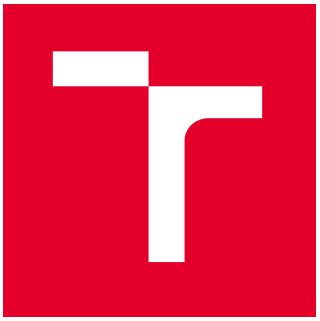


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

MĚŘENÍ SPOTŘEBY ENERGIE NA BÁZI OPEN SOURCE TECHNOLOGIÍ V AUTOMATIZACI

ENERGY METERING BASED ON OPEN SOURCE TECHNOLOGIES IN AUTOMATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Adam Szymanik**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Petr Fiedler, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

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

Ústav automatizace a měřicí techniky

Student: Bc. Adam Szymanik ***ID:*** 195437

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Měření spotřeby energie na bázi open source technologií v automatizaci

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je provést rešerši dostupných open source technologií a automatizačních platforem pro účely měření spotřeby (elektřina, voda, plyn, ...) a vyvinout demonstrační řešení na bázi zvolené platformy.

1. Srovnejte vybrané platforemy.
2. Zvolte vhodnou platformu nebo platfomy pro "off-line" automatizaci měření spotřeby.
3. Realizujte automatizační úlohu demonstrující možnosti zvolené platformy.
4. Zhodnoťte dosažené výsledky, rozeberte silné a slabé stránky zvoleného řešení.

DOPORUČENÁ LITERATURA:

[1] Dokumentace ke zvoleným platformám na základě doporučení zadávající firmy.

[2] https://csa-iot.org/all-solutions/matter/

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 17.5.2023

Vedoucí práce: doc. Ing. Petr Fiedler, Ph.D.

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

ABSTRAKT

Tato diplomová práce se zabývá popisem různých open source platforem, které můžou sloužit jako alternativy ke klasickým PLC a následným výběrem nejvhodnější platformy pro realizaci automatizační úlohy. Dále je v práci popsána tvorba úlohy ve vývojovém prostřední Codesys, která má za úkol měření spotřeby elektrické energie, výpočet predikované čtvrthodinové spotřeby a následnou regulaci. Taktéž je zde popsán postup instalace programu QC4 a konfigurace vytvořené vizualizace, která umožňuje zobrazení měřených a počítaných hodnot, zobrazení průběhu vývoje dat, přehledu událostí a alarmů a taky manuální odpojení spotřebičů. V závěru práce jsou shrnuty dosažené výsledky práce, nedostatky a výhody řešení.

KLÍČOVÁ SLOVA

Open-source, Linux, PFC200, Wago, Unipi Patron, Revolution Pi, Řídicí systém, Codesys, Měření spotřeby, Quick Control 4, Modbus TCP

ABSTRACT

This master thesis deals with description of open-source automation platforms, which can be used as alternatives to normal PLC systems and selectino of the most suitable platform for automation task. This thesis also describes creation of task for energy consuption measurement, prediction and control in Codesys developement system. Installation of QC4 software and created configuration is also described. The vizualization displays measured and calculated values, event and alarm log and allows for manual disconnection of appliances. The results, shortcomming and benefits are summarized in conclusion.

KEYWORDS

Open-source, Linux, PFC200, Wago, Unipi Patron, Revolution Pi, Control system, Codesys, Consuption measurement, Quick Control 4, Modbus TCP

SZYMANIK, Adam. *Měření spotřeby energie na bázi open source technologií v automatizaci*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2023, 75 s. Diplomová práce. Vedoucí práce: doc. Ing. Petr Fiedler, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Adam Szymanik
VUT ID autora:	195437
Typ práce:	Diplomová práce
Akademický rok:	2022/23
Téma závěrečné práce:	Měření spotřeby energie na bázi open source technologií v automatizaci

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

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

Brno 16.5.2023 .....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Petru Fiedlerovi, Ph.D a taky panu Ing. Rostislavu Fitzovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Tato diplomová práce se zabývá popisem a následným porovnáváním různých open source automatizačních platforem. Tyto platformy můžou poskytovat velice atraktivní alternativu ke klasickým PLC. Pořizovací cena takových PLC je často vysoká a programují se pomocí proprietárního softwaru, ke kterému je potřeba zakoupit licenci a tím se dále zvyšují náklady. Obzvlášť v dnešní době, kdy prakticky celý svět bojuje s vysokou inflací je nižší pořizovací cena pro zákazníka lákavá. Open source platformy umožňují větší flexibilitu použitého softwaru a taky aplikace těchto systémů, ať už se jedná o použití v průmyslu tak třeba i pro domácí automatizaci.

V první kapitole jsou popsány řídicí systémy společnosti Wago. Jedná se o modulární systém, kterého základem je procesorový modul, ke kterému se připojuje rozšiřující moduly s digitálními a analogovými vstupy a výstupy podle potřeb konkrétní aplikace. Dále lze připojit moduly pro komunikaci protokolem M-Bus, moduly pro měření třífázového výkonu, moduly s sériovým rozhraním RS-232 a další.

Další kapitola se věnuje řídicím systémům společnosti Unipi, přesněji produktové řadě Patron. Jedná se o kompaktní řídicí systémy s operačním systémem Linux založené na procesoru ARM-Cortex A53.

Poslední popsanou platformou je Revolution Pi společnosti Kunbus. Základem Revolution Pi je výpočetní modul Rasberry Pi. Tento základní řídicí modul je možné rozšířit o vstupní a výstupní moduly pomocí PiBridge konektoru a Connect konektorem o různé komunikační moduly.

Kapitola 4 se zabývá porovnáním výše zmíněných platforem podle požadavků realizované aplikace, kterou bude část systému Měření a Regulace pro halu odstavů tramvají v Olomouci, komunikačních možností, nákladového hodnocení a vývojového prostředí pro danou platformu.

