VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

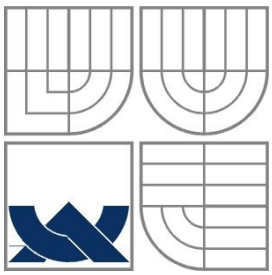
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

UŽIVATELSKÉ ROZHRANÍ PRO ŘÍZENÍ TECHNOLOGIÍ

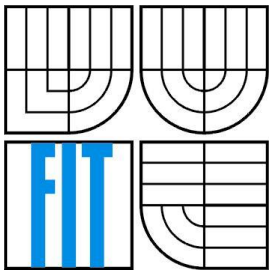
DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE BC. FILIP ZAPLETAL
AUTHOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

UŽIVATELSKÉ ROZHRANÍ PRO ŘÍZENÍ TECHNOLOGIÍ

USER INTERFACE FOR TECHNOLOGY CONTROL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — BC. FILIP ZAPLETAL
AUTHOR

VEDOUCÍ PRÁCE — PROF. DR. ING. PAVEL ZEMČÍK
SUPERVISOR

BRNO 2015

Abstrakt

Tato práce popisuje systémy a uživatelská rozhraní pro dohled a řízení technologických zařízení. Čtenáři přibližuje základní přístupy k tomuto problému, běžně používané komunikační protokoly a existující uživatelské rozhraní. Na základě zjištěných poznatků jsou definovány požadavky pro návrh moderního uživatelského rozhraní pro řízení technologií, zvláště v oblastech zpracování čisté a odpadní vody. Závěr práce tvoří popis implementace uživatelského rozhraní na základě definovaných požadavků a srovnání s existujícími systémy.

Abstract

This paper describes systems and user interfaces for monitoring and supervisory control of technological equipment. It describes basic approaches used to solve this problem, commonly used communication protocols and existing user interfaces. Based on those findings are defined requirements for design of a modern user interface for technology supervision and control, particularly in the area of clean and waste water distribution systems. The thesis concludes with a description of a user interface based on defined requirements and finally compares them with existing systems.

Klíčová slova

Uživatelské rozhraní, technologická zařízení, SCADA, PLC, komunikační protokoly, dohledové systémy, řízení technologií, automatizace.

Keywords

User interfaces, technological equipment, SCADA, PLC, communication protocols, supervisory systems, technology control systems, automation.

Citace

Zapletal Filip. Uživatelské rozhraní pro řízení technologií, diplomová práce, Brno, FIT VUT v Brně, 2015

Uživatelské rozhraní pro řízení technologií

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Prof. Dr. Ing. Pavla Zemčíka. Další informace mi poskytla firma GDF Mostkov spol. s r.o. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Filip Zapletal
26. května 2015

Poděkování

Rád bych poděkoval Prof. Dr. Ing. Pavlu Zemčíkovi za vedení této diplomové práce, poskytnuté rady a znalosti.

Dále bych rád poděkoval firmě GDF spol. s r.o. a jejím pracovníkům, za poskytnutí materiálů, zkušeností a rad, které byly pro výslednou podobu této diplomové práce velmi důležité.

Nakonec děkuji i Ing. Josefu Václavíkovi z Vodovodů a kanalizací Vsetín a panu Janu Schweidlerovi z Vodovodů a kanalizací Vyškov za zhodnocení vytvořené aplikace.

Obsah

1 Úvod

V posledních několika desítkách let se na celém světě, v čím dál více oblastech, využívá automatických řídicích systémů. Může se jednat o řízení výrobních linek v továrnách, správu a provoz inteligentních sítí v oblastech energetiky, dodávek pitné vody či zpracování vody odpadní, řízení provozu v dopravě a mnoho dalších oblastí využití. V poslední době se však automatizace rozšiřuje i mimo klasické výrobní a distribuční systémy, například do dnes populárních inteligentních domů.

Mezi hlavní cíle přesunu řízení těchto činností z člověka na automatizovaný systém patří úspora lidských zdrojů, větší úspora provozních nákladů díky lepší správě, kontrole a rychlejší reakci na problémy a v neposlední řadě i větší pohodlnost a menší náchylnost automatických systémů vůči chybám.

Ovšem ani u sofistikovaných řídicích systémů není možné se spoléhat pouze na ně, bez další kontroly a zásahů ze strany člověka, zvláště při neočekávaných či nestandardních situacích. Drtivá většina automatizovaných systémů má tak i nějakou formu uživatelského rozhraní pro dohled a správu systému. A právě těmito uživatelskými rozhraními se bude zabývat tato práce.

Cílem je podat čtenáři základní přehled o problematice, seznámit ho s existujícími postupy, systémy a komplexními řešeními problematiky a navrhnout uživatelské rozhraní, které je pokud možno, lepší než ta stávající. Minimálně si dává za úkol vnést do této, často velmi konzervativní oblasti, moderní pohled a technologická řešení. Většina uživatelských rozhraní pro řízení technologických procesů totiž vznikla často už před několika desítkami let, a svou koncepcí a použitými technologiemi často neodpovídají moderním požadavkům.

Návrh a především realizace takto komplexního a univerzálního systému však přesahuje rozsah běžné diplomové práce, proto je dále v textu problematika redukována na užší oblast použití. Široká oblast využití řídících a dohledových systémů, je tak omezena na využití při distribuci pitné vody a zpracování vody odpadní. Bude se také zabývat jen některými technologiemi a částmi celého systému, z nichž jsou vybrány ty, u kterých je největší potenciál pro případné inovace. Cílem tedy není vytvořit plně funkční, komplexní aplikaci použitelnou v široké praxi, ale spíše ukázat možnosti a nové způsoby realizace, které budou sloužit buď jako základ, nebo inspirace pro sofistikovanější systémy.

V první části práce se čtenář seznámí se základními pojmy z oblasti automatických řídicích systémů a to i na příkladech. Budou zde popsány standardizované komunikační prostředky, i existující uživatelská rozhraní pro dohled a správu. Ty budou dále zhodnoceny a na základě výsledků budou definovány požadavky, na základě kterých bude navrženo nové uživatelské rozhraní, které splní výše definované cíle. Poslední část práce se již zabývá samotnou realizací navrženého řešení a jeho hodnocením.

2 Systémy pro řízení technologií

V následující kapitole budou popsány některé z dnes existujících systémů pro řízení technologických celků a to jak z pohledu jejich uživatelských rozhraní, tak i z pohledu komunikačních protokolů a rozhraní. Jednotlivé systémy budou rozděleny do několika skupin dle způsobu používání a oblastí jejich nasazení. Budou zde popsány základní pojmy a principy chování, krátce budou čtenáři popsány systémy pracující v reálném čase a příklady jejich použití.

2.1 Úvod do SCADA

Systémy pro řízení technologických celků jsou v praxi označovány zkratkou SCADA, v originále *Supervisory Control and Data Acquisition*. Volně přeloženo *systémy pro dohled nad řízením a sběr dat*.

Důležitým aspektem u těchto systémů je možnost vzdáleného dohledu a ovládání nějakého zařízení, technologie. První systémy, které využívaly vzdáleného dohledu a ovládání, se začaly objevovat již na počátku 20. století. Jako moderní a plnohodnotné vzdálené řízení, lze ale chápat až systémy založené na mikropočítačích. Ty se prvně začaly k tomuto účelu používat v energetickém průmyslu . Dnes jsou SCADA systémy používány v mnoha odvětvích, mezi něž patří mimo jiná i tato:

- Energetický průmysl (elektrárny, rozvodné sítě)
- Vodovodní sítě a zpracování odpadních vod
- Distribuce a zpracování ropy a zemního plynu
- Petrochemický průmysl
- Komunikační sítě
- Průmyslové podniky a průmyslové řídicí systémy (výrobní linky, složité stroje)

Každé z těchto odvětví má na SCADA systémy různé nároky a požadavky. Mezi ně může patřit například spolehlivost a rychlost odezvy, kde jistě budou jiné požadavky u jaderné elektrárny než u vodovodních sítí, kde případná havárie nemusí nutně způsobit tak velké škody. Na druhou stranu jsou ale vodovodní sítě často velmi rozsáhlé a je u nich třeba počítat s přenosem dat na velké vzdálenosti a často i do míst, kde nejsou k dispozici žádné standardní komunikační linky. Naproti tomu u výrobní linky nelze předpokládat trvalý dohled z dispečerského pracoviště, a proto je tu kladen důraz spíše na bezobslužnou automatizaci, kde SCADA systémy slouží spíše pro dohledání poruch systému a pro komunikaci výrobní linky s okolím (například provázání s informačním systémem skladu, kde automaticky zaznamenáváme vyrobené kusy a spotřebované suroviny).

I přes různé požadavky v různých nasazeních mají ale SCADA systémy ucelenou sadu funkcí, mezi něž patří:

- Přenos dat a komunikace mezi jednotlivými PLC (*Programmable Logic Controller* – programovatelný logický automat)
- Vizualizace aktuálního stavu a měřených veličin
- Archivace stavu a měřených veličin
- Zpracování archivovaných dat
- Vyhodnocování a zpracování alarmů (nestandardních a chybových stavů zařízení)

Vzhledem k velké obsáhlosti tohoto tématu a i rozdílnosti výše zmíněných požadavků se v následujícím textu budeme primárně zaměřovat na systémy většího geografického rozsahu a systémy ne zcela kritickými z pohledu spolehlivosti. Nebudeme tedy uvažovat ani jaderné elektrárny, ani jednoduché výrobní linky a stroje. Naopak budeme uvažovat složitější distribuční síť, kterou může být například síť vodovodní a kanalizační. V následujícím textu si tedy popíšeme strukturu takové sítě a v ní používané prostředky.

2.2 Příklad technologické sítě

Jako ukázkový příklad byla zvolena vodovodní a kanalizační síť. Kombinace distribuce čisté a zpracování odpadní vody byla zvolena proto, že ve většině krajů jsou tyto dvě odvětví spravována stejnou společností a často i stejnými lidmi. Když pomineme samotnou topologii vodovodního potrubí a zaměříme se primárně na prvky, které má smysl řídit a sledovat, dostáváme následující seznam typů objektů (staveb):

- Úpravny pitné vody (ÚV)
- Vodojemy
- Čerpací stanice (ČS)
- Šachty a měřící objekty
- Čistírny odpadních vod (ČOV)
- Kanalizační čerpací stanice (KČS)

Výše uvedené typy objektů je možné rozdělit do třech kategorií, kterými jsou:

- **Objekty s trvalým dohledem** (stálá přítomnost obsluhy) přímo v místě – sem patří velké objekty typu ČOV a ÚV, kde je technologie a údržba tak složitá, že je zde třeba stálé přítomnosti obsluhy. Objekty bývají prostorově rozsáhlé (několik budov, velká prostranství). Aby bylo možné nad nimi provádět dohled, bývají zde umístěny takzvané dispečinky, tedy místa, kde je k dispozici ovládací a dohledové rozhraní (SCADA). Se systémem SCADA tyto objekty často komunikují po přímých drátových datových linkách, kterými mohou být buď průmyslové datové sběrnice (RS232, RS485, Profibus) anebo i klasický etherenet.

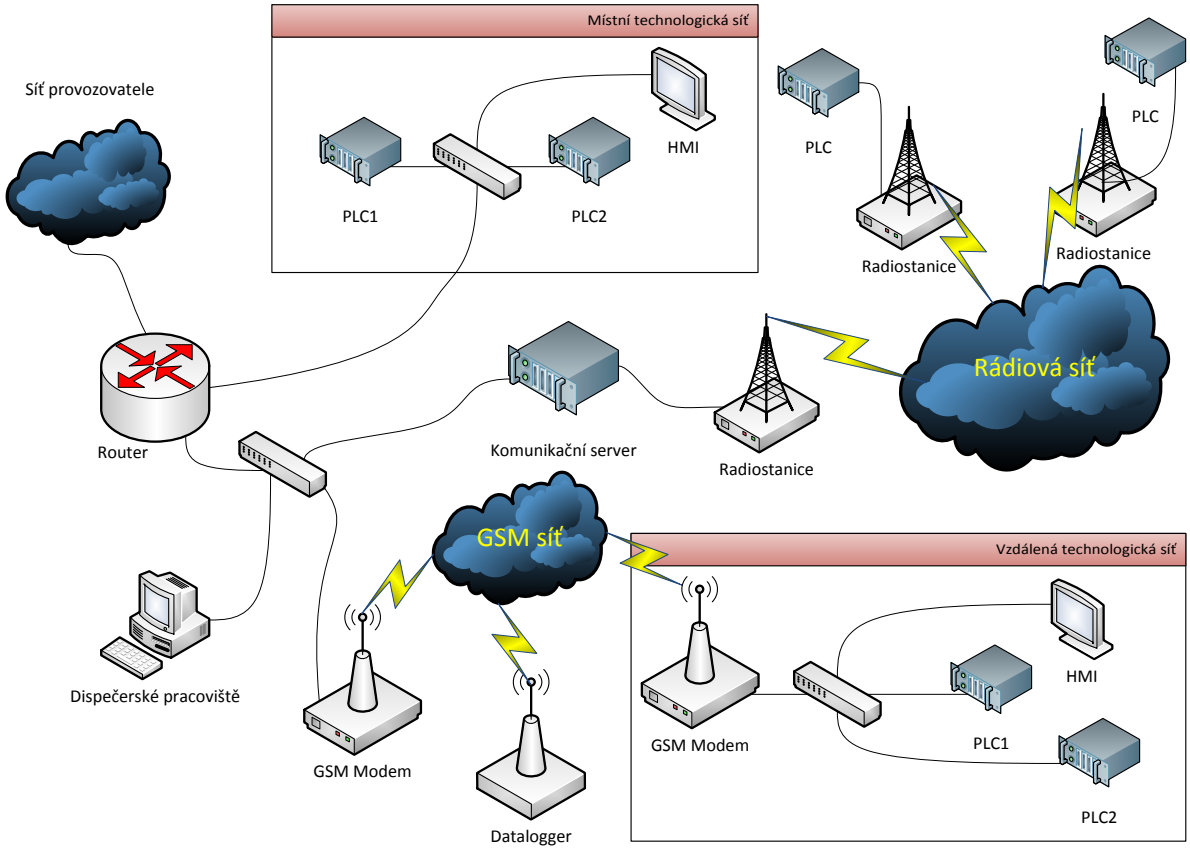
Velmi často tyto objekty mimo klasického přístupu, založeného na ovládání přes komunikační server a dispečerský počítač, používají i řízení pomocí dotykového ovládacího panelu. Ten je připojen přímo k PLC a nachází se přímo ve dveřích skříně elektrického rozvaděče. Tím je docíleno nezávislosti na okolních vlivech a je tak zavedena i redundance ovládání, pokud by došlo k poruše dispečerského počítače či komunikačních linek. Tyto panely jsou robustní kusy HW s velkou mechanickou odolností, mají však často malé úhlopříčky a jsou poměrně drahé.

- **Objekty se vzdáleným dohledem** – do této skupiny patří vodojemy a čerpací stanice (včetně těch kanalizačních). Na objektu tohoto typu není trvale žádný obslužný personál, řízení je navrženo tak, aby bylo zcela automatické, a je prováděn pouze vzdálený dohled. Velmi často tak, že je objekt hlídán z nějakého centrálního dispečinku, který má na starosti všechny objekty tohoto typu v daném regionu (okresu, kraji). V některých případech ani nemusí být na centrálním dispečinku prováděn dohled celodenní a je to pak řešeno formou hierarchie dispečinků. Existuje tedy více centrálních dispečinků, kde pouze jeden má celodenní provoz a v nočních hodinách se stará o objekty dispečinků ostatních. Dalším, a to velmi častým případem je model s využitím dispečinku mobilního. Je tedy určen pracovník, který sice není mimo klasickou pracovní dobu přímo na dispečinku, ale má s sebou jeho mobilní verzi (vzdálený přístup například přes GSM síť) a je o případných problémech informován prostřednictvím SMS zpráv. Komunikace systému SCADA s tímto typem objektů (tedy těch, kde není trvalá přítomnost obsluhy a jsou rozmístěny na velké geografické ploše) pak bývá realizována formou bezdrátové datové sítě. Tou může být standartní datová síť založená na GSM, ale i specializovaná autonomní rádiová síť na licencovaných frekvencích, jejíž provoz byl (zvláště dříve) levnější, bezpečnější a bylo možné jej použít i v místech bez GSM pokrytí.
- **Objekty bez elektrické přípojky** – poslední kategorií objektů jsou pak ty, které nemají žádné faktické řízení a slouží pouze pro monitorování stavu. Ty často nemají ani elektrickou přípojku a jsou realizovány formou bateriově napájeného *dataloggeru* (zařízení, které periodicky zaznamenává sledované veličiny a získaná data si ukládá do paměti). Ten pak prostřednictvím bezdrátové datové sítě (GPRS a následníci) ve stanovených intervalech odesílá nasbíraná data na centrální dispečink popsaný v předchozím bodu.

Důležitým prvkem v celém systému jsou tedy dispečerská pracoviště. Jejich topologie je často identická se strukturou samotného SCADA systému – tedy na každém dispečinku jsou často všechny, v předchozí kapitole popsané funkce (komunikace, archivace, vizualizace, vyhodnocování a zpracování dat). Z hlediska HW se tento uzel SCADA systému skládá z komunikačního a datového serveru, dispečerského pracoviště (počítač s vizualizací a řízením), komunikačních pojítek (radiomodemy, GSM modemy) a aktivních síťových prvků (propojují různé komunikační sítě, včetně propojení se sítí provozovatele).

Pokud si tedy dáme dohromady vše výše popsané, můžeme dostat schéma technologické komunikační sítě, kterou lze vidět na obrázku 1. Ta obsahuje lokální technologickou síť (rozsáhlý

objekt v místě dispečerského pracoviště), několik vzdálených objektů (vodojemy, ČS, KČS), vzdálenou technologickou síť (například menší ČOV, které nemá noční dohled řešen místně, ale vzdáleně) a několik dataloggerů.



Obrázek 1 - schéma technologické komunikační sítě

2.3 Systémy pracující v reálném čase a PLC

Prakticky každý dnes používaný řídicí systém je nějakým způsobem programovatelný. Klasické PC s běžným operačním systémem pro tyto účely není moc vhodné, především díky třem zásadním problémům:

- Klasické PC není z hlediska HW přizpůsobeno těmto nasazením. Primárně postrádá mechanizmus pro jednoduché připojení vstupních signálů, které jsou i dnes hlavně ve formě binárních, ale i analogových signálů. Běžné PC je také poměrně složité a tím pádem i náchylnější na chyby.
- Pro potřeby řídicích systémů, je třeba reagovat velmi rychle, respektive je třeba vždy reagovat v definovaném čase. Toho u běžných OS dosáhnout nelze, především díky nedeterministickému přepínání procesů, malému rozlišení priority procesů či pomalému přepínání kontextu, danému především celkovou složitostí OS.
- Dalším problémem je velikost klasických PC, jejich konstrukce a také spotřeba. Systémy jsou často umístěny v nepříznivých podmínkách (vysoké teploty, vysoké elektromagnetické rušení)

a musí být v bezúdržbovém provozu po dobu i několika desítek let, k čemuž klasické PC nejsou tak úplně určena.

Právě proto byly vyvinuty speciální řídící počítačové systémy, který získaly označení PLC (*Programmable Logic Controller*). Oproti běžným PC mají tyto vlastnosti:

- Vyšší kvalita použitých komponent
- K PLC jsou přímo připojitelné signály technologických procesů. Jde o digitální a analogové vstupy a výstupy. Systémy jsou často modulární, kde jsou samotné vstupy a výstupy umístěny na různých zásuvných kartách a tím pádem je možné si PLC poskládat tak, aby vyhovoval potřebám konkrétní aplikace.
- Mají operační systém uzpůsobený pro zpracování dat v reálném čase. To je často řešeno tak, že program zpracovávají cyklicky (viz dále).
- Operační systém v PLC je často velmi jednoduchý, což přispává k větší možnosti jeho verifikace a tím pádem eliminaci chyb.
- PLC pak mají implementován i *Watchdog*, tedy systém, který nezávisle monitoruje stav systému a v případě chyby provede reset do výchozího, funkčního stavu.

Výrobou PLC se zabývá velké množství společností a jsou nabízeny i ve velkém množství různých provedení. Můžou tak mít formu malých jednoduchých zařízení s několika vstupy a výstupy, v ceně několika tisíc korun, ale i složitých modulárních systémů i s desítkami tisíc vstupů či výstupů, s cenou v milionech korun. Příklad modulárního PLC vyšší třídy je na obrázku 2.

Obrázek 2 - Příklad modulárního PLC - Schneider Modicon M340 (www.schneider-electric.com)

Jak již bylo zmíněno, tak důležitým prvkem u řídicích systémů je to, aby fungovaly v reálném čase. To je zajištěno především architekturou operačního systému, pro kterou se vžilo označení *Real-time Operating Systems*. U RTOS je jedním ze základních požadavků, aby byla daná činnost dokončena v předem definovaném čase, který je velmi malý. Toho bývá často dosaženo tím, že systém pracuje v cyklech, které mají definovanou maximální dobu trvání dle priority daného procesu.

Každý cyklus pak sestává z několika fází, mezi něž patří především načtení stavu vstupů PLC do paměti, provedení logického programu (který v sobě nemůže mít další cykly – potenciální zdroje

zpoždění) a nakonec nastavení výstupů PLC podle výsledku programu. Detailnější popis PLC či *real-time* systémů přesahuje možnosti této práce a je lze nalézt ve specializované literatuře, například v nebo .

2.4 Komunikační protokoly a rozhraní

V této kapitole budou popsány různé komunikační protokoly a rozhraní používaná ve SCADA systémech. Komunikační protokoly můžeme rozdělit primárně podle toho, mezi kterými typy zařízení jsou používány. Základní rozdělení je do dvou skupin, a to na ty, které obstarávají komunikaci mezi PLC a komunikačním serverem (vyčítání dat z PLC – jsou často přímo závislé na výrobci a typu PLC), a na ty, které zprostředkovávají data dále (v jednotném formátu a nezávisle na PLC), tedy pro další vizualizaci a archivaci.

Jednotlivé protokoly můžeme ještě dělit dle úrovně ISO/OSI modelu, na protokoly pracující na vrstvě fyzické, linkové, případně vyšších). Protokoly, běžícími na fyzické vrstvě, se tato práce zabývat nebude a popíše pouze protokoly vrstev vyšších, které jsou často schopny pracovat nad různými komunikačními protokoly fyzickými, které jsou často realizovány formou sériových rozhraní (RS232, RS485) či je využito sítě ethernet.

2.4.1 Protokoly pro vyčítání dat a ovládání PLC

V dřívějších dobách byla tato oblast poměrně roztříštěna. Každý výrobce PLC si často vyvinul i vlastní komunikační protokoly, minimálně ty vyšších vrstev. V rozsáhlejších a nehomogenních sítích, které byly často spravovány různými organizacemi tak po jejich sloučení pod jednotnou správu, nastávaly komplikace v tom, že použitý SCADA systém často musel podporovat i několik desítek různých proprietárních komunikačních protokolů.

Proto se začali objevovat pokusy o standardizaci, a snahou bylo vytvořit jednotné komunikační prostředky, které by byly podporovány významnou částí výrobců nejrozšířenějších PLC. Pokud budeme ignorovat protokoly určené primárně pro komunikaci PLC s periferiemi (senzory a aktory), jsou dnes nejpodporovanějšími protokoly PROFIBUS a Modbus, kde dnes každé rozumné PLC podporuje alespoň jeden z nich.

Oba tyto protokoly jsou stavěné jako modulární a podporují několik komunikačních rozhraní nízké úrovně (fyzická vrstva). Mohou komunikovat pomocí:

- RS232, které se používá pro propojení s PC na krátkou vzdálenost.
- RS485, které se využívá pro propojování více PLC a komunikaci na delší vzdálenosti.
- Lze použít i optické datové spoje, pokud je třeba komunikovat na delší vzdálenosti či je třeba odolnost proti elektromagnetickému rušení.
- Lze použít i speciální typy sběrnic určených například pro výbušná prostředí.

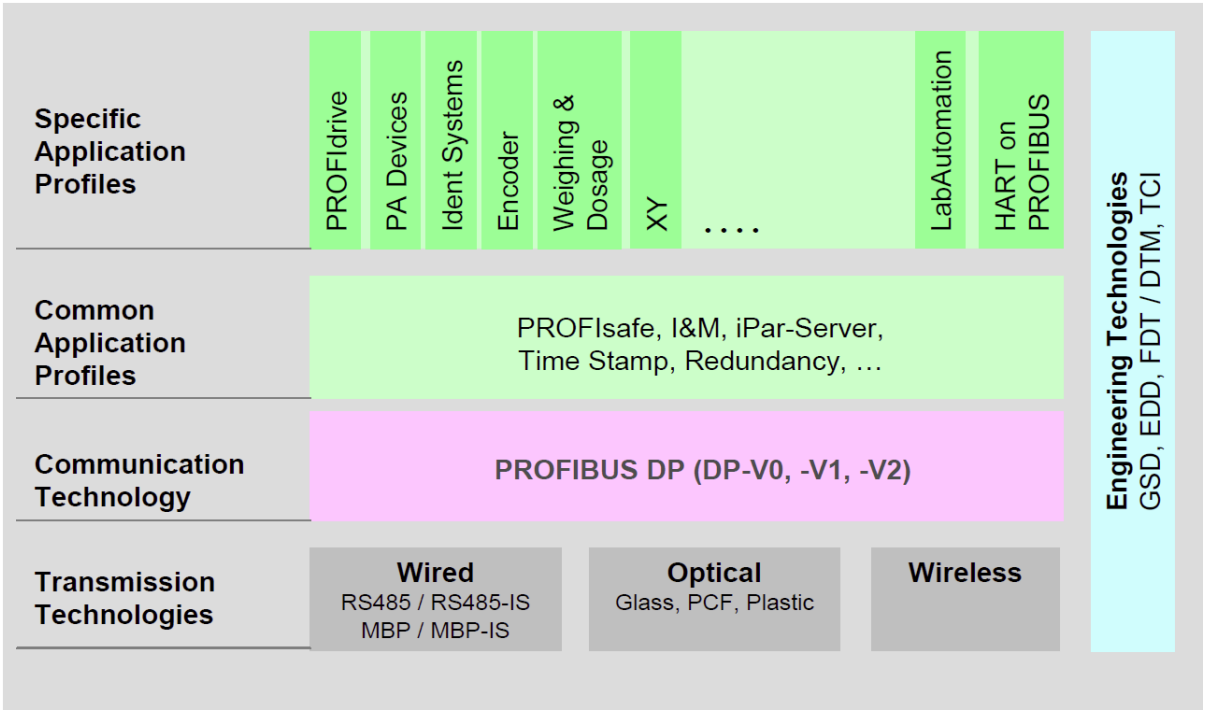
- A v neposlední radě mají oba protokoly i implementaci nad klasickými datovými sítěmi typu ethernet.

