizaci je potřeba mít program Reliance 4 Control, který je určen pro řídicí pracoviště. Pomocí tohoto programu lze například pracovat s aktuálními daty a sledovat jejich historii pomocí grafů, zobrazit všechny nastalé alarmy a diagnostikovat danou aplikaci. [21]

Pokud uživateli vystačuje do aplikace pouze nahlížet, tak je pro něj určen program Reliance 4 View. Tento program umožňuje vše, co Reliance 4 Control, kromě ovládání daného procesu. [21]

Jako datový server pro runtime moduly klientů a tenké klienty slouží program Reliance 4 Server, který má v sobě zabudovaný webový server. Poskytuje jednotlivá data a stará se o vykonávání všech povelů a žádostí. Nemá v sobě vizualizaci, takže funguje pouze jako datový server. Je spuštěn jako služba Windows. [21]

V případě požadavku mít vše na jednom místě existuje program Reliance 4 Control Server. Pro tento program je potřeba mít výkonnější počítač, jelikož v sobě obsahuje kromě společných funkcí runtime modulů také všechny funkce programů Reliance 4 Control a Reliance 4 Server a stará se tak o veškerou obsluhu. [21]

Tencí klienti

Umožňují vzdálený přístup k vizualizaci přes webový prohlížeč, mobil nebo tablet skrze síť internet. V nabídce jsou aktuálně dvě verze tenkých klientů. Název tenký klient se používá z důvodu menší nabídky funkcí oproti standardním klientům. [22]

Pro mobily a tablety je určen SCADA a HMI klient Reliance 4 Smart Client, který je optimalizován pro dotykové ovládání a zobrazuje vizualizaci pomocí webových stránek. Vytvoření vizualizace probíhá na datových serverech ve formátu HTML5. O uživatelské rozhraní se stará javascriptový framework jQuery Mobile. Oproti klasickému klientovi má tento klient omezené funkce runtime modulů. [22]

Další varianta je Reliance 4 Web Client, který umožňuje vzdálený přístup k vizualizaci pomocí internetu. Funguje na jazyku Java, takže jej lze použít na libovolném operačním systému a v libovolném internetovém prohlížeči. Používá datové servery Reliance 4 Server nebo Reliance 4 Control Server. Oproti klasickému klientovi má méně funkcí pro runtime moduly. [22]

OPC servery

Pro spojení softwarové aplikace s řídicím hardwarem je použita technologie, navržená přímo pro tento účel - OPC (OLE for Process Control). Jedná se o otevřený standard, který neustále vylepšují stovky různých firem a implementují do něj stále nové technologie. Má na starosti neustálé získávání dat z připojeného hardwaru. Pro spojení lze vybrat libovolný software a hardware, fungující na standardu OPC, aniž by bylo potřeba řešit komunikační drivery. Na výběr je mezi OPC klientem a OPC serverem. [23]

SCADA a HMI nebo MMI aplikace fungují jako OPC klient. Zpracovávají data získána z OPC serveru. Server OPC poskytuje klientům data, která vyčítá z jednotlivých hardwarových zařízení, pro něž je specificky vytvořen. Díky komunikaci skrze pevně definované rozhraní může kterýkoliv klient komunikovat s libovolným serverem, aniž by musel řešit, pro které zařízení byl server určen. Nabízeny jsou dva typy OPC serverů v závislosti na tom, zda uživatel chce, aby serverový modul přistupoval k parametrům vizualizace skrze standardní rozhraní OPC, nebo skrze rozhraní OPC UA. [23]

1.2.5 Systém Control Web

Programy vytvářené systémem Control Web mohou fungovat jako řídicí systém, datový server s webovými klienty, vizualizace nebo SCADA aplikace. Lze pomocí něj také propojit výrobní technologie s informačním systémem podniku nebo simulovat a modelovat různé procesy. [24]

Tvorba aplikace se provádí pomocí spojování různých komponent (virtuálních přístrojů). Vlastnosti těchto virtuálních přístrojů lze nastavit pomocí jejich parametrů. Pokud nastavení parametrů není dostatečné pro požadovanou funkčnost, lze požadované chování napsat pomocí skriptů. Vyvíjet a testovat aplikace lze pomocí Control Webu zdarma, ale pro jejich dlouhodobý běh je nutné zakoupit licenci. [24]

Díky používaní ovladačů lze připojit k systému jakýkoliv hardware. Část ovladačů je již součástí systému, některé je potřeba doinstalovat. K dispozici je například

ovladač pro Modbus, OPC, OPC UA, Simatic, různé typy PLC, komunikaci po sériové lince a mnoho dalších. V případě potřeby Control Web také podporuje .NET a umožňuje uživateli vytvořit si vlastní komunikační ovladač. [24]

Control Web také umožňuje pomocí svého grafického editoru vytvořit si vlastní statické a dynamické obrázky. Má v sobě rovněž integrovaný nástroj pro tvorbu aplikací strojového vidění VisionLab. Kromě toho spolupracuje s různými databázovými systémy a má v sobě zabudovaný webový server s podporou HTTP a HTTPS. [24]

1.3 Technologie firmy APEL

Firma APEL nabízí pro dispečerské řízení své vlastní řešení - mozaikovou stavebnici dispečerského panelu. Tato stavebnice se skládá z programových, elektrických a mechanických modulů. Lze ji sestavit do libovolné požadované struktury a jednoduše upravovat i v provozu. Zobrazování probíhá pomocí LED diod a díky své malé konstrukci lze vytvářet panely menších rozměrů. [25]

Mozaika se skládá z roštu a krytky. Do mozaiky je možné vkládat měřicí, regulační a signalizační přístroje od různých výrobců, které svými rozměry splňují násobky rozměru krytky (24x24 mm). Pro zobrazení požadované technologie na krytce se používá gravírování, sítotisk nebo barevné samolepicí fólie. Stav technologie je poté vyjádřen pomocí různých LED diod. LED diody mohou být umístěny samostatně nebo spojené tak, aby tvořily s průhlednou krytkou podsvícený prvek. Do krytky lze také vložit spínací prvky, jako jsou například ovladače nebo tlačítka, analogové a digitální měřicí přístroje a regulátory nebo 7 a 16segmentové displeje. [25]

1.3.1 Řízení mozaiky

V závislosti na velikosti technologie je na výběr z tří typů řízení mozaiky: [25]

- ovládání pomocí řídicího počítače skrze desku paměti obrazu prostředí PT;
- ovládání pomocí modulu sériové komunikace PSK;
- přímé připojení mozaiky k technologii skrze speciální desky elektroniky.

Řídicím počítačem se ovládají převážně systémy s velkým počtem signálů (typicky nad 1000 signálů). K tomuto počítači se přidávají do PCI sběrnice speciální desky paměti obrazu prostředí PT. Použité monitory a počítače jsou v průmyslovém provedení. [25]

Pro o něco menší aplikace s menším počtem signálů (typově do 1024 dvoubitových informací) je pro ovládání mozaiky určen modul sériové komunikace PSK. Modul má na starosti řízení podřízených vstupních a výstupních modulů SSK a VSK.

Skrze moduly SSK dokáže spínat relé a řídit signalizační prvky. Skrze moduly VSK dochází k vyhodnocování prvků z řídicího panelu nebo technologického procesu. [25]

K řízení malých technologií s malým počtem signálů (typově do 300 signálů) se používá přímé připojení dispečerské mozaiky. Pro toto připojení musí být v technologii nebo v nadřazeném systému k dispozici paralelní výstupy. Přímé připojení umožňuje vytvořit malé a levné dispečerské řízení. K připojení mozaiky k technologii je použita deska pro úpravu vstupů D-IN nebo deska optického oddělení OP-IZ200. Ke zjišťování poruch je použita deska POR 32. K připevnění LED diod na mozaice slouží desky konektorů K4 - K16. [25]

1.3.2 Hardware pro počítačem řízený panel

Deska paměti obrazu prostředí PT4

Je určena pro 32bitovou PCI sběrnici. K jednomu počítači lze připojit více desek. Může ovládat až 65 536 dvoubarevných LED diod a každé z nich nastavovat dvojí intenzitu svícení. Umožňuje řídit 3 páry kmitačů k modulaci světla LED diod. K této desce lze napojit až 8 desek MX1. [26]

Deska multiplexu MX1

Slouží k přesměrování dat z desky PT4 na zobrazovací moduly na dispečerském panelu. K této desce lze napojit až 32 zobrazovacích modulů. Rovněž umožňuje řízení dvou akustických návěstí. Na desce je umístěna kontrolní LED dioda, která signalizuje správnou funkčnost desky. [26]

Deska spínačů JS2

Dohromady s deskou JK1V nebo JK2 vytváří zobrazovací modul dispečerského panelu. Řízení probíhá pomocí desky MX1. Dokáže řídit svícení 256 dvoubarevných LED diod umístěných na deskách JK1V nebo JK2. [26]

Desky konektorů JK1V a JK2

Montují se na zadní stranu roštu dispečerského panelu. Mají na sobě 256 třípólových konektorů k připojení jednobarevných nebo dvoubarevných LED diod rozmístěných do matice 16x16. LED diody jsou napájeny přes spínače JS2. Na desku JK2 lze také napojit 7segmentový displej pomocí desek 7SEG1 a 7SEG20. Displej zabírá místo pro 4 dvoubarevné LED diody. [26]

Desky zobrazovačů 7SEG1 a 7SEG20

Deska 7SEG20 umožňuje připojení jednoho 7segmentového displeje k desce JK2, deska 7SEG1 umožňuje připojení dvou 7segmentových displejů k desce JK2. Displej se vkládá do konektorů na těchto deskách. Displej může být zelené, žluté nebo červené barvy. V případě potřeby delšího displeje je možné umístit více desek vedle sebe. Napájení displejů probíhá pomocí spínačů JS2. [26]

Deska akustické signalizace HOUK

Slouží k ovládání dvou nezávislých akustických návěstí. Pracuje s deskou MX1, ze které je zároveň přivedeno napětí a data. Pomocí WAGO svorek lze připojit a ovládat dvě piezosirény. Jsou zde rovněž kontakty dvou relé pro buzení výkonnějších akustických měničů. [26]

Deska optických oddělovačů OPC-24

Používá se u získávání dvouhodnotových signálů z technologie. Dokáže opticky oddělit 24 signálů a upravit jejich úroveň pro zpracování pomocí obvodů TTL. Signály mohou být na společném pólu nebo být galvanicky oddělené. Na desce jsou umístěny červené LED diody signalizující stav všech vstupů. Výstupy jsou vyvedené na 50pólový konektor pro plochý kabel. [26]

TMP deska přijímačů DMS a DO

Používají se pro příjem telemechanizačních značek pro maximálně 16 zařízení typu D0-100, DMS nebo DMS-I. Přijímače lze díky své nezávislosti programovat samostatně. Deska se připojuje pomocí osmibitové ISA sběrnice. Do jednoho počítače lze vložit více TMP desek. Přenosovou rychlost lze nastavit na 50, 100 nebo 200 Bd. Délka DMS a DMS-I značky je 40, 80 nebo 128 bitů. [26]

Deska přijímačů a vysílačů TMB

Je určena k příjmu a generování telemechanizačních značek pro zařízení typu D0-100, DMS a DMS-I. Má v sobě zabudované dva přijímače a čtyři vysílače, které jsou na sobě nezávislé a které lze samostatně programovat. Deska se připojuje pomocí osmibitové ISA sběrnice. Do jednoho počítače lze vložit více TMB desek. Vstupy a výstupy pracují s proudovou smyčkou 20 mA. [26]

1.3.3 Moduly pro sériovou komunikaci

Základní modul PSK

Pomocí sériového kanálu dokáže komunikovat s okolním hardwarem, jako je například počítač nebo PLC. Komunikační kanál může jít skrze RS-232, RS-422 nebo RS-485 a je galvanicky oddělený. Modul má konektor CAN 9 s kanálem RS-232 pro jeho konfiguraci a servis. K tomuto modulu lze připojit pomocí plochého 10pólového kabelu moduly typu SSK a VSK. Obsahuje rovněž tři vstupy pro test signalizačních prvků a pro realizaci poruchové signalizace, ale lze je využít i pro jiné účely. Pro ovládání akustického návěstí je zde jeden výstup, další je pro externí signalizaci poruchy komunikace a dva výstupy slouží pro řízený stabilizátor. [27]

Vstupní moduly VSK

Slouží pro zpracování galvanicky oddělených kontaktních vstupů na svorkách. Podle typu může modul zpracovat 8 nezávislých vstupů nebo 16 vstupů se společným vodičem. Podle vstupního napětí a polarity společného vodiče se dále rozlišují jednotlivé typy vstupních modulů. [27]

Výstupní moduly SSK

V závislosti na typu modulu lze pomocí nich budit 64 jednobarevných signalizačních LED diod nebo 32 dvoubarevných LED diod. Pomocí nich se také obstarává buzení 7segmentových displejů. Umožňují také spínaní 16 zátěží ke kladnému nebo zápornému pólu. [27]

1.3.4 Desky pro přímé připojení technologických procesů

Deska vstupů D-IN

Používá se k úpravě signálů z technologického procesu, aby je bylo možné zobrazit pomocí jednobarevných nebo dvoubarevných LED diod. Má v sobě rovněž zabudované obvody pro testování LED diod. Vstupy se připojují skrze WAGO svorky. Na výstupu je 12 třípolových konektorů pro připojení dvoubarevných LED diod. [28]

Deska poruchové signalizace POR32

Zachytává poruchová hlášení z technologie a zobrazuje je pomocí LED diod. Má 32 opticky oddělených vstupů a 32 výstupů pro LED diody, mikrořadič, výstup ovládající houkačku a spínací stabilizátor pro úpravu napájení. [28]

Desky konektorů

Různé typy desek umožňují na jejich přední část připojit různé typy a počty LED diod. Zadní strana těchto desek slouží například k propojení s deskami POR32 nebo D-IN. Umisťují se na zadní stranu mozaikového roštu. [28]

Deska galvanického oddělení RE 12

Deska má 12 oddělovacích relé. Lze přidat ještě 13. relé jako kontrolu napájení ostatních relé, které by hlásilo jeho ztrátu rozsvícením LED diody. Cívky relé jsou přemostěny skrze odrušovací diodu. Jeden vývod cívky je poté zapojen na společnou svorku (-) a druhý vývod je vyveden na svorkovnici. Zapínací kontakty jednotlivých relé jsou samostatně vyvedeny na dvojice svorek. [28]

Deska optického oddělení O-IN 16

Používá se k optickému oddělení vstupů dvoustavové signalizace skrze LED diody. Vstupy se aktivují záporným pólem, kladný pól je pro všechny vstupy společný. Obsahuje rovněž nezávislý vstup pro testování LED diod. Výstupy jsou kompatibilní s deskou vstupů D-IN. Lze ji také využívat ve spojení s deskami konektorů. [28]

OP-IZ200

Používá se pro převod stejnosměrného vstupního napětí o hodnotě 220 V na 24 V. Výstup je od vstupu galvanicky oddělený. Přítomnost vstupního napětí je indikována pomocí LED diody. [28]

1.3.5 Řídicí software

K řízení technologie a mozaiky využívá firma APEL svůj vlastní řídicí a informační systém QIRS. Pro řízení lze také využít drivery a knihovny pro připojení z jiných systémů. [29]

Základ programového vybavení tvoří síťový operační systém QNX s grafickou nástavbou QNX Photon. Vnější vzhled a ovládání je podobné systému Microsoft Windows. Pro tvorbu vizualizace se používá objektově orientovaná grafika s vlastním grafickým editorem. Do grafiky lze importovat grafické soubory různých typů. Má v sobě rovněž zabudovaný editor databáze a programy určené k testování hardwarové mozaiky. Lze skrze něj ovládat mozaiku z jiného systému. Dokáže komunikovat s okolím, kromě sériových a paralelních kanálů, také skrze počítačové sítě. [29]

Tento software má v sobě zabudované velké množství funkcí sloužících k práci s dispečinkem. Může například přímo ovládat vlastnosti dispečerské mozaiky, jako

je ovládaní jasu, barev a kmitání indikátorů a displejů nebo zobrazovat stav technologie. Umožňuje monitoring a zpracování alarmů a různých typů událostí, které lze následně vytisknout. Dokáže také evidovat práce na zařízení (B příkazy). Pro práci s veličinami obsahuje sadu různých funkcí, včetně speciálních funkcí pro jejich import a export. [29]

Konfigurace dispečerské mozaiky probíhá skrze sériový kanál. Konfigurační software obsahuje tabulku s adresací jednotlivých signálů, tabulku stavů a nastavení řízení. Všechny tabulky jsou uloženy v databázovém formátu dBaseIV a komunikace s tímto formátem probíhá přes jazyk SQL s přístupem skrze rozhraní ODBC. Během jeho vlastní instalace se nainstaluje také vlastní driver, zajišťující správu dat v tomto formátu. [29]

Pokud se uživatel potřebuje připojit k dispečerské mozaice z jiného systému, lze použít speciálně k tomu vytvořený komunikační protokol APEL-XL. Pro řízení mozaiky skrze prostředí Control Web a Control Panel byly vytvořeny DLL knihovny, ke kterým byly doplněny funkce pro ovládaní různých prvků a vlastností mozaiky (například intenzita svícení, testování prvků, sledování stavů atd.). Kromě těchto systémů lze propojit dispečerskou mozaiku se systémem InTouch, CIMPLICITY, SCS100 ABB, SAT 250 atd. [29]

1.4 Technologie firmy OHLA ŽS

Firma OHLA ŽS provádí svůj vlastní návrh, konstrukci a následnou montáž rozvaděčů do měníren. Tyto rozvaděče osazuje průmyslovými automaty firmy Teco, převážně řadou Foxtrot (například PLC Tecomat Foxtrot CP-1003 popsané v kapitole 2.1) pro menší aplikace a TC700 pro větší aplikace. Pro jejich programování využívá nástroj Mosaic. Kromě toho každý napáječ a usměrňovač obsahuje HMI panel značky Weintek (například typ MT6070iE popsaný v kapitole 2.2), který programují v softwaru EasyBuilder Pro. Pro klientské pracoviště se používají průmyslové embedded počítače od firmy Advantech. Na těchto počítačích běží SCADA aplikace pro ovládání měnírny, vytvořená v softwaru Promotic.

1.4.1 PLC Tecomat TC700

Jedná se o modulární řídicí systém pro střední a velké aplikace v oblasti průmyslové automatizace, technického zařízení budov a dopravy. Má zvýšenou provozní spolehlivost. Lze na něm aplikovat redundanci v několika úrovních. Obsahuje 32bitový RISC procesor. V základním provedení lze použít dva Ethernetové porty a 10 sériových kanálů. K PLC lze rovněž připojit komunikační modul s FSK modemem.

Decentralizované periferie mohou být spojeny na vzdálenost 300 m přes metalické propojení nebo 1 700 m pomocí optiky. Moduly lze vyměnit za provozu. [30]

1.4.2 Nástroj Mosaic

Mosaic je vývojový nástroj, sloužící k vytvoření aplikací pro PLC Tecomat. Funguje od operačního systému Windows 7. Dodržuje normu IEC 61131-3. Programování je možné pomocí blokových diagramů (FDB), reléových schémat (LD), pokročilého grafického programování (CFC), instrukčního jazyku (IL) a strukturovaného textu (ST). Má v sobě zabudovaný simulátor operátorských panelů a PLC. Mezi jeho debugovací nástroje patří kromě standardního krokování, trasování a přerušovacích bodů také PIDMaker, PanelMaker, GrafMaker a GrafPanelMaker. Pro vizualizaci má vestavěný WebMaker pro tvorbu webu a online komunikaci se SCADA Reliance. [31]

1.4.3 Software EasyBuilder Pro

Slouží k vytváření vizualizace pro HMI panely Weintek řady eMT3000, IE, mTV, cMT, XE a iER. Lze v něm vytvářet aplikace pro různá průmyslová prostředí. Pro usnadnění jejich tvorby nabízí příslušné grafické knihovny. Nabízí různé nástroje, jako jsou například Easy Simulator, Database Editor, cMT Viewer, Recipe Editor a mnoho dalších. Pro spojení s různými zařízeními, jako jsou PLC, servopohony, měniče atd. podporuje přes 300 komunikačních ovladačů. [32]

1.4.4 Software Promotic

Promotic je SCADA software, díky kterému lze vytvářet aplikace pro monitoring a řízení různých technologických procesů například v oblastech energetiky, chemického a potravinářského průmyslu, tepelného hospodářství a v mnoha dalších odvětvích. Funguje na operačních systémech Windows. Umožňuje vytvářet jak malé aplikace, tak také aplikace velkého rozsahu. Programování probíhá pomocí jazyků javascript nebo VBscript. Tyto jazyky se kromě událostního programování využívají také k přistupování do metod a vlastností jednotlivých objektů Promoticu nebo pro přístup k jiným softwarovým aplikacím. Pro diagnostiku a ladění aplikace se využívá takzvaný INFO systém. K prohlížení alarmů a událostí (eventů) je zde zabudován speciální systém alarmů a událostí a pro zobrazení historie určených proměnných je použit systém trendů. [33]

Pro vytváření grafických obrazů se používá editor grafiky, který obsahuje předdefinované grafické objekty, nebo si může programátor vytvořit nebo importovat vlastní. Promotic také umožňuje naprogramovat vlastnosti všech grafických objektů.

Obrazy aplikace lze prohlížet přes internetové prohlížeče díky automatickému generování dynamických HTML stránek a rovněž přes ně aplikaci ovládat. Přenos dat zde funguje přes HTTP nebo HTTPS protokol a je zajištěn Promotic Web serverem, který obsahuje také konfiguraci uživatelů a oprávnění. [33]

Díky své otevřenosti lze tento systém integrovat s dalším softwarem pomocí ODBC, ADO, XML, ActiveX, OPC, DDE, TCP/IP atd. Aplikace přistupuje k datům z externích zdrojů, jako je PLC, databázový server, vstupní PC karty a další. Z tohoto důvodu Promotic obsahuje kromě ovladačů pro standardní komunikační protokoly také velké množství vlastních komunikačních ovladačů pro různé značky PLC (Siemens, SAIA, Omron, Mitsubishi atd.), které lze navíc parametrizovat podle svých potřeb a disponuje také přenosy v sítích GSM a rádiových sítích. Propojení s komunikačními servery probíhá přes OPC, ActiveX a DDE. Pro webové aplikace se používají komunikační rozhraní s TCP/IP, HTTP, DCOM a XML. [33]

Aby uživatel nemohl vstoupit do vývojového prostředí aplikace, používá se zaheslování projektu. Pro zamezení nežádoucích zásahů uživatele do operačního systému je umožněno vyblokování kritických kláves systému. Běh aplikace je hlídán softwarovým watchdogem a v rámci aplikace lze nastavit omezení přístupu skrze nastavení uživatelů a jejich oprávnění. [33]

2 Návrh struktury celkového systému řízení

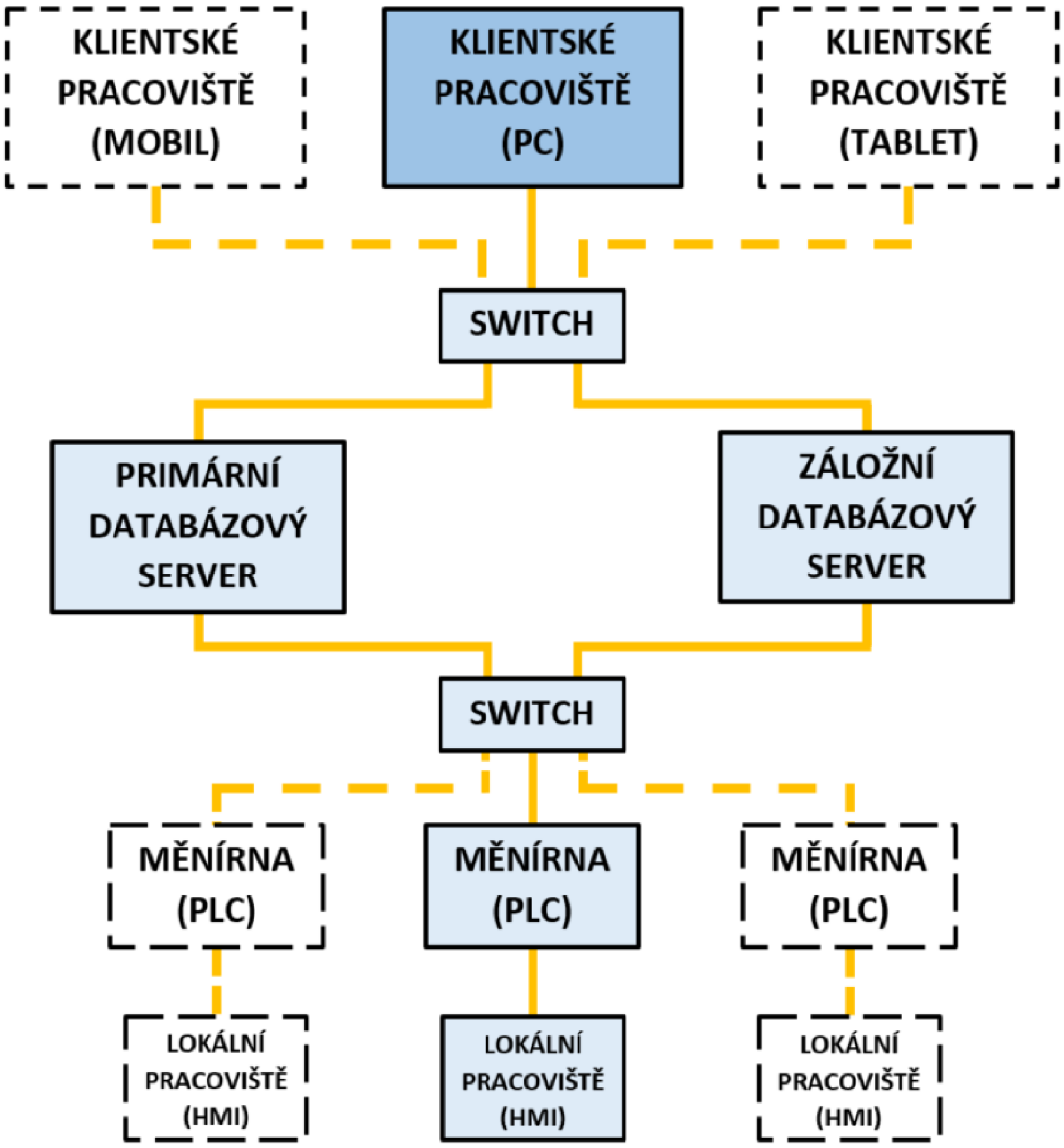
Vzhledem k tomu, že není k dispozici fyzická měnírna, je potřeba vytvořit její simulaci. Pro samotnou měnírnu bude nejvhodnější vybrat PLC a naprogramovat ho tak, aby simulovalo řízení vybrané měnírny pro daný tramvajový a trolejbusový úsek. Pro tuto měnírnu bude potřeba vytvořit simulaci lokálního pracoviště, které má za úkol ovládat měnírnu a kontrolovat její stav. K tomuto účelu by mohl posloužit HMI panel, napojený pomocí komunikačního rozhraní na řídicí PLC s měnírnou. Dispečerské pracoviště může v praxi řídit libovolný počet takto vytvořených měníren, ale vzhledem k tomu, že po softwarové stránce jsou tyto měnírny téměř identické a mám dostupný omezený počet hardwaru, tak budu vytvářet pouze jednu měnírnu, sestavenou z jednoho PLC a jednoho HMI panelu.

Měnírny budou napojeny skrze switch na primární a záložní databázový server, který bude uchovávat všechny informace o všech měnírnách a přes který se budou dané měnírny ovládat z klientského pracoviště (dispečink). V případě funkčnosti obou serverů bude povely zpracovávat pouze primární server a na záložní server se budou ukládat již pouze výsledky těchto povelů. V případě výpadku primárního serveru převezme tuto funkci záložní server. Pro tyto databázové servery by bylo vhodné použít přímo k tomu určený hardware, ale vzhledem k tomu, že nemám žádný k dispozici, tak budu vytvářet oba servery na klasických počítačích.

Klientské pracoviště se bude skládat z uživatelského počítače, který bude skrze switch napojen na oba databázové servery. Na tomto počítači poběží SCADA systém pro ovládání a celkovou správu všech měníren pro všechny úseky. Klientských pracovišť může být více a mohou být různých typů (například mobily, tablety, stolní počítače atd.), ale pro toto dispečerské řízení bude použit pouze jeden osobní počítač. Blokové schéma výsledného návrhu struktury celkového systému řízení včetně možných rozšíření je zobrazeno na obrázku 2.1.