Praktická část práce (kapitola 5) se zabývá realizací části systému Měření a Regulace. Realizovaná část má za úkol měření spotřeby elektrické energie a predikci čtvrthodinové spotřeby. Podle predikované spotřeby a nastavených maximálních hodnot budou následně odpojovány spotřebiče podle stanovených priorit. Součástí aplikace bude i vizualizace, která bude zobrazovat měřené a počítané hodnoty a taky umožňovat manuální odpojení spotřebičů. Dále bude systém integrován do již existujícího systému MaR, s kterým bude komunikovat protokolem Modbus TCP.

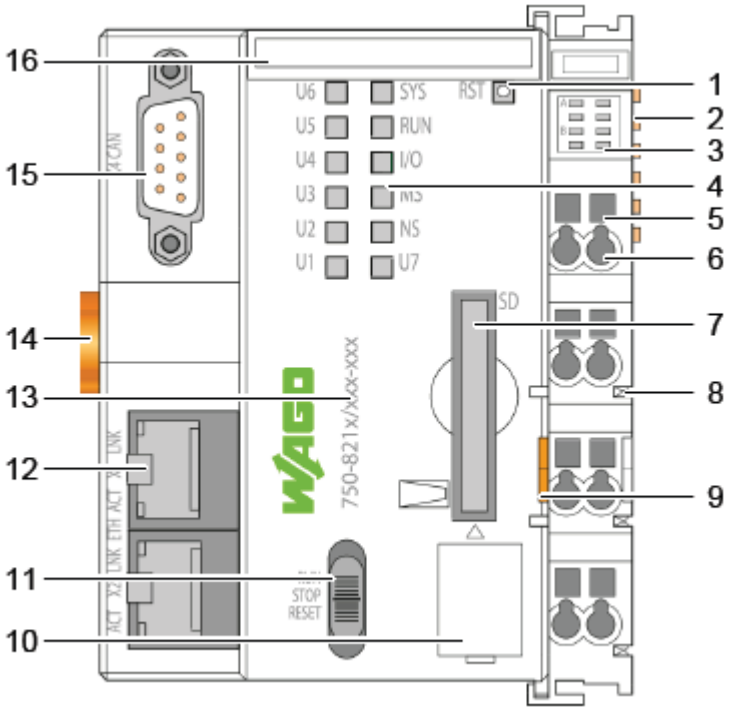
Ze zadání praktické částí vyplývají na řídicí systém určité požadavky. Jedná se o konkrétní počet digitálních a analogových vstupů a výstupů, schopnost systému komunikovat protokolem Modbus TCP. Podpora protokolu M-Bus není vyžadována, ale může být výhodou pro daný systém. Bez podpory protokolu M-Bus bude měření probíhat pomocí impulsních výstupů elektroměru.

Poslední kapitola pojednává o dosažených výsledcích práce, problémech zjištěných při realizaci úlohy a testovacím provozu. Dále jsou zmíněny nedostatky a výhody zvoleného řešení.

1 PFC100 a PFC200

Společnost Wago nabízí řadu řídicích systému. Pro malé aplikace nabízí společnost Wago řídicí systém Compact 100 s již integrovanými vstupy a výstupy. Procesorové moduly PFC100 a PFC200 najdou širokou škálu uplatnění díky modulárnímu systému vstupů/výstupů. Právě na tyto procesorové moduly je zaměřená tato kapitola. Obě řady nabízí Wago i ve verzi do extrémního prostředí. Tyto modely mají vetší odolnost proti vibracím, nárazům, přepětí a zvýšený pracovní rozsah teplot.[1]

Jádrem systému PFC100 a PFC200 je procesor Cortex A8 s frekvencí až 1 GHz. Paměť RAM u PFC200 dosahuje velikosti až 512 MB. Maximální velikost datové a programové paměti je až 1,5 GB[2]. Procesorové moduly obsahují vždy dva ethernetové porty a další komunikační rozhraní podle typu modulu, které jsou popsány v kapitole 1.1. Na následujícím obrázku je zobrazen příklad jak může vypadat procesorový modul PFC100 nebo PFC200[3].



Obr. 1.1: Příklad procesorového modulu PFC [3]

1. Tlačítko pro reset modulu
2. Datové kontakty sběrnice pro připojení I/O modulů
3. Stavové signálky zdroje napájení
4. Stavové signálky systému
5. Přístup pro otevření konektorů napájení

6. Konektory pro připojení napájení
7. Slot pro paměťovou kartu
8. Kontakty napájení pro I/O moduly
9. Uvolňovací západka I/O modulu
10. Kryt servisního rozhraní
11. Přepínač pracovního režimu
12. Ethernet port
13. Sériové číslo produktu
14. Zámek pro montáž na DIN lištu
15. Sériové rozhraní (volitelné)
16. Místo pro popisky Mini-WSB (volitelné)

Konkrétní typy se můžou od zobrazeného modulu lišit podle dalších komunikačních rozhraní, přítomností modulu zdroje napájení a podle ovládacích a zobrazovacích prvků[3].

1.1 Komunikační rozhraní

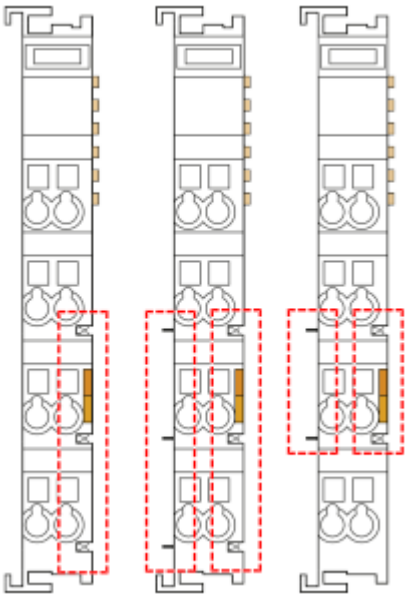
Každý procesorový modul z řady PFC100 a PFC200 disponuje dvěma ethernetovými porty a případně dalšími porty podle konkretního typu. U řady PFC100 je možnost doplnění sériového rozhraní pro komunikaci pomocí RS-232 nebo RS-485. Řada procesorových modulů PFC200 nabízí širší nabídku komunikačních rozhraní. Pro PFC200 nabízí Wago tyto konfigurace rozhraní:

- 4 x Ethernet
- 2 x Ethernet, 2 x SFP Port
- 2 x Ethernet, RS-232/-485
- 2 x Ethernet, CAN
- 2 x Ethernet, RS-232/-485, CAN
- 4 x Ethernet, CAN, CANopen, USB
- 2 x Ethernet, RS-232/-485, CAN, PROFIBUS DP Slave
- 2 x Ethernet, RS-232/-485, Mobile Radio Module
- 2 x Ethernet, RS-232/-485, CAN, CANopen, PROFIBUS Master

1.2 I/O moduly

Samotné procesorové modulu neobsahují žádné vstupy a výstupy. K tomu slouží modulární I/O systém WAGO, kterým lze připojit digitální a analogové vstupy a výstupy nebo také speciální komunikační moduly. Komunikace mezi řídicí stanicí a I/O moduly probíhá pomocí vnitřní datové sběrnice KBUS. Jedná se o 6 datových

kontaktů, které jsou označeny číslem 2 na obrázku 1.1. Maximální počet připojených modulů je omezen na 64. S použitím modulu k prodloužení vnitřní sběrnice je možné připojit až 250 I/O modulů. Procesorové moduly s označením ECO mají počet I/O modulů omezen na 4. Napájení modulů je distribuováno pomocí napájecích propojek na stranách každého I/O modulu, jak je zobrazeno na následujícím obrázku[3].



Obr. 1.2: Napájecí kontakty I/O modulů [3]

Na obrázku je zobrazen modul s nejmenší nabízenou šířkou, tj. 12 mm. Některé moduly můžou mít dvojnásobnou nebo čtyřnásobnou šířku[3].

1.3 Programování

Základním nástrojem pro konfiguraci, programování, simulaci a vizualizaci řídicích úloh je program WAGO e!COCKPIT, který je založen na bázi CODESYS 3[4]. CoDeSys je zkratkou proo Controlled Development System, jedná se o univerzální prostředí pro vývoj aplikací pro PLC. Programovat lze všemi standarními jazyky pro PLC dle normy IEC 61131-3 nebo pomocí objektově orientovaného programování. Norma IEC 61131-3 definuje tyto programovací jazyky[5]:

- IL (Instruction list) - seznam instrukcí
- ST (Structured text) - strukturovaný text
- LD (Ladder diagram) - liniové/reléové schéma
- FBD (Function block diagram) - schéma funkčních bloků
- SFC (Sequential function chart) - tabulka sekvenčních funkcí

- CFC (Continuous function chart) - volně propojované bloky

S novým firmwarem verze 23 je možné pro programování řídicích systémů PFC200 využít přímo prostředí Codesys 3, bez nutnosti zakoupení licence pro WAGO e!COCKPIT. Před programováním v prostředí Codesys je nutné nainstalovat soubory všech používaných zařízení. Dostupné z webových stránek `www.wago.cz` v sekci *Ke stažení*.

Na všech procesorových modulech od firmy Wago je operační systém Linux pracující v reálném čase. Tato možnost zajišťuje větší flexibilitu, použitím open-source softwaru, který lze přizpůsobit požadavkům každého uživatele[2].

1.4 Zabezpečení

Obě řady procesorových modulů poskytují zabezpečení v podobě šifrovaní SSL/TLS, připojení pomocí VPN nebo firewallu. Konfigurace připojení VPN lze provádět pomocí integrovaného web serveru, kde je spuštěný WBM (web-based management)[2].

1.5 Zhodnocení

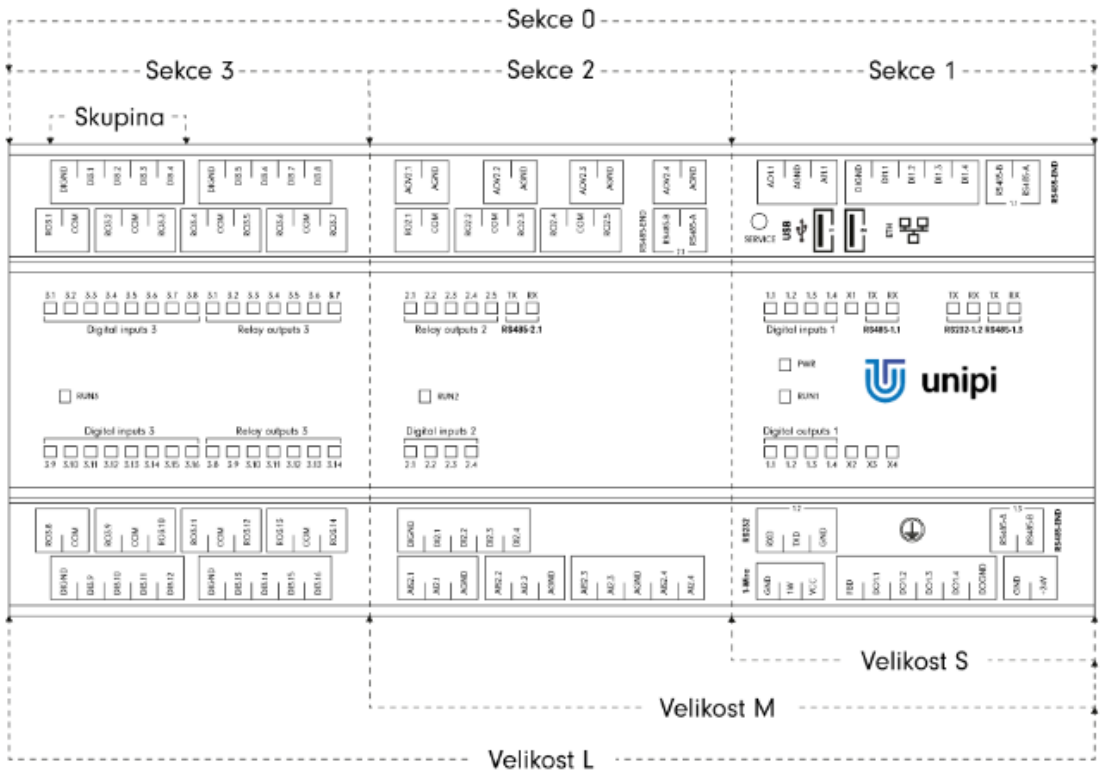
Výhodou řídicích systému Wago je velká nabídka rozšiřujících modulů. V kategorii vstupních a výstupních nabízí moduly s různým počtem vstupů/výstupů, moduly pro různá napětí, zapojení, pracovní podmínky, takže se najde řešení pro každou aplikaci. Další výhodou je vývojové prostředí, kterým je e!COCKPIT založený na bázi Codesys 3 nebo od firmwaru verze 23 již právě zmíněný Codesys 3.