Na použitém komunikačních rozhraní pak často záleží i podporovaná topologie, RS232 například umožňuje propojit jen 2 zařízení, RS485 je pak striktně sběrnicová (všechny prvky musí být propojeny za sebou a konce této sběrnice musí být ukončeny terminátory), kdežto síť ethernetová tvoří často topologii hierarchické hvězdy.

2.4.2 PROFIBUS

Protokol a komunikační sběrnice PROFIBUS se začala vyvíjet v roce 1987. Dnes je vývoj zaštiťován organizací *PROFIBUS & PROFINET International* (PI). V roce 2010 měla tato organizace přes 1400 členů ve 27 zemích a celkově bylo v uvedeném roce instalováno přes 30 milionů zařízení podporujících tento komunikační protokol.

Jak již bylo zmíněno, je tato sběrnice navržena jako vysoce modulární. To lze vidět i na obrázku 3, kde je tato modulární struktura znázorněna. Jde o vrstvový model, kde jeho primární část leží na aplikační úrovni ISO/OSI modelu. Jednotícím prvkem je samotná komunikační technologie *PROFIBUS DP*, která je schopná běžet nad různými přenosovými protokoly a poskytuje jednotné rozhraní pro vyšší vrstvy.



Obrázek 3 - Modulární struktura sběrnice PROFIBUS

Těmi jsou obecné komunikační profily, které opět určují rozhraní, kterému musí všechna zařízení, která daný profil implementují, rozumět. Určují způsob adresace, přenosu a formátu dat, poskytují nástroje pro správu a diagnostiku. Těchto profilů je několik, rozdíl spočívá především v oblasti, pro

kterou se požívá (automatizace, připojení vzdálených vstupů, sběr dat z vědeckých experimentů v laboratořích). Protokol dále poskytuje ještě vyšší úroveň abstrakce, kde se snaží pro podobná zařízení různých výrobců vytvořit jejich jednotné rozhraní.

Jádrem protokolu je ovšem samotná komunikační vrstva *PROFIBUS DP*. Ta podporuje několik logických topologií včetně poměrně propracované redundance, jak na úrovni spojů, tak i jednotlivých uzlů. Uzly pak mohou být buď typu *slave* (poskytují data na dotazy) a nebo *master,* který se na data z jednotlivých *slave* dotazuje. Dotazování může probíhat v několika režimech, v závislosti na *funkční verzi* (úrovni složitosti a vybavenosti) jednotlivých zařízení.

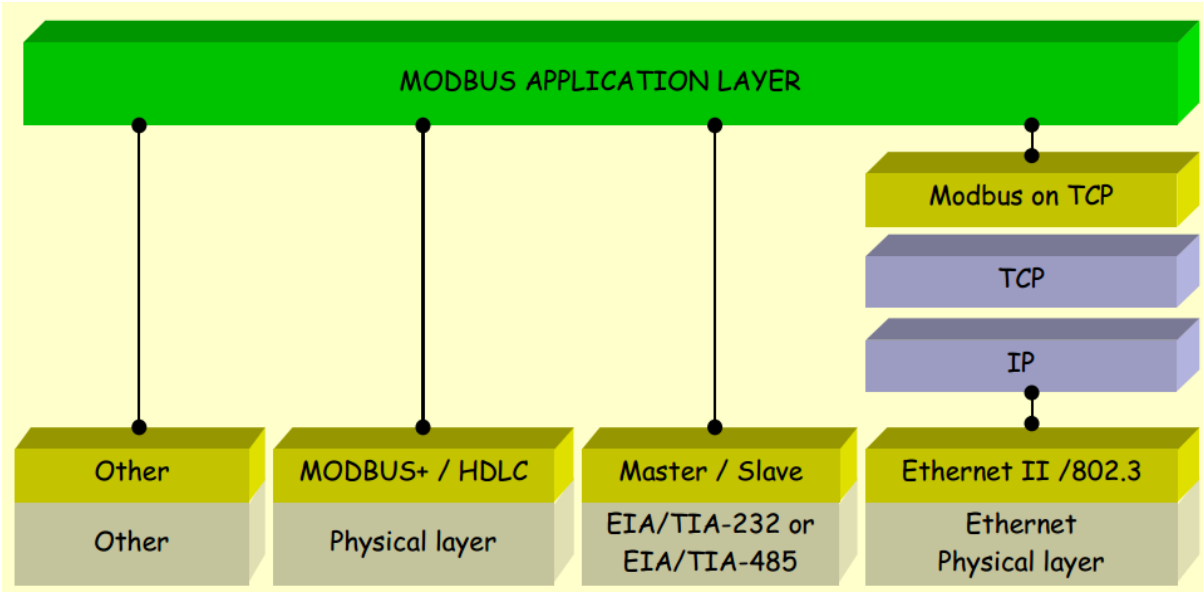
Základní verze DP-V0 podporuje cyklické dotazování jednotlivých *slave* v přesně daném pořadí, postupně jednotlivými *mastery* (může jich být i více). DP-V1 podporuje i dotazy acyklické, tedy *master* si může vyžádat data od kteréhokoli *slave* nezávisle na pořadí. DP-V2 pak podporuje i *broadcasty* (všesměrové dotazy) a jistou formu *multicastu* (dotaz pro skupinu zařízení) založenou na poskytovatelích a odběratelích dat. Vyšší úroveň s sebou samozřejmě nese i vyšší nároky na daná zařízení.

Jednotlivé komunikační uzly jsou adresovány buď pomocí hardwarových přepínačů (DIP, otočné), nebo pomocí k tomuto účelu vytvořeného konfigurační software. Kompletní popis struktury datového rámce přesahuje rozsah této práce, lze ho ovšem najít v .

2.4.3 Modbus

Komunikační protokol Modbus byl vyvinut společností *Gould Modicon,* později firmu koupil a o vývoj se dnes stará firma *Schneider Electric*. Tento protokol je, stejně jako PROFIBUS také velmi rozšířený a výrobci podporovaný, například již v roce 2004 ho jako primární komunikační protokol používalo více než 40% aplikací.

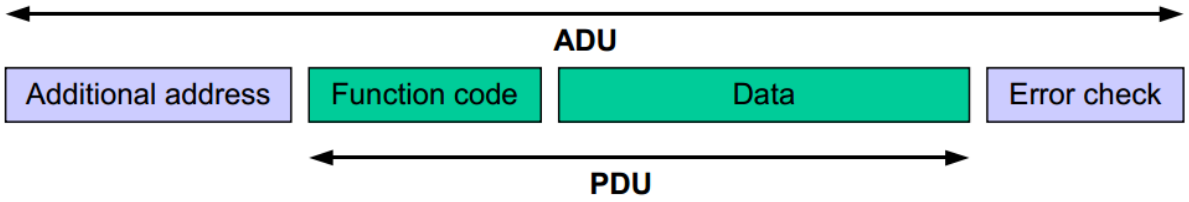
Protokol je opět modulární, a to až do té míry, že lze jako komunikační sběrnici na fyzické vrstvě použít téměř cokoliv. Standardně se jako komunikační médium používá RS232, RS485 či Ethernet, a to jak v metalickém, tak i optickém provedení. Speciální variantou je Modbus+, který je založený na algoritmu *token passing* a používá se primárně pro připojení periferií (senzorů, aktorů). Topologicky je protokol založený na modelu s jednou stanicí typu *master*, která řídí veškerou komunikaci a až 247 stanicemi typu *slave*.



Obrázek 4 - Modulární architektura sběrnice Modbus

Modbus podporuje dva režimu komunikace, a to v módu ASCII, kde jsou data přenášena pomocí i lidmi čitelného textu (určeno primárně pro testování), a v módu RTU, kde jsou data přenášena binárně.

Datový rámec protokolu (viz obrázek 5) má povinnou část (PDU) nezávislou na použitém komunikačním protokolu nižší vrstvy, a rozšířenou část (ADU) používanou například pro dodatečnou adresu na sběrnicových sítích a kontrolní součet (CRC), pokud zabezpečení neřeší vrstva přenosová.



Obrázek 5 - Obecný formát rámce protokolu Modbus

Samotná komunikace je založena na modelu dotaz – odpověď, kde se stanice *master* dotazuje a *slave* odpovídá. Takovým dotazem ale může být i příkaz, kterým lze data do stanice *slave* zapsat. Pokud nelze dotaz či příkaz z jakéhokoli důvodu provést, je místo odpovědi vrácen chybový kód. Každý dotaz v sobě nese i informaci o použité funkci, která může být následující:

- Čtení nebo zápis binárního výstupu (v terminologii Modbus nazývaném *coil*).
- Čtení analogového vstupu.
- Čtení nebo zápis paměťového registru.
- Další funkce pro řízení komunikace, testování a diagnostiku (často již závislé na komunikačním médiu).

Protokol je v základu 16 bitový a v rámci jedné zprávy lze přenést maximálně 127 slov (tedy 256 bajtů). Základní princip funguje tak, že pomocí tohoto protokolu je interní paměť zařízení (např. PLC) mapována na adresový rozsah 0 – 65535, ze kterého jsou pak data čtena (případně zapisována) po

větších blocích. Takto získaná data nejsou protokolem nijak interpretována, to je ponecháno na uživateli – vyšší vrstvě komunikačního modelu.

Obsah, struktura a význam jednotlivých datových či chybových zpráv protokolu již přesahuje rozsah této práce, a lze se o nich více dozvědět v .

2.4.4 Protokoly a rozhraní pro zpřístupnění dat

Cílem této skupiny protokolů a rozhraní je pokud možno co nejvíce odstínit rozdíly mezi jednotlivými typy a výrobci PLC. V praxi se používají dva přístupy, u kterých velmi záleží na potřebách a požadavcích provozovatele. Mezi největší rozdíly patří, zda provozovatel používá různé systémy různých výrobců, nebo se spokojí s použitím SCADA systému a všech navazujících programových prostředků z dílny výrobce jednoho.

V případě nutné kooperace více různých systémů je tedy zapotřebí ve všech z nich implementovat jednotné rozhraní, které by mělo být ideálně standardizováno a podporováno v nejběžnějších a nejpoužívanějších systémech. Tímto rozhraním se stala rodina standardů OPC, která dnes popisuje téměř všechny oblasti této problematiky. Proto je také popsána v následující kapitole.

Pokud používáme systém jednoho výrobce, tak i tam je nutné zvolit jistou míru abstrakce a sjednocení reprezentace dat. Způsob provedení však již bývá silně individuální a každý z výrobců si jej přizpůsobuje svým potřebám. I tyto systémy pak často implementují OPC klienta, lze je tedy připojit k systémům jiným, ale svá data dále již prostřednictvím OPC neposkytují. Tento pro jiné výrobce ne moc přívětivý přístup, často bývá součásti obchodní politiky a snahy znepříjemnit případný přechod zákazníka ke konkurenci.

2.4.5 Open Platform Communications - OPC

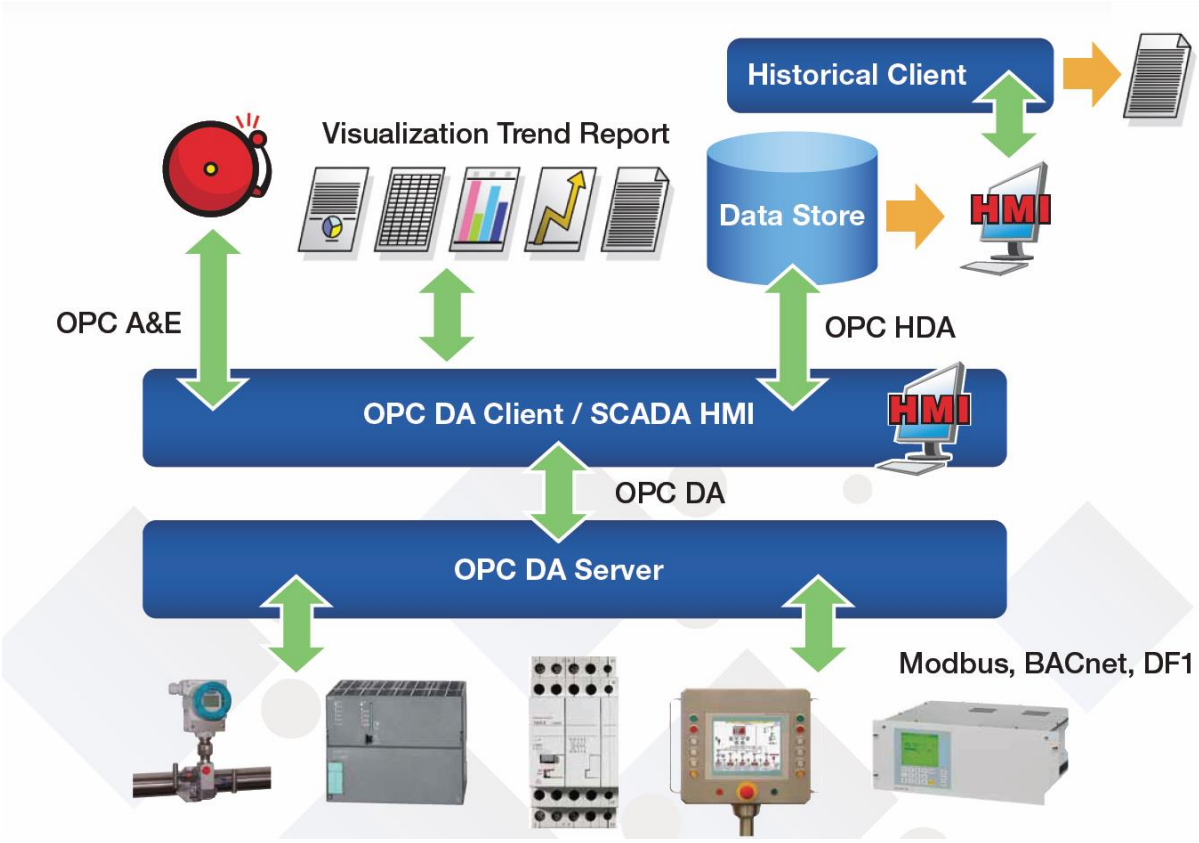
Open Platform Comunication, zkracované a obecně známé jako OPC (dříve měla zkratka význam *OLE for Process Control*) je skupina standardů pro průmyslovou automatizaci a výměnu telemetrických dat v reálném čase. Klade si za úkol sjednotit komunikační rozhraní mezi zařízeními (PLC) a SCADA systémy různých výrobců. Jedinou podmínkou je, aby výrobce PLC vytvořil pro své zařízení odpovídající *OPC Data Access server* a jakýkoli klient (SCADA systém, jiné PLC), v sobě obsahoval obecného OPC klienta.

Celá architektura OPC je založena na technologiích OLE, COM a DCOM (detailní vysvětlení těchto technologií přesahuje rozsah této práce) od společnosti Microsoft. Standard definuje skupiny objektů, rozhraní a metod, čímž je zajištěna právě výše zmíněná interoperabilita mezi různými systémy a zařízeními.

Skupina standardů OPC se dnes ovšem nezabývá pouze sjednocením datového rozhraní pro aktuální data, ale obsahuje i jiné moduly, mezi něž (mimo jiné) patří:

- **OPC Data Access** – Původní rozhraní pro přenos dat mezi jednotlivými PLC a SCADA systémy v reálném čase.
- **OPC Alarms & Events** – Přidává podporu generování a zpracování událostí a alarmů (nekontinuální datový tok).
- **OPC Historical Data Access** – Popisuje jednotné rozhraní pro ukládání a získávání historických (archivovaných) dat.
- **OPC XML-DA** – Podobně jako OPC DA (OPC Data Access) poskytuje aktuální data, ale ve formátu XML.
- **OPC Unified Architecture** – Nová skupina standardů, které se snaží o odpoutání se od technologie Microsoft (D)COM za účelem možnosti použití OPC i na jiných platformách. Výhodou je i použití modernějších programovacích technik a přístupů.

Příklad celého systému využívajícího OPC je ukázán na obrázku 6.



Obrázek 6 - Příklad architektury systému založeného na OPC

2.5 Příklady SCADA systémů

Systémů pro dohled, řízení a sběr dat je velké množství. Vychází to z rozdílných požadavků různých oblastní nasazení, z požadavků na velkou flexibilitu a nutnost úprav dle přání zákazníka, z velikosti zákazníka a také jeho finančních možností. Proto velmi často firmy, které tyto systémy ve větší míře nasazují, jsou i jejich vývojáři, nebo s některými z nich úzce spolupracují.

Existuje sice i několik větších a všeobecně používaných systémů, jakými jsou například *WinCC* od společnosti *Siemens, ClearSCADA* společnosti *Schneider Electric* či další. U nich je ale problém ve vyšší pořizovací ceně a minimální možnosti individuálních úprav systému. U velkých firem totiž je téměř nulová šance, že jako menší zákazník docílíte zapracování vašich požadavků ve formě úpravy či doplnění funkcionality.

Pokud je ale výrobce SW i tím, kdo jej nasazuje, nebo pokud mezi sebou mají výrobce a realizátor dohodu, tak lze SW poměrně dobře požadavkům jednotlivých zákazníků přizpůsobit. Pro mnoho menší firem, je totiž i zakázka v nižších cenových relacích zajímavá. Velkou roli pak hrají i jazykové bariéry, kde čeští zákazníci požadují kompletní lokalizaci celého SW, jeho dokumentace, ale především i technické podpory. Proto se velmi často využívají právě lokální produkty. Vzhledem k těmto skutečnostem, se další část práce bude zabývat právě českými produkty, a to těmi určenými primárně pro oblast použití ve správě vodovodních sítí a zpracování odpadních vod (tato oblast byla zvolena s ohledem na znalosti a zkušeností v daném odvětví, získané díky spolupráci se společností *GDF spol. s r.o.*).

2.5.1 WinControl

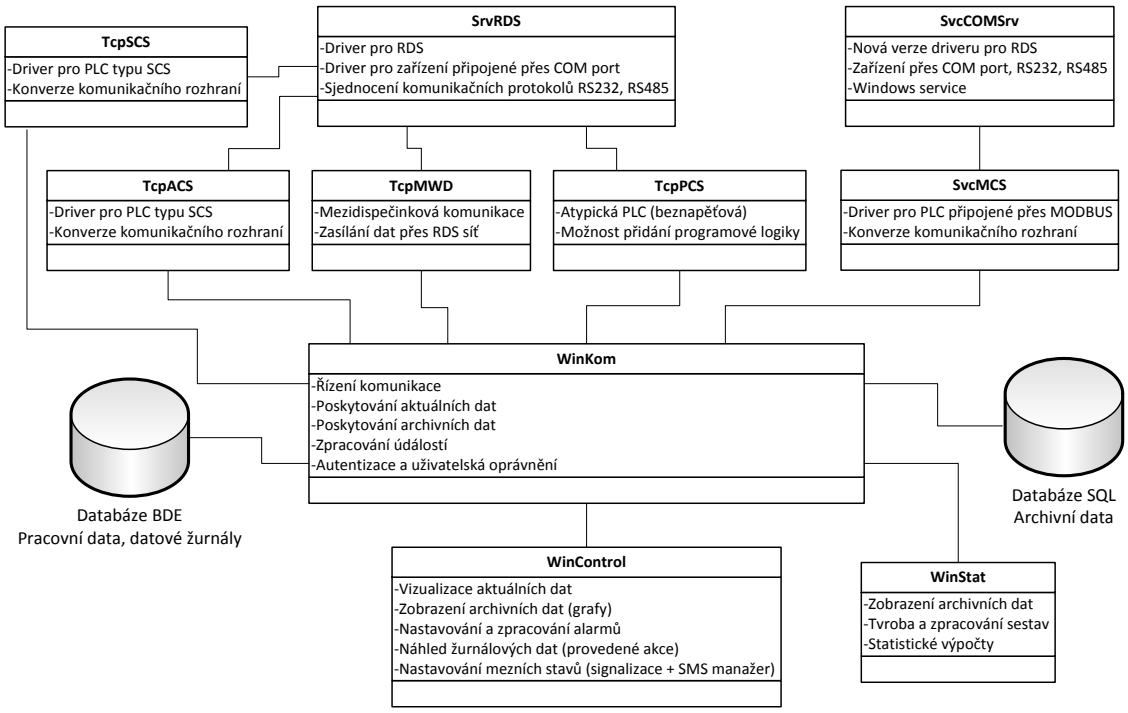
WinControl je SCADA systém vyvíjený společností *GDF spol. s r.o.* Je primárně zaměřen pro použití v oblasti zpracování čisté a odpadní vody, a tomuto nasazení je uzpůsobena i celá jeho architektura. Jde o systém již staršího data vydání, vznikl v roce 1997 jako nástupce dřívější verze určené pro operační systém MS-DOS (vzniklé v roce 1991). Tento systém je poměrně populární napříč celou Českou Republikou, je používán ve většině krajů s výjimkou severní Moravy, severních Čech a také velkých měst. Jeho upravená jazyková verze je nasazena i v některých částech Polska.

Bohužel byl u tohoto systému již před delší dobou (kolem roku 2005) zastaven aktivní vývoj, a od té doby probíhá pouze jeho údržba bez zásadních úprav funkcionality. I přesto je ale i dnes s úspěchem nasazován do nových lokalit k novým zákazníkům, z čehož lze usoudit jeho nadčasovost a již v době svého vzniku vysokou vyspělost.

Vzhledem k zastavenému vývoji a z dnešního pohledu poměrně starých a problematických technologií použitých při vývoji systému (SW je implementován v Delphi 6, jako primární úložiště používá databázi BDE) a absenci dnes požadovaných funkcionalit (webový přístup k vizualizaci dat, integrační nástroje), je jeho další používání v delším časovém horizontu spíše problémem. Je zde zmíněn primárně proto, že již ve své době obsahoval velké množství nadstandardních funkcí,

doplněných dobrou ergonomií, čímž dosahoval velmi vysoké produktivity práce a rychlosti reakce na případné problémy celého vodovodního a kanalizačního systému. Důležitým prvkem byly i přenosy dat mezi jednotlivými lokalitami po radiové síti (internetová spojení byla v dobách, kdy systém vznikal buď nedostupná, nebo nespolehlivá a drahá).

I přes dnes již zastavený vývoj byl systém před několika lety (přibližně v roce 2011) doplněn o možnost archivace dat do dnes již standardní databáze typu SQL místo staršího BDE. Dále byly přepsány některé dříve problematické moduly do platformy *Microsoft .NET* (viz. dále), čímž se odstranilo několik problematických starších modulů a zvýšila se tak stability systému, jehož celkovou strukturu si lze prohlédnout na obrázku 7.



Obrázek 7 – Základní struktura systému GDF WinControl

Jádro systému je tvořeno modulem *WinKom*. Ten se stará o veškerou správu komunikace s PLC, správu dat (ať těch aktuálních, tak i archivních), správu oprávnění a mnoho dalšího. Jako datové úložiště používá databáze založené na BDE. Ty, na rozdíl od těch dnešních a moderních, měly problémy s větším objemem dat, a proto je *WinKom* separoval do samostatných databází po jednotlivých dnech. To ovšem způsobovalo problémy při externí práci s daty (mimo systém *WinControl*), a tak byla později doplněna i možnost ukládání dat do databáze standardu SQL.

Veškerá komunikace s modulem *WinKomu* probíhá prostřednictvím protokolu TCP/IP. Jeho pomocí se k němu připojují klienti, které lze dle funkce rozdělit na ty, které data poskytují, nebo ty, které data konzumují.

Klienti, kteří data poskytují, jsou již závislí na použitém PLC. Protože byl systém vyvíjen i nasazován jednou společností, která vyvíjela i vlastní PLC (u SCS včetně HW, u ACS byl v zařízení firmy *Advantech* použit vlastní firmware), odpovídají první dva moduly (*TcpSCS* a *TcpASC*) právě

těmto dvěma typům PLC. Na první pohled se může tento způsob *vlastní cesty* z dnešního pohledu jevit jako ne příliš šťastný, opak je ale pravdou. Díky propojení celého vývojového cyklu, od tvorby PLC po tvorbu vizualizace, byl vývoj v tomto systému velmi rychlý, což mělo příznivý vliv na cenu a tedy i celkovou dostupnost a rozšíření systému.

Mimo vlastních PLC ale bylo třeba mít možnost do systému připojit i zařízení jiných výrobců. K tomu byl vyvinut modul *TcpPCS*. Ten může, mimo komunikace s rozličnými typy PLC i sám o sobě obsahovat nějakou přidanou programovatelnou logiku. Ta slouží právě k odstranění rozdílů a následnému sjednocení rozhraní a formátu dat. Modul *TcpPCS* obsahuje i implementaci protokolu *Modbus*, která je ale dnes nahrazena samostatným modulem. Důvodem k tomuto kroku byla již velká složitost tohoto modulu, který původně nebyl určen pro hromadné nasazení, které po přechodu z vlastních PLC na ty, komunikující přes *Modbus* nastalo.

Všechny výše zmíněné moduly mají kromě jejich běhové (runtime) verze i verzi konfigurační, která slouží k editaci vnitřních parametrů, ale často i k nastavování a správě samotných PLC. O propojení těchto komunikačních modulů s fyzickými zařízeními se poté staraly různé verze takzvaných *COM serverů*, které implementovaly protokoly nižších úrovní, tedy primárně RS232, RS485 či jejich zapouzdřené verze v protokolu RDS92, který je používán při rádiové komunikaci.