2.1 PLC pro měnírnu

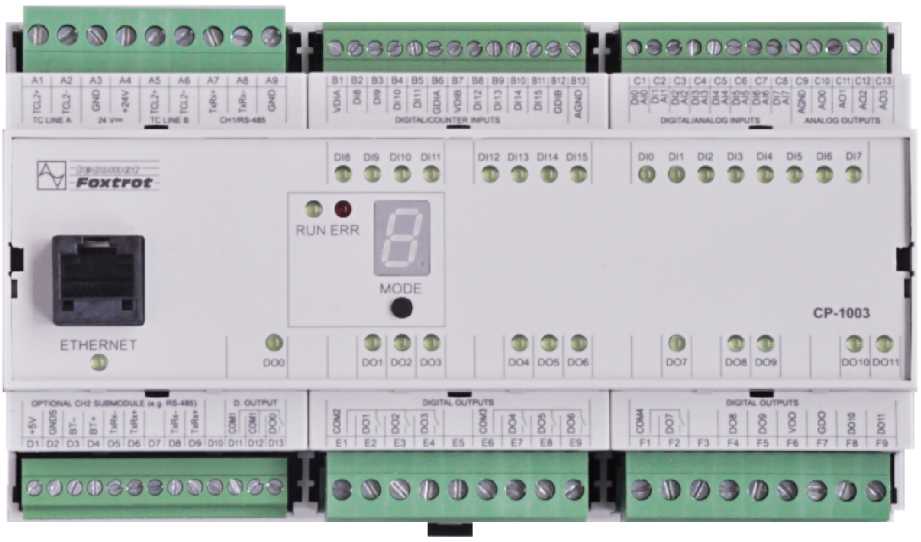
Pro simulaci měnírny jsem obdržel PLC Tecomat Foxtrot CP-1003 firmy Teco, který se nachází na obrázku 2.2. Jedná se o základní modulární programovatelný automat řady Foxtrot, který se umisťuje na DIN lištu. Je vybaven osmi rychlými binárními vstupy, u kterých lze nastavit rozhodovací úroveň, osmi víceúčelovými vstupy, které mohou být buď binární o hodnotě napětí 24 V, nebo analogové (například proudové, napěťové nebo pro pasivní teplotní čidla). Dále obsahuje čtyři analogové výstupy o hodnotě napětí ±10 V, čtyři rychlé tranzistorové výstupy s možností přímého připojení krokových nebo DC motorů a osm reléových výstupů. Konfigurace, diagnostika a programování požadovaných funkcí probíhá pomocí vývojového prostředí



Obr. 2.1: Blokové schéma navržené struktury systému řízení

Mosaic. Toto prostředí jsem popsal v kapitole 1.4.2. [34]

V základním modulu se nachází procesor řady L. Tato řada je určena pro aplikace vyžadující vysoký výkon. Dále obsahuje zálohovanou paměť CMOS RAM určenou pro uživatelské programy a registry, data, tabulky a DataBox. Kromě této paměti je vybaven také Flash pamětí pro zálohu uživatelského programu, slotem pro SD/SD-HC/MMC paměťovou kartu, rozhraním Ethernet, obvodem reálného času, až čtyřmi sériovými kanály, z nichž jeden je s pevným rozhraním RS-485 a další jsou s pozicí pro volitelný submodul. Pro připojení rozšiřovacích modulů ke zvýšení počtu vstupů a výstupů systému nabízí dvě systémová rozhraní TCL2. [34]



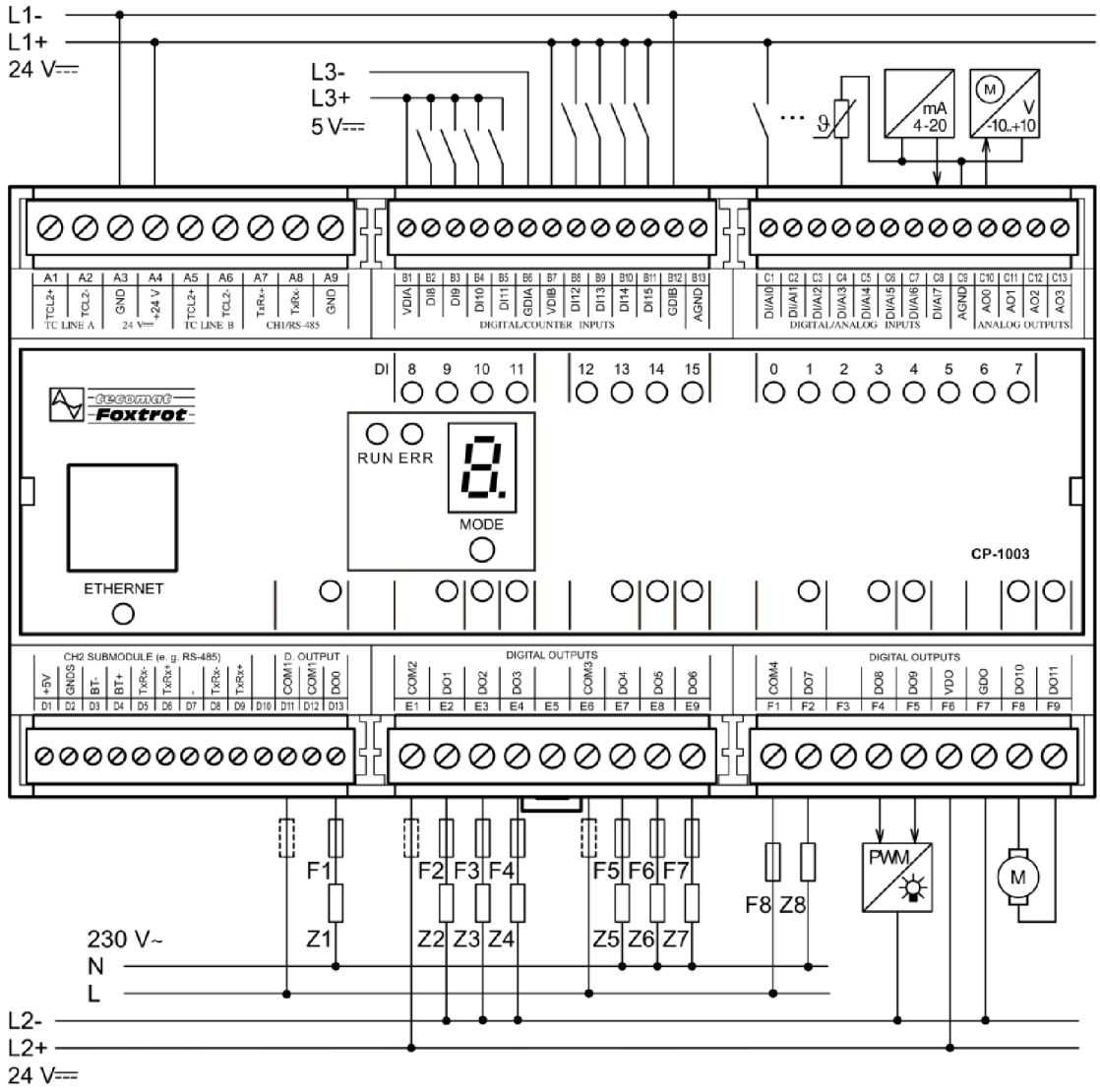
Obr. 2.2: PLC Tecomat Foxtrot CP-1003 [35]

2.1.1 Sestava základního modulu

Základní modul řady Foxtrot se skládá ze dvou částí. První část je tvořena centrální jednotkou s hlavním CPU systému, komunikačními kanály a systémovou sběrnicí TCL2 pro komunikaci s periferními moduly. Druhou periferní část tvoří deska IR-1062, na které je procesor, obsluhující vstupy a výstupy. Tento procesor se na systémové sběrnici hlásí pod stejným jménem, jako je název desky (IR-1062). [36]

2.1.2 Propojení PLC

Komunikace mezi PLC a nadřazeným PC, ostatními PLC nebo ostatními zařízeními je většinou realizována pomocí sériového přenosu. Řada Foxtrot rovněž podporuje základní přenos skrze síť Ethernet nebo průmyslovou síť EPSNET. Rozhraní Ethernet je 10/100 Mb a umožňuje provozovat zároveň více logických spojení. Na prvním asynchronním sériovém kanálu je pevně osazeno rozhraní RS-485 a na druhém sériovém kanálu si zákazník může vybrat mezi rozhraními RS-232, RS-422 nebo RS-485. Při použití rozhraní RS-485 na jedné úrovni sítě EPSNET může být délka sériové linky až 1 200 m a na této lince může být až 32 účastníků. Podporovány jsou rovněž jiné průmyslové protokoly a sběrnice, například Profibus DP, Modbus, CAN atd. Asynchronní komunikaci lze také zajistit univerzálními přenosovými kanály ovládanými přímo z uživatelského programu. Základní zapojení PLC je zobrazeno na obrázku 2.3. [36]



Obr. 2.3: Základní zapojení modulu CP-1003 [34]

2.2 HMI panel pro lokální ovládání měnírny

Pro lokální ovládání měnírny jsem obdržel HMI panel MT6070iE 1WG firmy Weintek Labs. Jeho vzhled se nachází na obrázku 2.4. Z přední strany má panel sedmipalcový barevný TFT displej s rezistivní dotykovou plochou a rozlišením 800 x 480 pixelů a jednu oranžovou LED diodu pro indikaci napájení. Na zadní straně panelu se nachází napájecí svorkovnice, dva USB 2.0 porty, dva COM porty, sokl pro pojistku a čtyři DIP přepínače základních funkcí (hardwarový reset celého panelu, zamknutí přístupu do nastavení panelu, spuštění kalibrace, normální režim atd.). Díky jeho tvaru je ideální pro montáž do dvířek rozvaděčových skříní. Programování a nastavování panelu probíhá skrze vývojový software EasyBuilder Pro, který jsem popsal

v kapitole 1.4.3. [37]

Obr. 2.4: HMI panel Weintek MT6070iE [37]

2.2.1 Připojení panelu

Oba COM porty mají devět pinů. První COM port (COM1) je určen výhradně pro RS-232. Druhý COM port (COM1/3) může mít zapojené současně dva dvouvodičové nebo čtyřvodičové RS-485. Tabulka jejich zapojení se nachází na obrázku 2.5. [37]

Pro RS-485 Weintek podporuje většinu komunikačních protokolů. Kromě klasického Modbus RTU jsou zde podporovány také sběrnice Siemens MPI, Cnet, Omron Host Link nebo SEW Movilink. Toto umožňuje připojení různých snímačů, měřicích zařízení nebo frekvenčních měničů. Z kategorie PLC zde lze připojit téměř všechny výrobce (například Siemens, Allen-Bradley, Omron, Mitsubishi atd.), stačí aby měly sériové rozhraní RS-232 nebo RS-485. [37]

Z USB portů je jeden typu miniUSB, který funguje jako USB Device a používá se pro připojení panelu k počítači. Skrze toto připojení může poté panel komunikovat s programovacím softwarem v počítači a lze do něj následně nahrávat vytvořený program. Druhý USB port je klasický port typu samec pracující jako USB Host a lze

COM PORT1

PIN#	Symbol	COM1[RS232]
1	Not used	
2	RxD	Received Data
3	TxD	Transmitted Data
4	Not used	
5	GND	Signal Ground
6	Not used	
7	RTS	Ready to send output
8	CTS	Clear to send input
9	Not used	

COM PORT2

PIN#	Symbol	COM1 [RS485]2w	COM1 [RS485]4w	COM3 [RS485]
1	Rx-	Data-	Rx-	
2	Rx+	Data+	Rx+	
3	Tx-		Tx-	
4	Tx+		Tx+	
5	GND	Ground		
6	Not used			
7	Data-			Data-
8	Data+			Data+
9	GND			Ground

Obr. 2.5: Zapojení COM portů pro MT6070iE [37]

skrze něj k panelu připojit USB flash disk. Kromě nahrávání programu nebo firmwaru lze tyto porty využít pro uložení a následný přenos naměřených dat v podobě excelovské tabulky do počítače. Slouží také k rozšíření vnitřní paměti panelu. [37]

3 Návrh struktury komunikačních rozhraní pro dispečink a měnírny

Klientské pracoviště má být napojené skrze switch na databázové servery pomocí protokolu IEC 60870-5-104 na Ethernetu. Identické připojení bude také mezi databázovými servery a PLC Tecomat Foxtrot CP-1003. Je potřeba navrhnout komunikační rozhraní mezi PLC a panelem HMI Weintek MT6070iE. Vzhledem k tomu, že jediné Ethernetové rozhraní na PLC bude využito pro připojení k databázovým serverům, tak pro připojení HMI panelu k PLC jsem zvolil sériové rozhraní RS-485 a bude potřeba vybrat komunikační protokol fungující na tomto rozhraní. Pro připojení HMI by šlo také použít sériové rozhraní RS-232, ale vzhledem k jeho horším přenosovým vlastnostem a dodržení jednotného způsobu zapojení HMI panelů ve firmě jsem zvolil rozhraní RS-485.

Při následné realizaci komunikačních rozhraní jsem zjistil, že software, který používám pro klientské pracoviště, nepodporuje přenos dat přes protokol IEC 104. Z tohoto důvodu jsem po konzultaci ve firmě a se souhlasem vedoucího práce vymyslel jiný způsob přenosu dat mezi databázovými servery a klientským pracovištěm. Tento způsob je popsán v kapitole 3.5. Protokol IEC 104 tak používám jen mezi PLC a databázovými servery.

3.1 Síť Ethernet

Ethernet je definován normou IEEE 802.3. Pod tuto síť se řadí velká skupina síťových technologií. Používá se převážně pro lokální sítě. Standardní Ethernet používá v ISO/OSI modelu první (fyzickou) vrstvu a druhou (linkovou) vrstvu. Je zde definován typ kabeláže, signály, formát adresace nebo typ přístupu k síti. [38]

Jako přenosové médium se používá kroucená dvojlinka (twisted pair) nebo optické vlákno (například 10Base-F). Kdysi byl využíván koaxiální kabel. U metalického kabelu se používá stíněná (STP - Shielded Twisted Pair) nebo nestíněná kroucená dvojlinka (UTP - Unshielded Twisted Pair). Optické vlákno může být jednovidové nebo mnohovidové. Při rychlosti Ethernetu 100 Mb/s se používá označení Fast Ethernet (například 100Base-TX), rychlost 1 Gb/s již nese označení Gigabit Ethernet (například 1000Base-T). [38]

Pro přístup k médiu se u Ethernetu standardně využívá metoda CSMA/CD (Carrier Sense Multiple Access With Collision Detection). Při více připojených zařízeních k médiu může komunikovat v jeden moment pouze jedno zařízení, ostatní mohou pouze naslouchat, jinak dojde ke kolizi. V praxi to funguje tak, že zařízení, které chce vysílat, zjišťuje, zda je linka volná, a pokud ano, tak vyšle rámec. Po

odeslání rámce zjišťuje, zdali nedošlo ke kolizi, a pokud ano, tak ukončí vysílání a odešle zprávu, že došlo ke kolizi. Následně počká náhodný čas a pokus opakuje. [38]

Pro přenos se používá half (poloviční) duplex nebo full (plný) duplex. U poloviční duplexu lze v jeden moment komunikovat pouze v jednom směru. U plného duplexu lze zároveň vysílat a přijímat. V dnešní době se používá převážně full duplex. U Ethernetu při kroucené dvojlince se používají dva nebo čtyři páry vodičů, kdy část slouží pro vysílání a část pro přijímání. Díky tomu se zredukuje zpoždění (je zde teoreticky dvojnásobná rychlost) a problémy způsobené metodou CSMA/CD. [38]

3.2 Protokol IEC 60870-5-104

Pro pochopení fungování protokolu IEC 60870-5-104 je nejdříve potřeba popsat IEC 60870-5 a protokol IEC 60870-5-101 (nebo také jen IEC 101). IEC 60870-5 je protokolový standard pro dálkové řízení, ochranu a ostatní telekomunikační funkce pro energetické systémy. IEC 60870-5-101 je standard pro monitorování, řízení a další související komunikační procesy, sloužící k automatizaci energetických systémů. [39]

3.2.1 IEC 60870-5-101

Skrze tento protokol lze přenášet data ze SCADA aplikace do energetického systému. Pro přenos dat je zde použita sériová komunikace, založená například na rozhraních RS-232 nebo FSK. Je založen na EPA architektuře (Enhanced Performance Architecture) a definuje pouze fyzickou, linkovou a aplikační vrstvu ISO/OSI modelu. [39, 40]

Podle typu dostupného komunikačního kanálu (point-point nebo point-multipoint) se používají dva typy komunikačních režimů. Pro komunikaci point-point se používá vyvážený režim, který pracuje jako full duplex a vzdálená koncová jednotka zde může odesílat data, aniž by musela čekat na požadavek z řídicího centra. Díky tomu je komunikace rychlejší. Pro komunikaci point-multipoint se používá nevyvážený režim, který pracuje jako half duplex. Zde data odesílá pouze vzdálená koncová jednotka, která obdržela požadavek z řídicího centra. Modul master se zde musí cyklicky dotazovat všech jednotek, zdali na nich nedošlo k nějaké změně. Nevyvážený režim lze použít také pro komunikaci point-point, ale prodlouží se délka odezvy. [41]

IEC 60870-5-101 definuje dva typy rámců - kontrolní rámec (s pevnou délkou) a rámec s proměnlivou délkou pro přenos aplikačních dat pro jednotlivé služby. Aplikační data se přenáší skrze ASDU (Application Service Data Units) a mohou být různého typu (například inicializace, periodický nebo náhodný přenos, časová

synchronizace, řídicí nebo testovací příkaz, přenos souborů atd.). Po každé odeslané zprávě se čeká na její potvrzení. [41]

Během obecného dotazování jsou zprávy odesílány bez časového razítka, protože obsahují pouze aktuální hodnotu zprávy vzdáleného zařízení. Oproti tomu když vzdálené zařízení odešle samovolně ASDU, tak k nim připojí časové razítko, aby řídicí centrum bylo schopné vytvořit sekvenci událostí v pořadí, v jakém tyto události nastaly na vzdálených zařízeních. [41]

3.2.2 IEC 60870-5-104

IEC 60870-5-104 (nebo také jen IEC 104) je mezinárodní standard vydaný IEC v roce 2000. Definuje se jako síťový přístup pro IEC 60870-5-101 používající standardizované transportní profily. Jedná se prakticky o rozšíření IEC 101 o použití vlastností transportní, síťové, linkové a fyzické vrstvy pro získání plného síťového přístupu. Díky tomu lze zároveň komunikovat mezi více zařízeními a službami. Dvě oddělené linkové vrstvy umožňují přenos dat jak přes sériovou linku, tak také pomocí Ethernetu. Aplikační vrstva u IEC 104 je založena na aplikační vrstvě pro IEC 101. Pro komunikaci skrze Ethernetové LAN sítě se používá TCP/IP s full duplexem. [39, 41, 42]

Zatímco IEC 101 čeká na potvrzení každé odeslané zprávy, IEC 104 předpokládá, že je komunikační kanál stabilní a že může odeslat maximální počet K zpráv bez čekání na potvrzení druhé strany. Oproti IEC 101 má IEC 104 menší počet konfiguračních parametrů a typů zprávy. Například IEC 104 nepodporuje 24bitová časová razítka, ale používá vždy pouze jen 56bitová. Pokus o synchronizaci vzdáleného zařízení přes TCP/IP skrze ASDU s časovou synchronizací není deterministický. Proto jsou v IEC 104 s TCP/IP pro časovou synchronizaci preferovány jiné protokoly, jako je například NTP nebo SNTP, případně pro větší přesnost jsou použity GPS hodiny s PTP nebo IRIG-B. [41, 42]

3.3 RS-485

Sériové rozhraní s maximální přenosovou rychlostí 10 Mb/s. Standard pro RS-485 neurčuje, jaký typ kabelů a konektorů má být použit, ale určuje pouze napěťové úrovně. Budič musí mít minimálně $\pm 1,5$ V a rozhodovací úroveň přijímače je poté ± 200 mV. Na jednom vedení může být až 32 vysílačů a 32 přijímačů pro half duplex. Délka vedení může být 1 200 m při rychlosti do cca 100 kb/s. Pro RS-485 se používá diferenciální zapojení kvůli potlačení vlivu rušení a jako vedení se používá kroucená dvojlinka. Pro eliminaci odrazů se používá na konci vedení terminátor o hodnotě 100 (120) Ω. [43]

3.4 Výběr komunikačního protokolu mezi PLC a HMI

3.4.1 Profibus DP

Profibus standard byl navržen speciálně pro vysokorychlostní I/O operace v automatizaci budov a ve výrobních závodech. Jedná se o otevřený standard, který se řadí mezi nejrychlejší protokoly typu FieldBus. Označení DP v jeho názvu říká, že se jedná o decentralizovanou periferii. To znamená, že se jedná o distribuovaná I/O zařízení, připojená skrze rychlou sériovou datovou linku k centrální řídicí stanici. Navržen byl na základně referenčního modelu ISO/OSI. [44]

Profibus DP funguje na principu master-slave na sériové lince RS-485. Zařízení typu slave se chová jako pasivní kvůli absenci přístupových práv k lince, takže může pouze pouze odpovídat modulu master nebo potvrzovat příchozí zprávy. Všechny slave stanice mají identickou prioritu a síťová komunikace probíhá pouze z modulu master, který se cyklicky dotazuje jednotlivých stanic. Na Profibusu DP existují dva druhy mastera. Master 1 řeší klasickou komunikaci nebo přenos dat s přiřazenými moduly slave. Master 2 je specializované zařízení, určené hlavně pro spouštění modulů slave a diagnostiku. Master 1 může být například PLC nebo PC se speciálním softwarem. Master 2 může být například notebook programátora, který uvádí daná zařízení do provozu. Komunikace master-master není povolena, kromě udělování práv k přístupu jinému modulu master. [44]

Profibus byl navržen tak, aby bylo odpovídání deterministické. I/O data, přenášená ze zařízení slave do zařízení master, mají délku a časování určenou v souboru GSD nebo v datové bázi slave zařízení. Modul master se po spuštění, resetu nebo po ukončení výpadku energie pokouší znovu spojit s moduly slave a až poté přistupuje k cyklické výměně dat. [44]

3.4.2 BACnet MSTP

BACnet je zkratka pro Building Automation and Control networks - používá se převážně pro automatizaci budov, kde dokáže spojit zařízení různých výrobců. Ve verzi MSTP se používá k připojení oblastních zařízení ke kontrolérům, routerům nebo řídicím aplikacím. MS v názvu říká, že se jedná o Master-Slave komunikaci a TP znamená Token Passing. Nejedná se pouze o komunikační protokol, ale také definuje a popisuje typy objektů a služeb, které BACnet zařízení podporují (takzvané BIBBS - BACnet Interoperable Building Blocks pravidla). Fyzická vrstva ISO/OSI modelu funguje na RS-485 a dovoluje připojit až 128 zařízení v jedné síti do maximální délky 1,2 km a rychlosti do 115 000 Bd. Většinou se používá rychlost 19 200 Bd, 38 400 Bd nebo 76 800 Bd. [45, 46]

Zprávy na MSTP síti se dělí do dvou kategorií. První kategorie jsou režijní zprávy, například předávání tokenu. Do druhé kategorie patří aplikační zprávy, které přenášejí požadovaná data. Pouze zařízení, které má aktuálně token, může inicializovat zprávu na aplikační vrstvě a odeslat ji na libovolné zařízení v síti, kromě zařízení typu master, které může odesílat a požadovat data kdykoliv. Některé zprávy následně vyžadují okamžitou odpověď, u některých lze odpovědět později. Zařízení, které zprávu přijalo, nemusí čekat na přidělení tokenu pro vyslání odpovědi, ale může odpovědět ihned. Na aplikační vrstvě je omezený počet zpráv, které může dané zařízení odeslat, než bude muset předat token dále. Nevýhodou předávání tokenu je, že zařízení mají omezenou šířku pásma a musí tak mít vnitřní frontu pro zprávy na aplikační vrstvě, které chtějí odeslat. Pokud následně dojde k zahlcení fronty než znovu obdrží token, aby dané zprávy mohly odeslat, začnou postupně mazat jednotlivé zprávy. [45, 46]

3.4.3 Modbus RTU

Modbus RTU (Remote Terminal Unit) je otevřený protokol, pracující na sériové lince na principu master-slave. Díky své jednoduchosti a spolehlivosti se hodně rozšířil v oblasti průmyslové automatizace a v systémech pro správu budov. Používá se typicky pro přenos dat z řídicích přístrojů do PLC nebo do archivačních systémů. Kromě toho je často používán pro propojení řídicích počítačů se vzdálenými zařízeními (RTU) v SCADA aplikacích. V minulosti se hojně využíval díky své nízké náročnosti na výpočetní výkon a paměť RAM. Modbusová zpráva má jednoduchou 16bitovou strukturu s kontrolním součtem CRC (Cyclic-Redundant Checksum). Díky kontrolnímu součtu jsou případné přenosové chyby odhaleny s účinností až 99%. [47]

Master-slave komunikace může probíhat jak na sběrnicích, tak po síti. Modbus pracuje na sedmé (aplikační) vrstvě ISO/OSI modelu. Funguje na principu žádost/odpověď a poskytuje služby definované funkčními kódy. Funkční kódy jsou součásti Modbusové zprávy PDU (Protocol Data Unit). Pro sestavení aplikační datové jednotky musí klient inicializovat přenos, který slouží k informování serveru o tom, jakou akci bude provádět. Požadavek inicializovaný zařízením master je zajištěn skrze aplikační protokol Modbusu. Následně je pole s kódem funkce zakódováno do jednoho bytu. Validní kód je v rozsahu 1-255, ale hodnoty 128-255 jsou rezervovány pro výjimky (chyby). Pro definování více akcí jsou u některých funkcí k dispozici podfunkce, například modul master může číst stavy skupiny diskrétních vstupů a výstupů. Při obdržení odpovědi ze slave zařízení modulem master je v poli pro kód funkce informace od zařízení slave, zdali odpověď obsahuje chybu, nebo vše proběhlo v pořádku. [47]

Při odesílání dat se nejdříve odesílá nejméně významný bit (LSB). Všechna zaří-

zení v síti musí v tomto pořadí také skládat jednotlivé přijaté byty. Všechna zařízení master a slave připojená na sběrnici musí mít nastavenou stejnou přenosovou rychlost (typicky se používá 9 600 Bd nebo 19 200 Bd). Data jsou buď ve formátu cívek, nebo registrů. Cívky se používají převážně pro reprezentaci stavů diskrétních vstupů a výstupů. Registry jsou pro 16bitová data bez znaménka a desetinného místa a mohou mít hodnotu pouze z rozsahu 0-65 535 (0-FFFF v hexadecimální soustavě). Registry se dělí na vstupní a holding registry. Vstupní registry byly zamýšleny pro použití u analogových vstupů, ale jelikož dnes již moc I/O zařízení pracujících na Modbusu není, tak fungují vstupní registry stejně jako holding registry. Holding registry se dnes využívají jako úložiště pro zařízení. Pakety na Modbusu RTU slouží pouze pro odesílání dat. Nedokáží odesílat parametry, jako je například rozlišení nebo jednotky. [47]

Na RS-485 může být maximálně 32 stanic. Slave zařízení jsou identifikovány číslem stanice, které se nachází před obecnou strukturou zprávy. Adresy mohou být v rozsahu 1-255, nula je rezervována pro broadcast. Modbus RTU je limitován na jednoho mastera na lince. [47]

3.5 Komunikace mezi databázovými servery a dispečinkem

Pro spojení Promotic aplikace běžící na databázových serverech s Promotic aplikací, na které funguje klientské pracoviště, jsou omezené možnosti. Nejlepší možnost byla použít takzvané sdílení XML dat. Funguje na principu klient-server, kdy může být pouze jeden server s daty a libovolný počet klientů, kteří tato data čtou a mohou do nich i zapisovat. Pro vytvoření serveru je potřeba přidat do aplikace objekt PmaWeb, který spojí všechny Web komponenty a začne je nabízet jako jeden Web server. Sdílení XML dat se následně realizuje prostřednictvím karty Web server objektu PmaData nebo PmaDataTable, kde se tyto objekty povolí jako Web komponenta. Pro mou aplikaci jsem použil objekt PmaData a objekt PmaWeb nakonfiguroval jako interní s protokolem HTTP. Na Web klientovi objekt PmaWeb být nemusí. Web klient XML dat se vytvoří přes metody ReadFromWeb a WriteToWeb objektů PmaData nebo PmaDataTable. Data lze poté přenášet v reálném čase. Jedinou nevýhodou tohoto způsobu přenosu dat je, že karta Data objektu PmaData musí ve všech aplikacích obsahovat identické proměnné. Z bezpečnostních důvodů XML přenos dat neumožňuje automaticky změnit počet dat. Přenos XML dat objektu PmaData probíhá hierarchickým způsobem, to znamená, že proměnné nemusí být standardního typu, jako je například integer, ale mohou to být také reference na data dalších objektů PmaData. Jako Web klient může existovat také HTML stránka.

Praktická realizace přenosu dat tímto způsobem je již popsána v podkapitolách 6.1.2 a 6.1.3 v rámci kapitoly pro návrh a realizaci dispečerského SCADA systému. [48]

3.5.1 XML jazyk

XML (Extensible Markup Language) je rozšířitelný značkovací jazyk. Je podobný jazyku HTML, oproti kterému ale nemá předdefinované značky. Tyto značky si může uživatel vytvářet podle svých vlastních potřeb. Díky standardizovanému formátu lze jednoduše přenášet data pomocí XML napříč různými systémy. Pro správnost XML souboru musí být dodržené správné formování dokumentu, musí být dodržena všechna pravidla syntaxe XML a je nutné, aby soubor splňoval všechna sémantická pravidla nastavená zpravidla v XML schématu nebo DTD (Document Type Definition). Pro odkazování na speciální znaky (například znak >, který se používá standardně pro značky) nabízí XML metody, zvané entity. V základu jich je pět, ale pomocí DTD lze přidat další. Na výpisu 3.1 lze vidět ukázku obsahu XML souboru, který jsem vygeneroval. Jedná se o proměnnou KomunikacePLC, kterou jsem vytvořil v rámci softwaru Promotic, a její vlastnosti (datový typ, hodnota, poznámka, přesnost, jednotka a datové rozšíření). [49]

Výpis 3.1: Ukázka obsahu vygenerovaného XML souboru

```
1  <?xml version="1.0" encoding="UTF-16"?>
2  <Document>
3    <Cfg Name="Default">
4      <Content ver="90019">
5        <Props Name="KomunikacePLC">
6          <Prop Name="DataType">11</Prop>
7          <Prop Name="Value"></Prop>
8          <Prop Name="Note"></Prop>
9          <Prop Name="Prec">0</Prop>
10         <Prop Name="Unit"></Prop>
11         <List Name="Extens"/>
12       </Props>
13     </Content>
14   </Cfg>
15 </Document>
```

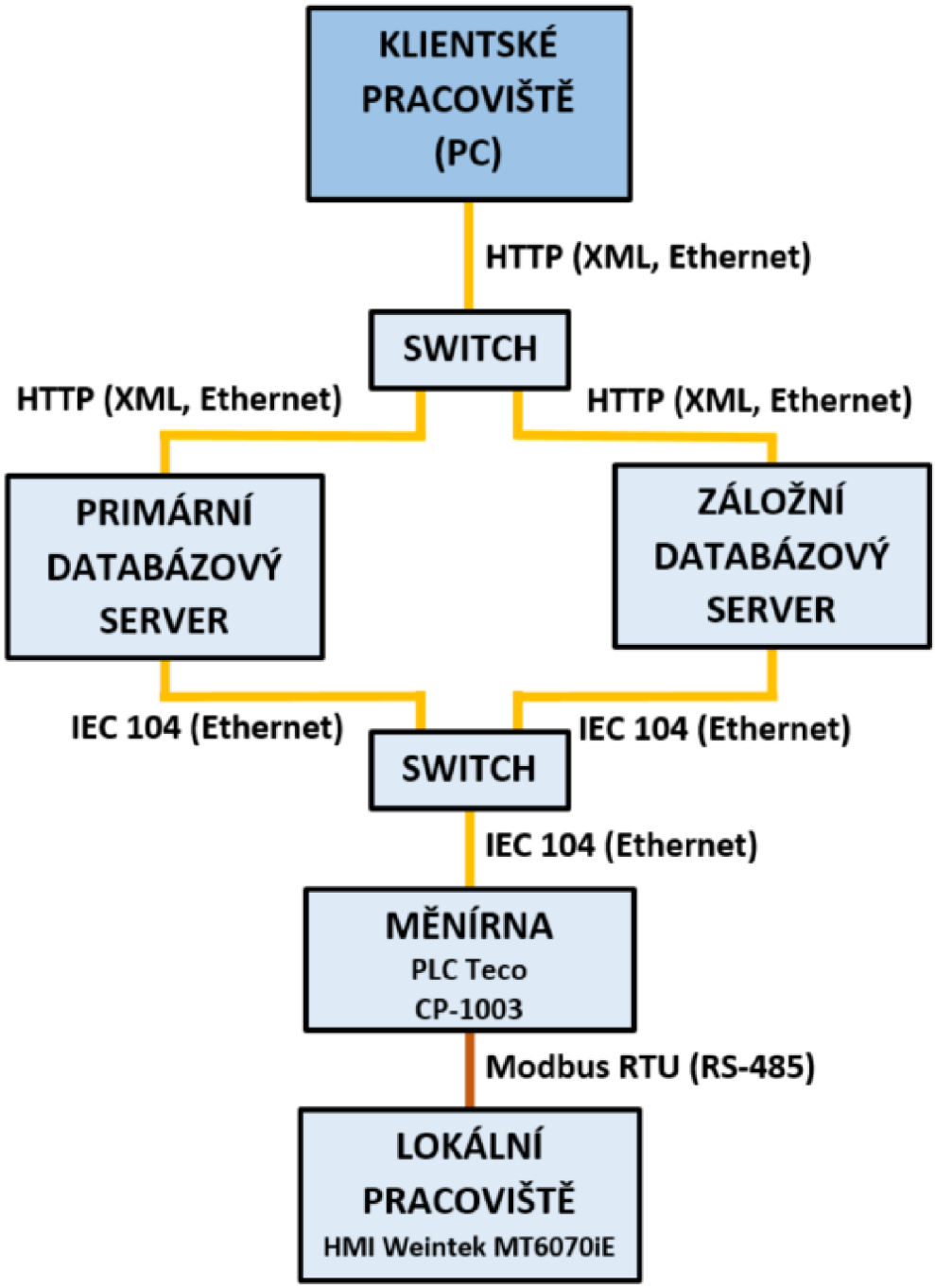
3.5.2 Protokol HTTP

Protokol HTTP (Hypertext Transfer Protocol) je protokol pro kolaborativní, distribuované a hypermediální informační systémy. Je založen na TCP/IP, funguje na

aplikační vrstvě. Může používat různá čísla portů, ale standardně pracuje na portu 80. Položil základ datové komunikace na internetu. Jedná se o obecný a bezstavový protokol, lze ho využít také pro jiné účely přes rozšíření jeho dotazovacích metod, hlaviček a chybových kódů. Používá se pro přenos dat po internetu, jako jsou obrázky, výsledky dotazů nebo HTML soubory. Specifikace HTTP určuje, jakým způsobem budou data na klientovi sestavena a odeslána na server a jakým způsobem server odpoví. Protokol má tři základní vlastnosti. Klient neudržuje spojení se serverem, ale pouze zahájí HTTP žádost, následně čeká na odpověď serveru a po doručení odpovědi se odpojí. Další vlastností je jeho nezávislost na médiu, kdy dokáže odeslat jakýkoliv typ dat, pokud jak klient, tak server dokáží pracovat s jejich obsahem. Do poslední vlastnosti patří jeho bezstavovost, klient se serverem na sebe zapomenou, jakmile ukončí zpracování požadavku. [50]

3.6 Výsledná struktura komunikačních rozhraní

Kvůli své jednoduchosti, spolehlivosti a pro zachování jednotného způsobu zapojení HMI panelů k PLC ve firmě, jsem pro mé připojení HMI Weintek MT6070iE k PLC Teco Foxtrot CP-1003 přes RS-485 zvolil z komunikačních protokolů popsaných v kapitole 3.4 Modbus RTU. Výsledná struktura zapojení se nachází na obrázku 3.1.