Nevýhodou může být výpočetní výkon, jelikož všechny procesorové moduly obsahují pouze jednojádrový procesor a taky pořizovací cena, která je vyšší ve srovnání s dalšími porovnávanými produkty.

2 Unipi Patron

Unipi je společnost, která má v nabídce hned několik řad řídicích jednotek. Tato kapitola se bude věnovat řadě programovatelných logických kontroléru Patron, založených na vlastním výpočetním modulu s procesorem i.MX 8M Mini s frekvencí až 1,8 GHz. Všechny modely v této řadě disponují stejnou 1GB pamětí RAM a 8GB interní pamětí eMMC[6].

Řada Unipi Patron je vyráběná ve velikostech S, M a L, kde velikost S je nejmenší a L největší. Velikost kontroléru určuje maximální počet vstupů a výstupů. Pro velikost S je to maximálně 8 vstupů a výstupů, pro velikost M je to 40 a pro velikost L až 70 vstupů a výstupů. Kromě klasických digitálních a analogových vstupů a výstupů nabízí Unipi Patron i reléové výstupy, kterými lze spínat zařízení napájené střídavým proudem nebo s vyšším proudovým odběrem[6].



Obr. 2.1: Velikosti Unipi Patron [6]

Označení sekcí na obrázku 2.1 je důležité v softwarové části při adresování vstupů a výstupů. Skrze speciální sekci 0 lze v některých případech přistupovat do všech tří sekcí zároveň[6].

Při nedostatečném počtu vstupů nebo výstupu na kontroléru je možnost připojit rozšiřující moduly Unipi Extension xS11 nebo xS51. Modul xS11 obsahuje pouze digitální vstupy a reléové výstupy, zatímco modul xS51 obshahuje menší počet digitální vstupů a reléových výstupů a navíc obsahuje čtyři analogové vstupy a čtyři výstupy. Rozšiřující moduly komunikují s kontrolérem pomocí protokolu Modbus RTU přes rozhraní RS-485[7].

2.1 Komunikační rozhraní

Všechny modely v řadě Unipi Patron mají několik společných rozhraní, do kterých patří dvojice USB 2.0 portů, 1-wire sběrnice a ethernetový port s maximální přenosovou rychlostí 100 Mb/s. Všechny modely obsahují taky RS-485, ale počet těchto rozhraní se liší podle modelu. Některé modely disponují sériovým rozhraním RS-232 nebo taky LTE modemem pro bezdrátové připojení k internetu nebo odesílání a přijímání SMS[6].

Mimo výše zmíněné komunikační možnosti nabízí Unipi i zakázkovou úpravu hardwaru. Úpravy zahrnují změnu počtu a typy vstupů/výstupů nebo doplnění různých komunikačních protokolů a technologií. V nabídce je např. RS485, RS232, 1-Wire, Ethernet, LTE, DALI, M-Bus, ZigBee, Z-Wave, WiFi, PoE, CAN, BACnet, Profinet, Profibus, EtherCAT, Bluetooth (low energy)[8].

2.2 Programování

Kontroléry Unipi jsou založeny na otevřeném operačním systému Linux a proto je k dispozici několik softwarových řešení. Oficiální softwarovou platformou dodávanou ke všem kontrolérům Unipi je Mervis a bude popsán v další části.

Další možností je využít dodávané API rozhraní pro přímý přístup k vstupům a výstupům. Pomocí tohoto API lze integrovat kontroléry Unipi do softwarového řešení třetích stran nebo do vlastního softwaru. Pro řešení třetích stran nabízí Unipi částečnou technickou podporu pouze pro Node-RED a to v podobě obrazů operačního systému[9].

2.2.1 Mervis

Mervis je softwarová platforma vyvíjená společností Unipi. Tato platforma je rozdělená na následujících částí[9]:

Mervis IDE

Mervis IDE je vývojové prostředí, které je základem celé Mervis platformy. Mezi nejdůležitější funkce patří konfigurace modulů, aktualizace běhového prostředí Mervis RT, programování pomocí funkčních bloků nebo strukturovaného textu podle normy IEC 61131-3, vytváření HMI rozhraní, vytváření projekty pro Mervis SCADA, propojování kontroléru s rozšiřujícími moduly a 1-wire senzory.

Mervis RT

Mervis RT je běhové prostředí, spuštěné přímo na kontroléru, pro vykonávání programu a zařizuje přístup k HMI pomocí web serveru.

Mervis DB

Mervis Database jak z názvu vyplývá je databáze pro ukládání dat z kontroléru a jejich následné analýzy pomocí Mervis SCADA. Službu nabízí Unipi jako cloudovou nebo lokální variantu.

Mervis Proxy

Mervis Proxy je služba zajišťující vzdálený přístup ke kontrolérům, které používají Mervis. Pomocí Mervis Proxy lze vyčítat a nastavovat hodnoty z/do kontroléru, vzdálené připojení, ladění, konfigurace přes Mervis IDE a propojení a výměna dat mezi kontroléry v různých sítích. Tato služba je nutností pro použití Mervis SCADA.