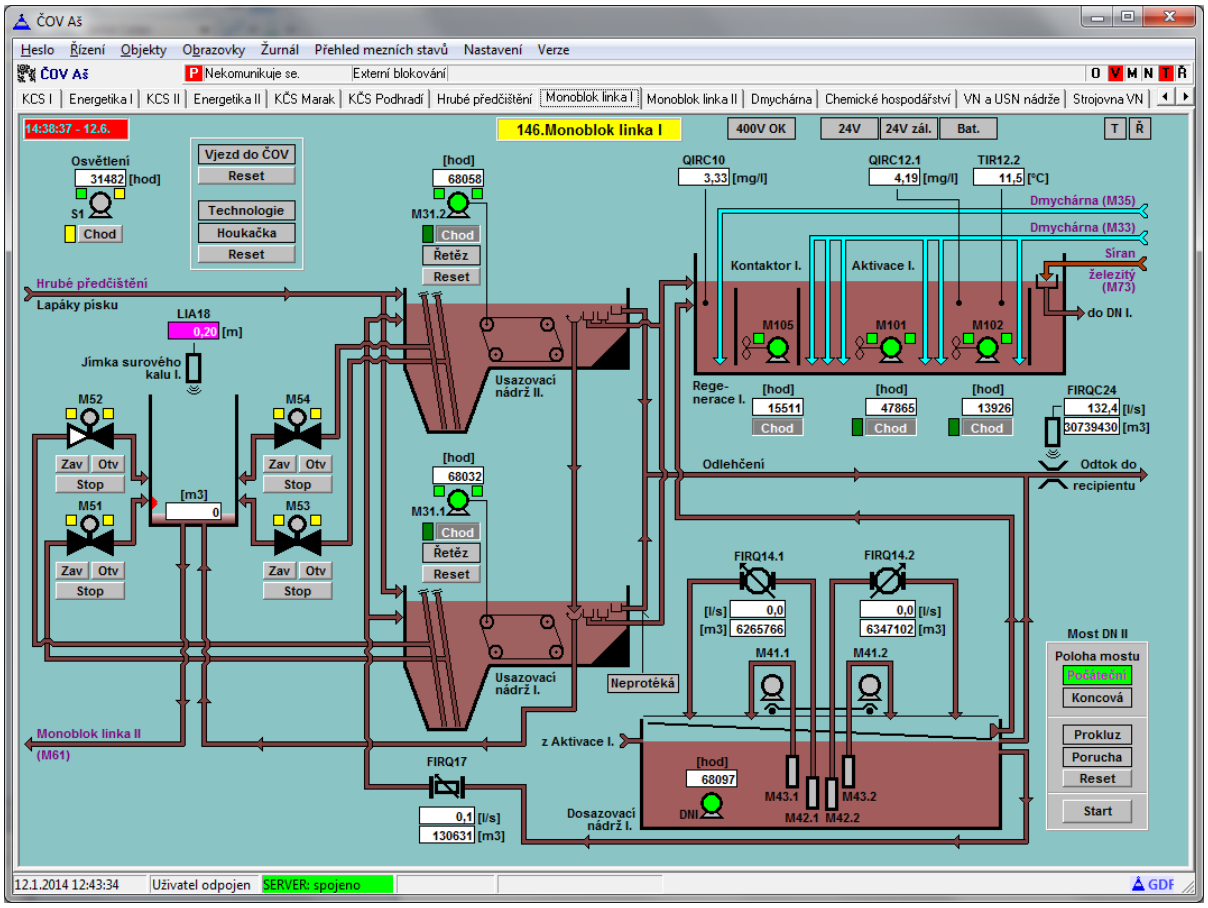
Mimo modulů určených pro komunikaci s PLC v systému existují i moduly pro komunikaci mezi jednotlivými dispečinky, a to nejen po klasickém TCP/IP, ale i po rádiové síti. K tomu slouží modul *TcpMWD*, který umožňuje mimo aktuálních dat přenášet i povely, a to vše při velmi nízkých datových tocích (obecná propustnost rádiové sítě je 9600kbit/s, přičemž jsou v ní i stovky uzlů).

Do systému byly dále doplněny již dříve zmíněné moduly pro komunikaci s PLC přes protokol *Modbus*. Jde o modul *SvcMCS*, který zajišťuje správnou interpretaci dat, a modul *SvcCOMSrv*, který je obdobou původních *COM serverů*. Oba jsou již na rozdíl od svých předchůdců (to byly klasické *WinForms* aplikace), napsány jako Windows služby a využívají technologie *Microsoft .NET*.

Jako datový model systém používá komplexní datové typy podobné strukturám. Tyto struktury (označované jako prvky) jsou pak skládány do hierarchických struktur – stromů. Pro identifikaci jednotlivých prvků, ale i jejich vlastností, slouží unikátní textové řetězce složené z názvů prvků. Díky tomu lze poměrně snadno adresovat různé úrovně abstrakce, od celého objektu (složeného i z několika PLC), přes konkrétní pohon (prvek symbolizující například čerpadlo), až po konkrétní vlastnosti daného prvku (tedy například stav signálu chod, či jeho provozní hodiny).

Druhou formu klienta, tedy toho, co data *WinKomu* konzumuje, je primárně aplikace, které dala celému systému jméno, a to *WinControl*. Tato aplikace složí pro vizualizaci dat, ovládání, prohlížení a rychlou analýzu archivovaných dat, a mimo jiné i o správu a vyhodnocování alarmů, včetně modulu pro zasílání SMS zpráv. Ukázku uživatelského rozhraní této aplikace si lze prohlédnout na obrázku 8. Samotná vizualizací dat je silně bitmapově orientovaná. Obsahuje tedy bitmapové obrázky (nádrže, technologické značky a symboly…) kombinované s aktivními prvky, které jsou však také umísťovány

na předem danou bitmapovou mřížku. To má za následek prakticky nulovou adaptabilitu na různá rozlišení a velikosti obrazovky.



Obrázek 8 – Ukázka uživatelského rozhraní a vizualizace v aplikaci WinControl

Pro tvorbu vizualizace slouží aplikace *WinGen*, která se vizuálně podobá aplikaci *WinControl*. Pro ukládání vizualizace je použita BDE databáze, a ta je vždy pro všechny objekty, které si přejeme mít v jedné vizualizační aplikaci, společná. To má několik výhod (vždy jen jedna platná verze vizualizace), ale i několik nevýhod. Mezi ně patří například nemožnost práce s danou databází (lokalitou, aplikací) současně ve více lidech, problémy při nasazování úprav (pokud se nasazení dané změny odloží, nastává problém), ale i samotná velikost databáze a s tím spojené dlouhé spouštění aplikace u rozsáhlejších nasazení.

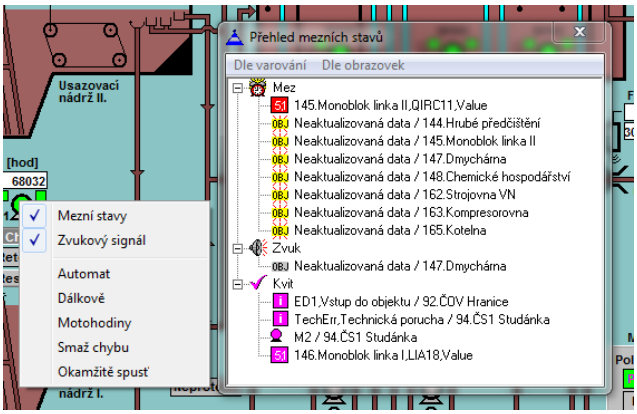
Mimo, již výše zmíněných bitmapových obrázků, které však nelze v aplikaci tvořit ani editovat, se vizualizace skládá i z velkého množství aktivních prvků. Ty jsou sdruženy do skupin, které korespondují s typy prvků použitými ve *WinKomu*. Aplikace pro tvorbu pak obsahuje průvodce, pomocí kterého lze skupinu indikačních a ovládacích prvků, například pro pohon, vytvořit. V průvodci pak lze nastavit velké množství vlastností, od velikosti, přes rozložení, až po přítomnost ovládacích prvků v závislosti na typu pohonu. Tyto skupiny jsou bohužel pevně dané, a nelze je bez nutnosti programátorského zásahu doplňovat či měnit.

I z toho důvodu také aplikace obsahuje i velké množství generických indikátorů, které jsou schopny zobrazovat text (číselné hodnoty), měnit barvu (binární stavy), ale měnit i polohu a velikost (indikace

hladiny či polohy komparátorů). Speciálním generickým prvkem je pak potrubí, které je také aktivní, ale lze implicitně tvořit pouze horizontálně či vertikálně rovné čáry. Pro tvorbu složitějších tvarů již aplikace přímou podporu nemá a je nutné je skládat z jednotlivých úseků či doplnit bitmapou.

Z aktivních prvků, doplněných o další detaily formou bitmapových obrázků, tak lze vytvořit základní obrazovku objektu. Jednotlivé obrazovky je možné vzájemně propojit pomocí odkazů, lze k nim ale přistupovat i pomocí lišty záložek. Ta je uživateli hojně využívána už jen proto, že ji lze ovládat pomocí šipek klávesnice a velmi rychle tak prohlédnout stav celého dispečerského systému.

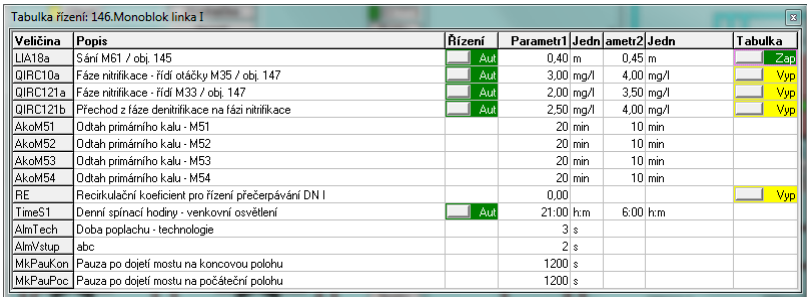
Ovládání je v aplikaci *WinControl* řešeno pomocí vícestavových tlačítek (stisknuto, neaktivní, zablokované) a kontextového menu (viz. obrázek 9), s jehož pomocí se provádí méně standardní akce. Pomocí kontextového menu tak lze na kterýkoli aktivní prvek nastavit mezní stav neboli alarm. U prvků zobrazujících analogovou veličinu je možné si nastavit alarm na minimum či maximum dané hodnoty, u veličin binárních je možné si nastavit alarm například na poruchu, přepnutí do ručního ovládání či cokoli jiného. Je možné si rovněž aktivovat i zvukové upozornění či zaslání SMS.



Obrázek 9 – Ukázka kontextového menu a přehledu alarmů

Právě aktivní mezní stavy si je možné zobrazit pomocí dialogu *Přehled mezních stavů* (viz. obrázek 9). Po kliknutí na odpovídající řádek přehledu je uživateli přepnuta obrazovka a přesunut kurzor na vizuální prvek, na kterém alarm vzniknul. Lze tak velmi rychle reagovat i bez detailní znalosti technologie a zdlouhavého vyhledávání.

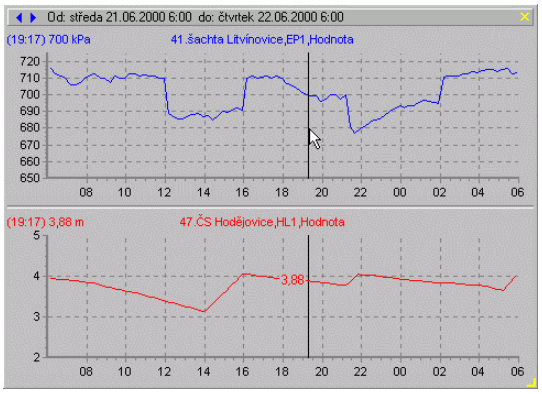
Pro nastavování detailních provozních parametrů technologie, je do systému zabudována i takzvaná *Tabulka řízení* (viz. obrázek 10). Ta umožňuje nastavování provozních parametrů přehledně a uceleně na jednom místě. Není tak třeba, tyto zřídka měněné veličiny, umisťovat přímo do vizualizace. Zápis do tabulky, lze stejně jako jiné možnosti ovládání, chránit pomocí jednoduchých uživatelských účtů zabezpečených hesly.



Veličina	Popis	Řízení	Parametr1	Jedn	ametr2	Jedn	Tabulka
LIA18a	Sání M61 / obj. 145	Aut	0,40	m	0,45	m	Zap
QIRC10a	Fáze nitrifikace - řídí otáčky M35 / obj. 147	Aut	3,00	mg/l	4,00	mg/l	Vyp
QIRC121a	Fáze nitrifikace - řídí M33 / obj. 147	Aut	2,00	mg/l	3,50	mg/l	Vyp
QIRC121b	Přechod z fáze denitrifikace na fázi nitrifikace	Aut	2,50	mg/l	4,00	mg/l	Vyp
AkoM51	Odtah primárního kalu - M51		20	min	10	min	
AkoM52	Odtah primárního kalu - M52		20	min	10	min	
AkoM53	Odtah primárního kalu - M53		20	min	10	min	
AkoM54	Odtah primárního kalu - M54		20	min	10	min	
RE	Recirkulační koeficient pro řízení přečerpávání DN I		0,00				Vyp
TimeS1	Denní spínací hodiny - venkovní osvětlení	Aut	21:00	h:m	6:00	h:m	
AlmTech	Doba poplachu - technologie		3	s			
AlmVstup	abc		2	s			
MkPauKon	Pauza po dojetí mostu na koncovou polohu		1200	s			
MkPauPoc	Pauza po dojetí mostu na počáteční polohu		1200	s			

Obrázek 10 – Tabulka řízení aplikace WinControl

Aplikace dále umožňuje, buď na samostatné obrazovce, nebo jako plovoucí okno, zobrazovat grafy průběhů sledovaných veličin. U grafů lze pak nastavovat jejich vlastnosti, jakými jsou horizontální i vertikální rozsahy (včetně automaticky vypočtených dle aktuálních hodnot), barvy, lze v nich zobrazit nastavené meze a samozřejmě se v nich i pohybovat v historii. Ukázka jednoduché sestavy grafů je na obrázku 11.



Obrázek 11 - Ukázka zobrazení grafů v systému WinControl

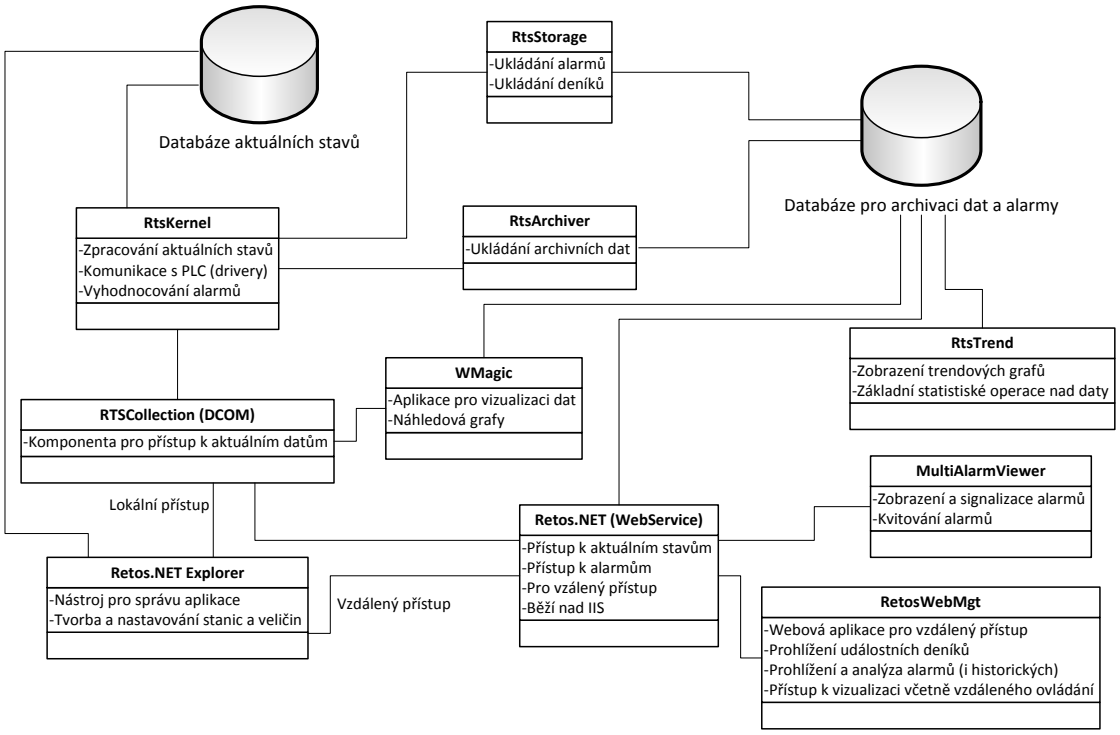
Mimo ucelené aplikace, pomocí které lze ovládat celý dispečink, systém obsahuje i modul určený pro analýzu a statistické zpracování dat – *WinStat*. Zákazníky žádané jsou také moduly pro export archivovaných dat do programu *Microsoft Excel*.

Celý systém *WinControl* je poměrně nezávislý a není náročný na hardwarové ani softwarové vybavení počítače, na kterém běží. Jedinou podmínkou je mít nainstalováno databázové prostředí BDE. Samotné aplikace si ukládají všechna svá data a nastavení do souborů, je tedy možné velmi elegantní migrace mezi různými počítači, spočívající v pouhém přenesení adresářové struktury. To je velmi výhodné například při poruše či výměně daného počítače – náhradu tak lze provést velmi rychle i bez předchozí přípravy. Systém ovšem umožňuje i duální provoz na více strojích, a to mnoha způsoby – lze využít záložního komunikačního serveru, oddělení komunikačního serveru od uživatelské aplikace či provoz vzdálených a mobilních dispečerských pracovišť.

2.5.2 Retos.NET

SCADA systém Retos.NET je rovněž vyvíjen českou společností, ostravskou *QLine, a.s.* Tento systém je také určen primárně pro oblast zpracování čisté a špinavé vody. Je s úspěchem nasazen v regionu severních Čech a severní Moravy, včetně města Ostravy. Má i několik dalších realizací spíše lokálního charakteru. Společností *GDF spol. s r.o.* byl zvolen jako jedna z možných náhrad již dříve zmíněného systému *WinControl*, a v minulých letech byl touto firmou nasazen v lokalitě severozápadních Čech (Cheb, Aš, Mariánské Lázně) a v oblasti Vsetínska.

Celý systém byl původně postaven na technologii DCOM společnosti Microsoft, ovšem díky problémům s touto technologií v rozsáhlejších nasazeních, od ní bylo upuštěno, a celý systém je postupně migrován do technologie *Microsoft .NET*.



Obrázek 12 – Struktura systému Retos.NET

Na obrázku 12 je znázorněna strukturu systému Retos.NET. Primární částí systému, který zabezpečuje veškerou komunikaci, konverzi dat a vyhodnocování alarmů je modul *RtsKernel*. Na rozdíl od dříve zmíněného systému *WinControl* je zde zvolen odlišný model komunikace s PLC. Komunikační a konverzní moduly, zde označované jako drivery (je jich k dispozici několik desítek) jsou realizovány jako dynamicky linkované knihovny (DLL) a jsou přímo součástí procesu jádra (*RtsKernelu*). Ten je, jako i ostatní moduly realizován formou Windows služby.

Jako datové úložiště, jsou pro aktuální data, konfiguraci a ostatní pracovní data využívány souborové databáze *Microsoft SQL Compact*, pro uložení datově náročnějších archivů a alarmů lze využít databázi *Microsoft SQL Server* nebo *Oracle*. O ukládání těchto dat se starají moduly *RtsStorage*

(alarmy) a *RtsArchiver* (archivní data). Archivů může být vytvořeno několik, s různou hustotou (četností ukládání) i hloubkou dat.

Pokud jde o datový model, tak ten je zde rovněž odlišný. Je nestrukturovaný, a obsahuje několik základních datových typů veličin, kterými jsou:

- Analogový vstup (AI) – načítaná číselná hodnota s plovoucí řádovou čárkou
- Binární vstup (BI) – načítaná binární hodnota
- Čítačový vstup (CI) – načítaná, ale i zapisovaná celočíselná hodnota
- Analogový výstup (AO) – zapisovaná číselná hodnota s plovoucí řádovou čárkou
- Binární výstup (BO) – zapisovaná binární hodnota
- Programově (pomocí skriptu či externí knihovny) vypočítávané varianty výše zmíněných

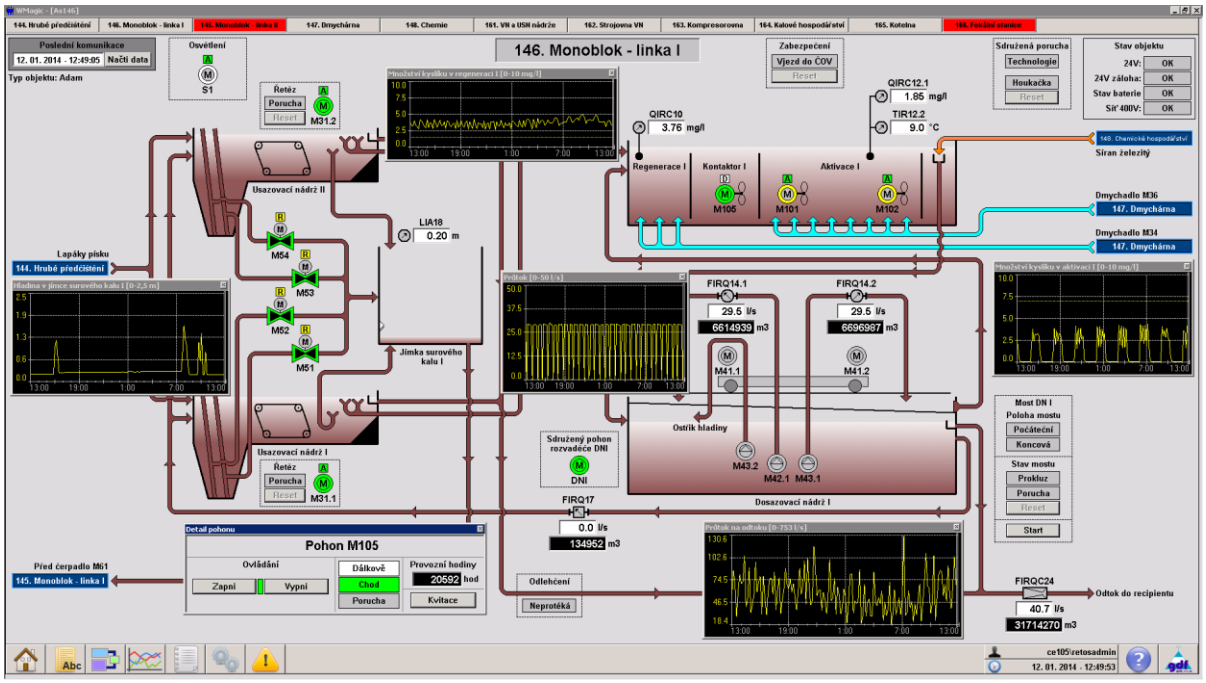
Pro adresaci se používají celočíselné indexy (pro každý typ nezávislé – lze mít tedy stejné indexy pro AI i BI). Tyto veličiny nelze nijak dále strukturovat, kromě pevně dané čtyřstupňové hierarchie, která se skládá z čísla uzlu (dispečinku, lokality), čísla stanice (PLC), typu hodnoty (AI, BI, CI…) a čísla (indexu) veličiny. Pomocí této hierarchie lze pak jednoznačně identifikovat každou veličinu v rámci celého systému. Tento přístup je sice mnohem variabilnější – lze v něm vytvořit téměř jakékoli datové variace, ale právě absence podpory strukturování na úrovni systému s sebou nese problémy při tvorbě a správě (je zde více datových bodů – menší přehlednost), a pokud chce uživatel použít nějakou systematičnost, je to čistě na něm (je nucen použít například odsazování adres pomocí offsetů).

Jak již bylo zmíněno, tak pro propagaci aktuálních dat (stavů), a to jak v rámci jednoho stroje, tak i mezi různými počítači, je použita technologie DCOM. Ta je ovšem silně závislá na ověřování pomocí Windows účtů. Tento fakt je i jedním z důvodu, proč je v celém systému využito právě systému uživatelských účtů poskytovaných Windows (ať již lokálních, tak i doménových). To s sebou ovšem nese poměrně vysoké požadavky na správnou konfiguraci celého systému, a rychlý přenos aplikace z jednoho počítače či serveru na jiný, je problémem. Jistou nevýhodou také může být nutnost použití Windows domény u složitějších síťových instalací.

Pro samotného uživatele, na rozdíl od systému *WinControl*, není k dispozici jedna aplikace, která poskytuje všechny služby, ale aplikací je několik. Pro správu a nastavení celého systému je určena aplikace *Retos.NET Explorer*, která se umožňuje se do systému připojit i vzdáleně. Stejně tak to umožňují i ostatní moduly, kterými jsou *WMagic*, zajišťující vizualizaci aktuálních dat a náhledových grafů, *MultiAlertViewer*, který je určen pro zpracování alarmů a *RtsTrend*, pro analýzu archivovaných dat a jejich zobrazení ve formě grafů. Systém umožňuje přístup k datům i pomocí webového rozhraní, k čemuž slouží webová aplikace *RetosWebMgt*. Ta umožňuje prohlížení vizualizace aktuálních dat, náhled a analýzu alarmů, a mimo jiné i prohlížení deníků, tedy záznamů provedených změn a povelů. Tato aplikace běží nad IIS (*Internet Information Service*), stejně jako další modul webového rozhraní pro vzdálený přístup k systému. Ten aktuálně poskytuje data pro prohlížečku alarmů a vzdálený přístup ke správě, v budoucnu by ale měl zcela nahradit DCOM.

Pro hromadné používání a především rychlou tvorbu aplikací, bylo vzhledem k absenci jakýchkoli vnitřních struktur dat, nutné zvolit jiný přístup. Ten je založen na využití offsetů mezi proměnnými. Nástroj pro vizualizaci pak pomocí maker umožňuje vytvořit takzvané vrstvy, tedy prototypy složitějších objektů, v nichž jsou jednotlivé veličiny adresovány právě s využitím makra, které udává počáteční adresu a ofset. Tak lze vytvořit téměř jakýkoli objekt. I tento přístup má však svá omezení. Nelze totiž vizualizaci tvořit bez podkladových dat – offsety se vypočítávají až v době běhu, a k tomu už je třeba, mimo samotné vizualizace dat, mít funkční i celý systém. Problém je ale i nízká variabilnost takto vytvořených vrstev, je tedy pro různé alternativy daného prvku (pohonu, servopohonu) vytvořit velké množství různých variací téže vrstvy, což pak způsobuje problémy při dodatečných úpravách (daný prvek má mnoho instancí, které je třeba upravit a následně otestovat).