Obr. 3.1: Výsledná struktura komunikačních rozhraní

4 Programové vybavení řídicího systému měnírny a její vizualizace

Vzhledem k tomu, že nebylo možné realizovat tuto práci na reálné měnírně, bylo potřeba vytvořit její přibližnou simulaci. Za normálního stavu má každý rozvaděč měnírny své vlastní PLC a HMI panel, ale já jsem měl k dispozici pouze jedno PLC Teco CP-1003, popsané v kapitole 2.1, a jeden HMI panel Weintek, popsaný v kapitole 2.2. Z tohoto důvodu jsem simulaci rozdělil do několika sekcí podle příslušných rozvaděčů měnírny.

4.1 Řídicí systém měnírny (PLC)

PLC jsem programoval v softwaru Mosaic, který je popsaný v kapitole 1.4.2. V manažeru projektu jsem vybral své PLC a v jeho hardwarové konfiguraci nastavil připojení přes RS-485 na kanálu 1 (CH1) pro HMI a dva UNI kanály (UNI0 a UNI1) pro Ethernet 1 (ETH1). V samotném projektu jsem vytvořil pro každou skupinu rozvaděčů svou vlastní programovou organizační jednotku (POU), do které jsem napsal kód pro řízení daného rozvaděče a POU pro ostatní funkce. Název programů (POU) se skládá z názvu rozvaděče, před kterým je předpona prg (například prgDP1). Pro simulaci napětí, proudu a teploty jsem si vytvořil vlastní funkční bloky. Kromě toho bylo potřeba vytvořit POU a inicializační funkci pro IEC 104. Kód jsem vytvářel v jazyce ST (strukturovaný text), který je podobný jazyku C. Deklaraci proměnných jednotlivých rozvaděčů jsem udělal pomocí univerzálních struktur, které jsem následně přiřadil k jednotlivým rozvaděčům. Příklad struktury pro RVS lze vidět na výpisu 4.1. Následné přiřazení v oblasti globálních proměnných lze vidět na výpisu 4.2. V programu jsem záměrně simuloval některé poruchy, aby bylo vidět měnírnu a jednotlivé rozvaděče také ve stavu poruchy.

Výpis 4.1: Ukázka struktury pro RVS

```
1  RVS_Struct : STRUCT
2            PodpetiGB11       : bool;    //000
3            PrepetiGB11       : bool;
4            PoruchaZND1       : bool;
5            StykacTVS_Zapnut  : bool;
6            StykacKONZ_Zapnut: bool;
7            StykacRT20_Zapnut: bool;
8            PritNapetiTVS     : bool;
9            PritNapetiKONZ    : bool;    //007
10           PritNapeti24V     : bool;    //008
```

```
11          TVS_Zapnout       : bool;
12          TVS_Vypnout       : bool;
13          KONZ_Zapnout      : bool;
14          KONZ_Vypnout      : bool;
15          Rezim             : bool; //Dalkove/Mistne
16          Rezerva00         : bool;
17          Rezerva01         : bool;     //015
18          NapetiGB11        : int;      //100
19          ProudGB11         : int;      //200
20          GB11PodpetiHodn   : int := 230;    //300
21          GB11PrepetiHodn   : int := 280;    //400
22    END_STRUCT;
```

Výpis 4.2: Přiřazení struktur k rozvaděčům

```
1  VAR_GLOBAL RETAIN
2    DP1   AT %R3000  : DP1_Struct;
3    DX1   AT %R3006  : DX1_Struct;
4    GU1   AT %R3016  : Usmernovac_Struct;
5    RVS1  AT %R3030  : RVS_Struct;
6    RZK1  AT %R3040  : RZK_Struct;
7    N1    AT %R3054  : Napajec_Struct;
8    N2    AT %R3066  : Napajec_Struct;
9    N3    AT %R3078  : Napajec_Struct;
10 END_VAR
```

Číslice v komentářích na výpisu 4.1 slouží pro orientaci v pořadí proměnných kvůli jejich následné adresaci v HMI, kterou vysvětluji v kapitole 4.2.1. Parametrem AT %R3XXX na výpisu 4.2 přiřazuji každému rozvaděči jeho adresu, jež se rovněž používá pro HMI.

4.1.1 prgHavStop_Deblok

POU kontroluje stisknutí havarijního STOP tlačítka pro měnírnu buď z dispečinku nebo z HMI panelu pro lokální řízení a pokud dojde k jeho stisknutí, zpracuje tento příkaz a vydá povel pro odpojení všech vypínačů v rozvaděčích. Zároveň s tím zablokuje vykonávání určité části řízení v jednotlivých POU a donutí je přejít do vypínacího stavu, kde se vykonají povely odpojení, včetně odpojení od napájení. Ukázka zpracování požadavku na havarijní stop se nachází na výpisu 4.3. Kromě toho se zde nachází také kód pro zpracování povelu k blokaci zemní ochrany.

Výpis 4.3: Zpracování havarijního stopu

```
1  //Havarijni stop
2  If DX1.HavarStop_Zapnout Then //Povel na havarijni
       vypnuti
3     DX1.HavarStop := true; //Aktivace
4     DX1.HavarStop_Zapnout := false;  //Nulovani povelu
5     DX1.HavarStop_Vypnout := false;  //Nulovani povelu
6  ElsIf DX1.HavarStop_Vypnout Then  //Povel na deaktivaci
       havarijniho vypnuti
7     DX1.HavarStop := false;  //Deaktivace
8     DX1.HavarStop_Vypnout := false;  //Nulovani povelu
9  End_If;
```

4.1.2 prgDX1

V programu pro rozvaděč DX1 probíhá simulace tří teplot. Jedná se o teplotu na měnírně, venkovní teplotu a teplotu transformátoru. Všechny teploty jsou ve °C. Všechny simulace probíhají přes volání funkčního bloku fbTeplota, popsaného v kapitole 4.1.9. Do funkčního bloku se předává parametr Offset s přednastavenou teplotou, od které začne simulace (hodnota je předána jako celé číslo, pro skutečnou teplotu se musí vydělit deseti). Simulovaná hodnota teploty se poté předává přes parametr VyslTepl již do konkrétní proměnné. Zdrojový kód prgDX1 se nachází na výpisu 4.4.

Výpis 4.4: Zdrojový kód prgDX1

```
1  PROGRAM prgDX1
2    VAR_INPUT
3    END_VAR
4    VAR_OUTPUT
5    END_VAR
6    VAR
7       Teplota1  : fbTeplota;
8       Teplota2  : fbTeplota;
9       Teplota3  : fbTeplota;
10   END_VAR
11   VAR_TEMP
12   END_VAR
13
14   //Simulace teplot
```

```
15    Teplota1(Offset := 400, VyslTepl => DX1.T_Menirna);
16    Teplota2(Offset := 220, VyslTepl => DX1.T_Venkovni);
17    Teplota3(Offset := 600, VyslTepl => DX1.T_Trafo);
18  END_PROGRAM
```

4.1.3 prgDP1

POU prgDP1 zpracovává veškeré povely týkající se rozvaděče DP1. Pro simulaci proudu se volá funkční blok fbNapetiProud popsaný v kapitole 4.1.8. Pokud hodnota proudu překročí určitou hranici, dojde k simulaci zapůsobení nadproudové ochrany na PS (podélná spojka). V případě havarijního zastavení se zde nachází kód pro celkové odpojení rozvaděče. Ukázka volání funkčního bloku fbNapetiProud a implementace proudové ochrany se nachází na výpisu 4.5. Do parametru Offset z tohoto výpisu se předává počáteční hodnota proudu a výsledná hodnota simulace je získávána z parametru VyslHodn. Z implementace proudové ochrany lze vidět, že proudová ochrana zareaguje od hodnoty 1 000 A.

Výpis 4.5: Simulace proudu a nadproudové ochrany v prgDP1

```
1  VAR
2     Proud   : fbNapetiProud;
3  END_VAR
4
5  //Simulace proudu
6  Proud(Offset := 500, VyslHodn => DP1.DP1_Proud);
7
8  //Simulace aktivace proudove ochrany
9  If DP1.DP1_Proud >= 1000 Then
10    DP1.PS_NadproudOchr := true;
11 Else
12    DP1.PS_NadproudOchr := false;
13 End_If;
```

4.1.4 prgRVS

V prgRVS se nachází kód, týkající se rozvaděče RVS1 a vlastní spotřeby (VLSP). Kromě simulace napětí a proudu na baterii GB11 se tady hlídá její podpětí a přepětí nebo se na základě hodnoty napětí na baterii určuje, zdali je ve VLSP přítomnost napětí 24 V (nastavuje se proměnná PritNapeti24V na true nebo false). Hraniční hodnoty pro podpětí a přepětí jsou přednastaveny, ale lze je změnit přes HMI panel.

Pokud aktuálně není havarijní stop, ve kterém dojde k rozpojení všech stykačů, tak se zde vykonávají povely pro jejich zapnutí nebo vypnutí. Na výpisu 4.6 je ukázka určování podpětí a přepětí na baterii.

Výpis 4.6: Určení podpětí a přepětí na GB11

```
1  If RVS1.NapetiGB11 <= RVS1.GB11PodpetiHodn Then
      //Podpeti na baterii
2     RVS1.PodpetiGB11 := true;
3     RVS1.PrepetiGB11 := false;
4  ElsIf RVS1.NapetiGB11 >= RVS1.GB11PrepetiHodn Then
      //Prepeti na baterii
5     RVS1.PodpetiGB11 := false;
6     RVS1.PrepetiGB11 := true;
7  Else  //Normalni stav
8     RVS1.PodpetiGB11 := false;
9     RVS1.PrepetiGB11 := false;
10 End_If;
```

4.1.5 prgRZK

Pokud na měnírně není havarijní stop, simulují se zde hodnoty proudu na jednotlivých zpětných kabelech. Když některý ze zpětných kabelů bude mít hodnotu proudu větší než 1500 A, aktivuje se porucha analogových vstupů. V případě havarijního vypnutí se hodnoty proudu všech zpětných kabelů nastaví na 0 A.

4.1.6 prgUsmernovac

Vykonává se zde veškerý kód, týkající se usměrňovače. Kromě klasického zpracování povelů pro vypínače a simulace napětí a proudu tady probíhá také simulace teploty transformátoru TU1 voláním funkčního bloku fbTeplota, který je popsaný v kapitole 4.1.9. Pokud dojde k překročení určité maximální nebo minimální hodnoty napětí nebo proudu, nastaví se proměnná, značící poruchu měření. Od hodnoty napětí 450 V na usměrňovači se také nastaví proměnná, indikující přítomnost napětí na usměrňovači. Podle hodnoty teploty se nastavují proměnné TrafoZvysTepl v případě zvýšené teploty a TrafoHavTepl v případě nebezpečné teploty, kde nebezpečná teplota je vyšší než zvýšená teplota. Tyto proměnné se následně využívají k poruchové signalizaci. Hranici pro zvýšenou a nebezpečnou teplotu lze nastavovat z HMI panelu. Ukázka kódu, týkajícího se teploty usměrňovače, se nachází na výpisu 4.7.

Výpis 4.7: Kód týkající se teploty usměrňovače

```
1  VAR
2     Teplota  : fbTeplota;
3     Napeti   : fbNapetiProud;
4     Proud    : fbNapetiProud;
5  END_VAR
6  //Simulace teploty
7  Teplota(Offset := 798, VyslTepl => GU1.Teplota);
8
9  If GU1.Teplota >= GU1.NastNebezpTepl Then //Dosazeni
      havarijni teploty
10    GU1.TrafoHavTepl := true;
11 ElsIf GU1.Teplota >= GU1.NastZvysTepl Then  //Dosazeni
      zvysene teploty
12    GU1.TrafoZvysTepl := true;
13    GU1.TrafoHavTepl  := false;
14 Else  //Normalni stav
15    GU1.TrafoHavTepl := false;
16    GU1.TrafoZvysTepl := false;
17 End_If;
```

4.1.7 prgNapajec

Program prgNapajec je společný pro všechny napáječe. Simulují se v něm hodnoty napětí a proudu na napáječích a hodnoty proudu na jejich kabelových odpojovačích. V případě překročení určitých hodnot napětí a proudů se nastavuje proměnná pro poruchu měření. Od určité hodnoty napětí se nastavuje proměnná PritNapeti, indikující přítomnost správné hodnoty napětí na napáječi. Dále se zpracovávají všechny povely na napáječích. Pokud je některý z kabelových odpojovačů v mezipoloze, nastaví se pro daný napáječ, na kterém tento stav vznikl, proměnná PorKabel, indikující poruchu kabelu. Kód, který toto zajišťuje, se nachází na výpisu 4.8.

Výpis 4.8: Nastavování proměnné pro poruchu kabelu

```
1  If (NOT (N1.KO1_Vypnut XOR N1.KO1_Zapnut)) OR (NOT
      (N1.KO2_Vypnut XOR N1.KO2_Zapnut)) Then  //Porucha
      kabelu
2     N1.PorKabel := true;
3  Else  //Normalni stav
4     N1.PorKabel := false;
```

```
5 End_If;
```

4.1.8 fbNapetiProud

Tento funkční blok používám pro generování hodnot proudu a napětí pro příslušné rozvaděče. Má jednu vstupní proměnnou Offset, která určuje počáteční hodnotu generovaného napětí nebo proudu a jednu výstupní proměnnou VyslHodn, do které se zapisuje vygenerovaná hodnota. Zdrojový kód je sestaven ze dvou časovačů TON s rozdílně nastavenými časy, které v daných intervalech inkrementují hodnotu v proměnné VyslHodn o předem nastavené číslo. Po dosažení určité hodnoty dojde k odečtení všech přičtených hodnot a proměnná VyslHodn se tak dostane zpět na svou počáteční hodnotu. Při prvotním zavolání funkčního bloku dojde k nastavení všech počátečních hodnot a spuštění prvního časovače. Výňatek zdrojového kódu, zobrazující inkrementaci pomocí časovačů a jejich parametry, je zobrazen na výpisu 4.9.

Výpis 4.9: Inkrementace hodnoty ve fbNapetiProud

```
1  //Inicializace casovacu
2  timerTON_1(IN := StartTON1, PT :=T#10s, Q => OutputTON1);
3  timerTON_2(IN := StartTON2, PT :=T#20s, Q => OutputTON2);
4
5  If OutputTON1 Then  //Sepnuti vystupu 1. casovace
6     VyslHodn := VyslHodn + 2;  //Inkrementace o 2
7     StartTON2 := true;
8     StartTON1 := false;
9  ElsIf OutputTON2 Then //Sepnuti vystupu 2. casovace
10    If VyslHodn >= Offset + 6 Then //Reset hodnoty
11       VyslHodn :=  VyslHodn - 6;
12    Else //Inkrementace o 2
13       VyslHodn :=  VyslHodn + 2;  //Inkrementace o 2
14    End_If;
15    StartTON1 := true;
16    StartTON2 := false;
17 End_If;
```

4.1.9 fbTeplota

Funkční blok slouží pro generování teploty ve °C pro požadované rozvaděče. Funguje na stejném principu jako funkční blok fbNapetiProud, popsaný v kapitole 4.1.8.

Jediný rozdíl oproti tomuto funkčnímu bloku je větší časová hodnota PT obou časovačů, takže k inkrementaci hodnoty nedochází tak často a místo výstupní proměnné VyslHodn se používá proměnná VyslTepl. Hodnota v proměnné VyslTepl je celé číslo, pro skutečnou hodnotu teploty je potřeba vydělit ji deseti.

4.1.10 Implementace IEC 104

Pro komunikaci s databázovými servery bylo potřeba importovat knihovny ComLib a Iec104sLib a následně implementovat IEC 104 do programu. PLC funguje jako IEC 104 server, zatímco databázové servery jsou klienti. Inicializaci IEC 104 jsem provedl ve funkci Init_IEC104, kterou jsem převzal z manuálu pro knihovnu Iec104sLib a následně ji upravil podle svých potřeb. Pro každou instanci (server) jsem musel vytvořit v globálních proměnných pole hodnot, sloužící pro spontánní odesílání dat. V prvním poli SendData_Spont jsem nastavil všechny parametry a druhé pole SendData_Spont_2 kopíruje první pole, takže ho stačilo pouze nadeklarovat. Ukázka části prvního pole a deklarace druhého pole se nachází na výpisu 4.10.

Výpis 4.10: Pole hodnot pro spontánní odesílání dat

```
1 SendData_Spont : ARRAY [1..130] OF
     tIec104s_InformationObject_ := [
2   //DP1
3   (TypeIdent := cM_SP_TB_1,  IOA1 := 01,  Used := True
       ), //1
4   (TypeIdent := cM_ME_TE_1,  IOA1 := 19,  Delta := 1.0,
       Used := True ) //19
5 ];
6
7 SendData_Spont_2 : ARRAY [1..130] OF
     tIec104s_InformationObject_;
```

Popis jednotlivých parametrů: [51]

- tIec104s_InformationObject - struktura pro deklaraci cyklicky nebo spontánně odesílaných objektů
- TypeIdent - typ objektu; cM_SP_TB_1 je jednobitová informace s časovou značkou CP56Time2a a cM_ME_TE_1 je 16bitová měřená hodnota s měřítkem a časovou značkou CP56Time2a
- IOA1 - dolní dva byty adresy objektu
- Used - informuje, zdali je objekt použitý
- Delta - velikost změny měřené hodnoty, která zapříčiní odeslání dat

Kopírování do druhého pole SendData_Spont_2 s použitím ukazatelů jsem implementoval přímo do funkce Init_IEC104. Vytvořený kód pro tento účel lze vidět na výpisu 4.11.

Výpis 4.11: Kopírování obsahu SendData_Spont do SendData_Spont_2

```
1  //Kopirovani dat pro druhy server
2  pDataSpont := ADR(DataSpont);
3  FOR i := 1 TO cIec104s_MaxSendDataSpont DO
4    IF (SendData_Spont[i].TypeIdent = cM_ME_TF_1) OR
        (SendData_Spont[i].TypeIdent = cM_ME_NC_1) THEN
5      pDataSpont^.valReal := SendData_Spont[i].valReal;
6    ELSE
7      pDataSpont^.valUdint := SendData_Spont[i].valUdint;
8    END_IF;
9    pDataSpont^.delta          := SendData_Spont[i].delta;
10   pDataSpont^.Descriptor.Ov :=
        SendData_Spont[i].Descriptor.Ov;
11   pDataSpont^.Descriptor.Bl :=
        SendData_Spont[i].Descriptor.Bl;
12   pDataSpont^.Descriptor.Sb :=
        SendData_Spont[i].Descriptor.Sb;
13   pDataSpont^.Descriptor.Nt :=
        SendData_Spont[i].Descriptor.Nt;
14   pDataSpont^.Descriptor.Iv :=
        SendData_Spont[i].Descriptor.Iv;
15   //Posunuti pointeru
16   pDataSpont := pDataSpont +
        sizeof(tIec104s_InformationObject_);
17 END_FOR;
```

Popis jednotlivých kopírovaných parametrů: [51]

- valReal/valUdint - hodnota objektů
- delta - velikost změny měřené hodnoty, která zapříčiní odeslání dat
- Descriptor - deskriptor kvality
  - Ov - přeplněno
  - Bl - blokováno
  - Sb - zaměněno
  - Nt - neaktuální
  - Iv - neplatné

V programu prgIEC104 jsem nejdříve inicializoval IEC 104 po startu a následně volal funkci Init_IEC104 pro oba klienty. Poté jsem přiřadil proměnné ze všech rozvaděčů do pole SendData_Spont a spustil obsluhu komunikace pro oba klienty (servery). Pro oba servery tady také testuji, zdali došlo k jejich připojení a tuto informaci odesílám na záložní server, aby věděl o případném výpadku primárního serveru a mohl ho tak nahradit. Ukázka kódu se nachází na výpisu 4.12.

Výpis 4.12: Ukázka kódu pro odesílání dat přes IEC 104

```
1  //Volani funkce pro oba klienty
2  Init_IEC104(DB_StartAdr := 0, DB_EndAdr := 65000,
      Iec_CFG := klient_1.CFG, DataSpont :=
      SendData_Spont[1]);      // klient_1
3  Init_IEC104(DB_StartAdr := 65001, DB_EndAdr := 128000,
      Iec_CFG := klient_2.CFG, DataSpont :=
      SendData_Spont_2[1]);      // klient_2
4
5  //Ukazka prirazeni do SendData_Spont
6  SendData_Spont[18].valUdint.0 := DP1.TVS_PusobOchr;
7  SendData_Spont[19].valUdint   :=
      int_to_udint(DP1.DP1_Proud);
8
9  //Obsluha komunikace
10 klient_1( chanCode := ETH1_uni0, coldRestart :=
      System_S.S2_4, hotRestart := System_S.S2_3);
11 klient_2( chanCode := ETH1_uni1, coldRestart :=
      System_S.S2_4, hotRestart := System_S.S2_3);
12
13 //Test spojeni se servery
14 Kom_klient1 := klient_1.connected;
15 Kom_klient2 := klient_2.connected;
```

Popis jednotlivých parametrů: [51]

- DB_StartAdr - adresa začátku bufferu zpráv v DataBoxu
- DB_EndAdr - adresa konce bufferu zpráv v DataBoxu
- CFG - konfigurační parametry
- chanCode - označení komunikačního kanálu, přes který probíhá komunikace
- coldRestart - studený restart (vymažou se data)
- hotRestart - teplý restart (data zůstanou zachována)
- connected - navázána komunikace

Pro zpracování povelů ze serverů jsem musel vytvořit v globálních proměnných pole hodnot RecvCommands pro příjem povelů. V programu prgIEC104 poté zjistím jednotlivé parametry povelu, zpracuji ho a zapíšu příslušnou hodnotu do proměnné daného rozvaděče. Ukázka deklarace části pole v globálních proměnných a následné zpracování povelu v prgIEC104 se nachází na výpisu 4.13.

Výpis 4.13: Ukázka kódu použitého pro příjem dat přes IEC 104

```
1  //Ukazka casti deklarovani pole hodnot pro prijem povelu
2  RecvCommands : ARRAY [1..50] OF
       tIec104s_InformationObjectCmd_ := [
3    (TypeIdent := cC_SC_NA_1, IOA1 := 208),//8
4    (TypeIdent := cC_SE_NB_1, IOA1 := 209) //9
5  ];
6
7  //Ukazka zpracovani povelu v prgIEC104
8  If ((klient_1.command.IsNewCmd AND
       klient_1.command.IsFound)) OR
       ((klient_2.command.IsNewCmd AND
       klient_2.command.IsFound)) Then
9      //DP1
10     If (RecvCommands[1].IOA1 = newCmd.IOA1) AND
          (RecvCommands[1].IOA2 = newCmd.IOA2) AND
          (RecvCommands[1].IOA3 = newCmd.IOA3) Then
11        If newCommands.DCS = 1 Then
12           DP1.PS_Vypin_Vypnout := true;
13        End_If;
14     End_If;
15 End_If;
```

Popis jednotlivých parametrů: [51]

- tIec104s_InformationObjectCmd_ - struktura pro deklaraci obdržených povelů
- TypeIdent - typ objektu; cC_SC_NA_1 je jednoduchý povel a cC_SE_NB_1 slouží pro nastavení měřené hodnoty
- IOA1, IOA2, IOA3 - části adresy objektu
- command - přijatý povel; IsNewCmd má hodnotu true po obdržení nového povelu, IsFound znamená, že byl nalezen v seznamu definovaných povelů
- DCS - SCO u jednoduchého povelu nebo DCO u dvojitého povelu

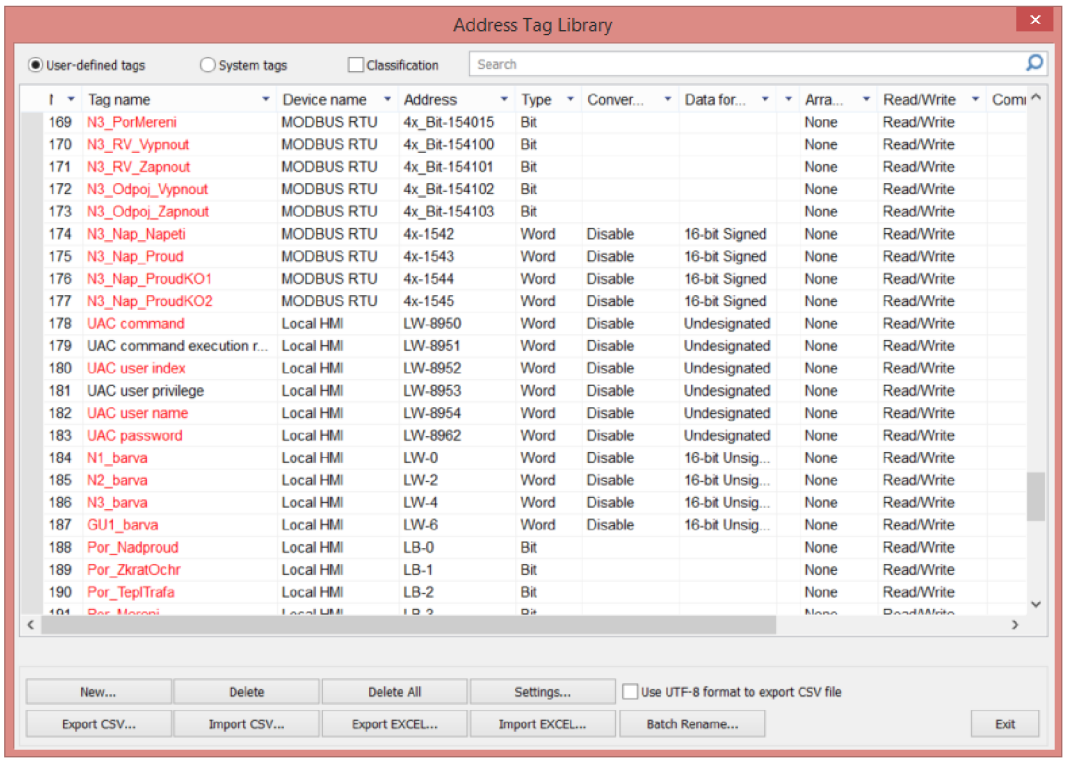
4.2 Lokální ovládání měnírny pomocí HMI panelu

K vytvoření vizualizace pro HMI panel jsem použil software EasyBuilder Pro popsaný v kapitole 1.4.3. Nejdříve jsem při nastavování projektu zvolil, že budu vytvářet projekt pro mnou použité HMI, čímž se mi nastavily základní vlastnosti projektu, jako je například rozlišení obrazovky. Poté jsem musel přidat nové zařízení pro Modbus RTU, přes které se napojím na PLC. Jeho nastavení lze vidět na obrázku 4.1.

Obr. 4.1: Nastavení pro Modbus RTU

4.2.1 Definování adres

V EasyBuilder Pro se místo standardních proměnných používají takzvané tagy s adresami. Jejich vzhled lze vidět na obrázku 4.2. Každé zařízení má své vlastní adresy.