Mervis SCADA

Mervis SCADA zajišťuje základní funkce pro sledování a správu ovládané technologie. Každá pracovní oblast má vlastní datový soubor definující způsob vizualizace dat a komunikace s terminálem. Mervis SCADA má tři režimy:

- seznam datových bodů
- technologické diagramy
- tabulky s historickými daty

Primárně je tato služba nabízená jako cloudová, ale existuje i možnost instalace na vlastním serveru[10].

2.3 Zhodnocení

Mezi výhody systému Unipi Patron patří nízká pořizovací cena řídicích systému a jejich dobrá výbava. Navíc možnost zakázkové úpravy, kterou lze zajistit komunikaci různými protokoly.

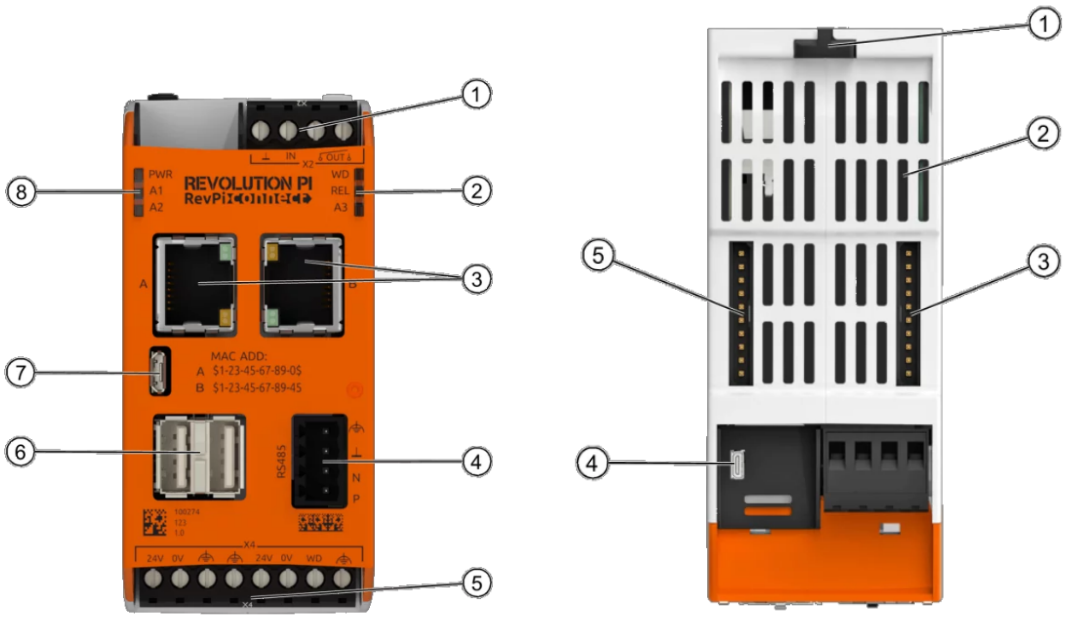
Digitální vstupy lze nakonfigurovat do režimu čítače, který je schopen měřit pulsy s daleko větší frekvencí než v režimu klasického digitálního vstupu. Tato funkce je určitě výhodou při měření pomocí impulsního výstupu.

Nevýhodou může být malá možnost konfigurace vstupů a výstupů, protože každá řídicí jednotka obsahuje určitý počet vstupů a výstupů. V případě, kdy daná aplikaci potřebuje pouze více vstupů nebo výstupu jednoho typu, je rozšíření možné modulem obsahující všechny typy vstupů a výstupů a tak jsou zbylé vstupy/výstupy nevyužité.

3 Revolution Pi

Revolution Pi nebo zkráceně RevPi je průmyslový počítač založen na Rasberry Pi od firmy Kunbus. Nevyužívá klasické Rasberry Pi, ale pouze výpočetní moduly, které připomínají SO-DIMM moduly paměti RAM používané v noteboocích. RevPi je nabízeno v několika modelech. RevPi Compact a RevPi Flat jsou kompaktní kontroléry s integrovanými vstupy, výstupy a komunikačními rozhraními. Naopak modely RevPi Core a RevPi Connect (RevPi Connect+ feat. Codesys) jsou pouze jako procesorové moduly se základním rozhraním. Vstupy, výstupy a další rozhraní je možné připojit pomocí dalších modulů[11].

Tyto modely jsou vybaveny Raspberry Pi Compute Module 4S nebo Compute Module 3+ podle verze. Modul Rasberry Pi Compute Module 4S je vybaven čtyřjádrovým procesorem ARM Cortex A-72 s frekvncí jádra 1,5 GHz a 1 GB paměti RAM typu DDR4. Rasberry Compute Module 3+ má čtyřjádrový procesor ARM Cortex A-53 s frekvencí 1,5 GHz a 1 GB paměti RAM typu DDR2. Interní eMMC úložiště má velikost od 8 do 32 GB[12].



(a) Pohled zepředu

(b) Pohled shora

Obr. 3.1: RevPi Connect[13]

Na obrázku 3.1 jsou zobrazeny dva pohledy modul RevPi Connect. Označení částí při pohledu zepředu, obrázek 3.1a:

1. X2 Svorkovnice
2. Stavové LED signálky

3. Dvojice ethernetových portů
4. 4-pinový RS-485 konektor
5. X4 Svorkovnice - pro připojení napájení a watchdogu
6. USB 2.0 porty
7. micro USB port
8. Stavové LED signálky

Označení částí na pohledu shora na RevPi Connect (obrázek 3.1b):

1. Zámek pro uchycení na DIN lištu
2. Ventilační otvory
3. ConBridge konektor
4. Micro HDMI
5. PiBridge konektor

V porovnání s RevPi Connect má modul RevPi Core poloviční šířku. Z toho vyplývá redukované množství komunikačních rozhraní. Neobsahuje rozhraní RS-485, svorkovnici X2 a jeden ethernetový port. V horní části je ConBridge konektor nahrazen druhým PiBridge konektorem[14].