Retos.NET ale poskytuje v oblasti vizualizace i jiné výhody, a to například možnost vytvoření vyskakovacích ovládacích oken (okno lze vidět v levé dolní části obrázku 13, společně s náhledem vizualizace) či možnosti zobrazení náhledových grafů. Mimo to poskytuje i mnohem širší možnosti při tvorbě vizualizace. Ta je sice stále z velké části bitmapově orientovaná, lze v ní ale již plnohodnotně vytvářet objekty nejběžnějších tvarů, na které lze aplikovat poměrně široké množství animací. Jistým problémem při použití animací je nutnost použití poměrně složitých adresovacích výrazů, zvláště pokud jde o nějakou složitější logiku v kombinaci s použitím maker a ofsetů.

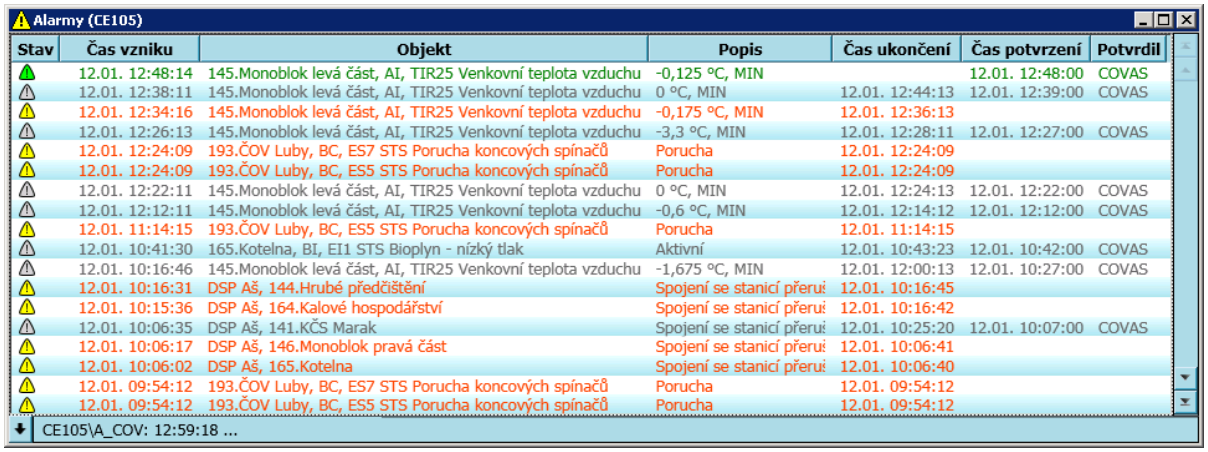


Obrázek 13 – Náhled vizualizace v systému Retos.NET

Aplikace ale poskytuje již plnohodnotné nástroje na tvorbu potrubí, které podporuje automatickou tvorbu ohybů. Navíc je každá obrazovka tvořena samostatným souborem, není tak problém práce ve více lidech na objektech jedné lokality, ani nasazování dílčích změn.

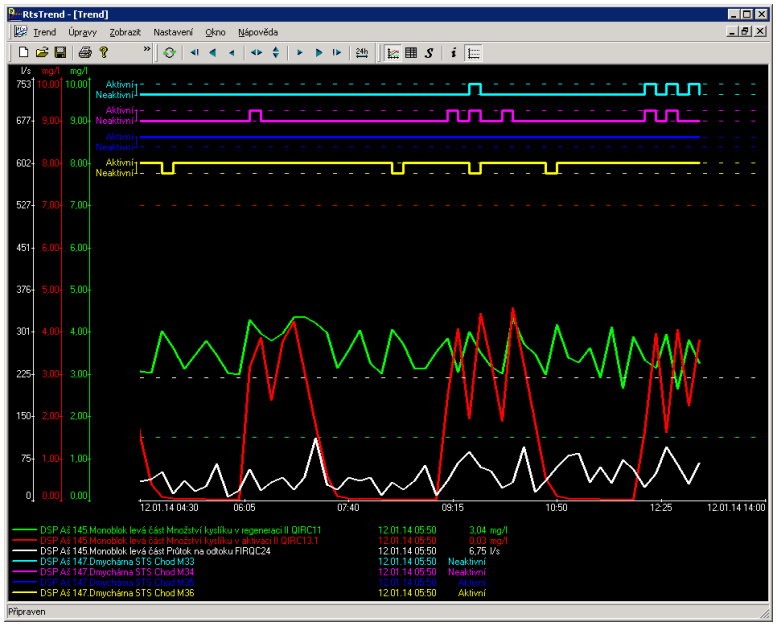
Kromě vizualizace je dalším klíčovým modulem aplikace pro zpracování alarmů. Jmenuje se *Retos.NET MultiAlertViewer* a se zbytkem systému komunikuje prostřednictvím webové služby

Retos.NET, a to i s více než jedním dispečinkem zároveň. Náhled aplikace je na obrázku 14. Je schopna zobrazit různé stavy alarmů. *Retos.NET* obecně rozlišuje alarmy ukončené nebo neukončené a potvrzené nebo nepotvrzené. V prohlížečce alarmů si pak lze zvolit libovolnou kombinaci těchto stavů a každému nastavit různou barvu. U alarmů lze také definovat různé závažnosti, jejichž pomocí lze alarmy filtrovat či pro vybrané závažnosti nastavit zvukovou signalizaci. Alarm lze potvrdit buď jednotlivě pomocí dvojkliku, nebo hromadně pomocí kontextového menu. U potvrzeného alarmu se pak zaznamenává čas a uživatel, který potvrzení provedl. Tím, že jde o samostatnou aplikaci, samozřejmě odpadá funkcionalita, kterou je rychlý přesun na místo vzniku alarmu – zde si ho musí uživatel najít svépomocí.



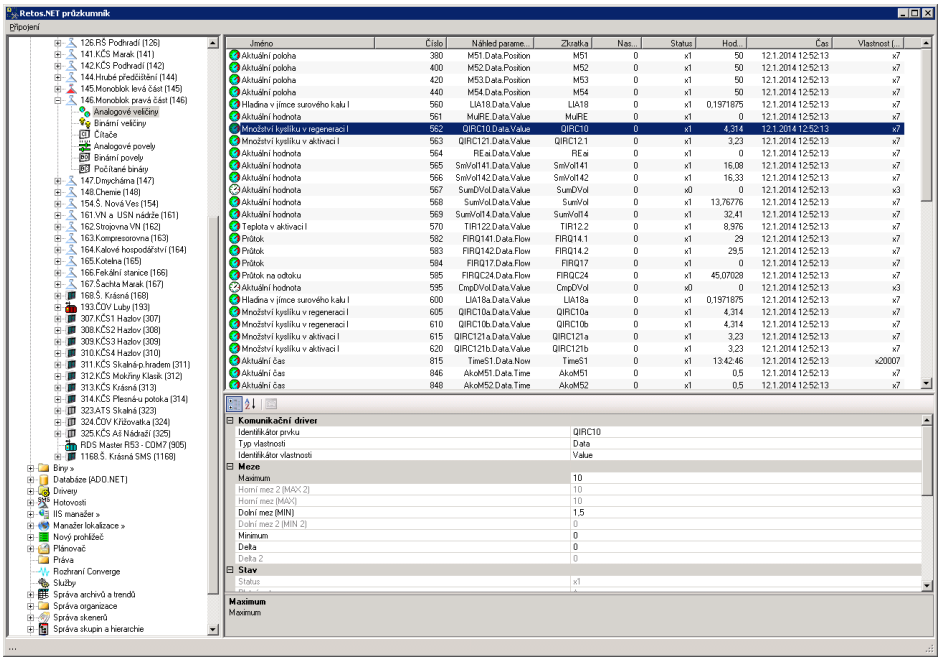
Obrázek 14 – Aplikace Retos.NET MultiAlertViewer pro práci s alarmy

Pro uživatele je k dispozici i aplikace pro tvorbu a prohlížení složitějších grafů *RtsTrend*. Ta umožňuje v jednom grafu zobrazit i několik různých typů veličin, každá může mít jiné jednotky a měřítko. Lze prohlížet i delší časové úseky a nechat si zobrazit základní statistické údaje. Samozřejmostí je možnost nastavení barev a automatická aktualizace dat. Ukázka vytvořeného grafu je na obrázku 15.



Obrázek 15 – Aplikace pro tvorbu a prohlížení grafů – RtsTrend

Poslední, zde zmíněnou aplikací, je Retos.NET Explorer. Ten nesloží přímo běžným uživatelům systému, ale jde o nástroj pro správu celého dispečinku. S jeho pomocí lze vytvářet nové stanice, veličiny, měnit jejich parametry, konfigurovat doplňkové funkce a mnoho dalšího. Jeho základní okno je na obrázku 16. V levé části obsahuje strom, který obsahuje všechny konfigurovatelné vlastnosti dispečinku. Aktuálně rozbalený je hierarchický pohled na aktuální stav veličin. V pravé horní části se zobrazují jednotlivé prvky dle polohy ve stromě vlevo. Na obrázku jsou to analogové vstupy pod stanicí *146.Monoblok, pravá část*. Jsou zde zobrazeny jejich základní vlastnosti, jako název, index, zkratka a aktuální hodnota. Detailní vlastnosti veličiny lze nastavit ve spodní části okna, případně v některém z dialogů vyvolaných pomocí pravého tlačítka myši.



Obrázek 16 – Retos.NET Explorer – základní nástroj pro správu systému

3 Motivace a návrh uživatelského rozhraní

Kapitola se skládá ze třech částí. V první, budou zhodnoceny stávající řešení v oblasti uživatelských rozhraní pro řízení technologií, budou popsány jejich slabé stránky a z nich vyvozeny požadavky, které by měl nově navrhovaný systém řešit.

Další část se zabývá výběrem vhodných technologií, které je možné použít při realizaci uživatelského rozhraní, a budou navržena možná řešení dříve definovaných požadavků. V poslední části kapitoly bude navržen způsob hodnocení výsledků práce a srovnání s existujícími systémy.

3.1 Zhodnocení stávajících řešení a motivace

Důvodů, proč se pouštět do návrhu a realizace nového uživatelského rozhraní pro řízení technologických procesů, je i přes velký počet existujících řešení několik. Většina systémů totiž ne zcela dobře řeší minimálně některé z níže uvedených požadavků, které lze od moderního uživatelského rozhraní pro řízení technologií očekávat.

Podpora různých zařízení, respektive hardwarových platforem

Většina existujících systémů je zaměřena pouze na konkrétní HW platformu. Systém určený primárně pro osobní počítače tak není uzpůsoben k tomu, aby byl provozován na jiných zařízeních. Je sice možné jej, nejčastěji formou webové aplikace, provozovat i na zařízeních jako je tablet, nebo mobilní telefon, ale vždy je to na úkor omezené funkcionality a snížení ergonomie. Podobně je tomu i u dotykových ovládacích panelů, které často také mají webové rozhraní. To je ale často řešeno velmi problematicky, prostřednictvím různých ActiveX doplňků či Java appletů, čímž jsou pak omezeny na konkrétní webové prohlížeče a platformy. V poslední době se v některých systémech objevují i nativní mobilní aplikace, ale ty jsou často pouze pro zobrazení aktuálních dat formou tabulek, tedy bez grafického znázornění technologie.

Moderní technické prostředky použité při vývoji aplikace

Velká část systémů je na trhu již poměrně dlouhou dobu a s tím souvisí i technická zastaralost a ignorování moderních technologií. Je to poměrně logický důsledek toho, že málokterý výrobce si troufne zahodit stávající práci a začít vyvíjet svůj produkt znovu na moderních technologiích. Velmi často se tak setkáme s tím, že jsou základy SCADA systémů postavené na technologiích starých více jak 15 let, které jsou jejich původními tvůrci už dávno označeny za přežité. Příkladem tak může být

technologie DCOM, kde se sice výrobci SCADA systémů alespoň snaží přicházet s náhradami, ale vlivem toho, že je odvětví poměrně konzervativní, se to daří zatím velmi pomalu.

Široká rozmanitost zobrazovacích zařízení

Většina SCADA systémů je historicky poměrně spjata s použitým zobrazovacím zařízením, nejčastěji monitorem, respektive jeho rozlišením. Je to dáno použitím rastrové grafiky a celého souřadnicového systému. V dřívějších dobách, kdy výběr monitorů a jejich rozlišení byl poměrně malý, to nebýval až takový problém. Naopak byl velký problém v reálném čase vykreslovat složitou vektorovou grafiku tak, aby ve výsledku vypadala dobře – s vyhlazováním hran, anti-aliasingem apod. To ale již nějakou dobu problém není, zvláště s příchodem akcelerace vykreslování přes grafickou kartu. Ale opět na to výrobci SCADA systémů reagují velmi pomalu nebo vůbec.

Vysoká otevřenost systému

Dalším, poměrně zásadním problémem, je v dnešní době poměrně malá otevřenost většiny systémů aplikacím třetích stran. Doplnění funkcionality či změna chování dle přání zákazníka, je většinou minimálně problematická, ne-li rovnou nemožná.

Špatná podpora dotykového ovládání

Většina SCADA systémů (mimo vestavných dotykových ovládacích panelů) není uzpůsobena pro moderní způsoby dotykového ovládání. Samozřejmě není problém je ovládat pomocí dotykového pera (označovaného jako stylus), ale to se zrovna neslučuje s moderními trendy, kde většina dnes prodávaných dotykových panelů používá pro snímání pozice dotyku prstu kapacitní technologii. Největším problémem jsou tak malé ovládací prvky a časté použití pravého tlačítka myši či plovoucích nápověd, tzv. tooltipů.

Bezpečnost

Kybernetická bezpečnost kritické infrastruktury je dnes stále větším tématem diskuzí v odborných, ale už i méně odporných – politických kruzích. Ač se většina systémů snaží mít alespoň nějakou úroveň zabezpečení, často není tak vysoká, aby se vůbec dala nazývat bezpečností. Je to dáno špatným návrhem komunikačních protokolů, které málokdy řeší šifrování či autentizaci na úrovni samotného protokolu.

Poměrně často je možné, aby se útočník při znalosti protokolu dostal do systému, včetně možnosti ovládání technologie. Dalším problémem je zastaralost použitých technologií, kde po ukončení podpory výrobcem zůstává neošetřeno poměrně velké množství bezpečnostních děr. Problémy se často řeší oddělením těchto systémů od jiných sítí, ale v praxi je to často uděláno nedůsledně, nebo je oddělení po čase přímo eliminováno, jako důsledek pohodlnosti a nedůslednosti.

Cenová politika

Cena SCADA systému sice není úplně technické téma, ale v praxi je často velmi důležité, ne-li nejdůležitější (s ohledem na dnešní způsob výběru dodavatelů). Pokud se vrátíme zpět k modelovému příkladu s využitím systému ve vodárenství, tak pro vodárenskou společnost na velkém městě, se statisíci zákazníky, není problém, aby do systému investovali řádově miliony korun. Zato pro malou okresní vodárnu, kterých je v ČR drtivá většina, to problém je. Firma má řádově desetinu zákazníků, ale často i složitější infrastrukturu. Přeci jen je jednodušší mít jeden vodojem nad městem, který zásobuje polovinu jejich zákazníků, než vodojemů desítky, pro každou zapadlou vesničku. U takových společností je i investice v řádu stovek tisíc korun velký problém.

3.2 Základní návrh a výběr technologií

Primárním cílem této práce je tedy navrhnout a vytvořit ukázku SCADA systému, který zohlední vlastnosti dříve popsaných systémů, pokusí se eliminovat jejich slabé články a v neposlední řadě přinést i nové způsoby řešení a celkově modernější přístup.

Jak již bylo zmíněno dříve, tak není vhodné tvořit systém příliš universální, je lepší se zaměřit na konkrétní oblast použití. Vzhledem k možnosti získat reálná provozní data a zkušenosti od firmy *GDF spol. s r.o.* a následně i možnosti, si výsledky práce ověřit v reálném prostředí, bude navržené rozhraní zaměřeno na využití ve vodárenství a zpracování odpadních vod.

Dle zadání by se práce měla zaměřit primárně na uživatelské rozhraní, ale to nemůže fungovat bez podpůrné infrastruktury pod ním. V následujícím textu tedy bude popsán celý systém, včetně částí určených pro sběr dat, jejich archivaci, práci s alarmovými stavy a v neposlední řadě i s možnostmi dalšího rozšíření. Práce si neklade za cíl vytvořit plně funkční SCADA systém s veškerou funkcionalitou, ale cílem je spíše ukázat směr a možnosti, které současné technologie nabízejí.

Výsledná aplikace by měla mít dvě části. Serverová by měla zajišťovat funkce spojené s komunikací s PLC, správou dat, archivací a vyhodnocování alarmů. Klientská aplikace by pak měla zobrazovat vizualizaci aktuálních dat a zajišťovat veškerou interakci s uživatelem.

3.2.1 Softwarová platforma

Jedním ze zásadních problémů při návrhu nového systému, je určit platformu, ve které se daný SCADA systém následně implementuje. Ta by měla mimo jiné podporovat následující funkce a vlastnosti:

- Podpora více typů HW zařízení (PC, tablet, možnost dotykového ovládání).
- Podpora pokročilých prostředků usnadňujících implementaci (objektové programování lambda výrazy, podpora práce s vlákny, *garbage collector*).
- Podpora relačních databází.
- Podpora pokročilých síťových komunikačních protokolů.

- Definice uživatelského prostředí pomocí deklarativního jazyka (např. XML) s možností interpretace za běhu (bez překladu).
- Dobrá podpora rozšiřitelnosti (připojování knihoven za běhu).
- Rozšíření (platforma by měla být snadno dostupná bez složité instalace).

Pokud si tedy shrneme výše uvedené požadavky, tak z běžně rozšířených programových a běhových prostředí, se nabízí *Java* a *Microsoft .NET Framework*. Vzhledem ke zkušenostem autora byl zvolen *.NET Framework*.

3.2.2 Komunikační rozhraní

Důležitou částí systému je rovněž komunikační rozhraní, respektive komunikační protokol mezi serverem a klientskou aplikací – ať už vlastní či externí. Právě kvůli možnosti využití tohoto rozhraní pro další rozšiřující aplikace, je vhodné použít takové, které bude pro ostatní aplikace co nejpřívětivější. Mělo by se tedy jednat o multiplatformní standard.

Další nutnou funkcí, kterou by mělo komunikační rozhraní poskytovat, je zabezpečení. Minimem je dodržovat základní bezpečnostní požadavky, jako je důvěrnost přenášených dat, jejich integrita, a ideálně i zajištění autentizace a autorizace.

Rozhraní by mělo být poskytováno v nějakém standardizovaném komunikačním protokolu a tím umožnit dobrou použitelnost aplikacemi třetích stran.

3.2.3 Uživatelské rozhraní

Při návrhu uživatelského rozhraní je třeba dbát na jeden ze základních požadavků, kterým je podpora i pro dotykové ovládání. To s sebou nese mimo jiné některá zásadní omezení, kterými jsou:

- Dostatečná velikost ovládacích prvků i pro ovládání prsty na zařízeních s menšími úhlopříčkami displeje.
- Nutnost implementovat rozhraní tak, aby jej bylo možné přizpůsobit velikosti a rozlišení obrazovky (volitelné zvětšení).
- Jednoduchost a intuitivnost.
- Omezené použití klávesnice a více tlačítek myši (pravé tlačítko, kolečko), případně náhrada funkcí pomocí dotykových gest.

Dalším faktorem při návrhu je ujasnění si, které informace je vhodné zobrazit a kdy. Pokud si vezmeme základní obrazovku dané stanice (objektu, PLC), tak na ní bude vhodné zobrazit tyto základní informace:

- Identifikaci objektu (název, číslo)
- Informaci o způsobu komunikace s objektem (čas posledního načtení dat, hlášení poruch komunikace)
- Grafický náhled obrazovky objektu, který by měl zobrazovat:

    - Schéma technologie, propojení trubek, nádrží…
    - Aktivní prvky, jako motory, servopohony a měření sloužící pro zobrazení aktuálního stavu technologie.
    - Názvy prvků, popisy.
- Tabulku alarmových stavů.

Vzhledem k tomu, že většina aktivních prvků na obrazovce objektu obsahuje i nějaké doplňující informace, jako jsou například detailní stavy poruch, informace o blokování pohonů, grafy měřených veličin a v neposlední řadě i prvky pro ovládání, je zapotřebí tyto informace a ovládací prvky někde zobrazit. Nabízí se několik způsobů.

Prvním je zobrazit je přímo na základní obrazovce objektu. To je poměrně jednoduchý způsob, na první pohled ne úplně špatný. Všechny informace jsou viditelné ihned, je možné aktivní prvky rovnou ovládat. Funguje to relativně dobře a přehledně, ale pouze do jisté složitosti objektu. Pokud objekt obsahuje mnoho aktivních prvků (pohonů, měření), je problematické umístit všechny detailní informace a ovládací prvky na jednu obrazovku. Bude jednoduše příliš velká a nebude ji možné zobrazit najednou. Využití posuvníků či rozdělení na více obrazovek není ideální, ani z hlediska přehlednosti, ani praktičnosti a ergonomie.

Druhou variantou je zobrazení detailů a ovládacích prvků ve vyskakovacím okně. Funguje to tak, že po kliknutí na aktivní prvek se zobrazí okno s chybějícími údaji. Není to špatný způsob, jelikož je poměrně přehledný, šetří prostor a nespornou výhodou je možnost mít takových oken otevřeno více. Je zde ovšem drobný problém s poněkud horší ergonomií práce s okny na dotykových zařízeních.

Třetí, v existujících aplikacích poněkud netradiční, ale nakonec i zvolenou metodou, je využití vyskakovacího bočního panelu. Informace a ovládací prvky, včetně případných grafů, jsou zobrazeny v relativně úzkém panelu v pravé části obrazovky. Pokud panel není potřeba, tak zmizí, například po kliknutí (dotyku) na prázdnou plochu obrazovky. Po kliknutí na aktivní prvek (pohon, měření) se panel znovu zobrazí. Toto řešení šetří místo na hlavní obrazovce, ale oproti vyskakovacím oknům se ovládá přímočařeji. Postrádá ovšem možnost zobrazení doplňkových dat z více aktivních prvků zároveň.

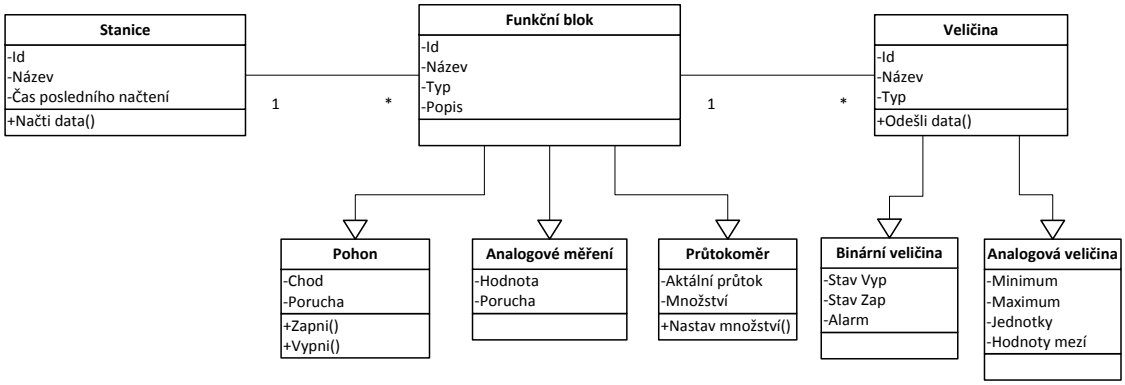
3.2.4 Návrh hierarchie dat

Důležitou částí návrhu je také ujasnění si základních datových prvků a jejich hierarchie. Zvláště se to týká těch prvků, které slouží pro reprezentaci dat načtených z PLC a popisují tak stav technologie daného objektu. Jak již bylo popsáno dříve, tak objektem lze chápat jednu ucelenou funkční jednotku, například vodojem, čerpací stanici či čistírnu odpadních vod.

Každý objekt je tedy třeba definovat nějakou datovou strukturou, nazvěme ji *stanice* (název objekt se v tomto kontextu dále používat nebude díky jeho mnohovýznamnosti).

Stanice by měla mít jméno a identifikátor. Protože stanice reprezentuje samotné PLC, měla by také zajišťovat načítání dat a udržovat informace o tom, kdy došlo k poslednímu platnému načtení a také obstarávat vyhodnocení výpadku spojení.

Každá stanice se pak skládá z dílčích prvků odpovídajících fyzickým zařízením, jakými jsou například pohony a měření. Ty lze souhrnně pojmenovat označením *funkční blok*. Ten by měl opět mít nějaký identifikátor, název a v neposlední řadě i popis – text upřesňující konkrétní instanci bloku. Příkladem popisu může být *Hladina ve vodojemu*, *Venkovní teplota* či *Čerpadlo vratného kalu*.



Obrázek 17 - Základní datové struktury

Ze základní třídy *funkčního bloku* jsou dále odvozeny třídy, reprezentující konkrétní funkci, například pohon, analogové měření nebo průtokoměr. Každá odvozená třída může obsahovat jinou vnitřní strukturu *veličin* a některé typy funkčních bloků i další doplňující funkce, jak je uvedeno na obrázku 17.

Jednotlivé *veličiny* mají také vlastní datovou strukturu, kterou opět tvoří identifikátor, název a typ. *Veličina* rovněž umožňuje zápis své hodnoty do PLC a může mít několik odvozených typů, které se liší především typem hodnoty, kterou reprezentují. U binárních veličin se určuje především význam jednotlivých binárních stavů, a zda není některý pro některý ze stavů vyhodnocován alarm. U analogových veličin je naopak důležité vědět rozsah (minimum a maximum), jednotky a hodnoty mezí pro signalizaci alarmových stavů po jejich překročení.

3.3 Návrh ověření a hodnocení výsledků práce

Pro hodnocení výsledků práce byly zvoleny dvě základní kritéria. První je založeno na definici požadavků na funkcionalitu, bezpečnost, ergonomii, celkovou použitelnost a následné vyhodnocení jejich splnění. Druhé kritérium vychází z hodnocení potencionálních uživatelů navrženého systému formou dotazníku, kdy ve spolupráci s firmou GDF spol. s r.o. bylo domluveno posouzení zaměstnanci přímo dotčených firem, v tomto případě pak zaměstnanců dispečerských pracovišť několika společností, pracujících v oboru výroby a distribuce pitné vody a zpracování vod odpadních.