I	Tag name	Device name	Address	Type	Conver...	Data for...	Arra...	Read/Write	Comi
169	N3_PorMereni	MODBUS RTU	4x_Bit-154015	Bit			None	Read/Write	
170	N3_RV_Vypnout	MODBUS RTU	4x_Bit-154100	Bit			None	Read/Write	
171	N3_RV_Zapnout	MODBUS RTU	4x_Bit-154101	Bit			None	Read/Write	
172	N3_Odpoj_Vypnout	MODBUS RTU	4x_Bit-154102	Bit			None	Read/Write	
173	N3_Odpoj_Zapnout	MODBUS RTU	4x_Bit-154103	Bit			None	Read/Write	
174	N3_Nap_Napeti	MODBUS RTU	4x-1542	Word	Disable	16-bit Signed	None	Read/Write	
175	N3_Nap_Proud	MODBUS RTU	4x-1543	Word	Disable	16-bit Signed	None	Read/Write	
176	N3_Nap_ProudKO1	MODBUS RTU	4x-1544	Word	Disable	16-bit Signed	None	Read/Write	
177	N3_Nap_ProudKO2	MODBUS RTU	4x-1545	Word	Disable	16-bit Signed	None	Read/Write	
178	UAC command	Local HMI	LW-8950	Word	Disable	Undesignated	None	Read/Write	
179	UAC command execution r...	Local HMI	LW-8951	Word	Disable	Undesignated	None	Read/Write	
180	UAC user index	Local HMI	LW-8952	Word	Disable	Undesignated	None	Read/Write	
181	UAC user privilege	Local HMI	LW-8953	Word	Disable	Undesignated	None	Read/Write	
182	UAC user name	Local HMI	LW-8954	Word	Disable	Undesignated	None	Read/Write	
183	UAC password	Local HMI	LW-8962	Word	Disable	Undesignated	None	Read/Write	
184	N1_barva	Local HMI	LW-0	Word	Disable	16-bit Unsig...	None	Read/Write	
185	N2_barva	Local HMI	LW-2	Word	Disable	16-bit Unsig...	None	Read/Write	
186	N3_barva	Local HMI	LW-4	Word	Disable	16-bit Unsig...	None	Read/Write	
187	GU1_barva	Local HMI	LW-6	Word	Disable	16-bit Unsig...	None	Read/Write	
188	Por_Nadproud	Local HMI	LB-0	Bit			None	Read/Write	
189	Por_ZkratOchr	Local HMI	LB-1	Bit			None	Read/Write	
190	Por_TeplTrafa	Local HMI	LB-2	Bit			None	Read/Write	

Obr. 4.2: Knihovna adres

Pod Local HMI jsem vytvářel pomocné proměnné pro řízení vizualizace a v MODBUS RTU jsem nadefinoval proměnné pro přenos dat mezi PLC a HMI. V PLC jsem každému rozvaděči přiřadil určitou adresu ve tvaru %R30XX (lze vidět na výpisu 4.2) a v HMI jsem v zařízení MODBUS RTU vytvořil proměnnou s příslušnou adresou ve tvaru 4x-15XX pro typ Word nebo 4x-_Bit-15XXXX pro typ Bit. K získání adresy pro HMI, patřící konkrétní proměnné v PLC, se vezme adresa rozvaděče z PLC, vydělí se dvěma a přičte se k ní jednička. Tímto je získána adresa pro Word. Pro Bit se ještě musí přidat dvě číslice, indikující pořadí proměnné v PLC, které mohou nabývat hodnot v rozsahu 00 - 15. V případě dosažení nejvyššího povoleného čísla se jen inkrementuje předchozí číslo v řadě. Například bitová proměnná v PLC na nulté pozici rozvaděče s adresou %R3000 bude mít v HMI adresu 4x-_Bit-150100 a proměnná na 17. pozici bude mít v HMI adresu 4x-_Bit-150201.

4.2.2 Grafická okna

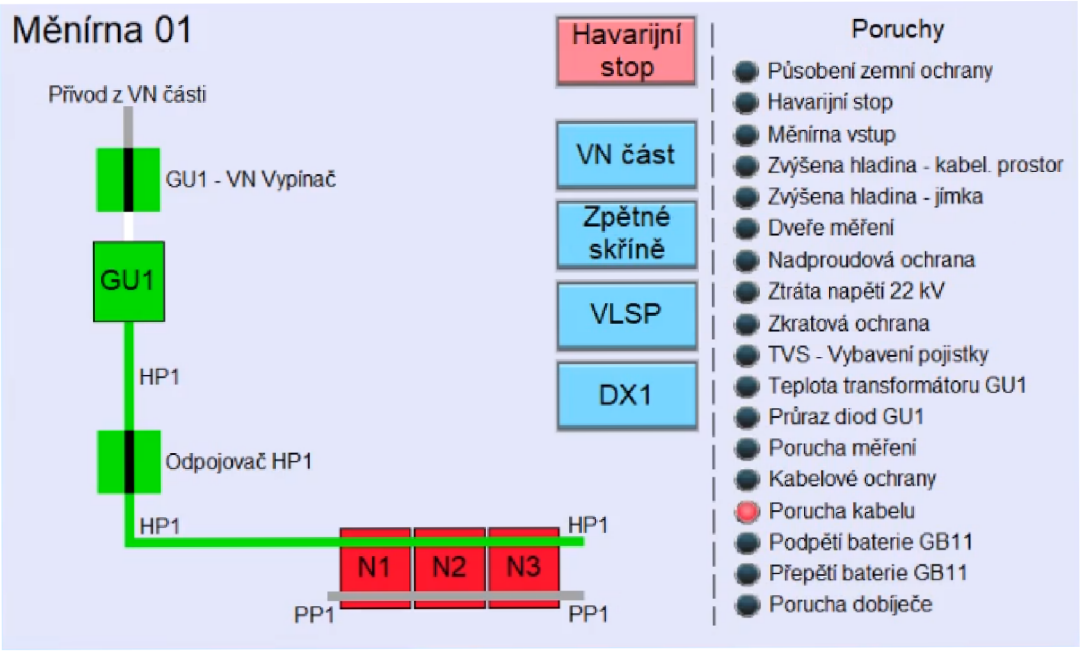
Vzhledem k tomu, že používám malý HMI panel, tak jsem po konzultaci s firmou ve všech grafických obrazech kromě přehledu měnírny nekreslil kompletní jednopólová schémata pro jednotlivé sekce měnírny, ale zobrazuji zde jednotlivé stavy pomocí jednoduchých indikátorů. Do grafických oken jsem také přidal sekce, kde uživatel může v rámci simulace nastavovat proměnné, které se standardně neovládají, ale jejich stavy jsou zjišťovány na základě senzorů (například stav odpojovače, který se standardně ovládá pouze ručně). Vizualizaci měnírny jsem rozdělil do několika grafických obrazů.

Hlavní přehled měnírny

Hlavní přehled měnírny je vytvořen v obraze WINDOW_010, který se zobrazuje jako první obraz po spuštění vizualizace na HMI panelu a lze jej vidět na obrázku 4.3. V tomto obraze je nakresleno zjednodušené schéma propojení usměrňovače s napáječi. Jednotlivé rozvaděče jsou představeny obdélníkovými obrazci, které mění barvu na základě jejich stavu. Pro ztrátu komunikace je to barva fialová, v případě poruchy na rozvaděči se jedná o barvu červenou, pokud je rozvaděč bez napětí, tak je bílé barvy a pod napětím má zelenou barvu. To samé platí pro nakreslené elektrické vedení kromě poruchy, kde je místo toho šedá barva pro neurčitý stav, pokud je některý z vypínačů nebo odpojovačů před daným úsekem v mezipoloze. Vypínače a odpojovače v přehledu nabývají čtyř stavů. Když není vypnuto ani zapnuto, zobrazuje se červený otazník na žlutém pozadí. V případě, kdy je aktivní obojí, zobrazují se dva červené vykřičníky na žlutém pozadí. Při vypnutém stavu je to černá čára na bílém pozadí a v sepnutém stavu černá čára na zeleném pozadí. V pravé části obrazu se nachází seznam poruch na měnírně a nalevo od tohoto seznamu je červené havarijní tlačítko pro vypnutí měnírny a modrá tlačítka pro přepínání mezi jednotlivými obrazy. V levé horní části se ještě nachází název měnírny.

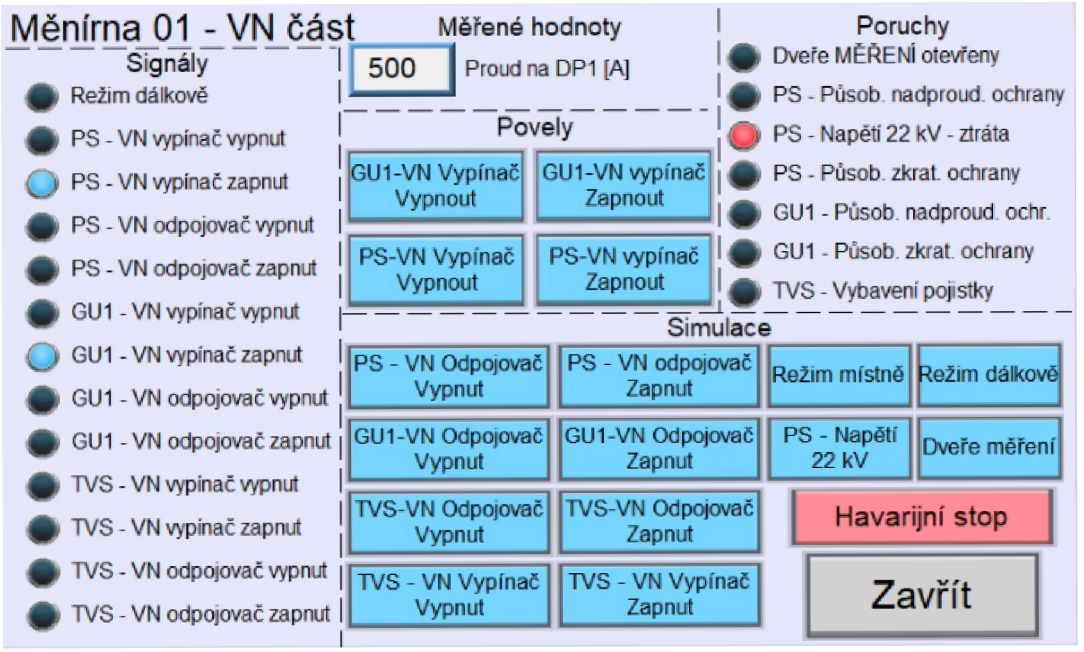
VN část

Obraz VN části (v projektu název VNcast) je rozdělen čerchovanou čárou na několik sekcí, jak lze vidět z obrázku 4.4. Vlevo nahoře se nachází název měnírny a aktuálně otevřeného obrazu. V levé části je seznam signálů pro VN část, představující různé prvky. Pokud je signál v nule (stav false), je jeho indikátor zhasnutý. V případě aktivního signálu se indikátor rozsvítí modře. V prostřední části obrazu lze vidět naměřené hodnoty na VN části a tlačítka pro spínání a rozpínáni vypínačů. Na pravé horní straně obrazu se nachází seznam poruch, týkajících se VN části měnírny, kdy pokud je porucha neaktivní, je její indikátor zhasnutý a v případě aktivní poruchy se rozsvítí červenou barvou. V pravé dolní straně obrazu se nachází sekce pro simulaci



Obr. 4.3: Přehled měnírny na HMI panelu

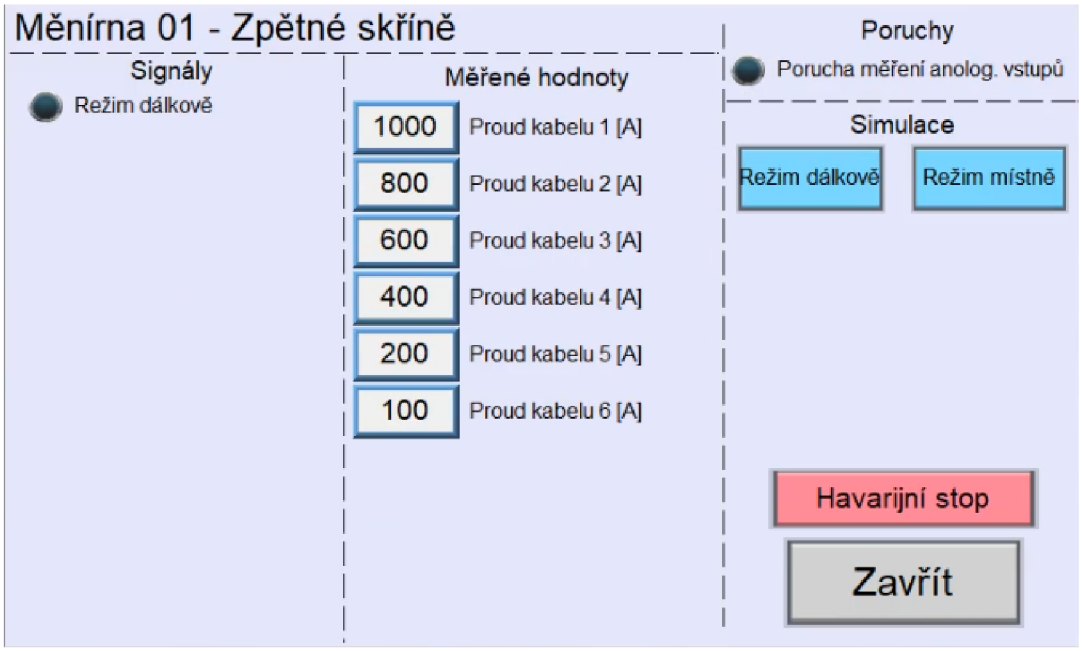
signálů, které nelze standardně ovládat. Tlačítka pro havarijní vypnutí měnírny a zavření obrazu VN části jsou umístěna v pravém dolním rohu.



Obr. 4.4: VN část na HMI panelu

Zpětné skříně

Obraz zpětných skříní měnírny (v projektu název ZpetneSkrine) je zobrazen na obrázku 4.5. Nachází se zde název měnírny a aktivního obrazu, signály, poruchy a měřené hodnoty týkající se zpětných skříní. Signály lze rovněž simulovat, v tomto případě pouze měnit režim ovládání z dálkového na místní a opačně. V pravém dolním rohu obrazu je umístěno tlačítko pro havarijní vypnutí měnírny a tlačítko pro zavření obrazu zpětných skříní, které dostane uživatele zpět na hlavní přehled měnírny.



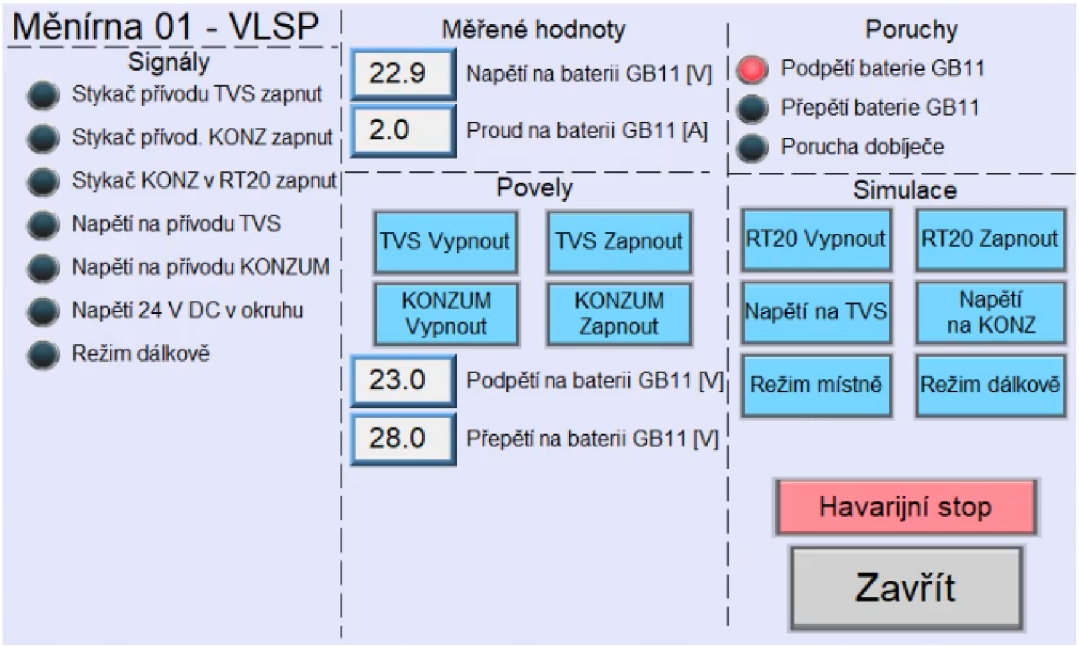
Obr. 4.5: Zpětné skříně na HMI panelu

Vlastní spotřeba (VLSP)

Přes obraz VLSP (v projektu název VLSP) může uživatel zjistit stavy jednotlivých signálů a naměřených hodnot ve vlastní spotřebě. Kromě standardního seznamu poruch, simulace jednotlivých signálů, tlačítka pro havarijní vypnutí měnírny a tlačítka pro zavření obrazu, se zde nachází také sekce Povely pro ovládání jednotlivých stykačů na vlastní spotřebě a nastavování hodnot podpětí a přepětí na baterii GB11. Obraz VLSP je zobrazen na obrázku 4.6.

DX1

Do vizualizace měnírny jsem také přidal speciální obraz pro rozvaděč DX1 (v projektu název DX1). Tento obraz lze vidět na obrázku 4.7. V tomto obrazu vidí uživatel



Obr. 4.6: Vlastní spotřeba na HMI panelu

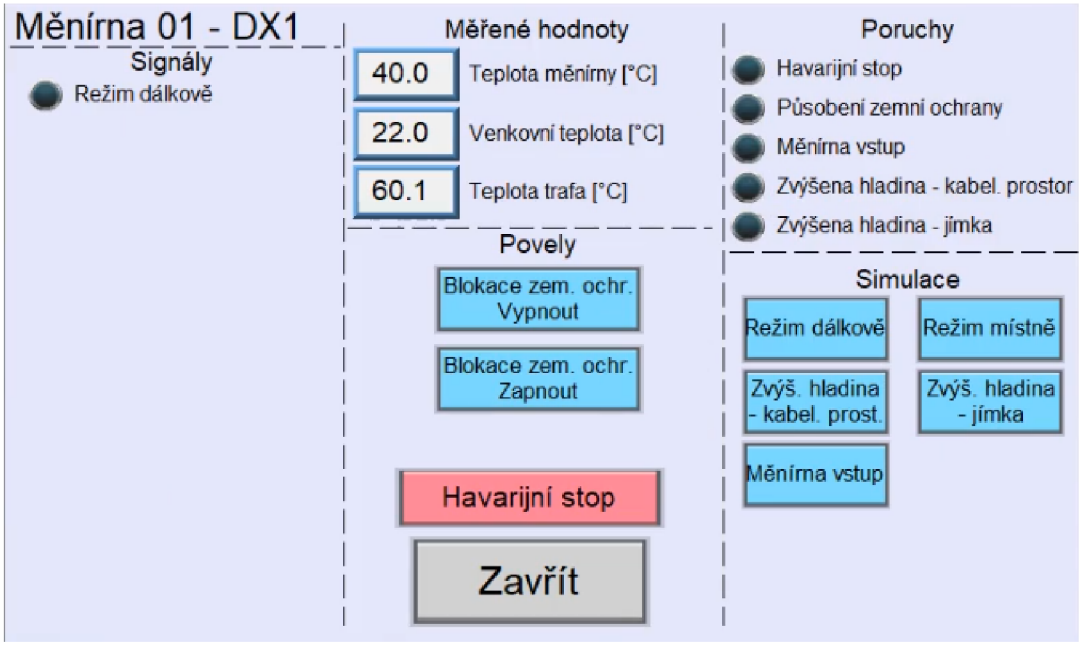
režim rozvaděče, aktuální teploty, týkající se celé měnírny, a konkrétní poruchy pro měnírnu a tento rozvaděč. Lze odtud ovládat blokaci zemní ochrany a simulovat režim ovládání nebo jednotlivé poruchy. Ve spodní části obrazu se uprostřed nachází tlačítka pro havarijní vypnutí měnírny a zavření obrazu rozvaděče DX1.

Detail usměrňovače

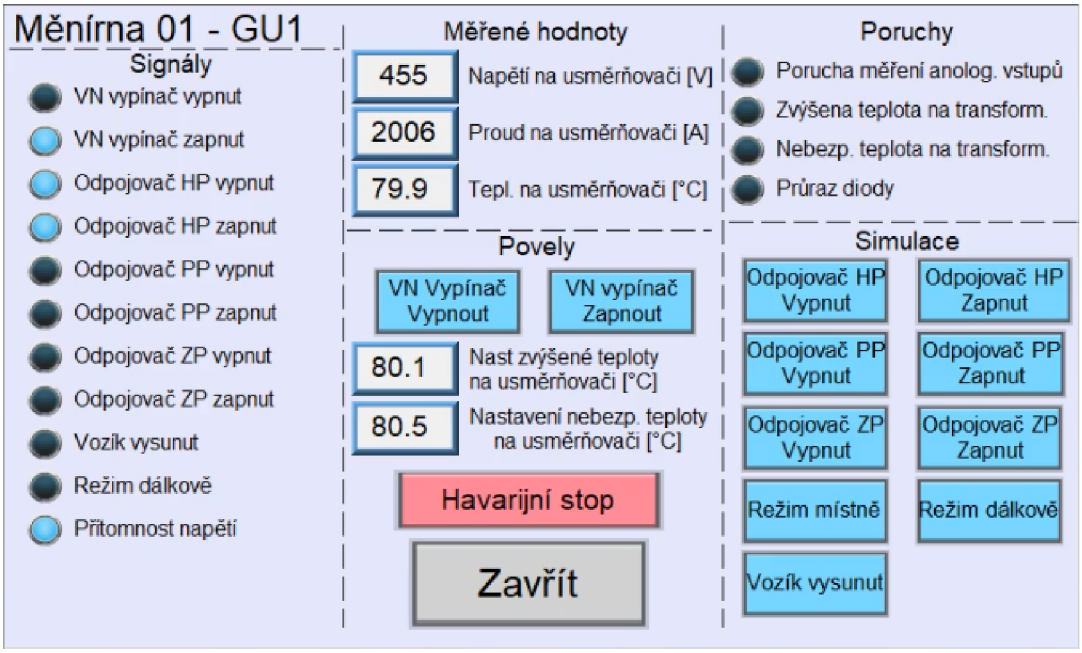
Po kliknutí na usměrňovač v přehledu se otevře okno s detailem usměrňovače, zobrazené na obrázku 4.8. Uživatel může v levé části obrazu vidět aktuální stav daného usměrňovače. Uprostřed se nachází indikátory s hodnotou napětí, proudu a teploty. Pod těmito indikátory lze pomocí povelů ovládat VN vypínač usměrňovače a nastavovat hranici pro zvýšenou a nebezpečnou teplotu tranformátoru TU1. Ve spodní části se nachází standardní tlačítka pro havarijní stop a zavření obrazu. Pokud na usměrňovači nastane nějaká porucha, rozsvítí se její indikátor v pravé horní části okna. Lze zde rovněž simulovat stavy jednotlivých prvků usměrňovače.

Detail napáječe

Pro zobrazení detailu napáječe z obrázku 4.9 stačí kliknout na jeho rozvaděč v hlavním přehledu měnírny. V levé části napáječe se nachází signály daného napáječe. Z měřených hodnot je zde napětí a proud na napáječi a proud na jednotlivých

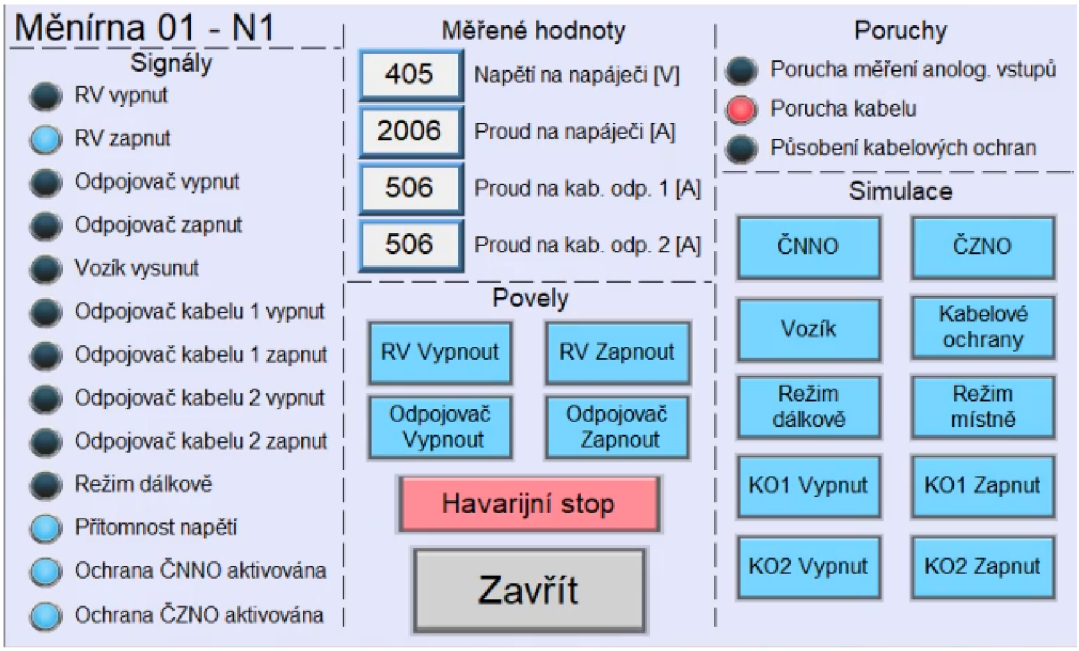


Obr. 4.7: Rozvaděč DX1 na HMI panelu



Obr. 4.8: Detail usměrňovače na HMI panelu

kabelových odpojovačích. Povely pro napáječ se nachází uprostřed v dolní polovině obrazu spolu s tlačítky pro havarijní zastavení měnírny a zavření tohoto okna. V pravé části se potom nachází seznam poruch na napáječi a simulace jednotlivých proměnných.



Obr. 4.9: Detail napáječe na HMI panelu

4.2.3 Řízení vizualizace

Veškerou vizualizaci lze řídit pomocí maker. Jejich programovací jazyk je podobný jazyku Visual Basic. V projektu jsem vytvořil čtyři makra pro specifické účely.

Makro Poruchy

Toto makro využívám pro obecné indikování poruch (sloučení více poruch stejné kategorie do jedné poruchy/indikátoru) v hlavním přehledu měnírny (WINDOW_010). Nejdříve získám hodnoty jednotlivých proměnných z MODBUS RTU pomocí funkce GetData a následně je spojím a uložím pomocí funkce SetData do speciálně vytvořené proměnné pro tento účel v Local HMI. Makro je nastavené na periodické vykonávání s intervalem 1,5 s. Ukázka kódu pro zvýšenou a havarijní teplotu transformátoru je zobrazena na výpisu 4.14.

Výpis 4.14: Ukázka kódu z makra Poruchy

```
1 bool nTemp, nSmaz, GU_ZvysTepl, GU_HavTepl
2 nTemp = 1
3 nSmaz = 0
4
5 'Ziskani promennych transformatoru
6 GetData(GU_ZvysTepl, "MODBUS␣RTU", "GU1_TrafoZvysTepl",
    1)
```

```
7  GetData(GU_HavTepl, "MODBUS␣RTU", "GU1_TrafoHavTepl", 1)
8
9  if GU_ZvysTepl or GU_HavTepl then 'Test stavu poruch
10   SetData(nTemp, "Local␣HMI", "Por_TeplTrafa", 1)
       'Nastaveni obecne poruchy
11 else
12   SetData(nSmaz, "Local␣HMI", "Por_TeplTrafa", 1)
       'Nulovani obecne poruchy
13 end if
```

Makro Stavy_barva

Používá se v hlavním přehledu měnírny (WINDOW_010), kde slouží pro nastavování barev rozvaděčům na základě jejich stavu, nastavování barev elektrického vedení a zobrazování příslušného stavu odpojovačů a vypínačů. Pro rozvaděče se testuje, zdali je navázána komunikace s PLC a pokud ne, tak se probarví fialově. Pokud komunikace funguje, zjišťuje se porucha na rozvaděčích a když jsou v poruše, tak se probarví červeně. V případě stavu bez poruchy změní barvu na zelenou pro stav pod napětím, nebo bílou pro stav bez napětí. Ukázka kódu pro napáječ N1 se nachází na výpisu 4.15. Makro se vykonává s periodou 1 s.

Výpis 4.15: Nastavování barev rozvaděčům

```
1  short nKom, nPor, nBezNap, nPodNap, nKomErr
2
3  nKom = 0
4  nPor = 1
5  nBezNap = 2
6  nPodNap = 3
7  'Ziskani dat
8  GetDataEx(nKomErr, "Local␣HMI", LW, 9401, 1)
9  GetDataEx(N1_KO, "MODBUS␣RTU", "N1_KabelOchr", 1)
10 GetDataEx(N1_Kab, "MODBUS␣RTU", "N1_PorKabel", 1)
11 GetDataEx(N1_Mer, "MODBUS␣RTU", "N1_PorMereni", 1)
12 GetDataEx(N1_Nap, "MODBUS␣RTU", "N1_PritNapeti", 1)
13
14 //N1
15 if nKomErr == 22 then 'Porucha komunikace
16   SetData(nKom, "Local␣HMI", "N1_barva", 1)
17 else
```

```
18    if N1_KO or N1_Kab or N1_Mer then 'Porucha rozvadece
19      SetData(nPor, "Local␣HMI", "N1_barva", 1)
20    else if N1_Nap then 'Pod napetim
21      SetData(nPodNap, "Local␣HMI", "N1_barva", 1)
22    else  'Bez napeti
23      SetData(nBezNap, "Local␣HMI", "N1_barva", 1)
24    end if
25  end if
```

Pro vypínače a odpojovače se zjišťují stavy jejich dvou proměnných zapnuto a vypnuto a na základě toho se nastaví hodnota do proměnné, která poté řídí jejich vzhled v obraze. Tato hodnota se také používá spolu se zjišťováním jejich stavů v makru pro nastavování barev jednotlivých úseků elektrického vedení, které může nabývat následujících stavů (hodnot):

- 0 = ztráta komunikace (fialová);
- 1 = neurčitý stav (šedá);
- 2 = bez napětí (bílá);
- 3 = pod napětím (zelená).