3.1 PiBridge

PiBridge konektor slouží pro připojení I/O modulů nebo RevPi Gate, které umožňují integraci RevPi do průmyslových sítí. Modely RevPi Connect obsahují pouze jeden PiBridge konektor a modely RevPi Core obsahují tyto konektory dva. Na každý konektor lze připojit maximálně pět modulů, z toho jeden modul může být typ RevPi Gate. Dvojnásobný počet PiBridge konektorů umožňuje RevPi Core připojit dvakrát více vstupů a výstupů a taky dva moduly RevPi Gate. Pomocí RevPi Gates se lze připojit k následujícím sítím:

- CANopen
- DeviceNet
- DMX
- EtherCAT
- Ethernet/IP
- Modbus
- Powerlink
- Profinet
- Profibus
- Sercos

Dále jsou k dispozici tři typy modulů pro rozšíření možností vstupů a výstupů:

- RevPi Digital I/O
- RevPi Analog I/O

- RevPi Multi I/O

Moduly digitálních vstupů a výstupů jsou nabízeny ve třech variantách. Prvním variantou je RevPi DIO, který má 14 vstupů a 14 výstupů. Druhou možností je RevPi DI, který obsahuje 16 digitálních vstupů a poslední variantou je RevPi DO s 16 digitálními výstupy. RevPi Analog I/O module má jedinou variantu a je vybaven dvěma analogovými výstupy, čtyřmi analogovými vstupy a dvěma vstupy pro měření teploty. Modul RevPi Multi I/O obsahuje kombinaci analogových a digitálních vstupů/výstupů. Konkrétně se jedná o 8 analogových výstupů, 8 analogových vstupů a 4 konfigurovatelné digitální vstupy/výstupy. Tyto digitální vstupy/výstupy podporují šest režimů:

- PWM vstup
- PWM výstup
- Měření pulsů
- Výstup pulsů
- Digitální vstup
- Digitální výstup

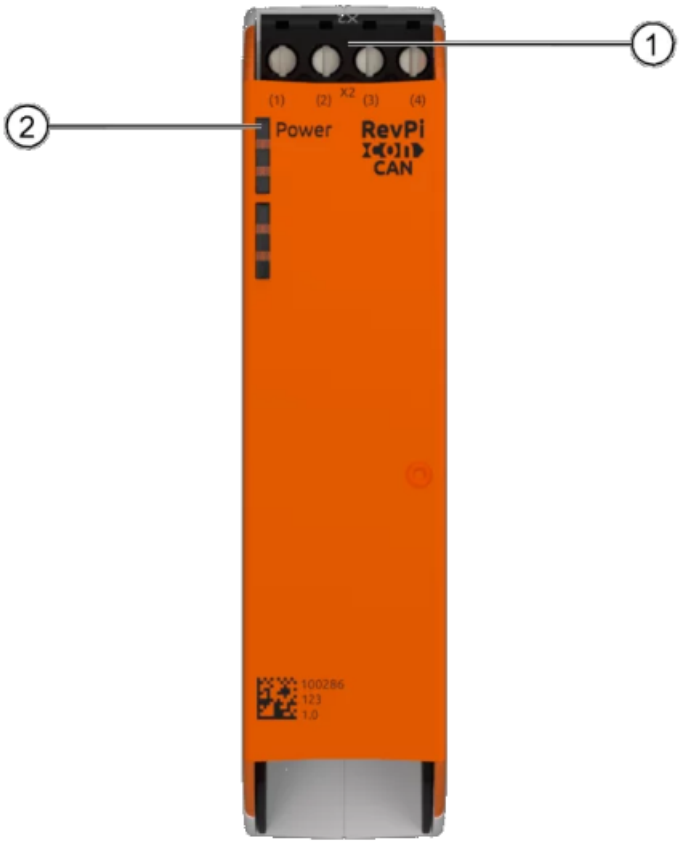
3.2 ConBridge

Konektor ConBridge je podobný k PiBridge konektoru, ale v případě jejich záměny může dojít ke zničení připojovaného rozšiřujícího modulu. Tento konektor se nachází pouze u modelu RevPi Connect (a Connect+ feat. CODESYS) a slouží k připojení komunikačních modulů RevPi Con M-Bus a RevPi Con CAN. Modul RevPi Con M-Bus podporuje bezdrátový protokol M-Bus, používaný například pro vzdálený odečet měřičů tepla, vody, plynu. K dispozici jsou dvě varianty tohoto modulu:

- RevPi Con M-Bus - komunikace probíhá na frekvenci 868 MHz s dosahem 800 m
- RevPi Con M-Bus VHP - komunikace probíhá na frekvenci 169 MHz s dosahem 20 000 m

Uvedené dosahy platí pro přímý dohled mezi vysílačem a přijímačem.

Pomocí RevPi Con CAN je možné se připojit k sběrnici CAN. Pro připojení k sběrnici se používá čtyřvodičová svorkovnice X2, označená číslem 1 na obrázku 3.2. Kontakty 2 a 3 jsou pro datové vodiče *CAN_H* a *CAN_L* a kontakty 1 a 4 slouží pro připojení zakončovacího odporu.



Obr. 3.2: Modul RevPi Con CAN [16]

3.3 Programování

Systémy Revolution Pi jsou softwarově otevřené a každý uživatel si může nainstalovat software, který mu nejvíc vyhovuje. Na všech zařízením je předinstalovaná upravená verze operačního systému Rasbian s podporou práce v reálném čase. Předinstalovný je i open-source software Node-RED, pomocí kterého lze provádět jednoduché automatizační úlohy. Node-RED plně podporuje rozšiřovací moduly RevPi[12].

3.3.1 PiCtory

PiCtory je webová aplikace, pomocí které se vytváří konfigurační soubor. Díky tomuto souboru může Revolution Pi komunikovat s rozšiřujícími moduly. Jsou zde informace o typu připojeného modulu, jeho pozici a základní nastavení podle typu[15].