3.3.1 Definice požadavků a systém hodnocení

V následující kapitole bude vytvořen souhrn požadavků na moderní SCADA systém, přičemž u jednotlivých bodů bude určena i jejich váha. Při hodnocení jednotlivých kritérií pak bude u každého bodu procentuálně určen podíl jeho splnění. Tato čísla pak budou vynásobena váhou jednotlivých kritérií a jejich vážený součet pak poskytne výsledné hodnocení celého systému.

Jednotlivé požadavky na systém a zároveň i hodnotící kritéria jsou vypsány v následující tabulce, pro přehlednost jsou rozděleny do několika základních kategorií. Jednotlivá kritéria a jejich váhy byly zvoleny tak, aby co nejvíce pokryly odlišnosti jednotlivých systémů. I když je tak základní funkčnost systému jedním z klíčových parametrů, nemá příliš vysokou váhu, protože ji musí splňovat všechny SCADA systémy – hodnocení je tedy primárně zaměřeno na přidanou hodnotu a funkce, které existujícím systémům spíše chybí.

Kategorie požadavku	**Upřesnění požadavku**	**Váha**
Základní funkčnost		**15%**
	Způsoby přenosu dat	5%
	Archivace dat (použité databáze – otevřené, proprietární)	5%
	Podpora alarmů, jejich provázání se zbytkem systému	5%
Použité technologie		**20%**
	Podporované HW platformy	5%
	Podpora alternativního (dotykového) ovládání	3%
	Podporované SW platformy	2%
	Podpora síťových protokolů	2%
	Podpora „tenkého klienta“ (většina dat by měla být v systému a ne lokálně)	3%
	Způsob tvorby vizualizace a způsob jejího uložení (otevřený nebo proprietární formát)	2%
	Podpora vlastních datových struktur	3%
Rozšiřitelnost a otevřenost		**20%**
	Možnosti přizpůsobení – funkční	5%
	Možnosti přizpůsobení – grafické	5%
	Možnost rozšíření funkcionality, propojení na systémy třetích stran	5%
	Podpora vlastních (nových) typů PLC	3%

	Dokumentace, otevřený kód, dostupnost rozhraní pro využití jinými SW systémy	2%
Grafické zpracování		**15%**
	Způsob vykreslování s hlediska HW akcelerace	5%
	Podpora vektorové grafiky	5%
	Možnost tvorby vlastních grafických šablon	5%
Bezpečnost		**12%**
	Podpora šifrovaného přenosu	3%
	Zabezpečení přístupu k systému (autentizace a autorizace)	5%
	Aktuální SW platforma (neaktuální knihovny mohou představovat bezpečnostní problém)	4%
Ergonomie ovládání		**18%**
	Dostatečná velikost ovládacích prvků, grafické názornost	5%
	Jednoduchost a intuitivnost ovládání	8%
	Prvky zvyšující přehlednost a rychlost reakce	5%

Tabulka 1 - Požadavky pro hodnocení a jejich váha
(každá kategorie má svou váhu, která je součtem vah jednotlivých požadavků)

3.3.2 Uživatelské hodnocení

Pro účely uživatelského hodnocení by bylo vhodné vypracovat jednoduchý dotazník, pomocí kterého by se dalo získat hodnocení od potencionálních uživatelů systému. Dotazník by se následně měl nechat vyplnit statisticky významným množství lidí, kteří se problematikou SCADA systémů zabývají, případně s nimi přímo pracují.

Protože je ale velmi komplikované se s takovými lidmi kontaktovat a navrženou aplikaci jim prezentovat (nelze to udělat veřejnou anketou přes internet), bude dotazník experimentálně předložen pouze několika vybraným lidem. Výsledky tak sice nebudou vypovídající, ale mohou naznačit trend.

4 Realizace navrženého rozhraní

Následující kapitola popisuje způsob implementace definovaných požadavků na SCADA systém, které vycházejí z návrhu, popsaného v přechozí kapitole. Bude popsána základní architektura systému, definována struktura dat a způsob jejich uložení, způsob realizace komunikačních rozhraní, ať už pro komunikaci s technologickými PLC, tak i s aplikacemi třetích stran. Dále se kapitola zabývá implementací vyhodnocování alarmových stavů, archivací dat a v neposlední řadě také realizací samotného uživatelského grafického rozhraní.

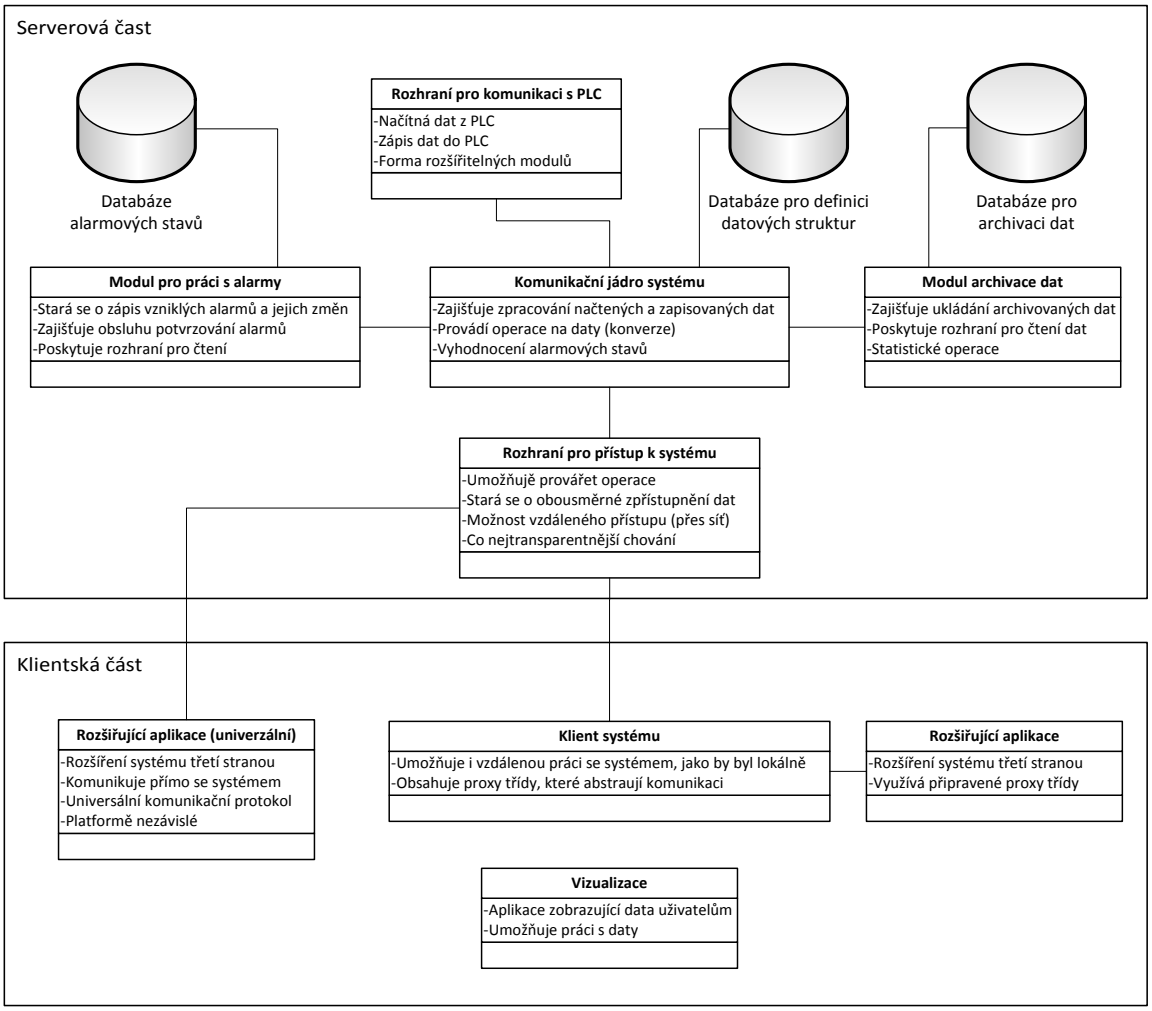
V závěru kapitoly je pak provedeno hodnocení vytvořené aplikace na základě, v předchozí kapitole definovaných, kritérií a je také provedeno srovnání s existujícími řešeními. Nakonec byl v rámci implementace vytvořen i dotazník pro uživatelské hodnocení, který byl experimentálně předán několika lidem z oboru, a následně bylo provedeno i hodnocení jeho výsledků.

4.1 Architektura systému

Jednou z prvních a snad i nejdůležitějších otázek, kterou si je třeba při realizaci systému položit, je jeho architektura. Už z principu by se u tohoto typu software mělo jednat o aplikaci typu klient-server. Vzhledem k tomu, že jedním z kritérií je použití na co nejrozmanitějších typech zařízení, měla by být klientská část co nejjednodušší. Tím pádem se drtivá většina operací bude provádět na serveru, klient se bude starat pouze o zobrazení a uživatelskou interakci.

Při zohlednění funkcí popsaných v úvodu této práce, pak dostaneme základní architekturu systému zobrazenou na obrázku 18. Serverovou část systému tvoří primárně komunikační jádro. To se stará o načítání dat z jednotlivých PLC (zdrojů telemetrických dat) za pomoci rozličných komunikačních rozhraní – *driverů*. Těch může být několik, v závislosti na použitém komunikačním protokolu. S jádrem systému budou jednotlivé *drivery* komunikovat pomocí předem definovaného universálního rozhraní. Mimo načítání dat ještě jádro provádí jejich konverzi do vnitřních datových struktur. Tyto struktury si ukládá do speciální (definiční) databáze. V neposlední řadě se jádro systému stará o vyhodnocování alarmových stavů. V tom spolupracuje s modulem pro správu alarmů a vzniklé alarmové stavy se ukládají do databáze.

Mimo komunikační jádro a alarmy je součástí navrženého systému i modul pro archivaci dat a práci s nimi a modul pro lokální i vzdálený přístup k systému z aplikace klientské a také aplikací externích. Rozhraní by mělo být poskytováno v nějakém standardizovaném komunikačním protokolu a tím umožnit dobrou rozšiřitelnost aplikacemi třetích stran. Jako nejvhodnější kandidát se, i s ohledem na vybranou softwarovou platformu, jeví použití protokolu SOAP (*Simple Object Access Protocol*), respektive jeho implementace ve WCF (*Windows Communication Foundation*).



Obrázek 18 – Základní architektura navrhovaného systému

Klientská část systému je tvořena knihovnou implementující nějakou formu *proxy* tříd, které budou zajišťovat co nejtransparentnější přístup do systému, ať se ním pracuje lokálně nebo vzdáleně, prostřednictvím síťové infrastruktury. Nad touto knihovnou pak bude rovněž možné tvořit externí aplikace rozšiřující základní funkcionalitu vytvořeného systému.

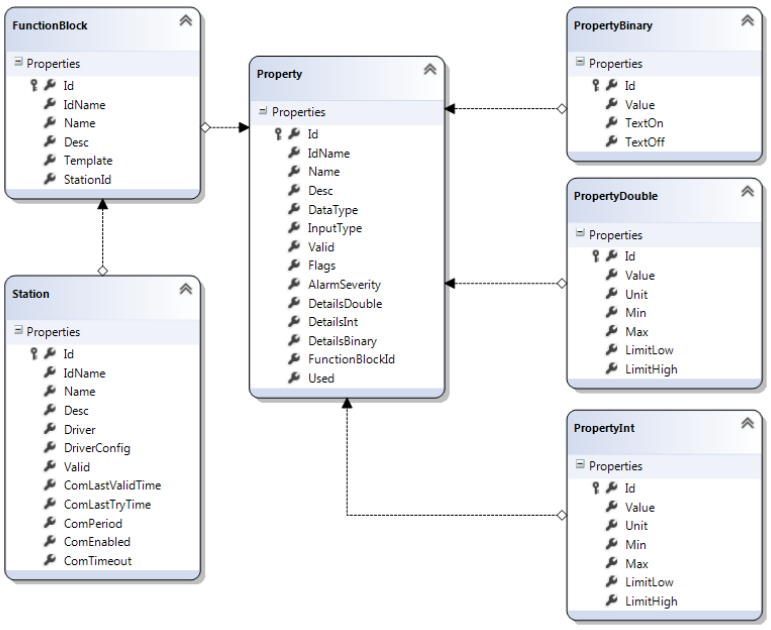
Knihovna bude využita i poslední části systému, kterou je samotná vizualizace. Ta se stará se o zobrazení aktuálního stavu technologie uživateli a o jeho zpětnou vazbu ve formě povelů pro ovládání technologie, či pro ovládání samotného systému.

4.2 Datová úložiště

Uložení persistentních dat je v aplikaci realizováno za použití databáze typu SQL. Pro jednoduchost byla zvolena databáze *Microsoft SQL Server*, ve variantě *Local Database*. Na cílovém počítači tak nemusí být nainstalován plnohodnotný databázový systém, ale pouze jeho část. Data jsou uložena v jednom datovém souboru (doplněném o transakční log), který se načte do paměti pouze v případě potřeby. Toto řešení má samozřejmě svá omezení, hlavně co se týče výkonu. V případě potřeby ale

není problém využít i plnohodnotnou databázi – stačí pouze upravit odpovídající připojovací řetězce, takzvané *connection stringy*.

Jak již bylo nastíněno v kapitole 4.1, tak SCADA systém používá databází několik. První z nich je databáze s definicí datových struktur (stanice, funkční bloky, veličiny), jejich aktuálního stavu a nastavení. Model této databáze je na obrázku 19.



Obrázek 19 - Model definiční databáze

Model vychází z návrhu v kapitole 4.1, jen je doplněn o více detailních informací. Jednotlivé dílčí typy funkčních bloků také nejsou definované přímo v databázi, ale za pomoci připravených šablon. Ty budou popsány detailněji v následujícím textu.

Zajímavá je také realizace uložení dat jednotlivých veličin (*Property*), kde data společná pro více datových typů jsou ve společné tabulce a údaje specifické pro konkrétní typ jsou v separátních tabulkách.

Nad touto databází jsou pak pomocí mechanizmu *LINQ to SQL* z *Microsoft .NET Frameworku* vytvořeny proxy objekty. Zjednodušeně řečeno se z databázové reprezentace dat (tabulky, atributy, relace) vytvoří jejich odpovídající reprezentace v datových strukturách konkrétního jazyka, v tomto případě jazyka *C#*. Vzniknou tak objekty a jejich atributy, relace jsou realizovány za pomoci referencí doplněných o seznamy (kolekce referencí na objekty). Framework se pak stará o vzájemné konverze dat, jejich načítání (i zpožděné až v době, kdy jsou data skutečně potřeba), ukládání a tvorbu nových záznamů. Snaží se to dělat maximálně transparentně tak, aby se s databázovými daty dalo pracovat jako s daty přímo v paměti dané aplikace.

Podobně jako definiční databáze jsou realizovány i databáze pro ukládání alarmových stavů a archivních dat. Na první pohled logičtější by se zdálo uložení všech těchto dat v databázi jediné, ale z praktického hlediska je lepší rozdělení. Výhodami tohoto řešení je možnost použití různých úložišť a

databázových systémů pro definice a data, možnost nezávislého zálohování, případně i přenos relativně malé definiční databáze mezi různými počítači. V případě havárie systému tak lze snadno a rychle rozběhnout základní funkcionalitu (zobrazení aktuálních dat, možnost ovládání) a obnovu dat archivních nechat na později.

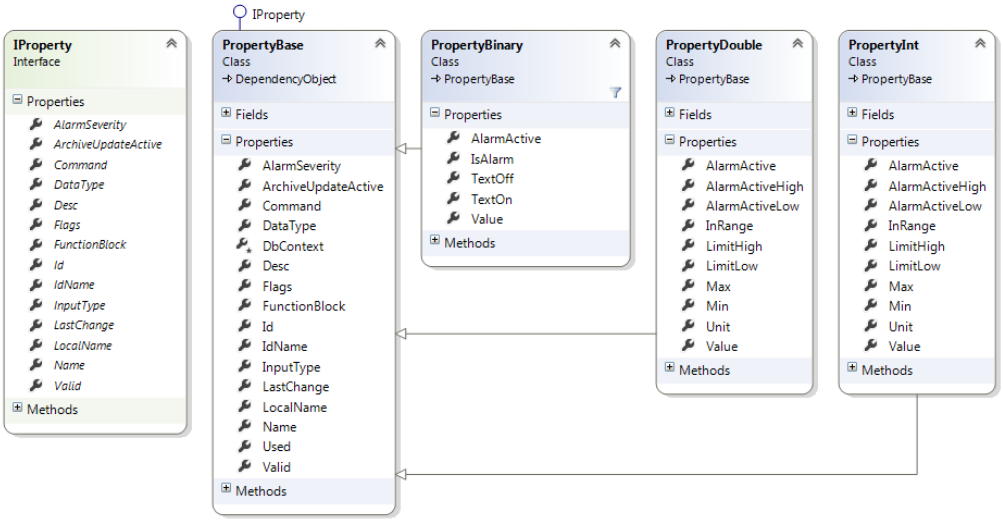
4.3 Datový a adresní model

Implementace datových struktur vychází z návrhu popsaného v kapitole 3.2.4 a rovněž i z databázového modelu popsaného v kapitole předchozí. Datový model se tak skládá z **veličin** (neboli vlastností funkčních bloků, odtud *property*), **funkčních bloků** (*function block*) a **stanic** (*station*).

4.3.1 Veličiny

Veličina, neboli *property*, je nejmenší datovou jednotkou schopnou pojmout data načtená či zapisovaná do technologického zařízení – PLC. Má určitou hierarchii (viz obrázek 20), která vychází z rozhraní *IProperty* definujícího základní atributy tohoto datového typu. Rozhraní *IProperty* je implementováno bázovou třídou *PropertyBase*, která sdružuje veškerou funkcionalitu společnou pro všechny odvozené datové typy veličin.

Z bázové třídy jsou pak odvozeny jednotlivé typy pro binární veličiny (*PropertyBinary*), veličiny s plovoucí řádovou čárkou (*PropertyDouble*) a veličiny celočíselné (*PropertyInt*). Odvozené typy implementují atributy, které mají význam pouze pro daný typ nebo se vzájemně vnitřní implementací liší. Například u *PropertyDouble* jsou všechny vnitřní datové typy atributů v datovém typu *Double*, kdežto u *PropertyInt* v 64 bitovém celém čísle.

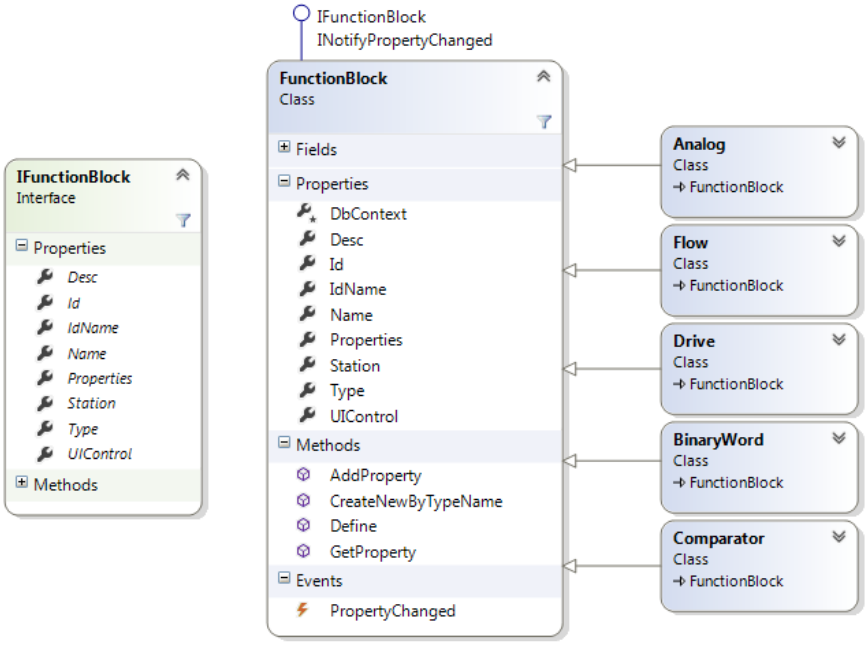


Obrázek 20 - Implementace datového modelu vlastnosti

Mimo základní implementaci atributů datová struktura *veličiny* řeší i replikaci svého stavu do databáze a zpět, stará se o vyhodnocování alarmových a mezních stavů, a nabízí možnost povelování, neboli odesílání (zapisování) dat do driveru a potažmo i PLC.

4.3.2 Funkční bloky

Funkční blok je datová struktura reprezentující konkrétní fyzické technologické zařízení, tedy například pohon, měřidlo průtoku nebo hladiny. Vychází z rozhraní *IFunctionBlock*, které je implementováno v bázové třídě *FunctionBlock* - viz obrázek 21.



Obrázek 21 - Implementace funkčního bloku

Od bázové třídy se dále odvozují konkrétní implementace jednotlivých typů. Ty ovšem nemusejí být statické, vytvořené pouze programově, ale teoreticky mohou být i dynamické, definované například ve formě doplňkové DLL knihovny či XML souboru.

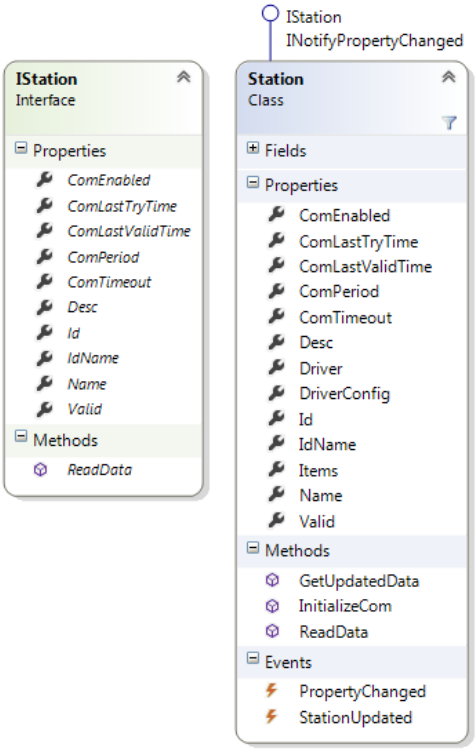
Celý mechanizmus funguje tak, že pokud uživatel v definici struktury stanice (viz dále) uvede, že daný prvek je typu *Drive*, aplikace prohledá veškeré aplikaci známé typy z definovaného jmenného prostoru a vytvoří objekt na základě tohoto typu. Typ *Drive* si pak, rovněž dynamicky, vytvoří svou strukturu potřebných veličin, kterou má interně definovanou. Tímto způsobem tak lze i uživatelsky vytvářet vlastní typu funkčních bloků. Nutno ale podotknout, že v rámci ukázkové aplikace není tento systém zcela dokončen. Problém tví převážně v tom, že aplikace načítá typy pouze z jedné DLL knihovny. To by ale bylo řešitelné například tím, že by se prostřednictvím konfiguračního souboru definoval seznam známých typů, společně s názvy a cestami k DLL knihovnám, ve kterých jsou uloženy. Rovněž by bylo možné vytvořit generický typ, který by seznam veličin, jejich názvů a typů načítal z přímo z konfiguračního souboru. Pak by ale nebylo možné, doplnit k funkčnímu bloku specifickou funkční logiku.

Funkční blok řeší i způsob informování zbytku systému o změnách dat. Data z většiny PLC jsou sice načítána periodicky, ale většina z nich se mění poměrně málo. Není tedy úplně vhodné všechna data při každém načtení posílat a vyhodnocovat napříč celým systémem. Řešením je použití mechanismu, kdy má funkční blok implementováno rozhraní *INotifyPropertyChanged*, které funguje

tak, že pokud je zaznamenána změna veličiny (její hodnoty či nějakého dalšího atributu), je vyvolána událost *PropertyChanged*, které je předán název dané veličiny. Na událost je možné dále reagovat, například odesláním nových dat ze serveru klientovy či přímo překreslením dané části grafického uživatelského rozhraní.

4.3.3 Stanice

Stanice je datový typ reprezentující konkrétní PLC, respektive fyzický objekt. Vychází z rozhraní *IStation*, implementovaného v třídě *Station*, jak lze vidět na obrázku 22.



Obrázek 22 - Implementace stanice

Mimo identifikátory obsahuje třída stanice i atributy a metody týkající se komunikace s PLC, jako jsou časy a periody komunikací, platnost načtených dat či metoda pro okamžité načtení dat.

Třída stanice opět implementuje rozhraní *INotifyPropertyChanged* s obdobnou funkcí jako u funkčního bloku. Mimo to ale implementuje i další způsob práce se změnami dat. Pro některé způsoby použití, typicky se to týká přenosu dat po síti, není úplně vhodné přenášet dílčí změny dat jednotlivých funkčních bloků ihned po změně. Zvláště pokud nejmenší jednotkou, kterou lze z PLC přečíst je celá stanice, u které se předpokládá, že bude změn dat více. Po dokončení načtení a zpracování dat z PLC je tak vyvolána událost *StationUpdated*. Na ni může být navázáno vyvolání komunikace a následný přenos dat ze severu na klienta. Je ale třeba mít k dispozici i konkrétní nová data, která je třeba poslat. To zajišťuje metoda *GetUpdatedData*, která za pomoci časových razítek změn dat u jednotlivých funkčních bloků určí ty, které je třeba přenést.

4.3.4 Adresní model

Jistou výzvou bylo zvolit vhodný adresovací model, který by umožňoval identifikaci daného prvku v rámci celého systému od stanice až po veličinu. Zvláštní ohled byl kladen na uživatelskou přívětivost a z možných variant tak byl zvolen model, kdy je adresace řešena pomocí textového řetězce a identifikačních názvů. Identifikační název by měl odpovídat skutečnému názvu prvku, tedy pokud je měření označené jako LIC1, tak LIC1 bude i identifikační název. Nemusí to být ale pravidlem a v některých případech je vhodnější zvolit název kratší a tím i uživatelsky přívětivější. Například u pohonu, kde je definován signál *Režim – dálkově* je vhodnější zvolit identifikačním název *modeRem*.

Ve výsledku pak lze z těchto identifikačních názvů složit adresu každé jednotlivé veličiny, která se skládá z identifikačních názvů *stanice, funkčního bloku* a *veličiny*, oddělených znakem lomítka. Adresovat lze samozřejmě i relativně, pokud tedy tvořím vizualizaci technologické obrazovku určité stanice, lze do adresy prvku vizualizace uvést pouze označení funkčního bloku. Prvek vizualizace funkčního bloku se pak skládá z dalších částí, například indikátoru zvoleného režimu, který je opět relativně adresován na konkrétní veličinu.