Makro Napajec_prirazeni

Abych nemusel vytvářet tři identické obrazy s detaily napáječe, protože v obraze nelze prvkům přiřazovat proměnné až na základě toho, pro který napáječ jsem otevřel jeho detail, vytvořil jsem toto makro. V obrazu detailu napáječe mají všechny prvky přiřazeny obecné proměnné začínající na Nx_, kde se za podtržítkem nachází konkrétní název proměnné (například Nx_RVZap). Každý z napáječů má přiřazenou svou vlastní bitovou adresu, která se nastaví po kliknutí na tento napáječ. Na základě této adresy se v makru do proměnných začínajících na Nx_ přiřadí konkrétní hodnoty pro zvolený napáječ a otevře se jeho detail. Pokud uživatel z obrazu vykoná nějaký povel, tak se tento povel zkopíruje z proměnné Nx_xxxx zpět do proměnné konkrétního napáječe. Ukázku přiřazení hodnot a otevření obrazu lze vidět na výpisu 4.16.

Výpis 4.16: Zpracování hodnot pro detail napáječe

```
1  bool N1, tmp, okno, zavreni, smaz
2  int tmp2
3  okno = 0
4  smaz = 0
5  GetData(N1, "Local␣HMI", LB, 10, 1)
6
```

```
7  if N1 then  'Test aktivniho napajece
8    GetData(tmp, "MODBUS RTU", "N1_PorMereni", 1)
9    SetData(tmp, "Local HMI", "Nx_PorMer", 1)
10   GetData(tmp2, "MODBUS RTU", "N1_Nap_Napeti", 1)
11   SetData(tmp2, "Local HMI", "Nx_Napeti", 1)
12
13   GetData(tmp, "Local HMI", "Nx_RV_Vypnout", 1)
14   if tmp then 'Zkopirovani aktivniho povelu
15     SetData(tmp, "MODBUS RTU", "N1_RV_Vypnout", 1)
16   end if
17
18   okno = 1
19 end if
20
21 GetData(zavreni, "Local HMI", LB, 41, 1)
22 if not zavreni then 'Otevreni okna
23   SetData(okno, "Local HMI", LB, 40, 1)
24 else  'Zavreni okna
25   SetData(smaz, "Local HMI", LB, 10, 1)
26   SetData(smaz, "Local HMI", LB, 40, 1)
27   SetData(smaz, "Local HMI", LB, 41, 1)
28 end if
```

Makro Simulace

Pokud chce uživatel simulovat nastavení nějaké hodnoty v napáječi přes obraz detailu napáječe, kterou standardně ovládat nelze, použije se toto makro. Uživatel stiskne tlačítko v detailu napáječe, tím dojde k převrácení bitové hodnoty příslušné obecné proměnné Nx_xxxx a zavolání makra Simulace, které hodnotu této proměnné již překopíruje do proměnné konkrétního napáječe. Ukázka překopírování jedné proměnné pro napáječ N1 se nachází na výpisu 4.17.

Výpis 4.17: Simulace proměnných napáječe

```
1 bool tmp, tmp2, N1
2 GetData(N1, "Local HMI", LB, 10, 1)
3
4 if N1 then
5   GetData(tmp, "MODBUS RTU", "N1_CNNOPovol", 1)
6   GetData(tmp2, "Local HMI", "Nx_CNNO", 1)
```

```
7     if tmp <> tmp2 then 'Prekopirovani zmenene hodnoty
8       SetData(tmp2, "MODBUS␣RTU", "N1_CNNOPovol", 1)
9     end if
10  end if
```

5 Návrh a realizace systému sběru a ukládání dat pomocí databázových serverů

Získávání dat z PLC a jejich následné ukládání do databází bude probíhat skrze SCADA software Promotic. Pomocí tohoto softwaru lze vytvářet aplikace pro řízení a vizualizaci různých procesů. Z databází, podporovaných Promoticem, jsem zvolil databázový SQL nástroj Firebird. Tento nástroj jsem zvolil, protože je z hlediska konfigurace a funkčnosti jednodušší než MS SQL Server a lze jej zdarma použít také pro komerční účely, aniž by byla nějak omezena jeho funkčnost. [52]

Aktuální hodnoty všech proměnných budou po vyčtení z PLC zapsány do datových objektů jednotlivých rozvaděčů přímo v rámci softwaru Promotic. Z těchto objektů již budou následně volány jednotlivé příkazy pro zápis do databáze na základě nastavených parametrů. Aplikaci budu vytvářet s využítím prototypů (PmaPrototype) a instancí (PmaInstance), aby ji bylo možné kdykoliv rozšířit o další měnírnu nebo rozvaděče bez potřeby velkých změn ve zdrojovém kódu.

5.1 Databázový nástroj Firebird

RDBMS Firebird vychází z databázového nástroje Borland InterBase 6.0. Jedná se o open source a lze jej využít zdarma také pro komerční účely. Funguje na všech hlavních operačních systémech. Pomocí Firebirdu lze vytvořit databáze o velikosti pár kB až několika GB, přičemž díky svým nízkým nárokům bude stále zachován dobrý výkon a databáze nebude ani náročná na údržbu. Existuje ve verzích Classic, Embedded a SuperServer. Verze Classic vytváří pro každé spojení jeden nezávislý proces. Verze SuperServer sdílí svou cache mezi jednotlivá spojení a využívá vlákna pro obsluhování každého spojení. Na operačním systému Windows lze spustit Firebird buď jako službu nebo jako klasickou aplikaci. K této databázi lze používat různé nástroje třetích stran, které obsahují různé grafické administrační nástroje nebo nástroje pro replikaci atd. [53]

Nástroj plně podporuje ACID transakce a uložené procedury a spouště (PSQL). Má TraceAPI, referenční integritu a multigenerační architekturu. Podporuje rovněž externí funkce (UDF knihovny). Umožňuje také rychlé obnovení bez potřeby transakčních logů. Přistupovat k databázi lze například přes dbExpress ovladače, ODBC, OLEDB, .NET, nativní/API a mnoha dalšími způsoby. Zálohy probíhají inkrementálně. Databáze obsahuje monitorovací a dočasné tabulky. Na spojení a transakce jsou použity triggery. Jsou zde rovněž plně implementovány kurzory v PSQL. [53]

5.2 Práce s databází v softwaru Promotic a technologie ADO

Pro přístup k databázím pomocí dostupných poskytovatelů připojení (ADO Provider) se používá technologie ADO od firmy Microsoft. Napojení na databázi se provádí přes objekt ADO Connection, který se definuje řetězcem ADO ConnectionString. Pomocí tohoto objektu lze realizovat příkazy SQL nad danou databází. V řetězci ADO ConnectionString se nachází jednotlivé parametry, jako je například poskytovatel připojení (provider), název databáze (database), adresa serveru (server), přihlašovací jméno a heslo (uid a pwd) atd., které umožňují objektu ADO Connection připojení k databázi skrze vybraného poskytovatele připojení. Každý ADO Provider určuje vlastní tvar a název těchto parametrů, ale kvůli kompatibilitě umí většina poskytovatelů zpracovat více různých názvů pro daný parametr. [54]

5.2.1 Objekt PmaAdo

Skrze něj se v Promoticu přistupuje k databázi s využitím technologie ADO. Díky němu jsou dostupné jednotlivé objekty a metody této technologie. Umožňuje také vykonávání příkazů SQL nad napojenou databází. Jeden objekt slouží k přístupu k jedné databázi, ale může obsahovat zároveň několik objektů AdoRecordset, které umožňují přístup k datům ve fyzických databázových tabulkách. Skrze objekt PmaAdo se rovněž zapisují a čtou data z fyzické databázové tabulky. Tento objekt obsahuje několik vlastností a metod. [55]

DbBeginTrans, DbCommitTrans a DbRollbackTrans

Metoda DbBeginTrans zahajuje novou transakci nad napojenou databází. Metoda DbCommitTrans potvrzuje transakci nad napojenou databází a metoda DbRollbackTrans ruší tuto transakci. Jejich syntaxe se nachází na výpisu 5.1. [56, 57, 58]

Výpis 5.1: Metody DbBeginTrans, DbCommitTrans a DbRollbackTrans [56, 57, 58]

```
1 'Obecna syntaxe
2 Object DbBeginTrans([String sParams])
3 Object DbCommitTrans([String sParams])
4 Object DbRollbackTrans([String sParams])
5
6 'Ukazka kodu
7 Dim oDb1, nErrVal
8 If 0 = oDb1.DbBeginTrans("return:map;").ErrorCode Then
9   'kod po uspesnem vykonani metody DbBeginTrans
```

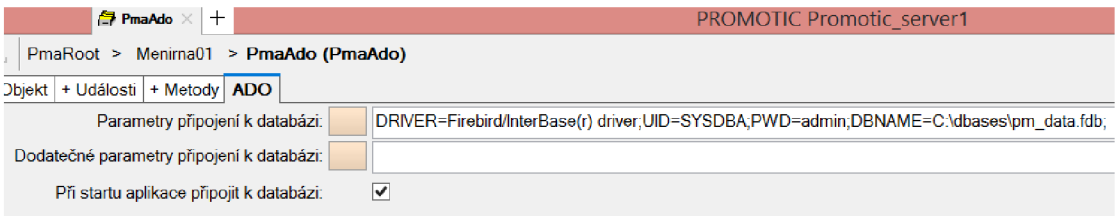
```
10
11    If nErrVal = 0 Then 'Test uspesneho vykonani
          predchoziho kodu
12      oDb1.DbCommitTrans
13    Else
14      Pm.Debug "Kod chyby: "&nErrVal
15      oDb1.DbRollbackTrans
16    End If
17  End If
```

Parametry v hranatých závorkách jsou nepovinné. Určují, zdali návratová hodnota metody má být mapa s více vrácenými hodnotami, nebo pouze jedna hodnota. Při vynechání těchto parametrů bude vracena pouze jedna prázdná hodnota (metoda bez návratové hodnoty). Mapa má následující návratové hodnoty: [56, 57, 58]

- Result - prázdná hodnota, metoda nic nevrátí
- ErrorCode - číselný kód chyby; pokud je hodnota 0, metoda proběhla úspěšně
- ErrorText - text chyby

DbOpen, DbConnectionString a DbConnectionParams

DbOpen je metoda pro připojení objektu PmaAdo k databázi. Připojení se realizuje přes objekt ADO Connection, který je součástí objektu PmaAdo. Parametry pro připojení k databázi se nachází ve vlastnostech DbConnectionString v podobě ADO ConnectionString a DbConnectionParams v podobě řetězce poté obsahuje dodatečné parametry připojení. Syntaxe této metody s ukázkou připojení objektu PmaAdoDb k databázi pm_menirna, fungující na MS SQL serveru, se nachází na výpisu 5.2. Je zde použit základní ADO Provider (SQLOLEDB) firmy Microsoft. V praxi není potřeba psát takto kód, ale stačí pouze zadat požadované parametry(vlastnosti) DbConnectionString a DbConnectionParams ve vlastnostech objektu PmaAdo. Ukázka takto zadaných parametrů pro mnou použitou databázi Firebird se nachází na obrázku 5.1. [59, 60, 61]



Obr. 5.1: Ukázka připojení databáze Firebird

Výpis 5.2: Metoda DbOpen a vlastnosti DbConnectionString a DbConnectionParams [59, 60, 61]

```
1  'Obecna syntaxe
2  Object DbOpen([String sParams])
3
4  'Ukazka kodu
5  Dim oDb1
6  Set oDb1 = pMe.Pm("/PmaAdoDb")
7  oDb1.DbConnectionString = "provider=SQLOLEDB;␣
       server=.\SQLEXPRESS;database=pm_menirna;␣
       uid=admin;pwd=pwadmin;"
8  oDb1.DbConnectionParams = ""
9  If Not oDb1.DbOpen() Then
10   'Kod, ktery se vykona pokud se pripojeni nezdari
11 End If
```

Parametry v hranatých závorkách jsou nepovinné. Určují, zdali návratová hodnota metody má být mapa s více vrácenými hodnotami, nebo pouze jedna hodnota. Při vynechání těchto parametrů bude vracena pouze jedna hodnota true v případě úspěšného připojení nebo false, pokud se připojení nezdařilo. Mapa má následující návratové hodnoty: [59]

- Result - příznak připojení k databázi; true - úspěšně připojeno, false - připojení selhalo
- ErrorCode - číselný kód chyby; pokud je hodnota 0, metoda proběhla úspěšně
- ErrorText - text chyby

Parametry vlastnosti DbConnectionString jsou následující (ale nemusí takto vypadat vždy, záleží na poskytovateli): [60]

- provider - říká, který ADO Provider se použije pro připojení k databázi
- server - jméno SQL serveru, na kterém běží databáze
- database - jméno databáze
- dsn - jméno zaregistrovaného ODBC DSN zdroje; pokud je uvedeno, není potřeba uvádět parametry server, database a driver
- driver - jméno ODBC ovladače
- uid - přihlašovací jméno pro připojení k databázi
- pwd - heslo

Parametry vlastnosti DbConnectionParams jsou následující: [61]

- connect - udává typ připojení (synchronní/asynchronní); implicitně je synchronní připojení, asynchronní připojení se nedoporučuje
- connectiontimeout - vyčkávání v sekundách na otevření spojení (hodnota je

přednastavena na 15 sekund)
- commandtimeout - vyčkávání v sekundách na provedení příkazu (hodnota je přednastavena na 30 sekund)

DbIsOpen

Tato metoda testuje, zdali je objekt PmaAdo připojen k databázi. Její syntaxe se nachází na výpisu 5.3. Metoda vrací hodnotu true, pokud je objekt připojen, nebo hodnotu false, pokud připojen není. [62]

Výpis 5.3: Metoda DbIsOpen [62]

```
1  'Obecna syntaxe
2  Boolean DbIsOpen()
3
4  'Ukazka kodu
5  Dim oDb1
6  Set oDb1 = pMe.Pm("/PmaAdoDb")
7  If Not pMe.DbIsOpen Then
8    pResult = pMe.DbOpen  'Pripojeni databaze pokud neni
       pripojena
9  Else
10   pResult = true
11 End If
```

DbClose

Slouží k odpojení připojeného objektu PmaAdo od databáze a zároveň uvolní všechny zapamatované objekty AdoRecordset. Jeho syntaxe se nachází na výpisu 5.4. [63]

Výpis 5.4: Metoda DbClose [63]

```
1  'Obecna syntaxe
2  Empty DbClose()
3
4  'Ukazka kodu
5  Dim oDb1
6  Set oDb1 = pMe.Pm("/PmaAdoDb")
7  oDb1.DbClose
```

DbExecute

Tato metoda slouží k provádění příkazů zadaných přes syntaxi jazyka SQL. Její syntaxe s příkladem přidání záznamu do tabulky se nachází na výpisu 5.5. [64]

Výpis 5.5: Metoda DbExecute [64]

```
1 'Obecna syntaxe
2 Object DbExecute(String sId, String sCommand, String
     sParams)
3
4 'Ukazka kodu
5 Dim oDb1
6 Set oDb1 = pMe.Pm("/PmaAdoDb")
7 oDb1.DbExecute "", "INSERT␣INTO␣table_name␣(name,␣value)␣
     VALUES␣('Voltage',␣230)", ""
```

Metoda má tři parametry: [64]

- sId - jednoznačný identifikátor, díky kterému jsou výsledná data v podobě objektu AdoRecordset zapamatována objektem PmaAdo; pokud je řetězec prázdný, objekt PmaAdo si je nezapamatuje; prázdný řetězec se rovněž uvádí, pokud daný příkaz nevrací žádná výsledná data
- sCommand - příkaz v jazyce SQL (například INSERT, SELECT, DELETE atd.)
- sParams - dodatečné parametry metody:
  - command - určuje, jak má ADO Provider vyhodnotit parametr sCommand (textový zápis, uložená procedura atd.)
  - execute - určuje, jak má ADO Provider provést dotaz SQL (synchronní, asynchronní atd.)
  - return - typ návratové hodnoty (mapa/jedna hodnota)

Pokud je návratová hodnota jedna hodnota, vrací se výsledná data v podobě objektu AdoRecordset, pokud dojde k chybě, tak se vrátí pouze hodnota Nothing (VBScript). Pokud se vrací objekt PmMap, tak má následující parametry: [64]

- Result - výsledná data ve stejném tvaru jako pro jednu hodnotu
- AffectedRows - počet řádků, kterých se týká daný příkaz v případě přidání, změny nebo smazání řádků
- ErrorCode - číselný kód chyby; pokud je hodnota 0, metoda proběhla úspěšně
- ErrorText - text chyby

LastErr a LastTextErr

LastErr je vlastnost objektu PmaAdo, která vrací číselný kód chyby poslední provedené metody nebo vlastnosti objektu PmaAdo. LastTextErr je poté její textový popis. Obě vlastnosti jsou pouze pro čtení. Jejich syntaxe jsou na výpisu 5.6. [65, 66]

Výpis 5.6: Vlastnosti LastErr a LastTextErr [65, 66]

```
1 'Obecna syntaxe
2 Long LastErr
3 String LastTextErr
4
5 'Ukazka kodu
6 Dim oDb1, nErr, sErr
7 Set oDb1 = pMe.Pm("/PmaAdoDb")
8 nErr = oDb1.LastErr
9 sErr = oDb1.LastTextErr
```

RsOpen

Metoda, která provádí SQL dotaz SELECT. Výsledná data jsou množinou záznamů v podobě objektu AdoRecordset. Pro provedení obecného SQL dotazu, který nevrací data jako objekt AdoRecordset, by se měla použít metoda DbExecute, jelikož u metody RsOpen nelze otestovat úspěšnost daného příkazu. Syntaxe metody se nachází na výpisu 5.7. [67]

Výpis 5.7: Metoda RsOpen [67]

```
1 'Obecna syntaxe
2 Object RsOpen(String sId, String sSource, String sParams)
3
4 'Ukazka kodu
5 Dim oDb1, oRs1
6 Set oDb1 = pMe.Pm("/PmaAdoDb")
7 Set oRs1 = oDb1.RsOpen("table_name", "SELECT␣*␣FROM␣
     table_name", "cursor:static;")
8 'Kod, ktery se ma nasledne vykonat
```

Metoda má tři parametry: [67]

- sId - jednoznačný identifikátor, díky kterému jsou výsledná data v podobě objektu AdoRecordset zapamatována objektem PmaAdo; pokud je řetězec prázdný, objekt PmaAdo si je nezapamatuje

- sSource - většinou dotaz v jazyce SQL (SELECT); může rovněž obsahovat název tabulky, volání procedury, název souboru s uloženým objektem AdoRecordset atd.
- sParams - dodatečné parametry metody:
  - cursor - určí typ kurzoru, který ADO Provider použije při otevření objektu AdoRecordset
  - lock - určí typ zámku, který ADO Provider použije při otevření objektu AdoRecordset
  - command - určuje, jak má ADO Provider vyhodnotit parametr sSource (textový zápis, uložená procedura atd.)
  - execute - určuje, jak má ADO Provider provést dotaz SQL (synchronní, asynchronní atd.)
  - return - typ návratové hodnoty (mapa/jedna hodnota)

Pokud je návratová hodnota jedna hodnota, vrací se výsledná data v podobě objektu AdoRecordset. Pokud dojde k chybě, tak se vrátí pouze hodnota Nothing (VBScript). Pokud se vrací objekt PmMap, tak má následující parametry: [67]

- Result - výsledná data ve stejném tvaru jako pro jednu hodnotu
- ErrorCode - číselný kód chyby; pokud je hodnota 0, metoda proběhla úspěšně
- ErrorText - text chyby

RsIsOpen

Metoda, která testuje, zdali existuje objekt AdoRecordset se zadaným identifikátorem. Syntaxe metody se nachází na výpisu 5.8. Metoda vrací hodnotu true, pokud objekt AdoRecordset se zadaným identifikátorem existuje a false, pokud neexistuje (nebyl otevřen nebo byl uvolněn metodou RsClose). [68]

Výpis 5.8: Metoda RsIsOpen [68]

```
1 'Obecna syntaxe
2 Boolean RsIsOpen(String sId)
3
4 'Ukazka kodu
5 Dim oDb1
6 Set oDb1 = pMe.Pm("/PmaAdoDb")
7 If oDb1.RsIsOpen("table_name") Then
8   'Dalsi kod
9 End If
```

RsGet

Metoda vrátí zapamatovaný objekt AdoRecordset se zadaným identifikátorem, který se vytvořil předchozím voláním metody RsOpen. Pokud objekt existuje, vrátí zapamatovaný objekt AdoRecordset s daným identifikátorem. Pokud objekt neexistuje, tak vrátí hodnotu Nothing (VBScript). Na výpisu 5.9 se nachází syntaxe této metody. [69]

Výpis 5.9: Metoda RsGet [69]

```
1 'Obecna syntaxe
2 Object RsGet(String sId)
3
4 'Ukazka kodu
5 Dim oDb1, oRs1
6 Set oDb1 = pMe.Pm("/PmaAdoDb")
7 Set oRs1 = oDb1.RsGet("table_name")
8 'Kod, ktery se ma nasledne vykonat
```

Parametr sId metody je jednoznačný identifikátor, pod kterým si PmaAdo zapamatoval objekt AdoRecordset. [69]

RsClose

Slouží k uvolnění zapamatovaného objektu AdoRecordset s daným identifikátorem. Pokud se vše správně uvolní, tak vrátí hodnotu true, pokud ne, tak vrátí hodnotu false. Na výpisu 5.10 se nachází syntaxe této metody. [70]

Výpis 5.10: Metoda RsClose [70]

```
1 'Obecna syntaxe
2 Boolean RsClose(String sId)
3
4 'Ukazka kodu
5 Dim oDb1
6 Set oDb1 = pMe.Pm("/PmaAdoDb")
7 oDb1.RsClose "table_name"
```

Parametr sId metody je jednoznačný identifikátor, pod kterým si PmaAdo zapamatoval objekt AdoRecordset. Pokud se zde uvede hodnota $all, tak se uvolní všechny AdoRecordset, které jsou zapamatovány. Při uvolnění AdoRecordset se uvolní zároveň také daný identifikátor. [70]

5.2.2 Objekt AdoRecordset

Patří pod technologii ADO od firmy Microsoft. Jedná se o množinu záznamů, která představuje výsledná data dotazu SQL. Počet záznamů v jednom objektu není pevně daný, ale všechny záznamy mají stejný tvar. Data jsou ve tvaru tabulky, jeden záznam představuje jeden řádek a sloupce se vytváří pomocí objektů AdoField. Maximálně jeden záznam může být v jeden moment označen jako aktuální, ale nemusí být jako aktuální označen žádný. Takto bývá označen záznam, se kterým se právě pracuje. Ty nejdůležitější vlastnosti objektu AdoRecordset jsou zobrazeny v seznamu na obrázku 5.2. Pro práci s tímto objektem existuje mnoho metod, seznam hlavních metod je zobrazen na obrázku 5.3. [71]

Objekt AdoRecord

Podle použitého ADO Providera může nastat situace, že výsledek dotazu SQL, který má pouze jeden řádek, nebude vrácen jako objekt AdoRecordset s jedním záznamem, ale jako objekt AdoRecord. Objekt AdoRecord reprezentuje jednořádkový výsledek SQL dotazu nebo jeden záznam objektu AdoRecordset. Tento objekt patří rovněž pod technologii ADO. Je vytvořen několika objekty AdoField a má následující vlastnosti a metody: [71, 72]

- Fields - vrátí objekt Collection, který obsahuje objekty AdoField
- Pm_LastErr - číselný kód chyby poslední provedené operace objektu AdoRecord
- Pm_LastTextErr - textový popis chyby poslední provedené operace objektu AdoRecord
- State - zjistí stav objektu AdoRecord (otevřený, zavřený atd.)

Objekt AdoField

V tomto objektu se nachází informace o sloupcích v objektech AdoRecordset a AdoRecord. Každý sloupec objektu AdoRecordset má svůj vlastní objekt AdoField. Obsahuje informace o celém sloupci, jako jsou například název nebo datový typ, a také informace o vlastní hodnotě aktuálního záznamu. V technologii ADO nese tento objekt název ADO Field. Obsahuje následující vlastnosti a metody: [73]

- Name - název sloupce
- Pm_LastErr - číselný kód chyby poslední provedené operace objektu AdoField
- Pm_LastTextErr - textový popis chyby poslední provedené operace objektu AdoField
- Value - hodnota sloupce aktuálního záznamu

AbsolutePage	Vrací nebo nastaví hodnotu, která určuje číslo stránky v objektu AdoRecordset.
AbsolutePosition	Vrací nebo nastaví hodnotu, která určuje pořadovou pozici aktuálního záznamu v objektu AdoRecordset.
ActiveCommand	Vrací objekt Command spojený s AdoRecordset.
ActiveConnection	Vrací nebo nastaví definici připojení, pokud je připojení uzavřeno, nebo aktuální Connection objekt, pokud je připojení otevřené.
BOF	Zjistí, zda pozice aktuálního záznamu je před prvním záznamem v objektu AdoRecordset
Bookmark	Vrací nebo nastaví záložku (bookmark). Záložka uloží pozici aktuálního záznamu
CacheSize	Vrací nebo nastaví počet záznamů, které lze uložit do mezipaměti.
CursorLocation	Vrací nebo nastaví umístění kurzoru.
CursorType	Vrací nebo nastaví typ kurzoru objektu AdoRecordset.
DataMember	Vrací nebo nastaví název datové oblasti, který se načte z objektu, na který odkazuje DataSource vlastnost.
DataSource	Určuje objekt obsahující data, která mají být reprezentována jako objekt AdoRecordset.
EditMode	Indikuje editační status aktuálního záznamu (hodnota ADO EditModeEnum)
EOF	Zjistí, zda pozice aktuálního záznamu je za posledním záznamem v objektu AdoRecordset
Fields	Vrací objekt Collection, který obsahuje objekty AdoField
Filter	Vrací nebo nastaví filtr pro data v objektu AdoRecordset.
Index	Vrací nebo nastaví název aktuálního indexu v objektu AdoRecordset.
LockType	Vrací nebo nastaví hodnotu, která určuje typ uzamčení při úpravách záznamu v objektu AdoRecordset.
MarshalOptions	Vrací nebo nastaví hodnotu, která určuje, které záznamy se mají vrátit na server.
MaxRecords	Vrací nebo nastaví maximální počet záznamů, které se mají vrátit na objekt AdoRecordset z dotazu.
PageCount	Vrací počet stránek s daty v objektu AdoRecordset.
PageSize	Vrací nebo nastaví maximální počet záznamů povolený na jedné stránce objektu AdoRecordset.
Pm_LastErr	Číselný chybový kód výsledku poslední provedené metody (vlastnosti) objektu AdoRecordset
Pm_LastTextErr	Textový popis výsledku poslední provedené metody (vlastnosti) objektu AdoRecordset
RecordCount	Vrací počet záznamů v objektu AdoRecordset
Sort	Vrací nebo nastaví názvy polí v AdoRecordset, na které se seřadí.
Source	Nastaví hodnotu řetězce nebo odkaz na objekt Command nebo vrátí String hodnotu, která označuje zdroj dat objektu AdoRecordset.
State	Zjistí, zda objekt AdoRecordset je otevřený, zavřený, připojující se, provádějící příkaz nebo čtoucí data
Status	Vrací stav aktuálního záznamu s ohledem na dávkové aktualizace nebo jiné hromadné operace.
StayInSync	Vrací nebo nastaví, zda se změní odkaz na podřízené záznamy, když se změní pozice nadřazeného záznamu.

Obr. 5.2: Vlastnosti objektu AdoRecordset [71]

5.3 Získávání dat z PLC a zpracování povelů

Vzhledem k tomu, že IEC 104 je změnový protokol, musí se po prvotním připojení serveru k PLC nebo po výpadku připojení vyslat z Promoticu na PLC generální dotaz, aby byla získána všechna aktuální data. Toto jsem učinil zavoláním komunikační zprávy MrDataRequest, která je součástí objektu IEC8705. Pomocí objektu IEC8705 zajišťuji komunikaci s PLC. Tento objekt se skládá z různých typů komunikačních zpráv. Po spojení s PLC se v něm vyvolá událost onConnect, ve které jsem vytvořil kód z výpisu 5.11 a která spustí časovač StartComm, jenž následně

AddNew	Začátek editačního režimu přidání záznamu
Cancel	Zruší provedení rozpracované metody.
CancelBatch	Zruší dávkovou aktualizaci.
CancelUpdate	Konec editačního režimu záznamu, zrušení změn
Clone	Vytvoří duplikát existujícího AdoRecordset.
Close	Zavře AdoRecordset.
CompareBookmarks	Porovná dvě záložky (bookmarks).
Delete	Smaže věty v objektu AdoRecordset
Find	Vyhledává záznam v objektu AdoRecordset, který splňuje daná kritéria
GetRows	Kopíruje více záznamů objektu AdoRecordset do 2-rozměrného pole
GetString	Vrací AdoRecordset jako řetězec.
Move	Přesun pozice aktuálního záznamu v objektu AdoRecordset o zadaný počet vět
MoveFirst	Přesun pozice aktuálního záznamu na první záznam v objektu AdoRecordset
MoveLast	Přesun pozice aktuálního záznamu na poslední záznam v objektu AdoRecordset
MoveNext	Přesun pozice aktuálního záznamu na následující záznam v objektu AdoRecordset
MovePrevious	Přesun pozice aktuálního záznamu na předcházející záznam objektu AdoRecordset
NextRecordset	Vymaže aktuální objekt AdoRecordset a vrátí další objekt AdoRecordset opakováním řady příkazů.
Open	Otevře databázovou oblast, která vám umožní přístup k záznamům v tabulce, výsledkům dotazu nebo k uloženému AdoRecordsetu.
Requery	Úplné opětovné načtení dat v objektu AdoRecordset
Resync	Obnovení existujících dat v objektu AdoRecordset
Save	Uloží objekt AdoRecordset do souboru nebo do objektu Stream.
Seek	Prohledá index objektu AdoRecordset a najde záznam, který odpovídá zadaným hodnotám.
Supports	Vrátí logickou hodnotu, která určuje, zda objekt AdoRecordset podporuje určitý typ funkce.
Update	Konec editačního režimu záznamu, provedení změn
UpdateBatch	Uloží všechny změny v objektu AdoRecordset do databáze. Používá se při práci v režimu dávkové aktualizace.

Obr. 5.3: Hlavní metody objektu AdoRecordset [71]

zajistí správné vykonání zprávy MrDataRequest. Jeho zdrojový kód se nachází na výpisu 5.12. Získaná data jsou datového typu variant a zapisují se do proměnných objektu Data v IEC8705 podle jejich datových rozšíření, ve kterých se nachází adresa příslušných proměnných z PLC. Následně tyto proměnné přiřazuji do proměnných konkrétních rozvaděčů pomocí události onDataReceive tohoto objektu.