3.3.2 RevPi Connect+ feat. CODESYS

Firma Kunbus nabízí speciální model RevPi Connect+ feat. CODESYS. Tento model je obdobný s RevPi Connect, ale navíc poskytuje možnost k programování využít vývojové prostředí CODESYS. K programování tohoto modelu je tak možné využít standardní jazyky pro programování PLC. Všechny rozšiřující moduly s výjimkou RevPi Con M-Bus (i verze VHP) jsou plně integrovány do prostředí CODESYS [17].

CODESYS Runtime podporuje mnoho protokolů na základě průmyslového Ethernetu jako například ProfiNet, EtherCat, EtherNet/IP, CANopen a tím i jednodušší integraci do těchto sítí [17].

Vizualizace je možné vytvořit pomocí CODESYS WebVisu. Vizualizace je spuštěná na webovém serveru a je možné ji otevřít s běžným internetovým prohlížečem z jakéhokoliv zařízení, které se nachází ve stejné sítí [17].

3.4 Zhodnocení

Řídicí systémy RevolutioPi mají k dispozici pouze M-Bus moduly, které umožňují komunikaci bezdrátovou variantou tohoto protokolu. Tato vlastnost může být podle účelu systému jako nevýhoda nebo výhoda.

Výhodou rozšiřujících modulů s digitálními vstupy pro RevolutionPi je možnost nakonfigurovat tyto vstupy do režimu čítače, který umožňuje zaznamenávat pulsy s frekvencí až 2 kHz.

4 Porovnání

Následující kapitola se věnuje porovnávání systémů popsaných v minulých kapitolách. Prvním kritériem jsou komunikační možnosti. Druhým kritériem je porovnána pořizovací cena, do které je zahrnut pouze hardware a nakonec jsou porovnána vývojová prostředí. Na základě těchto porovnání bude vybrána nejvhodnější platforma.

4.1 Komunikační možnosti

Komunikace tvoří důležitou část řídicích systému, protože systému umožňuje přijímat data z různých měřičů, odesílat povely a komunikovat s jinými zařízeními. V této části jsem se zaměřil hlavně na podporu protokolů Modbus a M-Bus, sériového rozhraní RS-485 a sběrnice CAN.

Řídicí jednotky PFC100 jsou k dispozici pouze se dvěma ethernetové porty a portem pro sériovou komunikaci. U modulů PFC200 existuje více možností komunikačních rozhraní, jako například sériové rozhraní RS-485 a sběrnice CAN. Oba typy modulů podporují komunikaci protokolem Modbus TCP a RTU. Protokol M-Bus tyto moduly přímo nepodporují, ale je možnost dokoupit modul M-Bus Master. Tato možnost přináší další zvýšení počátečních nákladů.

Protokol Modbus podporují řídicí jednotky Unipi Patron v obou verzích, díky ethernetovému portu a sériovému rozhraní RS-485. S podporou protokolu M-Bus je to podobné jako u Wago PFC100 a PFC200. Samostatné řídicí jednotky tento protokol nepodporují a musí se dokoupit převodník M-Bus na RS-485. Sběrnice CAN není součástí základních řídicích jednotek, ale Unipi nabízí zakázkovou úpravu hardwaru, při které je možnost nechat si sběrnici CAN doplnit. Tato možnost přináší další zvýšení pořizovací ceny. Výhodou jednotek Unipi je podpora sběrnice 1-wire, pro připojení teplotních nebo jiných senzorů.

Základní moduly Revolution Pi mají k dispozici sériové rozhraní RS-485 i ethernetové porty, takže s komunikací pomocí Modbus TCP ani RTU není problém. S protokolem M-Bus je situace horší, opět je k dispozici pouze při zakoupení rozšiřujícího modulu. Tento modul lze připojit pouze k řadě RevPi Connect, která je dražší než RevPi Core. Modul RevPi Con M-Bus podporuje pouze bezdrátovou verzi protokolu M-Bus. Pro podporu sběrnice CAN je potřeba připojit další rozšiřovací modul, který lze opět připojit pouze k řadě RevPi Connect.

4.2 Nákladové hodnocení

Velmi důležitým faktorem při výběru automatizační platformy je pořizovací cena. V následující tabulce je srovnání cen řídicích jednotek popsaných v předchozích

kapitolách. Ve sloupcích DI, DO, AI a AO jsou uvedené ceny za jeden digitální/analogový vstup/výstup. Kontroléry Unipi Patron mají vstupy a výstupy integrované a není možné spočítat cenu za jednotlivý vstup/výstup.

Uvedené ceny za vstupy/výstupy jsou počítány z ceny I/O modulu dělené počtem vstupů/výstupů. Pro Wago I/O moduly jsou nejvýhodnější šestnácti-kanálové digitální vstupní/výstupní moduly a osmi-kanálové analogové I/O moduly. Při výpočtu ceny I/O pro RevPi byla cena vypočítaná pro všechny moduly s výjimkou RevPi MIO, který má analogové i digitální vstupy a výstupy a nelze tak vypočítat jednotlivé ceny. Ceny v závorkách u DI a DO platí pro modul RevPi DIO, který má 14 vstupů a 14 výstupů. Pro přepočet z EUR na Kč byl použit kurz 1 EUR = 25 Kč.

Tab. 4.1: Přehled cen řídicích jednotek a I/O (Ceny jsou uvedené v Kč)

Výrobce	**Model**	**Cena CPU**	**DI**	**DO**	**AI**	**AO**
Wago	PFC100 PFC200	17 181 - 23 026 26 020 - 43 800	217	246	1140	1091
Unipi	S107 S207 M527 L527	10 790 12 990 16 890 21 590				
Kunbus RevPi	Connect+ (CODESYS) Connect S Connect+ Core S Core 3+	15 125 12 100 10 200 8 250 7 325	270 (193)	294 (193)	1169	

Mimo cen řídicích jednotek a I/O je vhodné ještě zmínit náklady na zprovoznění komunikace pomocí protokolu M-Bus. Modulární I/O systém Wago nabízí modul M-Bus master za cenu 8 626 Kč. Unipi nabízí na svém webu převodníky RS-485 na M-Bus začínající na 3 999 Kč. Revolution Pi podporuje pouze bezdrátovou verzi a cena rozšiřujícího modulu začíná na 144 EUR tedy přibližně 3 600 Kč, ale je potřeba vlastnit minimálně RevPi Connect+.