Tímto řešením adresace se docílí toho, že uživatel aplikace nemusí jednotlivé prvky složitě dohledávat jako v případě, kdy by byly adresovány pouze pomocí číselných referencí. Ty jsou samozřejmě v rámci zvýšení výkonu aplikace použity také, ale pouze interně.

4.3.5 Definice stanice

Jak bylo zmíněno v předchozích kapitolách, struktura *stanice*, včetně *funkčních bloků* a *veličin*, je uložena v definiční databázi. Podstatný je ale také způsob, jakým se tato struktura v definiční databázi vznikne.

Slouží k tomu definiční soubor ve formátu XML, jehož příklad je na obrázku 23. Kořenová značka *station* definuje stanici, u které je možné nastavit její identifikátor, periodu komunikace a driver (ovladač pro komunikaci s PLC). Je třeba rovněž definovat cestu ke konfiguračnímu souboru konkrétního driveru.

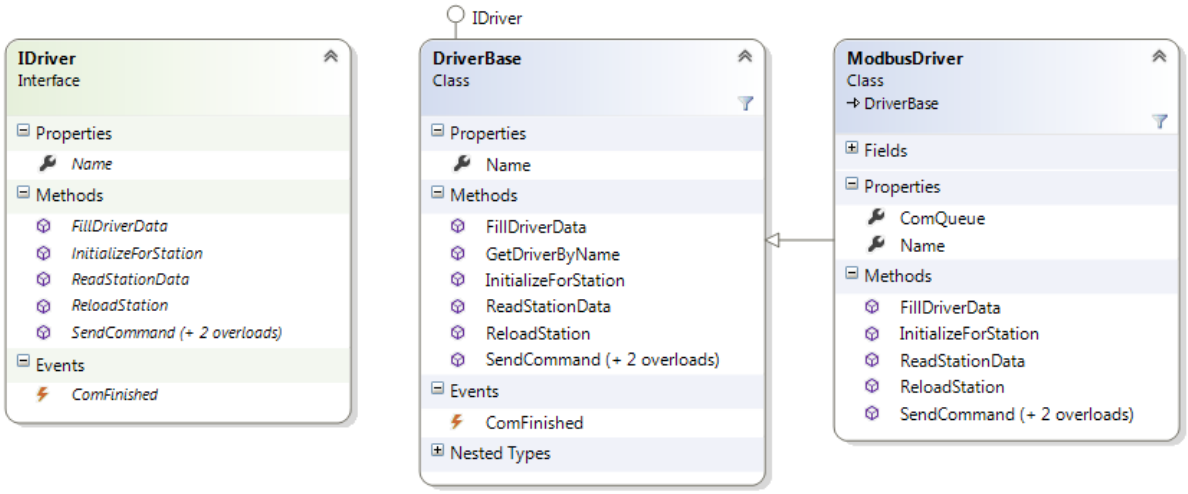
Stanice je dále tvořena XML značkami *item*, z nichž každá reprezentuje právě jeden funkční blok. U něj je nutné určit jeho typ, název a je možné doplnit i popis či identifikátor, pokud se liší od názvu funkčního bloku. Další vnořené XML značky již závisí na konkrétním typu funkčního bloku.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<station Id="1" IdName="1.VDJ_Slavkov" Name="1. VDJ Slavkov" Driver="Modbus"
         DriverConfig="SCADA_Test.Driver.xml" Period="1000" Timeout="800">
  <item Type="Drive" Name="M1">
    <blockCondition Id="0">Výpadek napájení 400V</blockCondition>
    <blockCondition Id="1">Vysoká hladina LIC1a</blockCondition>
    <blockCondition Id="2">Nízká hladina LIC2a</blockCondition>
  </item>
  <item Type="BinaryWord" Name="SYS">
    <bit0 alarm="1">Výpadek napájení 400V</bit0>
    <bit1 alarm="1">LZ1.1 - Minimální plovák</bit1>
    <bit2 alarm="1">LZ2.1 - Minimální plovák</bit2>
    <bit3 alarm="1">LZ1.2 - Maximální plovák</bit3>
  </item>
  <item Type="Analog" Name="LIC1" Desc="Hladina ve vodojemu">
    <minimum>0</minimum>
    <maximum>2,6</maximum>
    <limit-low>0,7</limit-low>
    <limit-high>2,3</limit-high>
    <unit>m</unit>
  </item>
  <item Type="Comparator" Name="LIC1a" Desc="Řídí výtlak M1">
    <minimum>0</minimum>
    <maximum>2,6</maximum>
    <unit>m</unit>
  </item>
  <item Type="Flow" Name="FIQ1" Desc="Průtok ze studny">
    <minimum>0</minimum>
    <maximum>100</maximum>
  </item>
</station>
```

Obrázek 23 - Příklad definičního XML souboru stanice

4.4 Vyčítání dat – komunikace s PLC

Pro čtení dat z jednotlivých zařízení, byl v systému navržen a realizován systém takzvaných *driverů*. Jako *driver* lze chápat unifikovaný modul, sloužící pro komunikaci s určitým typem zařízení, za pomoci určitého komunikačního protokolu. Driver má také za úkol konverzi načítaných a zapisovaných dat do vnitřních struktur systému.



Obrázek 24 - Rozhraní a třídy reprezentující drivery

Pro driver bylo vytvořeno rozhraní *IDriver* (viz obrázek 24), které reprezentuje základní atributy a metody, které každý *driver* musí implementovat. Kromě názvu *driveru* obsahuje následující metody:

- **InitializeForStation** – metoda vytvoří konkrétní interní instanci driveru pro danou stanici. Každý driver tak má jedinou instanci jeho samotného, ale tato jedna instance je schopná načítat data z více fyzických zdrojů – PLC. Jednotlivá čtení totiž musí být vzájemně synchronizovaná v případech, kdy probíhají přes sdílené médium, jakým je například sběrnice RS485. Součástí volání jsou i data pro mapování (adresaci) prvků mezi komunikačním protokolem, PLC a samotným SCADA systémem. Způsob mapování už je specifický pro konkrétní driver.
- **ReloadStation** – slouží pro obnovu mapování za běhu (při změně).
- **ReadStationData** – volání metody vyvolá načtení dat z PLC. Způsob načtení je už čistě na driveru, ale čtení dat se vždy provádí asynchronně. Volání tedy pouze zařadí požadavek na čtení do interní fronty a samotné čtení je realizováno až v době, kdy se požadavek dostane na řadu, respektive se uvolní komunikační linka. Načtená data si pak driver uloží do vnitřní paměti a vyvolá událost *ComFinished*.
- **FillDriverData** – tato metoda je volána po dokončení komunikace (načtení dat). Provádí se v ní konverze dat a jejich projekce na vnitřní datové struktury – *veličiny*. Metody pro čtení dat a jejich projekci (předání do systému) jsou oddělené záměrně, lépe se tak docílí možnosti paralelního čtení a zpracování dat.
- **SendCommand** – metoda slouží pro odeslání povelu – zápis dat ze SCADA systému do PLC. Opět jde o asynchronní proces, který je ovšem prováděn prioritně. Odeslání povelu je tak vždy zařazeno na začátek komunikační fronty. Způsob odeslání a převod dat je opět dán implementací konkrétního driveru. Metoda má několik variant, které slouží k odeslání různých datových typů. Ty korespondují s typy veličin, je tedy možné odeslat data binární, celočíselná a data s plovoucí řádovou čárkou.

V rámci ukázkové implementace byl realizován pouze driver pro komunikační protokol *Modbus* popsaný v kapitole 2.4.3. Při realizaci byla použita knihovna *Free .NET Modbus Library*, která implementuje samotný *Modbus* protokol a v driveru bylo implementováno pouze rozhraní mezi touto knihovnou a zbytkem systému.

4.4.1 Konfigurační soubor Modbus driveru

Jak již bylo napsáno v předchozí kapitole, každý typ driveru může mít zcela odlišnou konfiguraci a je tedy řešena přímo v daném driveru. Pro referenční implementaci driveru typu Modbus, vznikla i referenční podoba konfiguračního souboru, kterou si lze prohlédnout na obrázku 25.

Kořenovou XML značkou je *modbusStation*, u které je v atributech uveden způsob komunikace, *Modbus Id* a další atributy upřesňující reprezentaci dat. Kořenový XML prvek musí obsahovat alespoň jednu značku *portConfig*, která je závislá na zvoleném komunikačním protokolu (sériová linka či ethernet).

```
<?xml version="1.0" encoding="utf-8" ?>
<modbusStation Id="1" Type="tcp" Port="ETH" ModbusId="1"
           SwapByte="False" SwapWord="False" MaxRegs="100">
  <portConfig Port="502" Address="192.168.8.107" />

  <item IdName="M1/stRun" Address="30.0" Invert="False"/>
  <item IdName="M1/stErr" Address="30.1" Invert="False"/>

  <item IdName="M1/motohours" Address="36" Storage="Dword" Coef="0,001"/>

  <item IdName="M1/cmdOn" Address="31.0" Invert="False"/>
  <item IdName="M1/cmdOff" Address="31.1" Invert="False"/>

  <item IdName="SYS/bit0" Address="46.0" Invert="False"/>
  <item IdName="SYS/bit1" Address="46.1" Invert="False"/>
  <item IdName="SYS/bit2" Address="46.2" Invert="False"/>
  <item IdName="SYS/bit3" Address="46.3" Invert="False"/>

  <item IdName="LIC1/value" Address="8" Storage="Float"/>
  <item IdName="LIC1/error" Address="46.5"/>

  <item IdName="FIQ1/flow" Address="0" Storage="Float"/>
  <item IdName="FIQ1/volume" Address="2" Storage="Dword" Coef="0,001"/>
</modbusStation>
```

Obrázek 25 - Příklad konfiguračního souboru Modbus driveru

Konfigurační soubor dále obsahuje seznam vnitřních proměnných PLC reprezentovaných XML značkami *item*. Pomocí nich se pro *veličinu* určenou pomocí atributu *IdName* určí adresa paměťového registru a další parametry pro interpretaci dat.

4.5 Zpracování alarmových stavů

Jednou z primárních funkcí SCADA systému je upozornit uživatele (dispečera) na případné problémy. K tomu složí vyhodnocování mezních a alarmových stavů, které je v navrhovaném systému řešeno na úrovni jednotlivých veličin, a logicky liší se podle typu veličiny.

U binárních veličin se alarm vyhodnocuje na základě binárního stavu proměnné. Zda je daná veličina alarmovým stavem a má se tudíž jako alarm vyhodnocovat, se určuje v definičním souboru stanice. Při přechodu takovéto veličiny z logické 0 do logické 1, je vždy vygenerován nový alarm. Ten si s sebou v tuto chvíli nese informaci o veličině, která jej vyvolala a času vzniku. Každý vzniklý alarm je pak neprodleně zobrazen uživateli. Ten jej v tuto chvíli může potvrdit a tím zaznamenat, že o něm ví, a začít jej řešit. Po potvrzení přejde alarm do nového stavu, kdy již není prioritní a tento stav je uživateli signalizován méně výrazně a rovněž je zaznamenán čas potvrzení.

Alarm odezní ve chvíli, kdy veličina opět přejde do stavu logické 0. Je zaznamenán čas ukončení alarmu a po změně stavu je uživateli signalizován ještě méně výrazně, pouze pro zpětnou kontrolu. Pokud je alarm ukončen dříve, než jej uživatel potvrdí, zůstává signalizován jako velmi důležitý, dokud jej uživatel nepotvrdí.

U analogových veličin jsou alarmy vyhodnocovány tak, že u každé veličiny lze v definičním souboru stanice nastavit dolní a horní mezní hodnotu. Po překročení některé mezní hodnoty je vygenerován alarm, který v sobě nese nejen zdrojovou veličinu a čas, ale také informaci o tom, která

z mezí byla překročena a hodnotu, kterou veličina měla v době překročení meze. Alarm analogové veličiny má stejné stavy jako u veličiny binární (potvrzený, ukončený).

Všechny vzniklé alarmy včetně historie změn jejich stavů, jsou ukládány do databáze.

4.6 Archivace dat

Pro archivaci dat je možné zvolit dva způsoby realizace jejich ukládání. První ukládá vždy pouze změny hodnot oproti hodnotě předcházející. To není úplně špatná myšlenka – rozhodně šetří datový prostor, zvláště pokud se hodnoty v čase moc nemění. Je ovšem problém s poněkud horší interpretací takových dat, pokud z nich zobrazujeme jen část (je třeba data prohledávat hodně do minulosti). Druhým problém nastává ve chvíli, kdy se naopak data mění příliš často. To se děje zvlášť u systémů, kdy je SCADA systém přímo připojen k technologickým PLC a je možné data načítat, a tím pádem i ukládat, i několikrát za vteřinu. Děje se to zvláště u analogových měření, která se (i vlivem rušení) v nízkých řádek, mění poměrně často. Datová úspora pak padá, zvláště pokud chceme data archivovat po delší časové období.

Druhou metodou je periodické ukládání všech hodnot v určitém čase, tedy tvorba časových řezů napříč daty. Výhodou je, že lze poměrně snadno bez dalších operací získat obraz celého stavu v definovaném čase. Také lze taková data snadno zobrazovat, počítat nad nimi statistiku a snadno ovlivňovat jejich množství volbou vhodné periody archivace. Nevýhodou je pak často hrubší četnost dat a tedy možnost, že některé významné limitní hodnoty nebudou zaznamenány.

Vzhledem k výše popsaným faktům byla v ukázkové aplikaci implementována druhá varianta s periodickým ukládáním. V každém, periodou archivace daném cyklu, jsou tak posbírána všechna aktuální data a jsou uložena do databáze. Každý datový záznam se skládá z identifikátoru veličiny, času uložení a hodnoty. Možnost různých datových typů pro hodnotu je řešena tak, že databázová entita obsahuje více atributů různých typů, kdy vždy do jednoho z nich se hodnota uloží.

Archivní data lze následně číst pomocí speciální třídy *ArchiveReader*, která implementuje metody, kterými lze pro danou veličinu načíst tabulku dat v definovaném časovém období. V rámci ukázkové aplikace jsou archivní data zobrazena pouze ve formě grafů, ale je poměrně snadné je vyčítat i externí aplikací za pomoci SOAP rozhraní, které bude popsáno v následující kapitole.

4.7 Síťová komunikace a aplikační rozhraní

Z návrhu architektury systému je zřejmé, že podstatnou částí celého řešení je síťové komunikační rozhraní mezi serverovou a klientskou částí aplikace. Toto komunikační rozhraní je dále využitelné i pro zpřístupnění dat SCADA systému aplikacím třetích stran.

Síťová komunikace je řešena pomocí součásti *Microsoft .NET Frameworku* s názvem *Windows Communication Foundation*, která je rozšířenou implementací protokolu *SOAP*. Obě technologie jsou

pro samotnou implementaci a její pochopení natolik důležité, že jsou krátce popsány v následujících kapitolách.

4.7.1 SOAP - Simple Object Access Protocol

Jedná se o komunikační protokol založený na XML fungující na principu zasílání zpráv přes počítačovou síť. Základním komunikačním prvkem je tzv. zpráva (*message*), které se skládá ze SOAP obálky (*envelope*), což je kořenový element XML dokumentu posílaného ve zprávě. Zpráva může obsahovat i nepovinnou hlavičku, která je primárně určena pro doplňující informace týkající se způsobu přenosu, platnosti zprávy či dalších informací podobného charakteru. Je možné ji ale využít i pro doplňující informace o datech samotných.

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
    <n:priority>1</n:priority>
    <n:expires>2001-06-22T14:00:00-05:00</n:expires>
    </n:alertcontrol>
  </env:Header>
  <env:Body>
    <m:alert xmlns:m="http://example.org/alert">
      <m:msg>Pick up Mary at school at 2pm</m:msg>
    </m:alert>
  </env:Body>
</env:Envelope>
```

Obrázek 26 – Ukázka SOAP zprávy

SOAP zpráva se dále skládají z těla (*body*), které nese samotná data a případně i sekci, která řeší zpracování chybových stavů. Příklad SOAP zprávy je na obrázku číslo 26. Zaujme na něm i hojné využití XML jmenných prostorů.

Samotný přenos dat může probíhat přes značné množství komunikačních protokolů nižších vrstev. Způsob, jakým je samotná XML zpráva převedena na jeden z podporovaných protokolů, je definován jako tzv. *binding*. Mezi běžně podporované protokoly patří HTTP, HTTPS, TCP či SMTP, ale při splnění podmínek daných standardem je možné přenášet SOAP zprávy po téměř jakémkoli síťovém protokolu.

Jistou výhodou i nevýhodou tohoto komunikačního protokolu je použití jazyka XML. Výhody jsou v relativně snadné implementaci a jednoznačné interpretaci napříč platformami, především díky jeho široké podpoře. Je navíc i dobře uživatelsky čitelný, což rovněž usnadňuje implementaci.

Nevýhodou je pak poměrně velká režie, a to jak na straně množství přenášených dat (XML hlavičky, otevírací a uzavírací značky), tak i na straně potřeby vyššího výpočetního výkonu pro zpracování oproti protokolům s binární interpretací dat. Množství přenášených dat lze ale omezit například použitím komprese na úrovni podkladového komunikačního protokolu a větší spotřeba výpočetního výkonu dnes u většiny aplikací není až tak zásadní problém.

4.7.2 Windows Communication Foundation

Windows Communication Foundation (dále WCF) je jedna z implementací protokolu SOAP a v množství aspektů jej i rozšiřuje. Vznikla jako náhrada protokolu DCOM, po ukončení jeho podpory.

Základní koncept tkví v tom, že aplikace pomocí WCF poskytuje ostatním nějakou službu. Každá služba je definována několika prvky, mezi něž patří:

- **Adresa** – určuje, kde je služba umístěna.
- **Binding** – určuje jak je služba dostupná. Patří sem definice přenosového protokolu (HTTP, TCP…), definice zabezpečení zpráv (SSL, *SOAP message security*) a způsob kódování zprávy (textově, binárně). Služba může poskytovat více *binding* s různými nastaveními.
- **Kontrakt** – určuje, co a jak služba nabízí. Definuje tedy, zda je komunikace jednosměrná (dotaz - odpověď), či obousměrná (server posílá klientova zprávy i asynchronně), definuje podporované datové typy (komplexní) a v neposlední řadě definuje i podporované operace.

Druhou část WCF pak tvoří *Klientský kanál* (*Client's Channel*), který se k výše popsané službě připojuje. Je definován opět *adresou* služby, ke které se připojuje, *bindingem* (jedením konkrétním nebo skupinou) a rovněž i *kontraktem*.

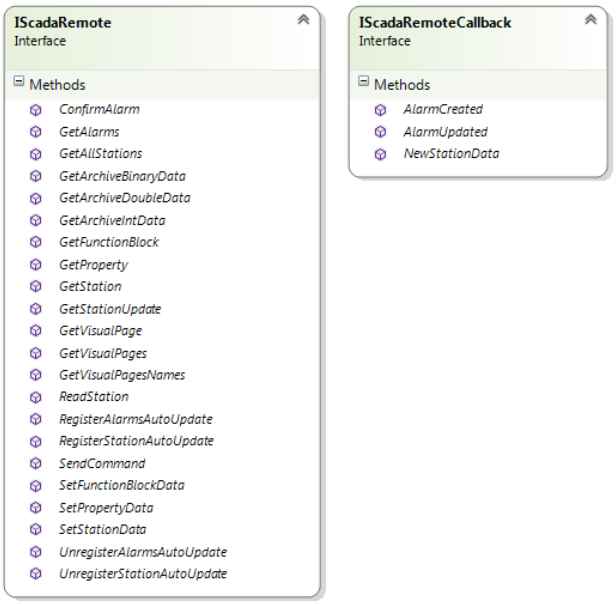
Z výše uvedeného tedy plyne, že klient musí explicitně vědět, co a v jakém formátu služba nabízí. Výhodou WCF ovšem je, že tyto informace lze snadno zjistit přímo načtením z konkrétní služby za pomoci protokolu WSDL (*Web Services Description Language*). Jeho hlavním cílem je právě popis výše popsaných služeb včetně *bindingu* a popisu *kontraktu*. Za pomoci specializovaných nástrojů si lze pak pouze ze znalosti adresy služby a místa, kde nabízí svá WSDL *metadata*, poměrně jednoduše vygenerovat kód klientské části aplikace. Výsledkem je například *proxy* třída, na které pouze voláme službou definované operace. Jsou vygenerovány rovněž službou definované a používané komplexní datové typy.

Použitím WCF lze tedy získat možnost přenosu dat na otevřených standardizovaných protokolech, podporu mnoha přenosových protokolů a v neposlední řadě tím vyřešíme i bezpečnost. Výhodou je jistě i možnost obousměrného asynchronního přenosu dat (jen na některých přenosových protokolech).

4.7.3 Implementace serverové části komunikačního rozhraní

Serverová část komunikačního rozhraní je realizována jako WCF služba, která je obousměrná, tedy umožňuje číst a zapisovat data na základě požadavků klienta, ale rovněž je zasílat klientovy asynchronně, pokud dojde v datech, které si klient označil pro sledování, k nějaké změně. Ve výchozím nastavení služba poskytuje rozhraní pomocí protokolu TPC-IP, který poskytuje vyšší výkon, podporu funkcí a menší režii, než například protokol HTTP. Použití protokolu TPC-IP však není podmínkou, je možné použít i jiné protokoly změnou nastavení *bindingu* v konfiguračním souboru. Tam lze také nastavit požadovaný způsob zabezpečení komunikační linky, jako je šifrování dat či způsob autentizace.

Ale nyní již k samotnému rozhraní WCF služby, respektive operacím, které nabízí. Ty si lze prohlédnou na obrázku 27. Ve zkratce jde o operace týkající se *stanic*, *funkčních bloků* a *veličin*, kdy lze pro danou veličinu načíst její aktuální stav či tento stav změnit. Další operace zpřístupňují archivní data a alarmy, případně vizualizační stránky.



Obrázek 27 - Operace rozhraní poskytované serverovou částí systému (vlevo) a operace pro zpětné zasílání asynchronních dat klientovi (vpravo)

Zvláštní skupinou operací jsou ty, které slouží pro registraci požadavků na automatické zasílání dat pomocí zpětného kanálu. Pokud si pomocí nich klient zaregistruje odběr změn z nějaké stanice, automaticky obdrží data této stanice pomocí operace *NewStationData* (viz obrázek 27), pokud dojde k jejich změně. Podobně funguje i automatické zasílání stavu alarmů.

Vzhledem ke komplikované implementaci standardních datových struktur (*stanice*, *funkční blok*, *veličina*), které obsahují značné množství atributů a funkcí určených pro interní zpracování, případně sloužících pro moduly driverů, byly pro účely přenosu po síti vytvořeny zjednodušené verze těchto datových struktur. Dalším důvodem pro vznik této, na první pohled duplicitní implementace, je i to, že standardní struktury používají pro reprezentaci kolekcí přímo reference na konkrétní objekty. WCF si s tím sice poradí, ale není to příliš praktické. V případě, že by si chtěl někdo načíst prvek stanice, který v sobě obsahuje funkční bloky, tak v případě přenosu přes WCF, se přenesou data všech obsažených funkčních bloků a veličin, což nemusí být vždy žádané. Struktury upravené pro přenos přes WCF tak obsahují pouze seznamy identifikátorů a je možné je tak načíst až když jsou skutečně třeba.

4.7.4 Implementace klientské části

Výhodou použití WCF, potažmo protokolu SOAP doplněného o WSDL je, že si lze klientskou část komunikačního rozhraní snadno vygenerovat v některém z mnoha prostředí a jazyků, které to umožňují. Tento generovaný kód však může posloužit pouze jako základ, zvlášť když je klientská aplikace

komplikovaná. Potom je vhodnější nad generovaným kódem vytvořit ještě nějakou formu zapouzdření, řešenou návrhovým vzorem nazývaným *proxy třída*.

Při implementaci tak vznikla knihovna, která zapouzdřuje celou síťovou komunikaci a veškeré dostupné objekty se snaží abstrahovat tak, aby uživatel nepoznal rozdíl, zda k daným objektům (stanicím, funkčním blokům, veličinám, alarmům, archivním datům) přistupuje lokálně, nebo vzdáleně. V rámci knihovny jsou řešeny i automatické aktualizace dat v klientské části, pokud dojde ke změně dat na straně serveru. Celý proces je samozřejmě optimalizován a jsou tak posílána pouze skutečně změněná data. Změny dat dále mohou vyvolat událost, respektive mechanizmus *INotifyPropertyChanged*, což dále způsobí například překreslení prvku ve vizualizaci.

Zajímavým problémem při řešení bylo, aby klientská část aplikace nemusela mít k dispozici aktuální databázi funkčních bloků, tedy podtříd odvozených z třídy *FunctionBlock*. Každá odvozená třída totiž může mít, a často i má jiné atributy a metody. Vzhledem k tomu, že se počítá s tím, že bude možné funkční bloky definovat externími DLL knihovnami, není přenášení všech těchto knihoven na klienta úplně ideální řešení.

V rámci funkčního bloku tak bylo využito mechanismu *DynamicObject* z *Microsoft .NET 4.0*, který umožňuje vytvořit objekt, který nemá pevně definovanou třídu. Jeho atributy a metody jsou vyhodnocovány až při běhu programu. Tím byl problém relativně snadno a elegantně vyřešen.

4.8 Uživatelské rozhraní aplikace

Jendou z klíčových částí SCADA aplikace je samotné uživatelské rozhraní. Je to pro uživatele nejviditelnější část celého systému, a to i co se technologické části týče, protože složí jako často jediný nástroj, se kterým člověk při práci nějakým způsobem pracuje. Při realizaci SCADA systému je třeba se zaměřit na tři oblasti návrhu, kterými jsou:

- **Samotné uživatelské prostředí** – tím je myšlen celkový vzhled aplikace, rozmístění a podoba ovládacích prvků.
- **Způsob návrhu a zobrazení technologických obrazovek** – každé technické zařízení (stoj, soustava potrubí a čerpadel…) je třeba uživateli nějak přiblížit. To se děje pomocí technologických obrazovek, které jsou schematickým znázorněním skutečnosti. Obrazovky musejí být uživatelsky definovatelné a samozřejmě musejí obsahovat nějaké další prvky, které uživateli graficky podají informaci o aktuálním stavu zařízení. Logicky by jedna technologická obrazovka měla odpovídat právě jedné *stanici*.
- **Definice a způsob použití grafických prvků** – každá technologická obrazovka se skládá z menších logických i fyzických celků, například motorů nebo analogových měření. Ty mají často jednotnou podobu, a proto je vhodné pro ně vytvořit nějakou formu šablony. Z šablony pak v technologické obrazovce vytvoříme konkrétní instance. Logicky grafický prvek odpovídá *funkčnímu bloku*, bude se tedy dále v textu označovat jako *vizualizace funkčního bloku*.