Výpis 5.11: Událost onConnect objektu IEC8705

```
1 Pm(pMe.Parent.PathName&"/Data/#vars/KomunikacePLC").Value
    = true   'Komunikace s PLC
2 'Povoleni casovace pro MrDataRequest
3 Pm(pMe.Parent.PathName&"/StartComm").Enabled = true
4
5 'Priprava dat pro zapis do databaze
6 Dim sDtiPars, aValues(4), sTime, sPK, sSource, sName
7 sDtiPars = "ColNames:ID,Datum,Zdroj,Promenna,Popis;␣
```

```
   ColTypes:S,D,S,S,S;Table:Menirna01_Eventy;␣
   ColPkNames:ID;ColPkTypes:S;"
sTime = Pm.Time
sSource = "Menirna01"
sName = "Kom_IEC104"
sPK = CStr(sTime)&"_"&sSource&"_"&sName

aValues(0) = sPK
aValues(1) = sTime
aValues(2) = sSource
aValues(3) = sName
aValues(4) = "Navázána␣komunikace␣primárního␣
   databázového␣serveru␣s␣PLC"

'Ulozeni udalosti (spojeni s PLC) do databaze
Pm(pMe.Parent.Parent.PathName&"/PmaAdo").Methods.DtiOper
   sDtiPars, "Open", Null, Null, Null
Pm(pMe.Parent.Parent.PathName&"/PmaAdo").Methods.DtiOper
   sDtiPars, "AddRow", aValues, Null, Null
Pm(pMe.Parent.Parent.PathName&"/PmaAdo").Methods.DtiOper
   sDtiPars, "Close", Null, Null, Null
```

Výpis 5.12: Časovač StartComm

```
Dim oComm, sFolder

sFolder = pMe.Parent.PathName
Set oComm = Pm(sFolder&"/IEC8705")

If Pm(sFolder&"/IEC8705/MrDataRequest").GetReady() Then
   'Test pripravenosti k odeslani zpravy
  If oComm.IsConnected Then 'Test spojeni
    pMe.Enabled = False
  End If
  Pm(sFolder&"/IEC8705/MrDataRequest").Run  'Odeslani
End If
```

Další z komunikačních zpráv objektu IEC8705 je Povel45, kterou jsem vytvořil pro odesílání povelů ze serveru na PLC. Po přijetí povelu z dispečinku se vyvolá v datovém objektu příslušného rozvaděče událost onItemAfterWrite, ze které se ná-

sledně zavolá tato zpráva a zapíšou se do ní adresa PLC (CmnAddr), adresa dané proměnné v PLC (IoAddr1) a hodnota proměnné/povelu (Value1). Po zavolání se zpráva a povel vynulují. Syntaxe jejího volání se nachází na výpisu 5.13.

Výpis 5.13: Volání komunikační zprávy pro povel

```
1  If Right(sUnit, 1) = "p" AND vValue Then  'Novy povel
2    nAddr = pEvent.Item.Extension("sv").Value 'Ziskani
       adresy z podpromenne
3    sFolder = pMe.Parent.Parent.Parent.PathName
       &"/CommIEC8705_104/IEC8705" 'Nastaveni cesty
4    If Pm(sFolder&"/Povel45").GetReady() Then 'Test
       pripravenosti zpravy
5      Pm(sFolder&"/Povel45/#sndvars/IoAddr1").Value = nAddr
6      Pm(sFolder&"/Povel45/#sndvars/CmnAddr").Value = 2
7      Pm(sFolder&"/Povel45/#sndvars/Value1").Value = 1
8      Pm(sFolder&"/Povel45").Run  'Vyslani povelu
9
10     'Nulovani zpravy
11     Pm(sFolder&"/Povel45/#sndvars/IoAddr1").Value = 0
12     Pm(sFolder&"/Povel45/#sndvars/CmnAddr").Value = 0
13     Pm(sFolder&"/Povel45/#sndvars/Value1").Value = 0
14   End If
15
16   pEvent.Item.Value = false 'Nulovani povelu na serveru
17 End If
```

5.4 Zápis dat do databáze a jejich čtení

Před samotným ukládáním dat do databáze Firebird v ní bylo potřeba jednorázově vytvořit dané tabulky. Pro tento účel jsem využil metodu DbExecute z kapitoly 5.2.1, do které jsem vložil příkaz SQL, vytvářející tabulku s danými parametry. Použité příkazy se nachází na výpisu 5.14. Jedna tabulka je vyhrazena pro všechny události, které se odehrály na měnírně. Do druhé tabulky se ukládají pouze alarmy.

Výpis 5.14: Vytvoření tabulek v databázi

```
1  pMe.Methods.DbExecute("create␣table␣Menirna01_Eventy␣(ID␣
     nchar(150)␣PRIMARY␣KEY,␣Datum␣timestamp,␣Zdroj␣
     nchar(15),␣Promenna␣nchar(50),␣Popis␣nchar(250))")
```

```
2 pMe.Methods.DbExecute("create␣table␣Menirna01_Alarmy␣(ID␣
    nchar(150)␣PRIMARY␣KEY,␣Datum␣timestamp,␣Zdroj␣
    nchar(15),␣Promenna␣nchar(50),␣Popis␣nchar(250))")
```

Obě tabulky mají následující parametry:

- ID - jedinečný identifikátor (primární klíč) záznamu, který se skládá z data a času zápisu a názvu zapsané proměnné
- Datum - datum a čas zápisu záznamu (časové razítko)
- Zdroj - název rozvaděče, ze kterého pochází daný záznam
- Promenna - název dané proměnné v rozvaděči
- Popis - popis dané události/alarmu

Požadavek na vložení záznamu do databázové tabulky se nachází v události onItemAfterWrite, která je součástí každého datového objektu jednotlivých rozvaděčů v rámci softwaru Promotic, do nichž jsou zapisovány aktuální hodnoty proměnných z PLC. Tato událost se vyvolá při každé změně hodnoty libovolné proměnné a obsahuje všechny informace o dané proměnné, která tuto událost vyvolala. V rámci této události zpracovávám dané informace o proměnné a upravuji je do formátu pro následné vložení do databáze. Následně volám metodu DtiOper objektu PmaAdo, do níž předávám tři parametry. První parametr je sDtiPars. Tento parametr udává informace o databázové tabulce, do které bude probíhat zápis. Druhý parametr určuje požadovanou operaci v metodě a poslední parametr aValues obsahuje data pro zápis v daném tvaru.

Metoda DtiOper následně podle požadované operace zavolá vybranou metodu objektu PmaAdo, která již ze zadaných hodnot vytvoří SQL příkaz a případně ještě pomocí další metody převede data do tvaru pro databázi Firebird. Následně zavolá metodu DbExecute z kapitoly 5.2.1, předá jí sestavený SQL příkaz a ta vše uloží do databázové tabulky. Na výpisu 5.15 se nachází ukázka tvaru parametru sDtiPars a volání metody DtiOper a následně výňatky kódu z jednotlivých metod použitých pro zápis záznamu.

Výpis 5.15: Výňatky kódu pro zápis do databáze

```
1 'Datovy objekt rozvadece
2 sDtiPars = "ColNames:ID,Datum,Zdroj,Promenna,Popis;␣
    ColTypes:S,D,S,S,S;Table:"&sMenirna&sType&";␣
    ColPkNames:ID;ColPkTypes:S;"
3 Pm(pMe.Parent.Parent.Parent.PathName
    &"/PmaAdo").Methods.DtiOper sDtiPars, "Open", Null,
    Null, Null
4 Pm(pMe.Parent.Parent.Parent.PathName
    &"/PmaAdo").Methods.DtiOper sDtiPars, "AddRow",
```

```
   aValues, Null, Null
Pm(pMe.Parent.Parent.Parent.PathName
   &"/PmaAdo").Methods.DtiOper sDtiPars, "Close", Null,
   Null, Null

'Case metody DtiOper pro pridani radku
Case "AddRow"
    aColNames = Pm.StringSplit(mDtiPars.ColNames,
       ",", "empty:1")
    aColTypes = Pm.StringSplit(mDtiPars.ColTypes,
       ",", "empty:1")
    pResult   =
       pMe.Methods.TableInsertRow(mDtiPars.Table,
       aColNames, vPar1, aColTypes)

'Metoda TableInsertRow (vytvoreni SQL prikazu pro
   pridani radku)
Dim i
For i = 0 To UBound(aTypes, 1)
  aValues(i) = pMe.Methods.ValueToSqlString(aTypes(i),
     aValues(i))
Next

If pMe.Methods.IsDbOpen() Then
  Dim sSQL, mMapRs, oRs
  sSQL       = "INSERT␣INTO␣" & sTable & "␣(" &
     Pm.StringJoin(aCols, ",") & ")␣VALUES␣(" &
     Pm.StringJoin(aValues, ",") & ");"
  Set mMapRs = pMe.DbExecute("", sSQL,
     "return:map;lock:optimistic;") 'Vykonani SQL prikazu
  pResult    = mMapRs.AffectedRows
End If
```

Vyčtení záznamů z databáze pro jejich následné uložení do tabulky událostí a alarmů na klientském pracovišti probíhá podobným způsobem jako jejich vložení. Do metody DtiOper se místo parametru AddRow předává parametr GetRows, sloužící pro operaci vyčtení. Tuto operaci jsem v metodě DtiOper upravil tak, aby dokázala vybírat pouze 50 záznamů od určitého data, nastaveného uživatelem, a nebyly tak vyčítány veškeré záznamy z databázové tabulky. Výsledná úprava se nachází

na výpisu 5.16.

Výpis 5.16: Výňatek kódu pro vyčtení záznamů z databáze

```
1 Case "GetRows"
2     DatumPred =
          Pm(pMe.Parent.PathName&"/Data/Data/#vars/Datum")
3     Datum = pMe.Methods.ValueToSqlString("D", DatumPred)
4     sSql = "SELECT␣" & mDtiPars.ColNames & "␣FROM␣" &
          mDtiPars.Table & "␣WHERE␣Datum>"&Datum&"␣ORDER␣BY␣
          Datum␣ROWS␣50"
5     pResult = pMe.Methods.TableGetRows(sSql)
```

5.5 Replikace databáze mezi primárním a záložním serverem

Za normálního stavu jsou oba servery napojeny na PLC a získávají z něj data nezávisle na sobě. Aktuální data poté zapisují do svých datových objektů pro jednotlivé rozvaděče, ale pouze primární server následně tato data ukládá navíc i do své databáze jako události nebo alarmy. Dispečerské pracoviště je za normálního stavu také napojeno na oba servery, ale aktuální data a události s alarmy získává pouze z primárního serveru, na který rovněž odesílá zpět povely pro ovládání měnírny. Pokud dojde k výpadku primárního serveru nebo výpadku spojení mezi tímto serverem a PLC nebo dispečinkem, veškeré role tohoto serveru převezme záložní server. Aby na záložním serveru byla identická data jako na primárním, musel jsem vytvořit replikaci těchto databázových serverů.

Firebird přímo v sobě nemá zabudovaný nástroj pro replikaci svých databází, proto jsem musel tyto databáze propojit se speciálním nástrojem třetí strany, který zajistí jejich replikaci. Pro tento účel jsem použil software SymmetricDS, popsaný v kapitole 5.5.1. V tomto softwaru jsem nastavil propojení mezi servery na master-master, aby byla možná synchronizace v obou směrech, pokud by došlo k výpadku primárního serveru a data by se tak zapisovala do databáze na záložním serveru. Dále jsem v programu nastavil IP adresy obou serverů, jejich názvy (engine.name), synchronizační adresu (sync.url), zařadil je do skupiny (group.id) a nastavil další parametry pro databáze jako je cesta, oprávnění atd. V databázích se následně nastavily synchronizační spouštěče (triggers), přes které SymmetricDS zajišťuje replikaci. V příloze na CD se nachází konfigurační soubory pro oba databázové servery.

5.5.1 SymmetricDS

Software SymmetricDS se používá pro replikaci databází. Jedná se o tzv. open source. Umožňuje replikovat databáze jednosměrně nebo oběma směry. Synchronizace databází může být spouštěna asynchronně, nebo může probíhat téměř v reálném čase. Mezi jeho výhody patří, že po obnovení komunikace dokáže synchronizovat data, která byla zapsána do databází během výpadku sítě, kdy nebyla možná okamžitá synchronizace. Umí pracovat na sítích s omezenou šířkou pásma. Dokáže mezi sebou propojit také databáze jiných značek (například Oracle, SQLite, MS SQL Server, MySQL, MariaDB, PostgreSQL atd.). Je založen na jazyce java, takže funguje na většině operačních systémů. Díky vydaným API ho lze rozšířit o různé funkce, řídit pomocí vzdáleného přístupu nebo implementovat přímo do aplikací. [74]

6 Návrh a realizace dispečerského SCADA systému

Pro tvorbu klientského pracoviště použiji software Promotic, ve kterém se realizují SCADA aplikace pro současné měnírny. Aplikace klientského pracoviště bude rozdělena na několik částí. Po spuštění aplikace se bude zobrazovat okno Přehled, ve kterém bude nakreslena mapa jednotlivých úseků a u každého z těchto úseků bude objekt, představující příslušnou měnírnu. Po kliknutí na daný objekt se již zobrazí přehled dané měnírny, na kterém se bude nacházet zapojení jednotlivých usměrňovačů a napáječů. Na horní liště bude možné přepínat se mezi jednotlivými okny měnírny, jako je její přehled, VN část, vlastní spotřeba a zpětná část. Dále se po kliknutí na libovolný usměrňovač nebo napáječ v přehledu zobrazí okno s jejich detailem.

Po kliknutí na jednotlivé prvky se zobrazí malé okno s nabídkou (povely) pro daný prvek. Tyto povely se následně vyšlou na databázový server, kde se zpracují. Řízení vlastností jednotlivých grafických prvků bude probíhat přímo v rámci klientské aplikace. Aplikace bude rovněž zajišťovat spojení s databázovými servery, realizovat přenos dat a určovat, který ze serverů se má aktuálně použít. Aplikaci budu vytvářet s použitím prototypů (PmaPrototype) a instancí (PmaInstance), aby mohla být v případě potřeby jednoduše rozšiřitelná o další měnírny nebo rozvaděče, aniž by bylo potřeba velkých změn.

6.1 Příjem dat z databázových serverů a odesílání povelů

6.1.1 Navázání spojení

Před začátkem přijímání aktuálních dat a databázových dat ověřuji komunikaci s oběma servery. Toto provádím přes časovač s názvem Komunikace s periodou 7 s, který volá objekt Sequencer_kom (PmaSequencer), sloužící k sekvenčnímu zpracování operací. Pomocí objektu Sequencer vytvářím nové pracovní vlákno s menší prioritou, v němž se již nachází mnou vytvořený kód pro test spojení. Ukázku kódu pro primární server lze vidět na výpisu 6.1. Do proměnné Komunikace_aktualni se ukládá stav komunikace (komunikace s primárním/záložním serverem, přepojování na druhý server, bez komunikace) a používá se k následné informaci uživatele. Implementace kódu v novém pracovním vlákně byla provedena z důvodu zamrzání aplikace, které způsobovala Pm metoda NetTestPC, pokud se nacházela v hlavním vlákně. Přenesením této metody do nového vlákna byly tyto problémy vyřešeny.

Pokud není komunikace navázána, dojde k zablokování prohlížení událostí a alarmů měnírny, jelikož tato data nejsou aktuálně dostupná.

Výpis 6.1: Test spojení se servery

```
1 'Test komunikace s primárním serverem
2 If Pm.NetTestPC(1, "192.168.1.10", 5000) Then
3   Pm("/Data/#vars/Komunikace_primarni").Value = true
4 Else
5   Pm("/Data/#vars/Komunikace_primarni").Value = false
6   Pm("/Data/#vars/Komunikace_aktualni").Value = 3
7 End If
```

6.1.2 Příjem dat

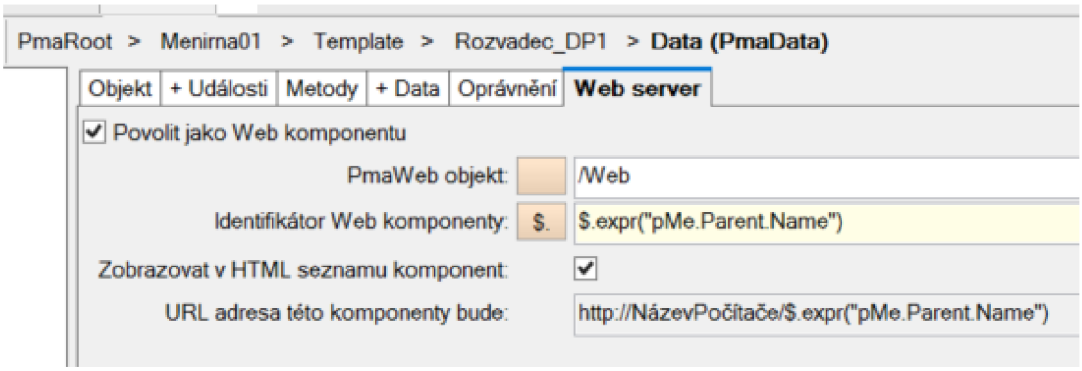
Příjem dat (historie událostí a alarmů na měnírně) z databází probíhá přímo přes objekty PmaAdo_prim a PmaAdo_zal (každý databázový server musí mít svůj vlastní PmaAdo), do jejichž parametrů pro připojení k databázi jsem nastavil IP adresy vzdálených databázových serverů. Objekt PmaAdo je popsán v kapitole 5.2.

Aktuální data jsou získávána přímo z Promotic aplikace, běžící na databázových serverech. Nejdříve bylo potřeba serverovou aplikaci změnit na Web server přidáním objektu Web (PmaWeb) a nastavit jeho parametry. Následně jsem připojil všechny datové objekty k tomuto objektu a tím povolil vzdálený přístup k proměnným těchto objektů metodou sdílení XML dat. Ukázka prostředí pro připojení k Webu se nachází na obrázku 6.1. V aplikaci klientského pracoviště (funguje jako Web klient) bylo potřeba vytvořit identické datové objekty, do kterých se ukládají proměnné z Web serveru. Pro čtení všech dat jsem vytvořil časovač ReadData, který s periodou 1 s vyčítá všechna data ze serveru a ukládá je automaticky do datových objektů dispečerské aplikace. Ukázka syntaxe pro čtení dat z primárního serveru a jejich uložení do datového objektu usměrňovače GU1 se nachází na výpisu 6.2. Nejdřív proběhne dotaz, zdali je Web klient připraven, a následně se spustí čtení z datového objektu na vzdáleném serveru. Parametrem fmt=purevalue metody ReadFromWeb určuji, že se mají vyčíst hodnoty všech proměnných datového objektu.

Výpis 6.2: Čtení dat z primárního serveru

```
1   Set oData = Pm(pMe.Parent.PathName&"/Data/GU1/Data")
2   If oData.WebClientIsReady Then
3     oData.ReadFromWeb
        "http://192.168.1.10/gu1/data.xml?fmt=purevalue",
        "oper", ""
4     Pm("/Data/#vars/Komunikace_aktualni").Value = 1
```

```
5    Else
6       'Pm.Debug "Web klient neni pripraven"
7    End If
```



Obr. 6.1: Propojení datových objektů s Webem

6.1.3 Odesílání povelů

Pokud uživatel zadá povel (například pro vypnutí vypínače), zavolá se mnou vytvořená metoda WriteData, do které předám zdroj povelu (do kterého datového objektu povel patří) a název požadované proměnné (povelu). V metodě se zvolí, na který server bude povel odeslán ke zpracování a prostřednictvím Web klienta se povel odešle. Nejdříve se pro Web klienta nastaví zdroj dat (datový objekt), následuje dotaz na připravenost zdroje dat a poté již dojde k samotnému zapsání povelu na Web server. Syntaxe metody se nachází na výpisu 6.3. Zde používám v metodě WriteToWeb parametr fmt:full, přes který této metodě oznamuji, že budu posílat pouze konkrétní proměnnou a její hodnotu a následně do parametru vars uložím název této proměnné, která se vyčte z datového objektu. Pokud se povel nepodaří odeslat, zobrazí se okno z obrázku 6.15 s upozorněním na tuto skutečnost.

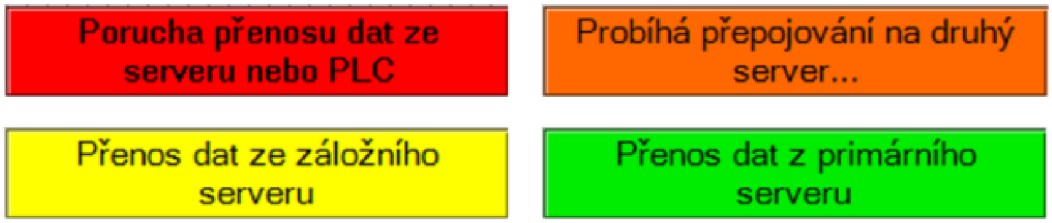
Výpis 6.3: Odeslání povelu na Web server

```
1  Dim oData
2
3  Set oData = Pm(pMe.PathName&"/"&sSource&"/Data")
4
5  If Pm("/Data/#vars/Komunikace_primarni") AND
      Pm(pMe.Parent.PathName
      &"/KomunikacePLC/#vars/KomunikacePLC") Then
6    If oData.WebClientIsReady Then
```

```
7       oData.WriteToWeb
           "http://192.168.1.10/"&sSource&"/data.xml?",
           "fmt:full;vars:"&sParam&";user:oper;"
8     Else
9       Pm("/Workspace").OpenView "/Povel_odpoved",
           "target:_blank;modal:1;scrollbar:0;"
10    End If
11  ElseIf Pm("/Data/#vars/Komunikace_zalozni") AND
       Pm(pMe.Parent.PathName
       &"/KomunikacePLC/#vars/KomunikacePLC") Then
12    If oData.WebClientIsReady Then
13      oData.WriteToWeb
           "http://192.168.1.20/"&sSource&"/data.xml?",
           "fmt:full;vars:"&sParam&";user:oper;"
14    Else
15      Pm("/Workspace").OpenView "/Povel_odpoved",
           "target:_blank;modal:1;scrollbar:0;"
16    End If
17  Else   'Nefunkcni komunikace
18    Pm("/Workspace").OpenView "/Povel_odpoved",
         "target:_blank;modal:1;scrollbar:0;"
19  End If
```

6.2 Grafická okna

V horní části všech oken se nachází lišta s několika tlačítky a různými informacemi. Tuto lištu lze vidět například na obrázku 6.3. První tlačítko Přehled dostane uživatele na mapu jednotlivých úseků a základní přehled všech měníren, zobrazený na obrázku 6.3. Druhé tlačítko Login slouží k přihlášení uživatele. Pomocí třetího tlačítka Logout se může uživatel odhlásit. Další tlačítko je Info, skrze které se přistupuje do INFO systému a poslední tlačítko Stop slouží k zastavení aplikace a jeho aktivace je omezena pouze na oprávněné uživatele. Následně se na liště nachází informace o aktuálně přihlášeném uživateli, aktuální datum a čas a stav spojení s primárním a záložním databázovým serverem, s informacemi o aktuálním přenosu dat. Pokud servery komunikují, tak je zobrazen zelený obdélník s jejich názvem. Pokud dojde k výpadku serveru, tak se obdélník daného serveru probarví červeně. Informace o přenosu dat ze serveru má čtyři stavy, jež jsou zobrazeny na obrázku 6.2.



Obr. 6.2: Stav přenosu dat ze serveru

6.2.1 Přehled

Nachází se zde schéma trakčního vedení s rozdělením jednotlivých úseků a příslušných měníren. Toto okno se zobrazuje při spuštění aplikace a lze se na něj rovněž dostat po kliknutí na tlačítko Přehled na levé straně horní lišty. Jeho vzhled je zobrazen na obrázku 6.3. Každý úsek a objekt měnírny se probarvuje podle jejích aktuálních stavů, které jsou v okně rovněž popsány. Zde jsou některé úseky záměrně v poruše nebo bez napětí a komunikace, aby bylo vidět jednotlivé stavy. Po kliknutí na objekt měnírny se uživatel dostane do přehledu konkrétní měnírny. Pokud na měnírně dojde k poruše, zobrazí se uživateli okno alarmů z obrázku 6.4 s danou poruchou. Toto okno se může zobrazit v libovolném obraze. Červená barva zde znamená, že porucha je stále aktivní, modrá barva znamená, že porucha již zanikla. Kód pro vyvolání tohoto okna se nachází v datovém objektu každého rozvaděče. Pro zobrazení stavu poruchy na měnírně a jejích jednotlivých prvcích jsou na PLC záměrně simulovány různé typy poruch.

6.2.2 Přehled měnírny

Po kliknutí na objekt měnírny v obraze Přehled se otevře přehled dané měnírny, zobrazený na obrázku 6.5 a nahoře v obrazu se objeví lišta pro přepínání mezi jednotlivými sekcemi měnírny. Na této liště se také nachází název této měnírny, tlačítka pro zobrazení historie událostí a alarmů, zapnutí a vypnutí blokace zemní ochrany a havarijní stop tlačítko. Přehled měnírny se skládá z přehledu všech poruch na měnírně v pravé části obrazu (aktivní porucha je zvýrazněna červenou barvou), usměrňovače (GU1), napáječů (RUV.Tx) a jejich vzájemného propojení. Každý z těchto rozvaděčů kromě jejich jednopólového schématu obsahuje také obecné informace o sobě, jako je jejich název a název ulice, kterou napájí, aktuální stav, režim ovládání, hodnotu proudu a teploty, stav vypínačů, odpojovačů a vozíku (vysunutý vozík lze vidět např. u napáječe RUV.T2).

Elektrické vedení mezi jednotlivými prvky může mít několik barev, které charakterizují jeho stav. V případě ztráty komunikace se veškeré vedení probarví fialově

Obr. 6.3: Okno Přehled

Obr. 6.4: Okno alarmů.

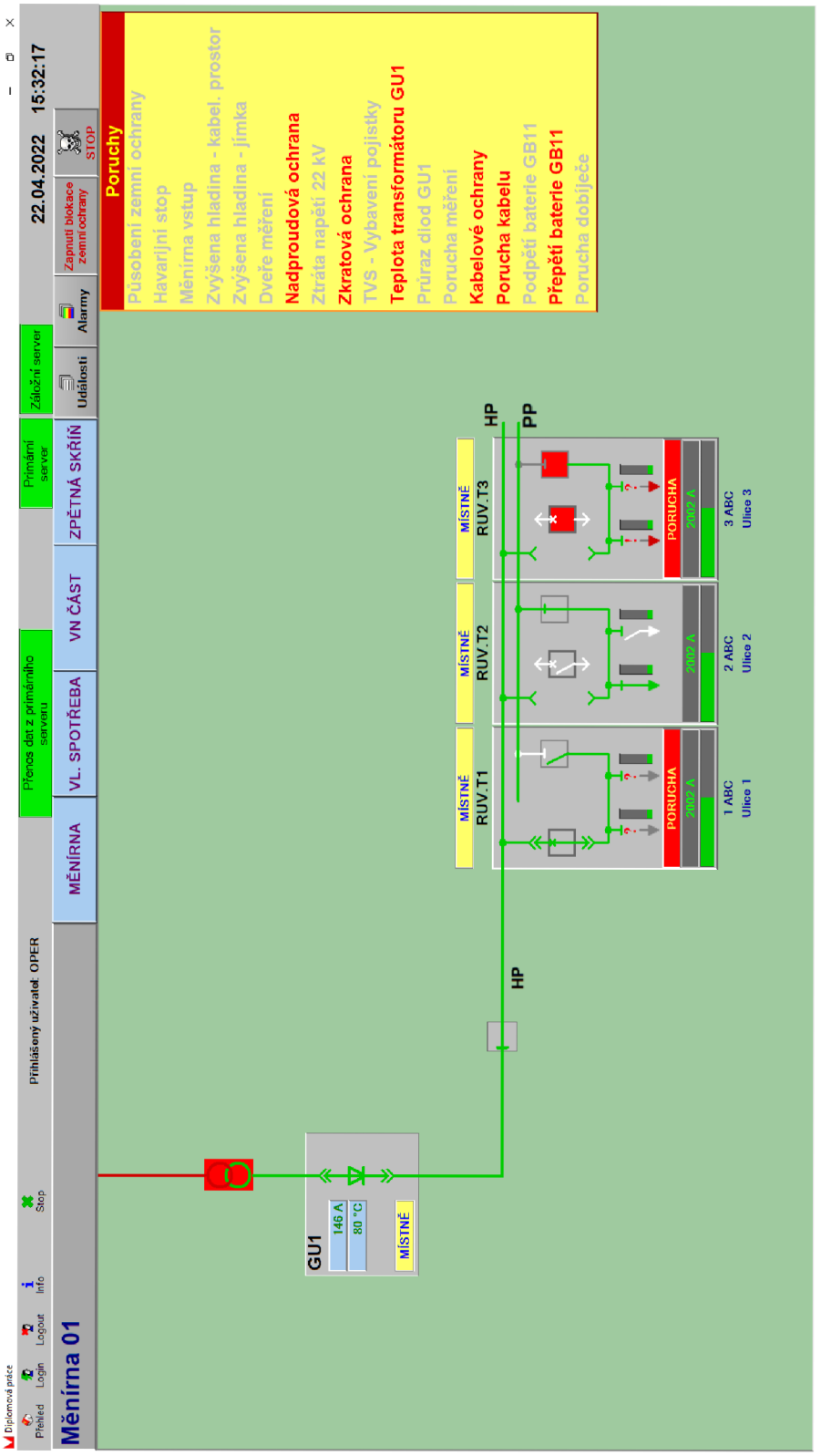
(obrázek 6.6). Pokud je některý z vypínačů nebo odpojovačů v mezipoloze, je jeho barva červená (například u napáječe RUV.T3) a vedení za tímto prvkem bude šedé barvy (neurčitý stav), protože ne na všech místech jsou voltmetry pro měření napětí, takže zde nelze určit, zdali je vedení pod napětím. Pokud je vedení bez napětí, tak je bílé barvy a pokud je vedení pod napětím, tak je tmavě červené barvy pro VN část a zelené barvy pro stejnosměrnou část. Takto se probarvují vedení, vypínače a odpojovače ve všech obrazech měnírny.