Pro jednoduchost a přehlednost je vhodné, aby pro tvorbu všech výše uvedených částí uživatelského rozhraní, bylo možné použít stejný nástroj. Jako vhodné se jeví použití jazyka XAML, což je deklarativní jazyk založený na XML, určený právě pro tvorbu uživatelských rozhraní. Mimo definice vzhledu je v něm možné modelovat i chování, kdy v závislosti na hodnotě proměnné, je možné měnit grafickou podobu, a naopak lze definovat i chování, kdy po nějaké akci (například kliknutí na grafický prvek) je možné nastavit hodnotu definované proměnné či vyvolat definovanou metodu.

Toto chování popisuje návrhový vzor nazývaný *Model View ViewModel* (MVVM), který je odvozen od návrhového vzoru *Model View Controler* (MVC). Úkolem MVVM je oddělit vzhled a chování uživatelského rozhraní od aplikační logiky a dat. Vazba mezi *Modelem* (aplikační logika) a *View* (pohledem - uživatelským rozhraním) se provádí za pomoci *ViewModel*, který pomocí tzv. *binding* (deklarativních vazeb) mapuje prvky uživatelského rozhraní na prvky aplikační logiky. Více se o tomto návrhovém vzoru dočtete v .

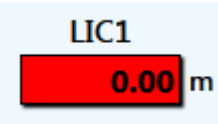
4.8.1 Vizualizace funkčního bloku

Pro každý funkční blok si lze vytvořit několik různých vizualizačních prvků, a to tak, že se vytvoří nová třída implementující rozhraní *IScadaUIControl*. Pomocí atributů tohoto rozhraní se určuje adresa konkrétní instance funkčního bloku a typ bočního postranního panelu, pokud se má po kliknutí na prvek nějaký vyvolat. Grafickou podobu a chování pak lze definovat přímo za pomoci jazyka XAML. Zjednodušený příklad definice *displeje* je na obrázku 28.

```
<UserControl x:Class="Scada.UIControls.Display">
    <Grid>
        <TextBlock Text="{Binding Path=Name" FontWeight="SemiBold" FontSize="16" Margin="0,-2,0,0"/>
        <Label Content="{Binding Path=value.Value}">
            <Label.Style>
                <Style TargetType="Label">
                    <Style.Triggers>
                        <DataTrigger Binding="{Binding Path=value.AlarmActive}" Value="True">
                            <Setter Property="Background" Value="Red"></Setter>
                        </DataTrigger>
                        <DataTrigger Binding="{Binding Path=value.InRange}" Value="False">
                            <Setter Property="Background" Value="MediumSlateBlue"></Setter>
                        </DataTrigger>
                        <DataTrigger Binding="{Binding Path=error.Value}" Value="True">
                            <Setter Property="BorderBrush" Value="Red"></Setter>
                            <Setter Property="BorderThickness" Value="3"/>
                        </DataTrigger>
                    </Style.Triggers>
                    <Style.Setters>
                        <Setter Property="Background" Value="White"/>
                        <Setter Property="BorderBrush" Value="Black"/>
                        <Setter Property="BorderThickness" Value="1"/>
                    </Style.Setters>
                </Style>
            </Label.Style>
        </Label>
        <Label Content="{Binding Path=value.Unit}"></Label>
    </Grid>
</UserControl>
```

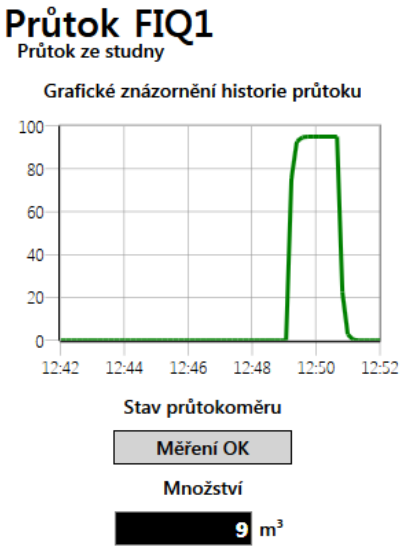
Obrázek 28 - Příklad definice vizualizace funkčního bloku (zjednodušeno)

Displej obsahuje textové pole z názvem funkčního bloku, ohraničené políčko, pro zobrazení hodnoty a indikaci mezních nebo alarmových stavů, a textové pole s jednotkami, ve kterých je veličina měřena. Grafickou podobu *displeje* je možné si prohlédnout na obrázku 29, kde je znázorněn i alarmový stav – pozadí *displeje* je červené.



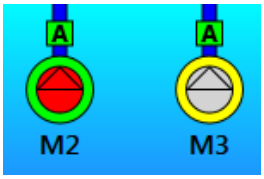
Obrázek 29 - Výsledná grafická podoba displeje

Jak již bylo zmíněno dříve, každá vizualizace funkčního bloku může mít postranní panel, který se otevře po kliknutí na definovanou část vizualizace. V tomto postranním panelu je možné zobrazit detailní informace o daném funkčním bloku. V příkladu panelu průtokoměru na obrázku 30, je zobrazeno celkové množství protečené vody, indikátor chyby měření a v neposlední řadě také grafický průběh archivovaných dat momentálního průtoku. Postranní panel se definuje za pomoci jazyka XAML.



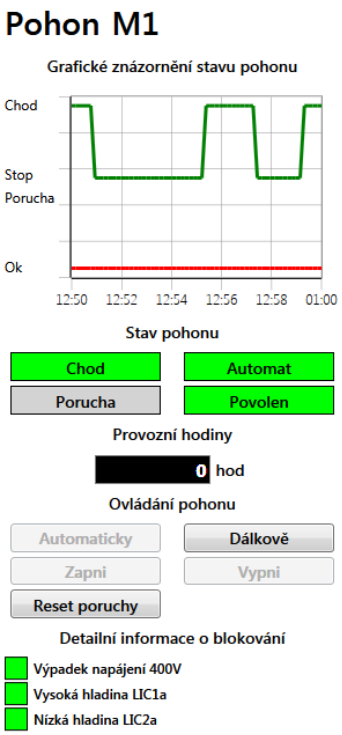
Obrázek 30 - Příklad postranního panelu pro průtokoměr

Nejsložitějším funkčním blokem, včetně vizualizace a postranního panelu, je funkční blok pohonu. Jeho vizualizace se skládá z čtverečku označujícího režim (mění se písmeno a barva), vnějšího mezikruží, které signalizuje, zda je pohon povolen (zda by se měl spustit v automatickém režimu), a vnitřní části, která různými barvami signalizuje poruchu, chod či klidový stav. Vzhled funkčního bloku a jeho postranního panelu si lze prohlédnout na obrázcích 31 a 32. I v tomto případě je grafická podoba a chování definováno pouze za pomoci jazyka XAML.



Obrázek 31 - Grafická podoba funkčního bloku pohonu v různých stavech.

Grafické podoba vizualizace jednotlivých funkčních bloků byla navržena tak, aby byla co nejvíce intuitivní a přehledná, a přitom poskytovala uživateli co největší množství informací. Použitá barevná schémata a technologická označení vycházejí z normy ČSN ISO 14617.



Obrázek 32 - Postranní panel funkčního bloku pohonu – zobrazení stavu, blokovacích podmínek, grafické historie a tlačítek pro ovládání

4.8.2 Vizualizace technologických obrazovek

Vizualizace technologické obrazovky by měla co nejvěrněji, ale přitom přehledně zobrazit skutečnou podobu technologie. Proto je mnohem vhodnější použít schématické znázornění, než reálný vzhled (fotky či detailní grafické zpracování technologických zařízení).

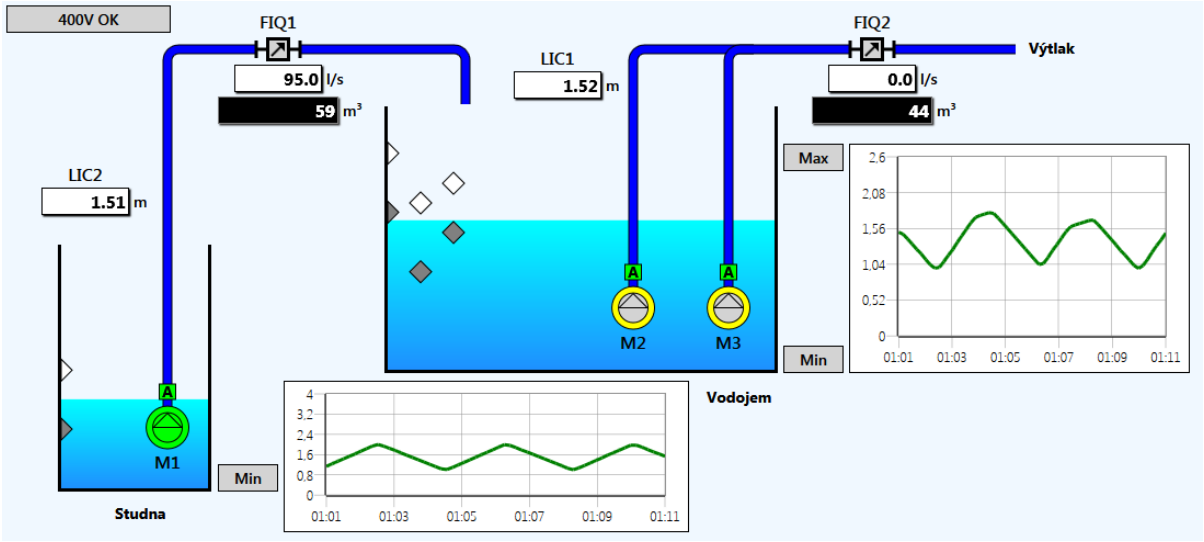
Jako prostředek pro definici vzhledu a chování obrazovek byl použit opět jazyk XAML. Výhodou jeho použití je možnost v rámci technologické obrazovky využít jak existujících prvků vizualizace funkčních bloků, tak i jiných grafických prvků. Těmi mohou být všechny grafické prvky dostupné v rámci *Windows Presentation Foundation* či v jiných kompatibilních knihovnách.

Pro samotnou tvorbu vizualizace je možné využít buď přímo jazyk XAML v jeho textové podobě, nebo některý z existujících nástrojů pro návrh uživatelského prostředí, jako je *Microsoft Visual Studio* či *Microsoft Expression Blend*. V budoucnu by samozřejmě bylo vhodnější vytvořit speciální editor přímo pro návrh vizualizace, jehož výstupem bude rovnou XAML soubor.

Pro snadnou údržbu je vhodné mít definiční soubory všech obrazovek v systému umístěné na jednom místě, tedy v serverové části aplikace. Klient má pak možnost získat seznam všech technologických obrazovek a v případě potřeby si některou z nich stáhnout ze serveru a vykreslit.

Příklad výsledné podoby technologické obrazovky je na obrázku 33, ukázka jejího definičního XAML souboru pak na obrázku 34. Zajímavou částí vizualizace je grafická podoba nádrže, v níž se dle aktuální hodnoty měření animuje výška hladiny. Pro rychlejší orientaci jsou v nádrži pomocí

kosočtverců znázorněny i aktuální výšky řídících hladin jednotlivých pohonů. Ty jsou rovněž animované a zobrazují skutečnou nastavenou hodnotu načtenou z PLC.



Obrázek 33 - Příklad vizualizace technologické obrazovky

Pro usnadnění tvorby technologických obrazovek byl vytvořen i prvek potrubí (*Pipe*), které sám vytváří efekt zakulacených rohů v ohybech. V rámci technologické obrazovky je také možné přímo zobrazit grafický průběh archivovaných hodnot.

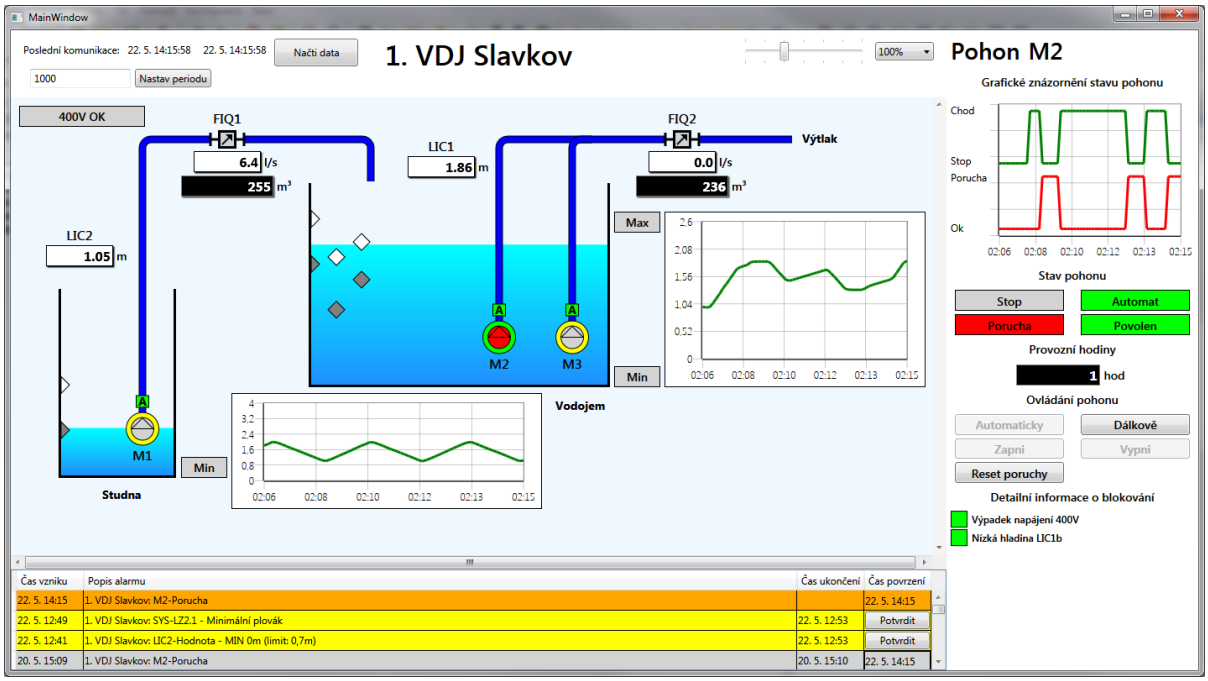
Jednotlivé prvky vložené do technologické obrazovky je třeba propojit s konkrétními instancemi funkčních bloků. K tomu slouží atribut *Address* u jednotlivých prvků vizualizace. Stačí do něj uvést identifikátor (viz kapitola 4.3.4) daného funkčního bloku.

```
<Canvas xmlns:s="clr-namespace:Scada.UIControls;assembly=ScadaUIControls"
        Background="AliceBlue" Width="1100" Height="500"
        s:Page.Name="Vodojem Slavkov" s:Page.Station="1">

    <s:Pipe Color="Blue" Points="50,350 50,0 130,0" Canvas.Left="107" Canvas.Top="50" />

    <s:Drive Address="M1" DriveType="PumpUp" Canvas.Left="117" Canvas.Top="352"/>

    <s:Flow Address="FIQ1" Canvas.Left="203" Canvas.Top="15"/>

    <s:Display Address="LIC1" Format="0.00" Canvas.Left="471" Canvas.Top="48" />
    <s:Tank Address="LIC1" Height="243" Canvas.Left="355" Canvas.Top="102" Width="359">
        <s:Tank.Comparators>
            <s:ComparatorAddress Address="LIC1a"/>
            <s:ComparatorAddress Address="LIC1b"/>
            <s:ComparatorAddress Address="LIC1c"/>
        </s:Tank.Comparators>
    </s:Tank>

    <s:Indicator Address="SYS" Bit="bit0" Canvas.Left="10" Canvas.Top="10"
        ColorOff="LightGray" ColorOn="Red" TextOff="400V OK" TextOn="Výpadek 400V"/>

    <s:Graph Address="LIC1" Canvas.Left="779" Canvas.Top="136" Height="209" Width="311"/>

</Canvas>
```

Obrázek 34 - Příklad způsobu definice vizualizace technologické obrazovky

4.8.3 Celkový vzhled a chování aplikace

Dalším krokem po dokončení realizace dílčích částí, jako je vizualizace funkčních bloků, technologických obrazovek a postranních panelů, je poskládání těchto částí do ucelené podoby ve formě klientské aplikace. Výsledná podoba je obrázku 35.



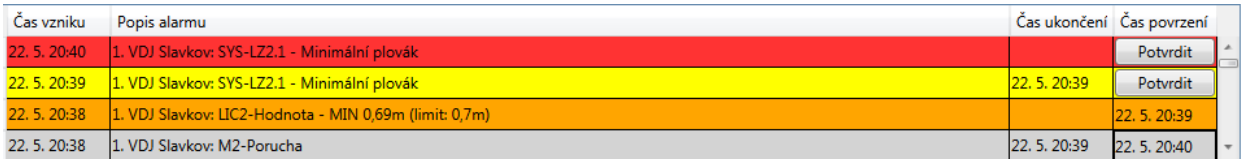
Obrázek 35 - Celkový vzhled klientské části aplikace

Hlavní část tvoří centrální panel, ve kterém je zobrazena vizualizace vybrané technologické obrazovky. Pro zachování vysoké použitelnosti napříč různými HW zařízeními, s různými velikostmi a rozlišeními obrazovek, je tuto část možné libovolně zvětšovat a zmenšovat pomocí ovládacích prvků v pravé horní části obrazovky. V případě úrovně zvětšení, kdy se již celá technologická obrazovka do okna aplikace nevejde, je možné se v technologické obrazovce posouvat, buď pomocí posuvníků nebo pomocí myši či gest v případě dotykového ovládání.

Levou část okna tvoří vysouvací postranní panel. Ten se zobrazí pouze, pokud uživatel klikne na vizualizaci funkčního bloku, která nějaký panel má přiřazen. Po kliknutí mimo funkční blok, se panel opět schová, aby byla maximalizována plocha pro zobrazení vizualizace technologie.

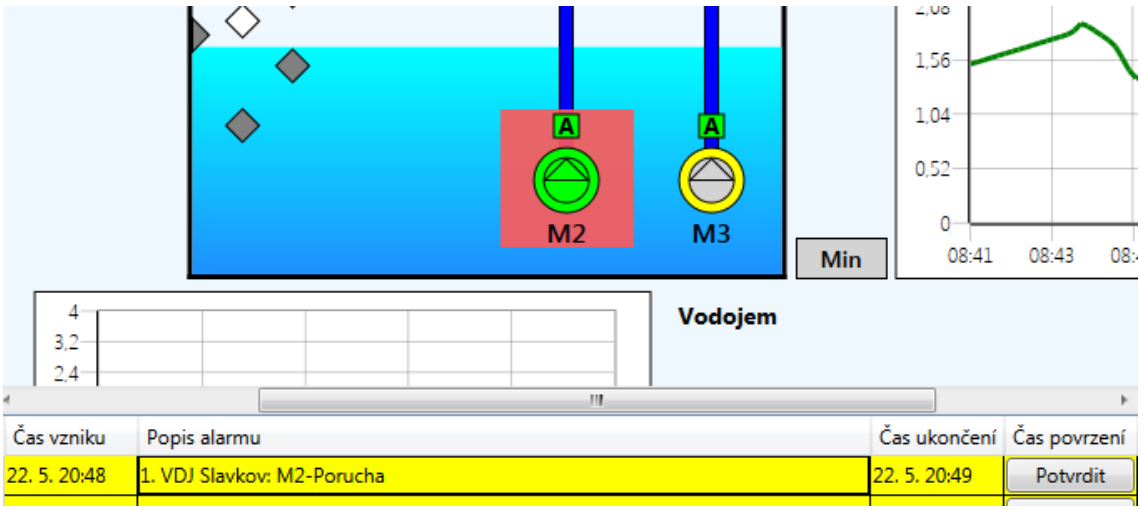
V horní části uživatelského prostředí jsou zobrazeny informace týkající se celé *stanice*, jako je její název, datum a čas posledního úspěšného načtení dat, posledního pokusu o načtení dat a rovněž i tlačítko pro vyvolání načtení dat.

Poslední část hlavního okna obrazovky je tvořena prohlížečkou alarmových stavů. Na obrázku 36 jsou zobrazeny všechny možné stavy alarmů tak, jak jsou indikovány uživateli. Červenou barvou je zobrazen trvající nepotvrzený alarm, oranžovou trvající potvrzený alarm, žlutou ukončený nepotvrzený alarm a nakonec šedou barvou ukončený a potvrzený alarm. Pomocí tlačítka u vybraného alarmu je možné jej potvrdit.



Obrázek 36 - Prohlížečka alarmových stavů

Pro zvýšení uživatelské ergonomie a rychlosti reakce je do systému doplněna funkce pro lokalizaci alarmu. Po kliknutí na vybraný alarm je uživateli *funkční blok*, který jej vyvolal, zvýrazněn přímo v technologické obrazovce, tak jak je to vidět na obrázku 37. V případě, že prvek není v zobrazovaném výřezu obrazovky, je výřez posunut tak, aby byl prvek pokud možno uprostřed okna aplikace.

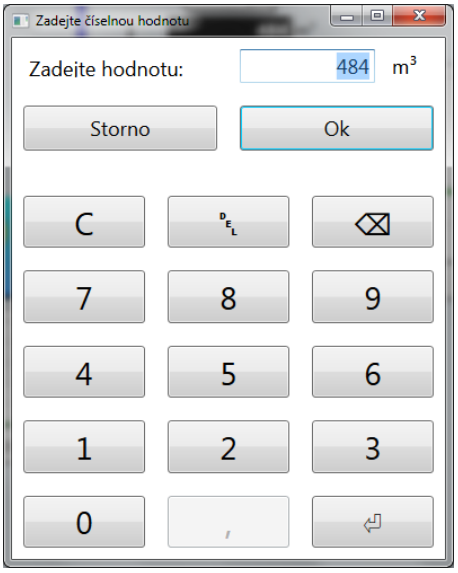


Obrázek 37 - Funkce lokalizace alarmového stavu

4.8.4 Dialog pro zadávání hodnot

Důležitou částí funkce uživatelského prostředí je i možnost zadávání dat do systému. Hodnoty, které lze zadat (zapsat do PLC) jsou ve vizualizaci znázorněny displejem s invertovanými barvami (černé pozadí, bílý text). Po kliknutí na displej je zobrazeno dialogové okno z obrázku 38, které navrženo speciálně pro snadné zadávání hodnot pomocí dotykového ovládání. Je sice možné použít i vestavěnou softwarovou klávesnici, u té je ale nevýhoda v tom, že sama zakrývá velkou část obrazovky. Proto se toto řešení jeví jako vhodnější.

Vytvořený dialog má mimo standardních tlačítek pro zadávání číslic i několik speciálních funkcí, kdy lze danou hodnotu přímo vynulovat, smazat první či poslední číslici a potvrdit zadání. Dialog se navíc může lišit dle toho, zda je zadávaná hodnota celočíselná či jde o hodnotu s plovoucí desetinnou čárkou.



Obrázek 38 - Dialog pro zadávání hodnot

4.9 Demonstrační aplikace – simulace vodojemu

Při vývoji jakékoli SCADA aplikace není v praxi možné ji testovat přímo na reálném zařízení. Pokud navíc daná SCADA aplikace má umožnit uživateli si vyzkoušet co největší množství funkcí, je třeba vytvořit nějakou formu simulace reálného chování technologických zařízení.

Lze to řešit dvěma způsoby. Buď je simulační režim přímo součástí SCADA aplikace, ale pak nelze funkce SCADA systému ověřit komplexně (včetně *driverů* pro komunikaci s PLC), nebo je možné chování simulovat přímo v PLC. Nevýhodou je však nutnost nějaké PLC přímo vlastnit, což nemusí být zrovna nejlevnější.

Já jsem při tvorbě zvolil druhou možnost, protože jsem využil nabídky firmy GDF spol. s r.o., která mi poskytla jak PLC pro testování, tak i SW pro jeho programování. Byla tedy vytvořena simulace chování vodojemu s dočerpáváním vody ze studně (viz obrázek 34). Program pro PLC *Schneider Modicon M340* (viz obrázek 2) byl vytvořen v software *Unity Pro S* .

Výhodou použití tohoto PLC a software je možnost spuštění programu PLC v simulačním režimu i na běžném počítači – odpadá tak nutnost mít v době prezentace funkcí SCADA systému připojené fyzické PLC. Nevýhodou je pak poměrně vysoká cena a tím pádem i malá dostupnost vývojového nástroje *Unity Pro S*.

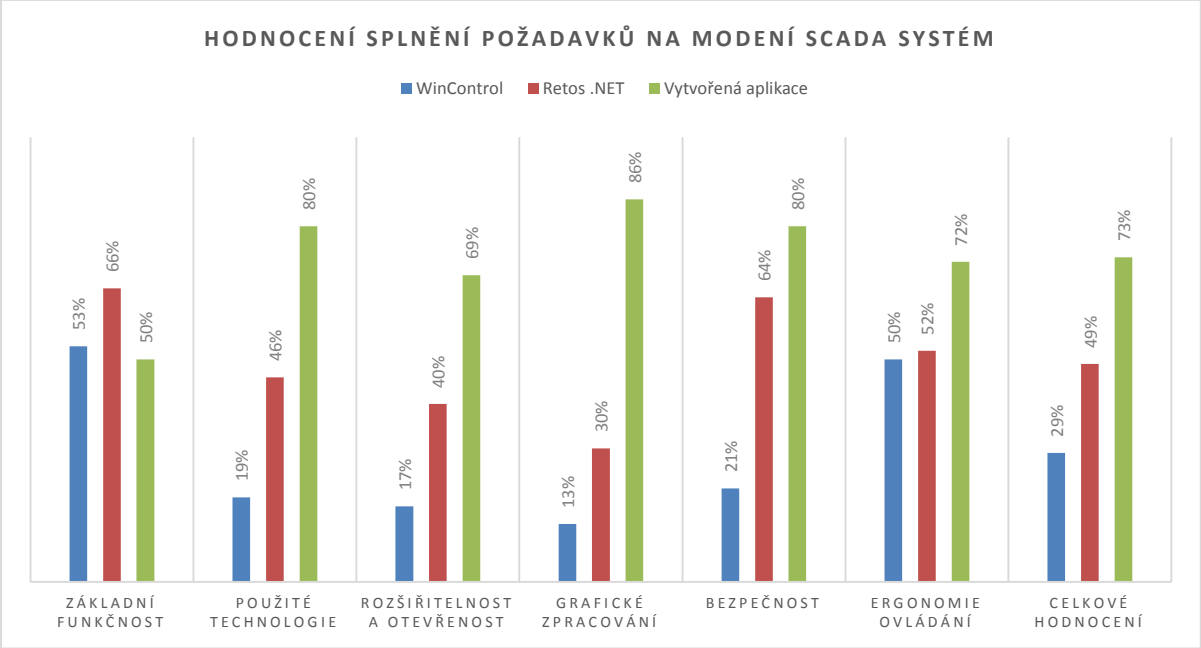
Data demonstrační aplikace jsou, buď z reálného PLC nebo i ze softwarově emulovaného zařízení, do SCADA systému čtena a zapisována pomocí protokolu *Modbus*. Zdrojové kódy vytvořené simulace jsou součástí DVD, které je přílohou této práce.