6.2.3 VN část

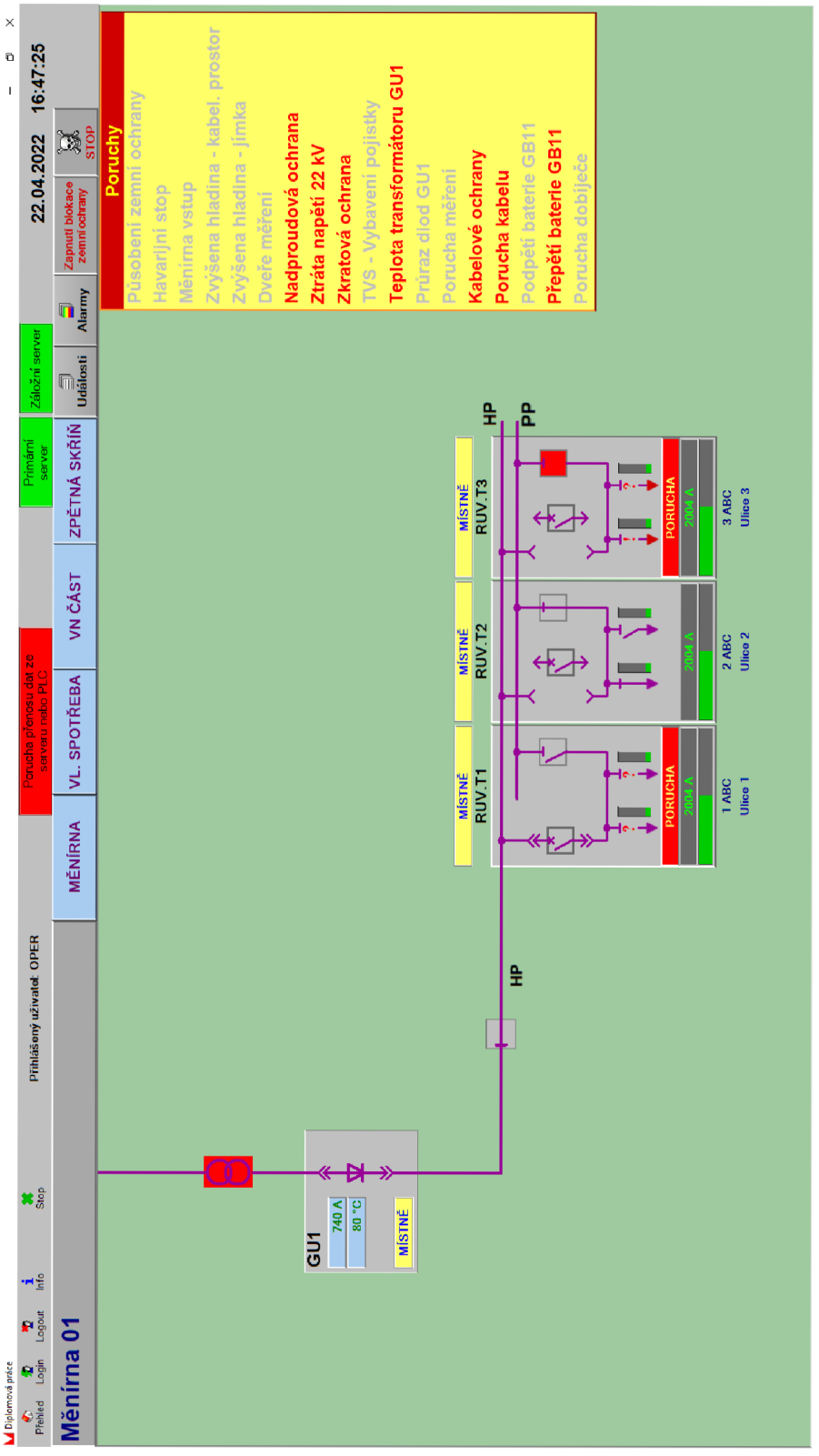
Pokud uživatel klikne v horní liště na tlačítko VN část, zobrazí se mu obraz VN části měnírny, zobrazený na obrázku 6.7. V levé části se nachází přívod napětí 22 kV přes podélnou spojku (PS), které je následně rozvětveno a přivedeno na trakční transformátor TU1, kde je transformováno na menší napětí pro usměrňovač GU1 a druhá větev je přivedena na transformátor vlastní spotřeby (TVS). Pokud by transformátor TU1 dosáhl nastavené hranice pro zvýšenou teplotu, jeho okolí by se probarvilo žlutě a vyhlásila by se porucha a pokud by jeho teplota následně dosáhla k nastavené hodnotě havarijní teploty, jeho okolí by změnilo barvu na červenou. Podélná spojka a přívody k jednotlivým transformátorům se skládají každý z jednoho odpojovače a jednoho VN vypínače, umístěného v sérii. Vypínače jsou motorické, a tak je lze oproti ručním odpojovačům ovládat vzdáleně pomocí povelů (vypnout/zapnout). Pokud jsou přívody nebo podélná spojka ovládány místně, zobrazí se u nich žlutý obdélník s modrým nápisem MÍSTNĚ a v případě poruchy se zobrazí červený obdélník s žlutým nápisem PORUCHA. Na pravé straně obrazu se nachází přehled všech poruch, týkajících se VN části.

6.2.4 Vlastní spotřeba

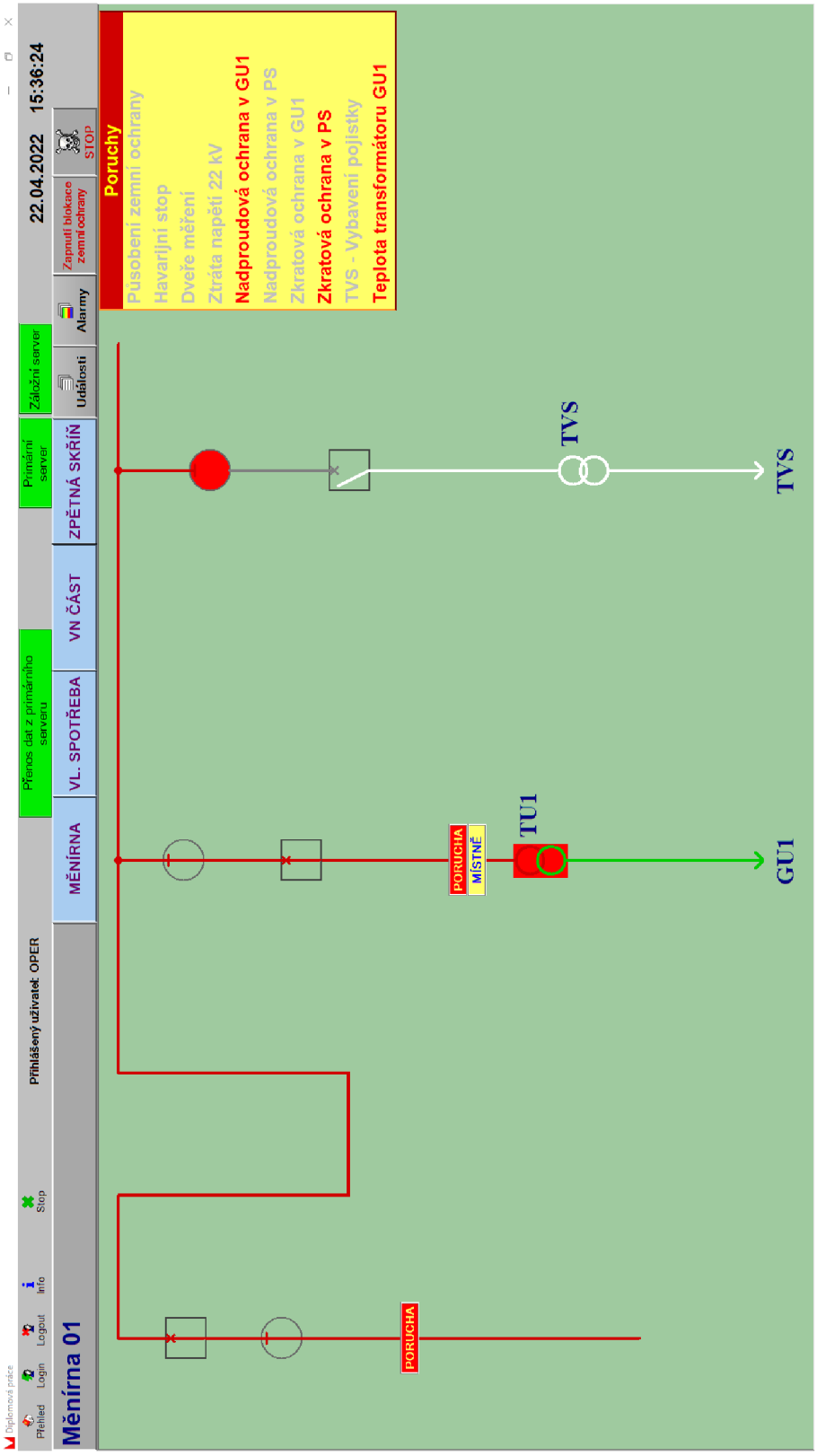
Po otevření obrazu vlastní spotřeby (VLSP) přes horní menu měnírny (tlačítko VL. SPOTŘEBA) se zobrazí její jednopólové schéma se dvěma rozvaděči - RVS1 a RU1. Vzhled obrazu je zobrazen na obrázku 6.8. Rozvaděč RVS1 slouží jako zdroj pomocného napájení 230/400 V AC pro ostatní rozvaděče (například pro napájení servomotorů sloužících k vysunutí vozíku). Rozvaděč RU1 se používá pro napájení technologií, požadujících stejnosměrné napětí 24 V. V rozvaděči RU1 se nachází baterie GB11, u které se v obraze zobrazuje její aktuální hodnota napětí a proudu. Pokud je pozadí baterie nebo diody ZND1 v RU1 zeleně, jsou tyto prvky v pořádku, červená barva znamená poruchu. Kromě toho je vlevo nahoře indikátor, zdali jsou rozvaděče aktuálně řízeny místně nebo dálkově. V pravé části obrazu se nachází poruchová signalizace pro VLSP. Na obrázku 6.8 lze vidět, že byla aktuálně simulována porucha přepětí baterie GB11.



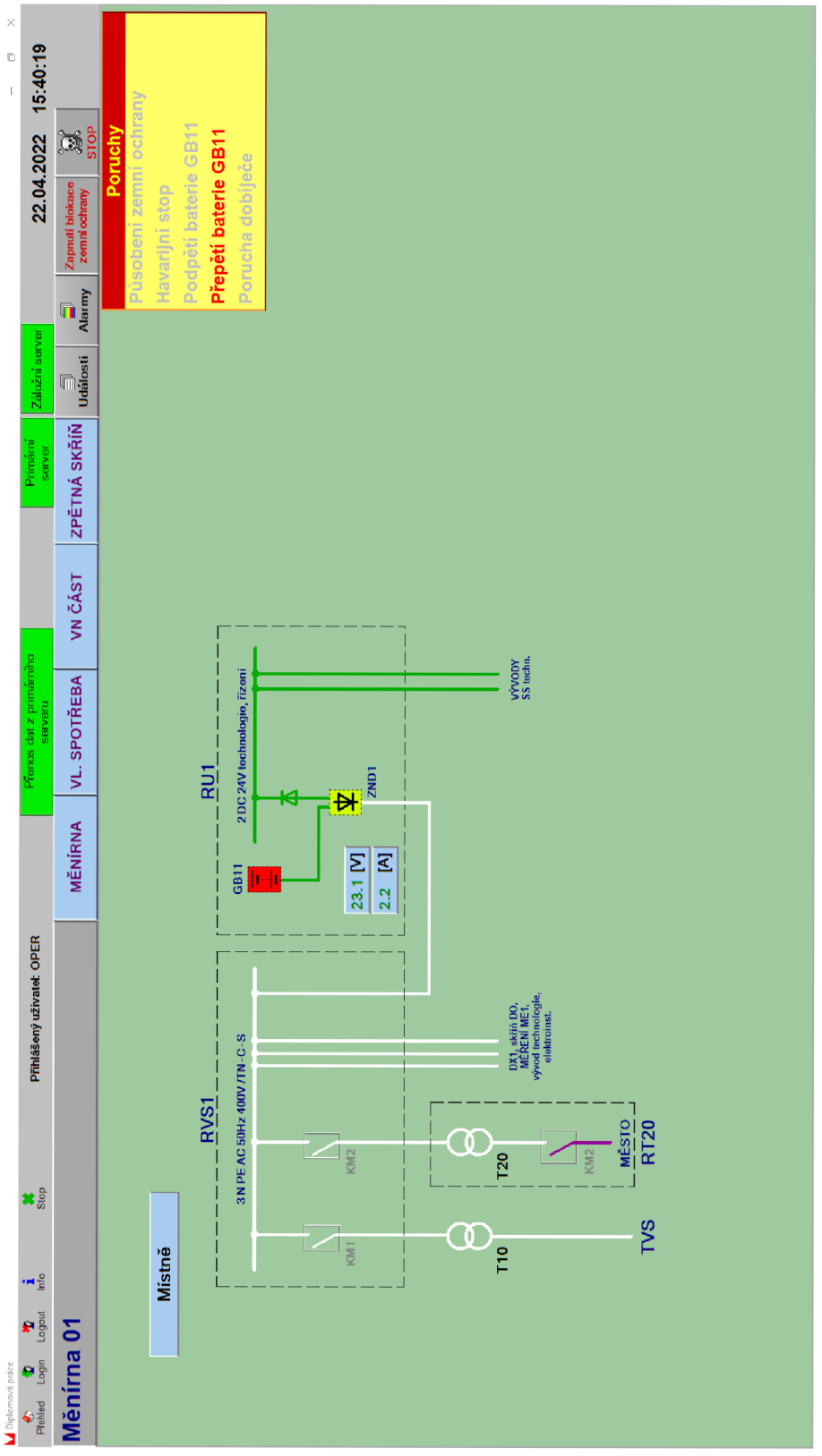
Obr. 6.5: Přehled měnírny



Obr. 6.6: Ztráta komunikace na měnírně



Obr. 6.7: VN část



Obr. 6.8: Vlastní spotřeba

6.2.5 Zpětná část

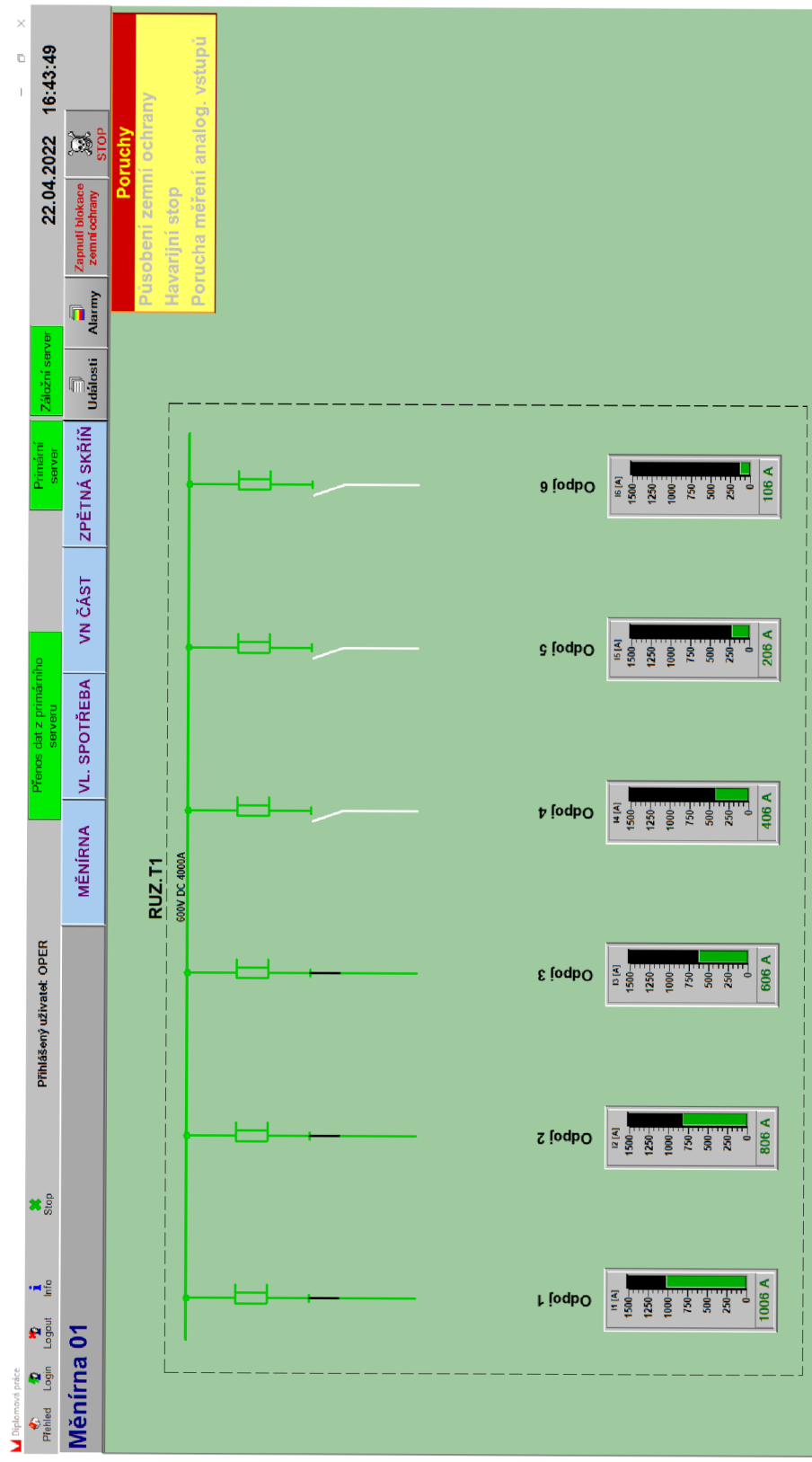
Obraz s jednopólovým schématem zpětných kabelů RUZ.T1 se zobrazí po kliknutí na tlačítko měnírny ZPĚTNÁ SKŘÍŇ. Jeho vzhled je zobrazen na obrázku 6.9. Je zde vidět šest zpětných kabelů s jejich názvy a hodnotou proudu na jednotlivých kabelech. Každý z těchto kabelů má jeden kabelový odpojovač, který zobrazuje simultánní stav podle nastavení obsluhy. Toto nastavení se provádí pomocí konfiguračního souboru baseset.ini aplikace klientského pracoviště a nemusí představovat aktuální skutečný stav technologie. V pravé části obrazu se nachází poruchová signalizace, týkající se zpětné části.

6.2.6 Detail usměrňovače

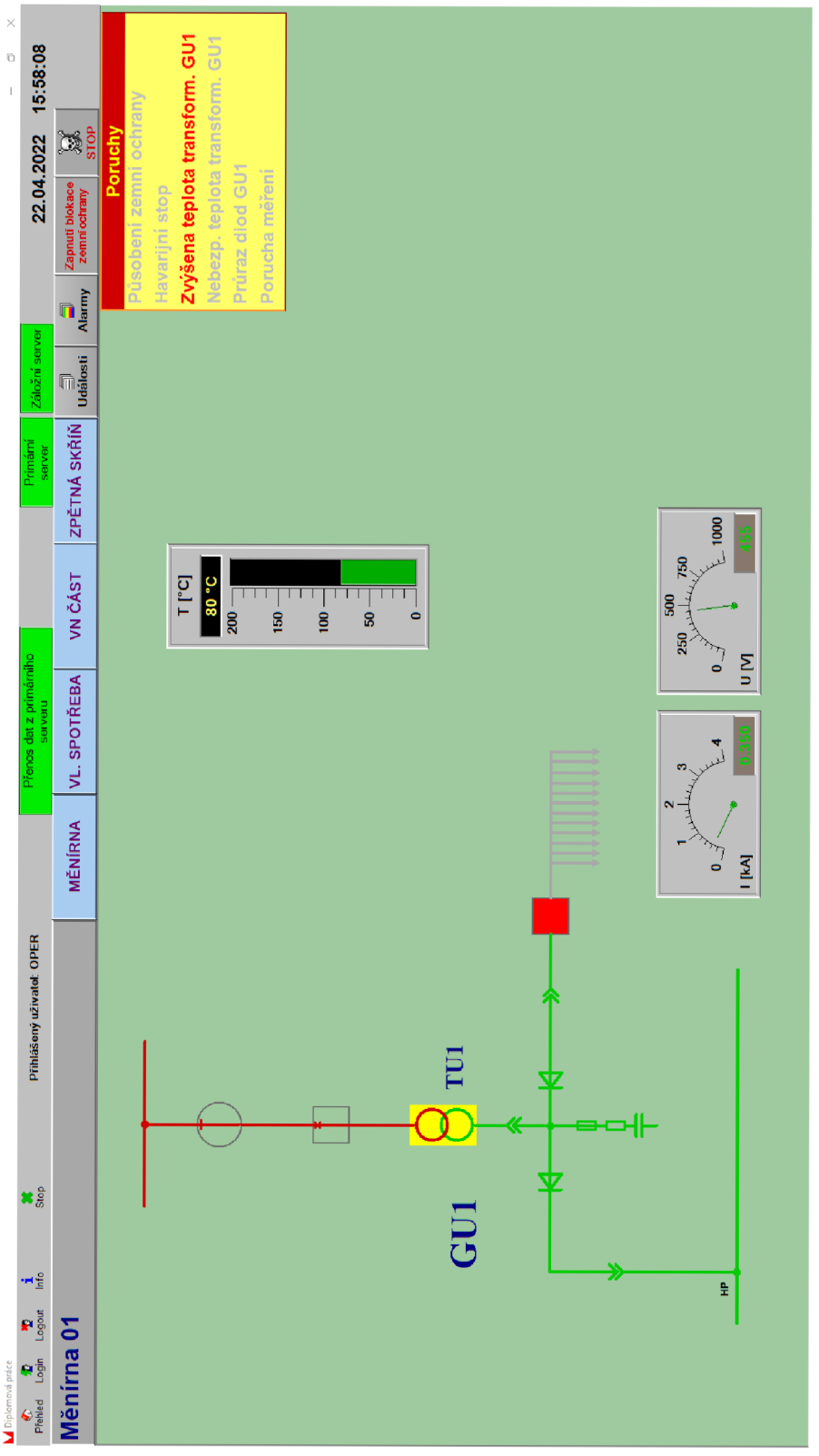
Pokud si uživatel chce zobrazit detail usměrňovače, stačí kliknout v přehledu měnírny na jeho rozvaděč. Poté se otevře okno, zobrazené na obrázku 6.10. Nachází se zde nakreslené jednopólové schéma usměrňovače s přívodem z VN části. Vozík usměrňovače je v tomto případě zasunut a pod napětím, v případe vysunutí vozíku se střed usměrňovače oddělí od okolního vedení a pokud na něj nebude přivedeno testovací napětí, tak se probarví bílou barvou (stav bez napětí). Odpojovač na zpětné skříně je na obrázku v mezipoloze. Vývod z usměrňovače je připojen na hlavní přípojnici (HP), která dále pokračuje přes odpojovač HP (lze jej vidět v přehledu měnírny - obrázek 6.5), ovládaný z usměrňovače GU1, na jednotlivé napáječe. Pro transformátor TU1 je zde indikátor zvýšené a havarijní teploty ve formě probarvení pozadí transformátoru a indikátor aktuální naměřené teploty. Dále se v tomto obraze nachází indikátory aktuálního naměřeného napětí a proudu na usměrňovači. V pravé horní části obrazu se nachází poruchová signalizace pro tento usměrňovač (aktuálně lze vidět zvýšenou teplotu transformátoru TU1, který má žluté pozadí).

6.2.7 Detail napáječe

Po kliknutí na rozvaděč napáječe v přehledu měnírny se zobrazí obraz s detailem daného napáječe, který je zobrazen na obrázku 6.11. V tomto případě se jedná o napáječ RUV.T2. Je zde nakresleno jeho jednopólové schéma s indikátory hodnoty proudu a napětí na napáječi. V dolní části napáječe se nachází kabelové odpojovače s indikátory hodnoty proudu, který jimi prochází. Pod kabelovými odpojovači se nachází malé trojúhelníkové obrazce, které, kromě standardních nastavených barev pro vedení, mohou změnit barvu na červenou, pokud by došlo k zapůsobení kabelových ochran na vývodu. Na obrázku 6.11 lze vidět, že hlavní přípojnice (HP) a napáječ jsou aktuálně pod napětím a jelikož je z napáječe sepnut odpojovač na pomocnou přípojnici (PP), tak je i tato přípojnice pod napětím. Vozík napáječe je v tomto



Obr. 6.9: Zpětné skříně



Obr. 6.10: Detail usměrňovače

případě vysunut a bez napětí. Kabelový odpojovač K1 se nachází v sepnutém stavu a kabelový odpojovač K2 je rozepnutý. Ve spodní části obrazu se nachází tabulka se seznamem aktivních ochran (ČZNO a ČNNO), obě ochrany jsou aktuálně neaktivní. Seznam poruch, týkajících se daného napáječe RUV.T2, je umístěn v pravé horní části obrazu.

6.2.8 Okno událostí a alarmů

Uživatel si může zobrazit historii událostí a alarmů na měnírně pomocí tlačítek Události a Alarmy na horní liště měnírny. Pro události se zobrazí okno z obrázku 6.12. Pro alarmy se zobrazí okno z obrázku 6.13. Jedná se o identické okno, ale podle zvoleného tlačítka se mu nastavují parametry tak, aby zobrazil buď alarmy, nebo události. Při otevření okna se zobrazí výchozí hodnoty - prvních 50 událostí/alarmů od data před sedmi dny. Uživatel může prohlížet události a alarmy jiného data výběrem požadovaného data přes tlačítko Změnit datum. Pokud je ztraceno spojení se servery, nelze tato okna zobrazit.

6.2.9 Povely jednotlivých prvků

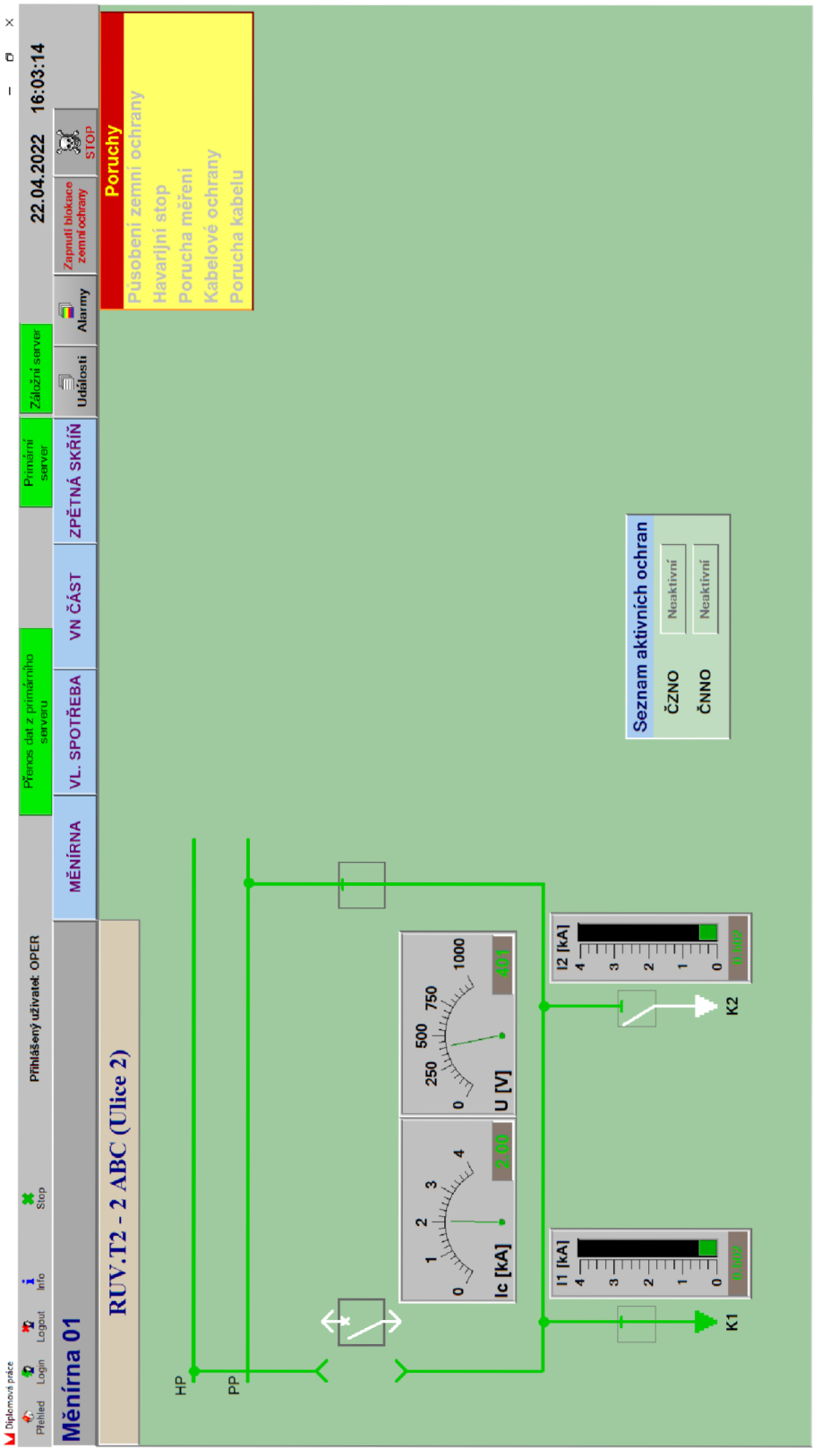
Pokud chce uživatel změnit stav některého z vypínačů, stačí na něj dvojklikem poklepat a zobrazí se mu okno z obrázku 6.14. Popis v tomto okně se mírně liší podle toho, na který vypínač uživatel klikl. V tomto případě se jedná o rychlovypínač (RV) napáječe N1 (RUV.T1). Uživatel má na výběr ze tří možností - zapnout, vypnout a storno. Pokud je vypínač například v zapnutém stavu jako na obrázku 6.14, je povel pro zapnutí zašedlý a nelze na něj kliknout.

Pokud povel nelze aktuálně vykonat, například z důvodu ztráty komunikace se serverem nebo PLC a nebo pokud je webový klient aktuálně zaneprázdněn, zobrazí se uživateli upozornění z obrázku 6.15 a uživatel tak musí povel vykonat znovu.

6.3 Řízení vlastností grafických prvků

Grafické prvky v obrazech jsou řízeny několika způsoby. Každý rozvaděč má svůj vlastní speciální datový objekt o názvu Napeti, který se skládá z několika proměnných (proměnné jsou pojmenovány písmeny abecedy) a každé písmeno odpovídá určité sekci příslušného elektrického vedení. Pomocí ukládaní daného čísla do těchto proměnných se v obrazech následně mění barva vedení. Čísla jsou následující:

- 0 = ztráta komunikace (fialová);
- 1 = neurčitý stav (šedá);
- 2 = bez napětí (bílá);

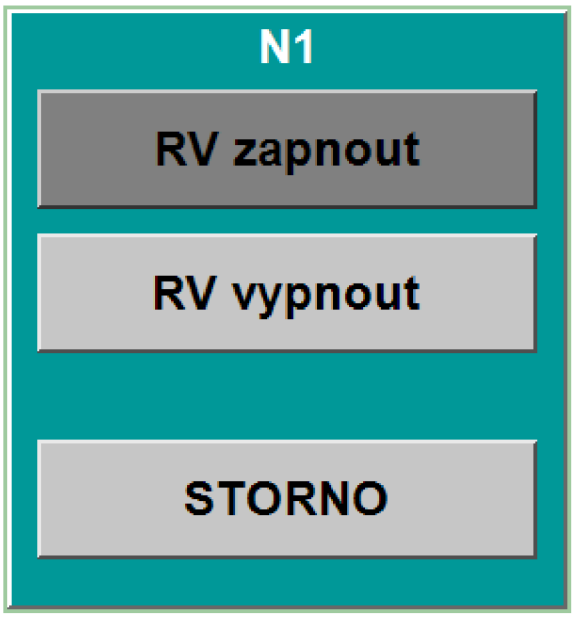


Obr. 6.11: Detail napáječe

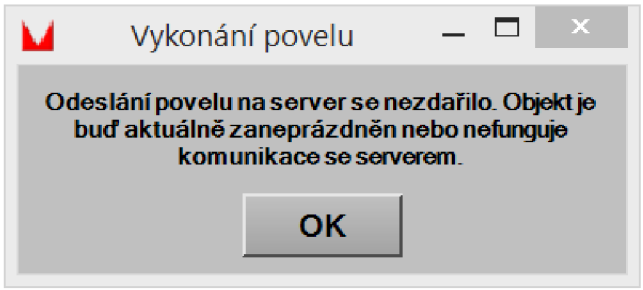
Obr. 6.12: Okno událostí

ID	Datum	Zdroj	Proměnná	Popis
20. 4. 2022 10:59:42_N2_PorMereni	20.04.2022 10:59:42	N2	PorMereni	Porucha měření analogových vstupů
20. 4. 2022 10:59:42_N1_PorMereni	20.04.2022 10:59:42	N1	PorMereni	Porucha měření analogových vstupů
20. 4. 2022 10:59:31_GU1_PorMereni	20.04.2022 10:59:31	GU1	PorMereni	Porucha měření na analogových vstupech
20. 4. 2022 10:59:31_N3_PorMereni	20.04.2022 10:59:31	N3	PorMereni	Porucha měření analogových vstupů - konec
20. 4. 2022 10:59:31_N2_PorMereni	20.04.2022 10:59:31	N2	PorMereni	Porucha měření analogových vstupů - konec
20. 4. 2022 10:59:31_N1_PorMereni	20.04.2022 10:59:31	N1	PorMereni	Porucha měření analogových vstupů - konec
20. 4. 2022 10:59:26_N3_PorMereni	20.04.2022 10:59:26	N3	PorMereni	Porucha měření analogových vstupů
20. 4. 2022 10:59:26_N2_PorMereni	20.04.2022 10:59:26	N2	PorMereni	Porucha měření analogových vstupů
20. 4. 2022 10:59:26_N1_PorMereni	20.04.2022 10:59:26	N1	PorMereni	Porucha měření analogových vstupů
20. 4. 2022 10:49:28_RVS1_PodpetiGB11	20.04.2022 10:49:28	RVS1	PodpetiGB11	Podpětí na baterii
20. 4. 2022 10:49:24_N3_PorMereni	20.04.2022 10:49:24	N3	PorMereni	Porucha měření analogových vstupů - konec
20. 4. 2022 10:49:24_N2_PorMereni	20.04.2022 10:49:24	N2	PorMereni	Porucha měření analogových vstupů - konec
20. 4. 2022 10:49:24_N1_PorMereni	20.04.2022 10:49:24	N1	PorMereni	Porucha měření analogových vstupů - konec
20. 4. 2022 10:49:14_N3_PorMereni	20.04.2022 10:49:14	N3	PorMereni	Porucha měření analogových vstupů
20. 4. 2022 10:49:14_N2_PorMereni	20.04.2022 10:49:14	N2	PorMereni	Porucha měření analogových vstupů
20. 4. 2022 10:49:14_N1_PorMereni	20.04.2022 10:49:14	N1	PorMereni	Porucha měření analogových vstupů
20. 4. 2022 10:47:42_GU1_PorMereni	20.04.2022 10:47:42	GU1	PorMereni	Porucha měření na analogových vstupech
20. 4. 2022 10:47:37_N3_PorMereni	20.04.2022 10:47:37	N3	PorMereni	Porucha měření analogových vstupů
20. 4. 2022 10:47:37_N2_PorMereni	20.04.2022 10:47:37	N2	PorMereni	Porucha měření analogových vstupů
20. 4. 2022 10:47:37_N1_PorMereni	20.04.2022 10:47:37	N1	PorMereni	Porucha měření analogových vstupů

Obr. 6.13: Okno alarmů



Obr. 6.14: Okno pro povel



Obr. 6.15: Upozornění na nevykonání povelu

- 3 = pod napětím (zelená/tmavě červená).

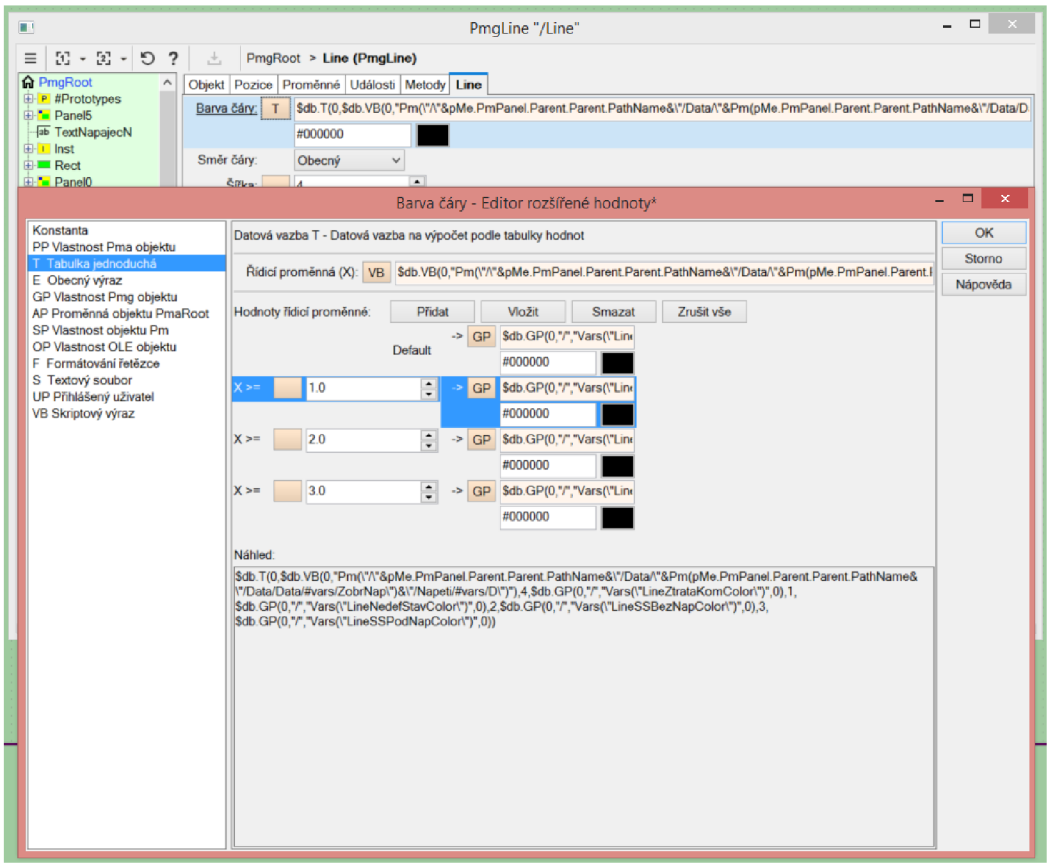
Nastavování těchto čísel probíhá v události onTick časovače Aktualizace (objekt PmaTimer) s periodou 1 s. V této události se zjišťuje nejdříve stav komunikace a následně se zjišťují stavy jednotlivých proměnných, které informují, zdali je nějaká část vedení pod napětím (v daném úseku se nachází voltmetr). Poté se zjišťuje stav jednotlivých vypínačů a odpojovačů a podle toho se nastaví příslušná hodnota proměnné v následujícím úseku za těmito vypínači/odpojovači. Pokud v úseku není žádný voltmetr, jeho hodnota se určí z hodnoty předchozího úseku a stavu oddělovacího prvku mezi nimi. Ke zjednodušení určování hodnot pro úseky, oddělené vypínači a odpojovači, se z událostí volá univerzální metoda Nap_spinac, do které se předá stav daného prvku a hodnota předchozího úseku a metoda následně vrátí hodnotu pro požadovaný úsek. Volání metody a její syntaxe se nachází na výpisu 6.4.

Událost onTick časovače Aktualizace obsahuje také kód pro nastavení barvy úseku a objektu měnírny v hlavním přehledu.

Výpis 6.4: Volání a syntaxe metody Nap_spinac

```
1  'Volani metody
2  Pm(sFold&"/E").Value =
       Pm(pMe.Parent.PathName).Methods.Nap_spinac(Pm(sFolder
       &"/Odpoj_Zapnut"), Pm(sFolder&"/Odpoj_Vypnut"),
       Pm(sFold&"/D"))
3
4  'Metoda Nap_spinac
5  '0 - ztráta komunikace
6  '1 - neurčitý stav
7  '2 - bez napětí
8  '3 - pod napětím
9
10 If (vyp AND NOT zap) OR predch_vet = 2 Then
11   pResult = 2
12 ElseIf zap AND NOT vyp Then
13   pResult = predch_vet
14 Else
15   pResult = 1
16 End If
```

Jednodušší kód lze psát přímo do jednotlivých grafických prvků. Každý grafický prvek má svou vlastní sadu událostí (například stisknutí tlačítka nebo klávesy, změna hodnoty, otevření obrazu atd.) a také prostor pro vytváření metod. Kromě toho lze kód psát také přímo do parametrů grafického prvku místo přímého zápisu hodnoty. Tyto vlastnosti jsem používal například pro nastavování viditelnosti jednotlivých prvků, definování barev objektů, čtení a nastavování názvů objektů z konfiguračního souboru a mnoho dalšího. Na obrázku 6.16 lze vidět ukázku tohoto prostředí, v tomto případě se jedná o příklad s nastavením barvy čáry (elektrického vedení).



Obr. 6.16: Ukázka prostředí pro nastavení vlastností grafických prvků

Závěr

V této práci jsem na začátku stručně vysvětlil, jak fungují dispečerská tabla a co to jsou měnírny. Následně jsem udělal průzkum firem, které vytváří dispečerské řízení měníren. Z nich jsem vybral dvě firmy s rozdílnými řešeními a popsal jimi používané technologie. První popsaná firma ZAT používá pro řídicí systémy své vlastní komponenty a svůj vlastní systém SandRA, nebo rovněž pracuje se systémy ostatních výrobců. Pro tvorbu centrálních dispečinků používá svůj systém SandRA nebo SCADA software Wonderware, Cimplicity, Reliance a Control Web. Druhá firma APEL používá pro dispečerské řízení jejich vlastní mozaikovou stavebnici dispečerského panelu, která se skládá z různých hardwarových komponentů sloužících pro různé účely, takže ji lze skládat do různých velikostí a podle různých požadovaných funkčností. Pro řízení technologie a mozaiky používá firma APEL svůj vlastní řídicí a informační systém QIRS, anebo se k dispečerské mozaice lze připojit také z jiných systémů (například Control Web). Kromě toho jsem v kapitole 1.4 popsal také technologii firmy OHLA ŽS, se kterou jsem spolupracoval a jejíž komponenty a software jsem používal.

Ve druhé části práce jsem navrhl strukturu celkového systému řízení a popsal další možná rozšíření. Bylo zde potřeba vytvořit simulaci měnírny. Pro to jsem zvolil PLC, na kterém funguje daná měnírna a HMI panel, který simuluje její lokální řízení. Z firmy jsem pro tyto účely obdržel PLC Teco Foxtrot CP-1003 a HMI Weintek MT6070iE. Klientské pracoviště a databázové servery běží na klasických počítačích, protože nemám k dispozici specializovaný hardware. Navržená struktura se nachází na obrázku 2.1.

Kromě navržené struktury celkového systému řízení jsem také navrhl a definoval strukturu komunikačních rozhraní pro centrální dispečink MHD a měnírny a popsal jsem jednotlivé použité komunikační rozhraní a protokoly. Pro spojení klientského pracoviště s databázovými servery a následně také s PLC jsem zvolil Ethernet a na něm komunikační protokol IEC 104 na úseku mezi PLC a databázovými servery. Ke spojení databázových serverů s klientským pracovištěm jsem musel, po konzultaci s firmou a se souhlasem vedoucího, nakonec vybrat metodu sdílení XML dat softwaru Promotic, protože zde protokol IEC 104 nebyl podporován. Pro spojení PLC s panelem HMI jsem zvolil připojení přes sériovou linku a rozhraní RS-485. Pro toto spojení jsem kvůli jednoduchosti, spolehlivosti a zachování jednotného způsobu propojení zvolil komunikační protokol Modbus RTU. Výsledná struktura komunikačních rozhraní je zobrazena na obrázku 3.1.

Na PLC jsem vytvořil přibližnou simulaci měnírny. Standardně má každý rozvaděč na měnírně své vlastní PLC. Vzhledem k tomu, že jsem měl k dispozici pouze jedno PLC, tak jsem místo toho pro zdrojový kód jednotlivých rozvaděčů vytvořil

jejich vlastní oddělené programy (POU), popsané v kapitole 4.1. Pro simulaci měřených hodnot jsem si vytvořil funkční bloky, které mi generovaly v určitých časových intervalech hodnoty. K implementaci IEC 104 na PLC jsem využil funkci z knihovny firmy TECO pro tento protokol a tuto funkci jsem si následně upravil podle svých potřeb. Pro samotné odesílání dat a příjem povelů jsem musel vytvořit několik polí dat, se kterými následně v PLC pracuji. Lokální řízení funguje na HMI panelu, kam jsem přidal funkci simulace některých proměnných, které standardně ovládat nelze, ale jsou řízeny výstupy ze snímačů. Protože jsem měl k dispozici pouze malý HMI panel, tak jsem na něm po konzultaci s firmou nevytvářel jednopólová schémata, jak se to aktuálně dělá u měníren, kde jejich lokální řízení běží na PC. Místo toho jsem stavy jednotlivých prvků zobrazoval pomocí jednoduchých indikátorů. V kapitole 4.2.2 lze vidět jednotlivá okna vytvořené vizualizace. Chování vizualizace jsem řídil s využitím maker.

Pro realizaci databázových serverů jsem se rozhodl použít software Promotic, který získává data z PLC a následně je zpracuje a uloží do databáze. Pro vytvoření databází jsem zvolil databázový nástroj Firebird, který je zdarma také pro komerční účely, aniž by byla nějak omezena jeho funkčnost. V Promoticu se k databázím přistupuje přes objekt PmaAdo s využitím technologie ADO firmy Microsoft. Nejdříve jsem v kapitole 5.2.1 popsal tento objekt a jeho vlastnosti s metodami. Následně jsem napsal, jak získávám data z PLC pro databázové servery a jak na serverech zpracovávám povely. V předposlední části se nachází způsob zápisu dat do databáze a jejich následné čtení pro klientské pracoviště. Vzhledem k tomu, že Firebird v sobě nemá zabudovanou replikaci svých databází, tak jsem se v poslední části kapitoly věnoval jejich replikaci s využitím externího softwaru SymmetricDS.

V poslední části této práce jsem navrhl a realizoval dispečerskou SCADA aplikaci pro klientské pracoviště. Nejdříve jsem popsal zpracování dat ze serveru s odesíláním povelů zpět na server, poté jsem v kapitole 6.2 ukázal a popsal jednotlivá okna aplikace a nakonec jsem popsal řízení jejich vlastností. U tvorby grafických oken jsem se snažil, aby technologie a schémata v nich zobrazená vzhledově odpovídala technologii a schématům v aplikacích aktuálních měníren.

V této diplomové práci se mi povedlo splnit všechny body zadání a vytvořit tak kompletní dispečerské řízení měnírny MHD. Po spojení jednotlivých částí do jednoho celku jsem celou práci úspěšně otestoval a vše je funkční. V příloze na CD se nachází krátké video, ve kterém tuto funkčnost demonstruji. Na tomto videu se nachází obraz z HMI panelu a aplikace klientského pracoviště, kde postupně zkouším různé povely jak v aplikaci, tak na HMI panelu a na obou stranách lze pozorovat jejich úspěšné vykonání. Následně na videu zobrazuji jednotlivá okna vizualizace HMI panelu a SCADA aplikace, včetně přístupu k datům z databází.

Literatura

[1] ZAT: *Produkty a řešení* [online]. c2021 [cit. 2021-10-06]. Dostupné z: https://www.zat.cz/cz/produkty-a-reseni.htm

[2] ZAT: *Automatizace technologických procesů: Inteligentní průmyslové systémy* [online]. c2021 [cit. 2021-10-6]. Dostupné z: https://www.zat.cz/cz/automatizace-technologickych-procesu.htm#scrollTarget=tbs&active=dalsi-prumyslove-podniky

[3] ZAT: *Řídicí systém SandRA* [online]. c2021 [cit. 2021-10-06]. Dostupné z: https://www.zat.cz/cz/ridici-system-sandra.htm

[4] ZAT: *Procesní stanice řady SandRA Z200* [online]. c2021 [cit. 2021-10-09]. Dostupné z: https://www.zat.cz/cz/procesni-stanice-rady-sandra-z200.htm

[5] ZAT: *Procesní stanice řady SandRA Z210* [online]. c2021 [cit. 2021-10-09]. Dostupné z: https://www.zat.cz/cz/procesni-stanice-rady-sandra-z210.htm

[6] ZAT: *Procesní stanice řady SandRA Z100: SandRA Z101 a Z102* [online]. c2021 [cit. 2021-10-09]. Dostupné z: https://www.zat.cz/cz/procesni-stanice-rady-sandra-z100.htm

[7] ZAT: *Stanice vzdálených vstupů a výstupů* [online]. c2021 [cit. 2021-10-16]. Dostupné z: https://www.zat.cz/cz/stanice-vzdalenych-vstupu-a-vystupu.htm#scrollTarget=tbsh&active=tab-2

[8] ZAT: *Speciální přístroje pro energetiku řady SandRA Z110* [online]. c2021 [cit. 2021-10-16]. Dostupné z: https://www.zat.cz/cz/specialni-pristroje-pro-energetiku-rady-sandra-z110.htm

[9] ZAT: *Vizualizace a dispečerská pracoviště* [online]. c2021 [cit. 2021-10-16]. Dostupné z: https://www.zat.cz/cz/vizualizace-a-dispecerska-pracoviste.htm

[10] ZAT: *Projektová databáze Pertinax* [online]. c2021 [cit. 2021-10-17]. Dostupné z: https://www.zat.cz/cz/projektova-databaze-pertinax.htm

[11] ZAT: *Projektová databáze Pertinax a systémy HMI/SCADA v řídicích systémech ZAT.* Automa [online]. 2013(11) [cit. 2021-10-17]. Dostupné z: https://www.zat.cz/data/folders/193.pdf

[12] ZAT: *Pertinax6: Nástroj pro programování řídicích stanic ZAT* [online]. 2014 [cit. 2021-10-18]. Dostupné z: https://www.zat.cz/data/folders/194.pdf

[13] ZAT: *PertinaxDiag* [online]. c2021 [cit. 2021-10-18]. Dostupné z: https://www.zat.cz/cz/pertinaxdiag.htm

[14] *Wonderware System Platform* [online]. [cit. 2021-10-23]. Dostupné z: https://www.zat.cz/data/folders/146.pdf

[15] *CIMPLICITY: Proficy HMI/SCADA*. GE Digital [online]. [cit. 2021-10-24]. Dostupné z: https://www.ge.com/digital/applications/hmi-scada/cimplicity

[16] *Reliance 4: Přehled.* Reliance: Industrial SCADA/HMI system [online]. c2021 [cit. 2021-10-30]. Dostupné z: https://www.reliance-scada.com/cs/products/reliance4-scada-hmi-system#page=overview

[17] *Postmort.* Reliance: Industrial SCADA/HMI system [online]. c2021 [cit. 2021-10-30]. Dostupné z: https://www.reliance-scada.com/cs/products/features-and-benefits-of-reliance/postmort

[18] *Vývojové prostředí Reliance 4 Design.* Reliance: Industrial SCADA/HMI system [online]. c2021 [cit. 2021-10-31]. Dostupné z: https://www.reliance-scada.com/cs/products/reliance4/development-environment

[19] *Reliance 4 Combi Package.* Reliance: Industrial SCADA/HMI system [online]. c2021 [cit. 2021-10-31]. Dostupné z: https://www.reliance-scada.com/cs/products/reliance4/reliance4-combi-package

[20] *Komunikační drivery.* Reliance: Industrial SCADA/HMI system [online]. c2021 [cit. 2021-10-31]. Dostupné z: https://www.reliance-scada.com/cs/products/reliance4/communication-drivers

[21] *Runtime moduly systému Reliance 4.* Reliance: Industrial SCADA/HMI system [online]. c2021 [cit. 2021-10-31]. Dostupné z: https://www.reliance-scada.com/cs/products/reliance4/runtime-software

[22] *Tencí klienti systému Reliance 4.* Reliance: Industrial SCADA/HMI system [online]. c2021 [cit. 2021-10-31]. Dostupné z: https://www.reliance-scada.com/cs/products/reliance4/thin-clients

[23] *OPC servery.* Reliance: Industrial SCADA/HMI system [online]. c2021 [cit. 2021-11-01]. Dostupné z: https://www.reliance-scada.com/cs/products/opc-servers

[24] *Control Web: Prostředí pro tvorbu a provozování řídicích, vizualizačních nebo SCADA aplikací.* Moravské přístroje [online]. [cit. 2021-11-06]. Dostupné z: https://www.mii.cz/art?id=972&lang=405

[25] *Dispečerská mozaika.* APEL [online]. [cit. 2021-11-07]. Dostupné z: https://www.apel.cz/mozaika.php?level=2

[26] *Hardware: Desky pro počítačem řízený panel.* APEL [online]. [cit. 2021-11-07]. Dostupné z: https://www.apel.cz/poc_desky.php?level=4

[27] *Hardware: Moduly pro sériovou komunikaci.* APEL [online]. [cit. 2021-11-08]. Dostupné z: https://www.apel.cz/moduly.php?level=4

[28] *Hardware: Desky pro přímé připojení technologických procesů.* APEL [online]. [cit. 2021-11-08]. Dostupné z: https://www.apel.cz/prime_desky.php?level=4

[29] *Řídící a informační systém.* APEL [online]. [cit. 2021-11-14]. Dostupné z: https://www.apel.cz/software.php?level=3

[30] *Katalog produktů: PLC Tecomat TC700.* TECO: Advanced Automation [online]. c2017 [cit. 2022-04-24]. Dostupné z: https://www.tecomat.cz/products/cat/cz/plc-tecomat-tc700-1/

[31] *Mosaic - pro vývoj PLC programu dle standardu IEC 61131-3.* Teco: Advanced Automation [online]. c2017 [cit. 2022-04-24]. Dostupné z: https://www.tecomat.cz/ke-stazeni/software/mosaic/

[32] *EasyBuider Pro.* WEINTEK [online]. c2014 [cit. 2022-04-24]. Dostupné z: https://www.weintek.com/globalw/Software/EasyBuilderPro.aspx

[33] *Co je PROMOTIC.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2022-04-28]. Dostupné z: https://www.promotic.eu/cz/pmdoc/WhatIsPromotic/WhatIsPromotic.htm

[34] *Tecomat Foxtrot, Programovatelné automaty: Základní dokumentace modulu CP-1003* [online]. 3. vydání. Kolín, červen 2017 [cit. 2021-12-10]. Dostupné z: https://catalog.tecomat.cz/produkt/cp-1003#download

[35] Teco: *CP-1003* [online]. Kolín, c2021 [cit. 2021-12-10]. Dostupné z: https://catalog.tecomat.cz/produkt/cp-1003

[36] *Programovatelné automaty Tecomat Foxtrot CP-1003, CP-1013* [online]. 6. vydání. Kolín, březen 2017 [cit. 2021-12-11]. Dostupné z: https://catalog.tecomat.cz/produkt/cp-1003#download

[37] VOJÁČEK, Antonín. *TEST - Levný 7"HMI panel Weintek MT6070iE - 1.díl hardware.* Automatizace.hw.cz: rady a poslední novinky z oboru [online]. 20. Duben 2015 [cit. 2021-12-12]. Dostupné z: https://automatizace.hw.cz/hmi-systemy/test-levny-7-hmi-panel-weintek-mt6070ie-1dil-hardware.html

[38] BOUŠKA, Petr. *Ethernet - CSMA/CD, kolizní doména, duplex.* www.SAMURAJ-cz.com [online]. c2005-2021, 03.08.2007 [cit. 2021-12-18]. Dostupné z: https://www.samuraj-cz.com/clanek/ethernet-csmacd-kolizni-domena-duplex/

[39] *What is IEC 104?: IEC 104 — IEC 60870–5-104.* iGrid T&D: Smart solutions for smart grids [online]. [cit. 2021-12-18]. Dostupné z: https://www.igrid-td.com/smartguide/communicationprotocols/iec-60870-5-104/

[40] *IEC 60870-5-101.* IPCOMM GmbH [online]. c2021 [cit. 2021-12-18]. Dostupné z: https://www.ipcomm.de/protocol/IEC101/en/sheet.html

[41] *Introduction to the IEC 60870-5-104 standard.* ENSOTEST: Energy Software & Testing [online]. c2021 [cit. 2021-12-18]. Dostupné z: https://www.ensotest.com/iec-60870-5-104/introduction-to-the-iec-60870-5-104-standard/

[42] *IEC 60870-5-104.* IPCOMM GmbH [online]. c2021 [cit. 2021-12-18]. Dostupné z: https://www.ipcomm.de/protocol/IEC104/en/sheet.html

[43] FIEDLER, Petr. *RS-232/422/485: RS-485.* Brno, 2003.

[44] *Úvod do PROFIBUSu DP- 3 dílný seriál - 1. díl: Co je Profibus.* PROFIBUS-PROFINET.CZ [online]. c2021 [cit. 2021-12-19]. Dostupné z: https://www.profibus-profinet.cz/12-blog/profibus/clanky/15-uvod-do-profibusu-dp-3-dilny-serial

[45] *BACnet - What is BACnet MSTP?* CHIPKIN [online]. c2004-2021 [cit. 2021-12-22]. Dostupné z: https://store.chipkin.com/articles/bacnet-what-is-bacnet-mstp

[46] *Why Choose BACnet IP Over BACnet MS/TP.* KMC Controls [online]. c2021, September 24, 2019 [cit. 2021-12-22]. Dostupné z: https://www.kmccontrols.com/blog/why-choose-bacnet-ip-over-bacnet-ms-tp/

[47] *MODBUS RTU.* RTA: Real Time Automation [online]. c2021 [cit. 2021-12-19]. Dostupné z: https://www.rtautomation.com/technologies/modbus-rtu/

[48] *Sdílení XML dat.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2022-05-08]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Subsystems/Comm/XML/XML.htm

[49] *XML introduction.* MDN Web Docs [online]. c1998-2022, Last modified: Mar 19, 2022 [cit. 2022-05-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction

[50] *HTTP - Overview.* Tutorials Point [online]. [cit. 2022-05-09]. Dostupné z: https://www.tutorialspoint.com/http/http_overview.htm

[51] *Knihovna Iec104sLib: TXV 003 62.01* [online]. Druhé vydání. TECO, září 2012 [cit. 2022-05-07]. Dostupné z: https://www.tecomat.cz/modules/DownloadManager/download.php?alias=txv00362_01

[52] *FireBird.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-23]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Subsystems/Db/FireBird/FireBird.htm

[53] CANTU, Carlos H. *Poznejte Firebird za 2 minuty.* Přeložil Jiří ČINČURA. FirebirdNews [online]. únor/2010 [cit. 2021-12-26]. Dostupné z: https://www.firebirdnews.org/docs/fb2min_cz.html

[54] *Databázové možnosti v systému PROMOTIC.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-24]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Subsystems/Db/Database.htm

[55] *Objekt PmaAdo (ADO databáze).* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-24]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PmaAdo.htm

[56] *DbBeginTrans - metoda objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-24]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/DbBeginTrans.htm

[57] *DbCommitTrans - metoda objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-24]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/ DbCommitTrans.htm

[58] *DbRollbackTrans - metoda objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-24]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/DbRollbackTrans.htm

[59] *DbOpen - metoda objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-25]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/DbOpen.htm

[60] *DbConnectionString - vlastnost objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-25]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/ DbConnectionString.htm

[61] *DbConnectionParams - vlastnost objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-25]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/ DbConnectionParams.htm

[62] *DbIsOpen - metoda objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-25]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/DbIsOpen.htm

[63] *DbClose - metoda objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-25]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/DbClose.htm

[64] *DbExecute - metoda objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-25]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/DbExecute.htm

[65] *LastErr - vlastnost objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-25]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/LastErr.htm

[66] *LastTextErr - vlastnost objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-25]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/LastTextErr.htm

[67] *RsOpen - metoda objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-25]. Dostupné z:

https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/RsOpen.htm

[68] *RsIsOpen - metoda objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-25]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/RsIsOpen.htm

[69] *RsGet - metoda objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-25]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/RsGet.htm

[70] *RsClose - metoda objektu PmaAdo.* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-25]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/PropMeth/RsClose.htm

[71] *Objekt AdoRecordset (ADO Recordset).* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-26]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/AdoRecordset/AdoRecordset.htm

[72] *Objekt AdoRecord (ADO Record).* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-26]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/AdoRecord/AdoRecord.htm

[73] *Objekt AdoField (ADO Field).* PROMOTIC: SCADA visualization software [online]. MICROSYS [cit. 2021-12-26]. Dostupné z: https://www.promotic.eu/cz/pmdoc/Objects/Pma/PmaAdo/AdoField/AdoField.htm

[74] *About SymmetricDS.* SymmetricDS [online]. JumpMind, c2022 [cit. 2022-05-05]. Dostupné z: https://www.symmetricds.org/about/overview

A Obsah přiloženého CD

- `/` ..... kořenový adresář přiloženého CD
  - `Madzia_DP.pdf` ..... text diplomové práce
  - `Madzia_demonstrace_funkcnosti.mp4` ..... video s demonstrací funkčnosti
  - `Madzia_simulace_menirny.zip` ..... simulace měnírny na PLC a její vizualizace
  - `Madzia_databazove_servery.zip` programy pro databázové servery, konfigurační soubory pro jejich replikaci a vytvořená databáze
  - `Madzia_dispecink.zip` ..... SCADA aplikace klientského pracoviště