4.10 Vyhodnocení a ověření výsledků práce

Na základě požadavků definovaných v kapitole 3.3.1 bylo provedeno hodnocení existujících aplikací *WinControl*, *Retos .NET* a v rámci této práce vytvořené aplikace z hlediska splnění požadované funkcionality a vlastností. Výsledky hodnocení jsou v tabulce 2, graficky pak na obrázku 39.

Na první pohled je zřejmé, že ve většině kategorií požadavků je vytvořená aplikace hodnocena mnohem lépe než ty stávající. Je to ovšem logické, neboť při definici požadavků jsem se zaměřil na funkcionalitu a vlastnosti, se kterými mají stávající systémy potíže. Nelze tak říci, že navržený a vytvořený systém je obecně lepší, než ty stávající – je lepší pouze v uvedených kritériích.

Největší problémy má vytvořená aplikace se základní funkčností. Podporuje totiž pouze jeden komunikační protokol, a i když lze doplnit další, tak pro nasazení do reálného provozu je to nedostačující. Aplikace také postrádá prostředky pro komplexnější analýzu archivovaných dat, a samostatný editor vizualizace funkčních bloků a technologických obrazovek. Bylo by také třeba doplnit možnost si uživatelsky (bez programování) vytvářet vlastní funkční bloky. S touto možností je sice počítáno (je pro ni v aplikaci podpora), ale není do detailu implementována.



Obrázek 39 - Graf hodnocení splnění požadavků na moderní SCADA systém

Pokud jde o hodnocení jednotlivých kategorií, tak si vytvořená aplikace vedla dobře v *použitých technologiích*, primárně díky cílení na více typů hardwarových zařízení a způsobů ovládání, a také díky použití pokročilých možností, které poskytuje platforma *Microsoft .NET*. Dobré *rozšiřitelnosti* zase pomohla alespoň základní podpora tvorby vlastních komunikačních driverů, funkčních bloků a jejich vizualizace.

Do vysokého hodnocení v oblasti grafického zpracování promluvilo použití vektorové grafiky společně s akcelerovaným vykreslováním prostřednictvím *Windows Presentation Foundation* a jazyka

XAML. Vysoké hodnocení v oblasti bezpečnosti je zase dáno využitím platformy *Microsoft .NET* a *Windows Communication Foundation*. Je tak vidět, že použitím moderních softwarových prostředků lze vyřešit značnou část požadavků na funkcionalitu, otevřenost a bezpečnost.

Požadavek na systém	Váha	Hodnocení		
		WinControl	Retos .NET	Vytvořená aplikace
Základní funkčnost	**15%**	**53%**	**66%**	**50%**
Způsoby přenosu dat	5%	50%	90%	20%
Archivace dat (použité databáze – otevřené, proprietární)	5%	30%	70%	50%
Podpora alarmů, jejich provázání se zbytkem systému	5%	80%	40%	80%
Použité technologie	**20%**	**19%**	**46%**	**80%**
Podporované HW platformy	5%	20%	50%	80%
Podpora alternativního (dotykového) ovládání	3%	10%	30%	90%
Podporované SW platformy	2%	30%	40%	70%
Podpora síťových protokolů	2%	50%	40%	90%
Podpora „tenkého klienta“ (většina dat by měla být v systému a ne lokálně)	3%	20%	70%	90%
Způsob tvorby vizualizace a způsob jejího uložení (otevřený nebo proprietární formát)	2%	20%	60%	80%
Podpora vlastních datových struktur	3%	0%	30%	60%
Rozšiřitelnost a otevřenost	**20%**	**17%**	**40%**	**69%**
Možnosti přizpůsobení – funkční	5%	10%	20%	60%
Možnosti přizpůsobení – grafické	5%	30%	60%	90%
Možnost rozšíření funkcionality, propojení na systémy třetích stran	5%	10%	50%	70%
Podpora vlastních (nových) typů PLC	3%	20%	30%	60%
Dokumentace, otevřený kód, dostupnost rozhraní pro využití jinými SW systémy	2%	10%	30%	50%
Grafické zpracování	**15%**	**13%**	**30%**	**86%**
Způsob vykreslování s hlediska HW akcelerace	5%	10%	20%	90%
Podpora vektorové grafiky	5%	10%	20%	90%
Možnost tvorby vlastních grafických šablon	5%	20%	50%	80%
Bezpečnost	**12%**	**21%**	**64%**	**80%**
Podpora šifrovaného přenosu	3%	20%	30%	80%
Zabezpečení přístupu k systému (autentizace a autorizace)	5%	30%	80%	80%
Aktuální SW platforma (neaktuální knihovny mohou představovat bezpečnostní problém)	4%	10%	70%	80%
Ergonomie ovládání	**18%**	**50%**	**52%**	**72%**
Dostatečná velikost ovládacích prvků, grafické názornost	5%	30%	50%	70%
Jednoduchost a intuitivnost ovládání	8%	50%	60%	80%
Prvky zvyšující přehlednost a rychlost reakce	5%	70%	40%	60%
Celkové hodnocení		**29%**	**49%**	**73%**

Tabulka 2 - Výsledky hodnocení splnění požadavků na moderní SCADA systém

Druhá část hodnocení vytvořené aplikace spočívala v hodnocení samotnými uživateli. Pro tento účel vznikl dotazník, který je přílohou této práce. Je zaměřen na získání základních informací o zkušenostech se SCADA systémy a hodnocení jednotlivých funkcí navržené aplikace. Poskytuje i možnost pro zpětnou vazbu, kdy lze od uživatelů zjistit, které funkcionalita jim chybí, nebo by si ji představovali odlišně.

Vytvořená aplikace byla předvedena, Ing. Josefu Václavíkovy, který je vedoucím centrálního dispečinku ve firmě Vodovody a kanalizace Vsetín, a panu Janu Schweidlerovi, který je vedoucím dispečerského pracoviště Vodovodů a kanalizací Vyškov. Poté jim byl předložen připravený dotazník. Oba vyplněné dotazníky jsou v příloze této práce.

Ačkoliv jsou tato hodnocení spíše subjektivního charakteru a ze dvou vyplněných dotazníků nelze vyvádět jasné závěry, je z odpovědí dotyčných osob zřejmé, že základní myšlenky a jejich realizace, se alespoň těmto dvěma potencionálním uživatelům vesměs líbí. Z odpovědí na dotazy, jsou také zřejmé některé další funkce, které by ocenili, a jistě by bylo vhodné s nimi počítat, pokud se v této práci bude pokračovat.

5 Závěr

Cílem této práce bylo seznámit se s problematikou uživatelských rozhraní pro řízení technologií a na základě získaných znalostí uživatelské rozhraní navrhnout a implementovat. Cíl byl splněn a v rámci této diplomové práce tak byly nastudovány základní pojmy, řešení a postupy z oblasti uživatelských prostředí pro řízení technologických procesů, byly popsány modelové případy použití, popsány některé z nejčastěji používaných komunikačních protokolů, ať již určených pro komunikaci přímo s programovatelnými logickými automaty, nebo určenými pro komunikaci jednotlivých částí systému. Byly popsány vybrané existující systémy používané v této oblasti a bylo u nich provedeno zhodnocení silných a slabých stránek. Na základě hodnocení byly definovány základní i některé detailní požadavky, které posloužili jako základ pro návrh vlastního řešení uživatelského rozhraní.

Z návrhu pak vzešla samotná realizace ukázkové aplikace, která je sama o sobě použitelná pro dohled nad jednoduchými řídicími systémy. Aplikace se skládá ze dvou částí, první (serverová) se stará o načítání dat z technologického procesu, jejich zpracování, archivaci a vyhodnocení alarmových stavů. Druhá (klientská) získaná data vizualizuje a slouží tak jako rozhraní mezi systémem a uživatelem. Jednotlivé části aplikace mohou běžet na různých zařízeních, protože vzájemně komunikují za pomoci standardizovaného komunikačního rozhraní. Výsledná aplikace pak byla, i za pomoci hodnocení samotných uživatelů, porovnána s existujícími řešeními. Je tedy možné konstatovat, že všechny definované cíle této diplomové práce byly splněny.

Při implementaci ukázkové aplikace vniklo přibližně 10100 řádků kódu, v 76 třídách a zhruba 20 definičních souborů prvků uživatelského rozhraní v jazyce XAML. Pro uložení dat byly navrženy 3 databáze s cekem 8 tabulkami a 64 atributy. Celé aplikace pak byla rozdělena do 4 DLL knihoven a 2 spustitelných souborů.

V rámci studia a samotné realizace jsem se jako autor seznámil s velkým množstvím technologií, komunikačních protokolů a standardů používaných v prostředí průmyslové automatizace technologických procesů. Dále jsem se seznámil s pokročilými metodami návrhu a především realizace vlastních komunikačních rozhraní, databázových úložišť a uživatelských rozhraní.

Vytvořená aplikace má pouze základní potřebnou funkcionalitu, která je doplněna o ukázku nových a moderních způsobů řešení některých funkcí, které existující systémy postrádají, nebo je neřeší úplně ideálně. Aplikace však zatím není úplně použitelná v praxi, především díky chybějící širší podpoře protokolů pro komunikaci s programovatelnými řídícími automaty, absenci pokročilejšího editoru pro tvorbu vizualizace a chybějícím doplňkovým modulům, především pro práci s daty. Právě na tyto oblasti by se měl zaměřit ten, kdo se rozhodne na výsledky této diplomové práce navázat, ať formou přídavných modulů, nebo přepracováním a doplněním těch existujících.

Citovaná literatura

1. Gaushell, Denis J. und Darlington, Henry T. *Supervisory Control and Data Acquisition. PROCEEDINGS OF THE IEEE.* 1987, Bd. 75.

2. Bailey, David und Wright, Edwin. *Practical SCADA for Industry.* s.l. : Newnes, 2003. 0080473903.

3. Vojtek, Robert. *Centrální dispečerské pracoviště VaK Vsetín - Zadávací dokumentace.* Mostkov : GDF spol. s r.o., 2013.

4. Gordon R. Clarke, Deon Reynders, Edwin Wright (B.Sc.). *Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems.* s.l. : Newnes, 2004.

5. Bolton, William. *Programmable Logic Controllers.* s.l. : Newnes, 2009. 0080961851.

6. Wolfgang A. Halang, Krzysztof M. Sacha. *Real-time Systems: Implementation of Industrial Computerised Process Automation.* s.l. : World Scientific, 1992. 9810210647.

7. Acromag Incorporated. *Introduction To ProfiBus DP.* [Online] 2002. http://www.diit.unict.it/users/scava/dispense/II_270/ProfibusIntroduction.pdf.

8. Modbus Organization. *MODBUS APPLICATION PROTOCOL SPECIFICATION. The Modbus Organization.* [Online] 26. Duben 2012. http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf.

9. PROFIBUS Nutzerorganisation e.V. *PROFIBUS System Description - Technology and Application.* Karlsruhe : PROFIBUS Nutzerorganisation e.V., 2010.

10. OPC Foundation. *OPC Data Access Automation Specification.* [Online] 3. Únor 1999. http://www-ad.fnal.gov/controls/opc/OPC_DA_Auto_2.02_Specification.pdf.

11. GDF spol. s r.o. *WinControl - Obsluha dispečinku.* 2013.

12. Sharp, John. *Microsoft Visual C# 2010 Step by Step.* s.l. : Microsoft Press, 2010.

13. QLine, a.s. *Retos.NET - Systémová dokumentace.* 2011.

14. Vojtek, Robert. *Centrální dispečerský systém CHEVAK Cheb - projektová dokumentace.* Mostkov : GDF spol. s r.o., 2012.

15. QLine, a.s. *Retos.NET - Průvodce vizualizací WMag.* 2013.

16. Gudgin, Martin, et al. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). W3C.* [Online] 27. Duben 2007. http://www.w3.org/TR/soap12/.

17. Microsoft Corporation. *Windows Communication Foundation Architecture Overview. Microsoft Developer Network.* [Online] Březen 2006. https://msdn.microsoft.com/en-us/library/aa480210.aspx.

18. Assunti, Simone. *Free .NET Modbus Library.* [Online] Prosinec 2013. https://code.google.com/p/free-dotnet-modbus/.

19. Christensen, Eric, et al. *Web Services Description Language (WSDL) 1.1. W3C.* [Online] 15. Březen 2001. http://www.w3.org/TR/wsdl.

20. Rusina, Alexandra. *Dynamic in C# 4.0: Creating Wrappers with DynamicObject. MSDN Blog.* [Online] 19. Říjen 2009. http://blogs.msdn.com/b/csharpfaq/archive/2009/10/19/dynamic-in-c-4-0-creating-wrappers-with-dynamicobject.aspx.
21. Český normalizační institut. *ČSN ISO 14617 - Grafické značky pro schémata.* Praha : Český normalizační institut, 2004.
22. Schneider Electric. *Unity Pro - IEC Programming Software for Modicon PACs.* [Online] http://www.schneider-electric.com/products/ww/en/3900-pac-plc-other-controllers/3950-pacs/548-unity-pro/.

Seznam použitých obrázků

Seznam příloh

- Šablona nevyplněného dotazníku pro uživatelské hodnocení vytvořené aplikace
- Vyplněné dotazníky pro hodnocení
- DVD obsahující:
    - Zdrojové texty této práce v elektronické podobě
    - Programovou dokumentaci
    - Zdrojové kódy vytvořené aplikace
    - Zdrojové soubory Unity Pro S demonstrační aplikace pro PLC
    - Připravené demo aplikace v binární podobě včetně konfigurace
    - Krátké komentované video z ukázkou aplikace
    - Instalační soubory s knihovnami potřebnými pro spuštění

Dotazník pro uživatelské hodnocení SCADA systému

Jméno a příjmení:

Firma:

Pracovní pozice:

Datum:

Stávající systém a obecné dotazy

1. V jakém oboru pracujete a co je náplní vaší práce?
2. Které SCADA systémy znáte a momentálně používáte?
3. Máte k aktuálně používanému systému nějaké výhrady? Jaké?
4. Umožňuje váš SCADA systém vzdálený přístup z různých typů zařízení (chytrý telefon, tablet)? Ocenily byste tuto funkcionalitu?

Navrhovaný SCADA systém

Za pomoci stupnice 1 až 5 (1 – nejlepší) proveďte hodnocení níže uvedených částí systému:

1. Způsob definice datová struktury.
2. Způsob propojení a adresace PLC.
3. Možnost a způsob definice vlastních datových struktur – funkčních bloků.
4. Možnost a způsob definice vlastních grafických prvků.
5. Použití stejného rozhraní na více typech zařízení (PC, tablet).
6. Grafické podoba uživatelského rozhraní:
    a. Celkový vzhled.
    b. Vzhled aktivních prvků (měření, pohony).
    c. Vzhled a funkce postranního uživatelského panelu.
    d. Způsob zobrazení alarmových stavů.
    e. Způsob zobrazení dat (grafů).
7. Ergonomie uživatelského rozhraní.
    a. Způsob ovládání.
    b. Velikost ovládacích prvků.
    c. Rozložení ovládacích prvků.
    d. Způsob zobrazení detailů a ovládání pomocí postranního panelu.
    e. Funkce zvýraznění prvku ve vizualizaci po označení alarmového stavu.
    f. Způsob zadávání číselných hodnot.
    g. Celkový způsob ovládání na PC.
    h. Celkový způsob ovládání na tabletu.

Doplňující slovní hodnocení

1. Jak se vám celkově navržený systém líbí?
2. Máte nějaké výhrady či poznatky ke grafickému zpracování?
3. Máte nějaké výhrady či poznatky ke způsobu ovládání a ergonomii?
4. Máte nějaké jiné výhrady?
5. Jaké je vaše celkové hodnocení systému?

Návrhy pro doplnění a vylepšení systému

1. Máte nějaké návrhy pro doplnění funkcí či jakékoli další nápady, kterými by se dal systém vylepšit?

Dotazník pro uživatelské hodnocení SCADA systému

Jméno a příjmení: Jan Schweidler
Firma: Vodovody a kanalizace Vyškov, a.s.
Pracovní pozice: vedoucí dispečerského pracoviště
Datum: 20. května 2015

Stávající systém a obecné dotazy

1. V jakém oboru pracujete a co je náplní vaší práce?

Pracuji na vodohospodářském dispečinku VaK Vyškov. Mou povinností je zajištění jeho fungování. Administrativně zajišťuji správu dat a vypracování statistik. Zajišťuji dodávku vody do vodojemů a trubní sítě. Dohlížím a zasahuji do řízení dispečinku. Vyhledávám ztráty v trubní síti, v případě poruch operativně zajišťuji dodávku z jiného zdroje. Objednávám servis, náhradní díly a organizačně zajišťuji řešení poruch a předávám informace o poruchách jednotlivým uživatelům. Snažím se optimalizovat a zvyšovat spolehlivost dodávky vody, dbám na úspory provozních nákladů.

2. Které SCADA systémy znáte a momentálně používáte?

Používáme SCADA GDF, spol.s r.o.-WinControl, QLine a.s.-Retos.
Znám SCADA Schneider Elektric CZ, s.r.o., Moravské přístroje, a.s., VaE Controls, s.r.o.

3. Máte k aktuálně používanému systému nějaké výhrady? Jaké?

Postrádám možnost editace obrazovek a technologických schémat uživatelem (u SCADA GDF).

4. Umožňuje váš SCADA systém vzdálený přístup z různých typů zařízení (chytrý telefon, tablet)? Ocenily byste tuto funkcionalitu?

K používaným SCADA systémům mám přístup pomocí notebooku s modemem nebo přes telefon. U systému GDF mohu spustit nainstalovanou aplikaci, u QLine je umožněn přístup přes webový prohlížeč. Mohu se též připojovat přes aplikaci vzdálená plocha. Připojení přes tablet nebo chytrý telefon považuji za standard.

Navrhovaný SCADA systém

Za pomoci stupnice 1 až 5 (1 – nejlepší) proveďte hodnocení níže uvedených částí systému:

1.	Způsob definice datová struktury.	2
2.	Způsob propojení a adresace PLC.	1
3.	Možnost a způsob definice vlastních datových struktur – funkčních bloků.	1
4.	Možnost a způsob definice vlastních grafických prvků.	1
5.	Použití stejného rozhraní na více typech zařízení (PC, tablet).	1
6.	Grafické podoba uživatelského rozhraní:	
	a. Celkový vzhled.	1
	b. Vzhled aktivních prvků (měření, pohony).	1
	c. Vzhled a funkce postranního uživatelského panelu.	1
	d. Způsob zobrazení alarmových stavů.	2
	e. Způsob zobrazení dat (grafů).	1

7. Ergonomie uživatelského rozhraní.
   a. Způsob ovládání. 1
   b. Velikost ovládacích prvků. 1
   c. Rozložení ovládacích prvků. 1
   d. Způsob zobrazení detailů a ovládání pomocí postranního panelu. 1
   e. Funkce zvýraznění prvku ve vizualizaci po označení alarmového stavu. 1
   f. Způsob zadávání číselných hodnot. 1
   g. Celkový způsob ovládání na PC. 1
   h. Celkový způsob ovládání na tabletu. 1

Doplňující slovní hodnocení

1. Jak se vám celkově navržený systém líbí?

Může se rovnat s námi používaným systémem.

2. Máte nějaké výhrady či poznatky ke grafickému zpracování?

Grafika neobsahuje zbytečné pohyblivé značky motorů, tekoucí vody... Obrazovka je tak přehledná, nedochází k odvádění pozornosti.

3. Máte nějaké výhrady či poznatky ke způsobu ovládání a ergonomii?

Nemám. Ovládání je intuitivní a přátelské.

4. Máte nějaké jiné výhrady?

Nemám.

5. Jaké je vaše celkové hodnocení systému?

Hodnotím vysokou známkou: 1.

Návrhy pro doplnění a vylepšení systému

1. Máte nějaké návrhy pro doplnění funkcí či jakékoli další nápady, kterými by se dal systém vylepšit?

Osobně bych přivítal viditelné mezní stavy v zobrazovaných grafech.

VODOVODY A KANALIZACE Vyškov, a.s. 17
682 01 Vyškov, Brněnská 410/13

Jan Schweidler
vedoucí vodohospodářského dispečinku
Vodovody a kanalizace Vyškov, a.s.
682 01 Vyškov, Brněnská 410/13

tel: +420 517 324 932
mobil: +420 721 656 660
e-mail: schweidler@vakvyskov.cz
web: www.vakvyskov.cz

Dotazník pro uživatelské hodnocení SCADA systému

Jméno a příjmení: **Josef Václavík**

Firma: **Vodovody a kanalizace Vsetín, a.s.**

Pracovní pozice: **technik Centrální dispečink**

Datum: **18. 5. 2015**

Stávající systém a obecné dotazy

1. V jakém oboru pracujete a co je náplní vaší práce?
   - **vodárenský dispečink**
   - **řízení a distribuce dodávky pitné vody, správa ASŘTP a telemetrických přenosů**
2. Které SCADA systémy znáte a momentálně používáte?
   - **WinControl, Promotic, RetosNT**
3. Máte k aktuálně používanému systému nějaké výhrady? Jaké?
   - **WinControl – uzavřený systém, bez možnosti úprav**
   - **Promotic – grafy prvků umístěny mimo obrazovku objektu, nepřehledné grafika**
   - **RetosNT – načítání dat a zasílání blokovací podmínek neprobíhá souběžně; při sestavování statistik je nepřehledný výběr ze všech prvků daného objektu**
4. Umožňuje váš SCADA systém vzdálený přístup z různých typů zařízení (chytrý telefon, tablet)? Ocenily byste tuto funkcionalitu?
   - **Ano, SCADA RetosNT toto umožňuje, ovšem grafika se nepřizpůsobí rozlišení daného zařízení**
   - **Ano, tato funkcionalita by v dnešní době měla být již standardem**

Navrhovaný SCADA systém

Za pomoci stupnice 1 až 5 (1 – nejlepší) proveďte hodnocení níže uvedených částí systému:

1. Způsob definice datová struktury. (**2**)
2. Způsob propojení a adresace PLC. (**2**)
3. Možnost a způsob definice vlastních datových struktur – funkčních bloků. (**2**)
4. Možnost a způsob definice vlastních grafických prvků. (**2**)
5. Použití stejného rozhraní na více typech zařízení (PC, tablet). (**2**)
6. Grafické podoba uživatelského rozhraní:
   a. Celkový vzhled. (**1**)
   b. Vzhled aktivních prvků (měření, pohony). (**1**)
   c. Vzhled a funkce postranního uživatelského panelu. (**2**)
   d. Způsob zobrazení alarmových stavů. (**2**)
   e. Způsob zobrazení dat (grafů). (**3**)

7. Ergonomie uživatelského rozhraní.
    a. Způsob ovládání. (**1**)
    b. Velikost ovládacích prvků. (**1**)
    c. Rozložení ovládacích prvků. (**1**)
    d. Způsob zobrazení detailů a ovládání pomocí postranního panelu. (**1**)
    e. Funkce zvýraznění prvku ve vizualizaci po označení alarmového stavu. (**1**)
    f. Způsob zadávání číselných hodnot. (**1**)
    g. Celkový způsob ovládání na PC. (**1**)
    h. Celkový způsob ovládání na tabletu. (**1**)

Doplňující slovní hodnocení

1. Jak se vám celkově navržený systém líbí?

   Grafika systému je přehledná a ovládání intuitivní. Líbí se mi praktické rozmístění jednotlivých komponent systému (prvky grafiky, grafy a alarmy). Systém se přizpůsobuje velikost používaného zařízení (PC, NTB, tablet). Způsob nastavování parametrů a mezí veličin je snadný.

2. Máte nějaké výhrady či poznatky ke grafickému zpracování?

   Grafické zpracování je na vysoké úrovni. Zvolené barevné schéma je příjemné a netahá oči. Kladně hodnotím grafické prvky vytvořené na základě technických značek, které napomáhají rychlému se zorientování se i ve složitých obrazovkách.
   Absence „pouťové" grafiky, točivých, blikajících a nezdařených 3d napodobenin strojů je vítána.

3. Máte nějaké výhrady či poznatky ke způsobu ovládání a ergonomii?

   Ovládání systému je intuitivní a vhodně integrované do přehledných grafických prvků. Velikost a umístění ovládacích prvků je zvolena, tak že neruší celkový vzhled a přesto jsou na první pohled rozpoznatelné.

4. Máte nějaké jiné výhrady?

   Ovládání systému z tabletu není tak komfortní jako z NTB nebo PC.

5. Jaké je vaše celkové hodnocení systému?

   Systém je navržen s důrazem na přehlednost a jednoduchost ovládání. Použité programovací prostředky umožňují rychlý návrh a sestavení systému. Důležitá je otevřenost systému, kdy má uživatel možnost editace či doplnění nových prvků a vazeb mezi mimy.

Návrhy pro doplnění a vylepšení systému

1. Máte nějaké návrhy pro doplnění funkcí či jakékoli další nápady, kterými by se dal systém vylepšit?
   - **vazba mezi alarmem a prvkem, kdy po kliknutí na alarm přejde kurzor na prvek, kterého se alarm týká**
   - **možnost zápisu poznámky k prvku, či objektu**

Vodovody a kanalizace Vsetín, a.s.
dispečink Vsetín
755 01 VSETÍN, Jasenická 1106