V tabulce 4.2 se nachází cenové porovnání popsaných platforem, tak aby obsahovaly potřebný počet vstupů a výstupu pro realizaci úlohy měření a regulace spotřeby. Celkový počet požadovaných vstupů a výstupů je následující:

- 5 + 5 digitálních vstupů
- 4 + 5 digitálních výstupů
- 5 analogových vstupů
- 4 analogové výstupy

Druhé číslo u digitálních vstupů a výstupu a počet analogových vstupů a výstupu označuje rezervu pro plánované rozšíření aplikace. Komunikace pomocí protokolu M-Bus není požadováná a je pouze výhodou a tak jsem v tabulce uvedl sumu zahrnující cenu s požadovanými vstupy a výstupy a sumu systémů s podporou protokolu M-Bus.

Tab. 4.2: Porovnání cen řídicích jednotek pro praktickou část (Ceny jsou uvedené v Kč)

	Wago		**Unipi**		**Kunbus**	
	PFC200	26 019	M207	15 990	RevPi Connect	15 125
	DI	3 478	xS51	7690	DIO	5 414
	D0	3 943				
	AI	14 575			AIO	9 341
	AO	7 112				
Suma		**55 128**		**23 680**		**27 888**
M-BUS		8 626		6 799		4 343
Suma (M-BUS)		**63 754**		**30 479**		**32 223**

Z nákladového hlediska jsou nejvýhodnější řídicí systémy Unipi Patron. RevolutionPi od společnosti Kunbus je cenově velmi srovnatelné. Nejdražší jsou systémy od společnosti Wago. Za zmínku stojí to, že uvedené ceny jsou z oficiálního ceníku, ale Wago poskytuje svým zákazníkům různé rabaty. V případě větších odběrů by se ceny mohly stát konkurence schopnými.

4.3 Vývojové prostředí

Vývojové prostředí je dalším důležitým faktorem při výběru řídicího systému. Dobře známé prostředí může ušetřit spoustu času a tedy i financí při vývoji programu.

Primárním vývojovým prostředím systémů Wago je aplikace e!COCKPIT. Jádrem aplikace je CODESYS 3, který podporuje standardní programovací jazyky pro PLC dle normy IEC 61131-3. Jedná se o velmi rozšířené prostředí a tím snižuje čas potřebný na seznámení se softwarem a zvyšuje možnosti uplatnění. Od nového firmwaru verze 23, je ukončená podpora prostředí e!COCKPIT a je k dispozici prostředí CODESYS 3. To znamená i snížené náklady za licence k software e!COCKPIT.

Unipi dodává k většině svých řídicích systém svůj vlastní software Mervis, včetně vývojového prostředí Mervis IDE a není tak třeba platit za jiné vývojové prostředí.

U tohoto prostředí poskytuje Unipi plnou technickou podporu a garantuje funkčnost se všemi jednotkami Unipi. Nevýhodou může být, že se jedná o proprietární software a programátor bude potřebovat určitý čas na seznámení. Přenesení vytvořeného programu z Mervis IDE do jiného prostředí tak může činit problém. Pro programování jednotek Unipi lze využít i prostředí CODESYS, ale Unipi nezaručuje plnou funkcionalitu.

Na řídicích systémech Revolution Pi je předistalovaný software Node-RED. Tento software je vhodný pro vytváření jednoduchých úloh. V případě složitějších aplikací bych zvolil jiný software nebo si připlatit za verzi, která podporuje prostředí CODESYS.

Na všech zmíněných platformách běží operační systém Linux a nabízí se tak možnost použití vlastního softwaru. Na základě vývojových prostředí bych hodnotil nejlépe systémy Wago i přestože se jedná o placený software. Pokud se jedná o vývoj pouze jedné aplikace, je možnost použít bezplatnou 30denní verzi aplikace. Protože je e!COCKPIT založen na CODESYSu, není kód vázán na konkrétní hardware, ale lze ho relativně lehce použít i pro jiný hardware programovatelný prostředím CODESYS.

4.4 Zhodnocení

Hardwarově si jsou všechny platformy velmi podobné. Všechny platformy mají sériové rozhraní, minimálně jeden ethernetový port s podporou komunikace pomocí protokolu Modbus TCP. Pro komunikaci protokolem M-Bus je vždy potřebný další modul nebo převodník. Počet digitálních a analogových vstupů/výstupů je možné v případě potřeby rozšířit pomocí modulů.

U řídicích systému řady Unipi Patron a rozšiřujících modulů pro RevolutionPi je možné nakonfigurovat digitální vstupy do funkce čítače. V tomto režimu je vstup schopen přijímat pulsy s daleko vyšší frekvencí a to je při měření spotřeby impulsním výstupem výhodou. Společnost Wago sice nabízí moduly s funkcí čítače, ale to přináší další zvýšení pořizovacích nákladů.

Pořizovací náklady systému Wago jsou nejvyšší ze všech porovnávaných možností, ale zase není zapotřebí kupovat licence k dalším softwarům a vývoj může probíhat v prostředí Codesys 3. K řídicím jednotkám Unipi je navíc software Mervis, který podporuje základní programovací jazyky normy IEC 61131-3. Některé služby systému Mervis jsou poskytovány jako cloudové služby, ale Unipi nabízí i možnost instalace na vlastním serveru.

Cena systému RevolutionPi je jenom lehce vyšší než u jednotek Unipi a lze si vybrat model s podporovou prostředí Codesys. Pro jiné modely je nutno zvolit jiný software, jako například NodeRed.

Jako nejvýhodnější platformy bych zvolil řídicí systémy Unipi Patron, z důvodu nejnižší pořizovací ceny a taky je k systému dodáván software, takže není zapotřebí dalších licencí.

5 Praktická úloha

Praktická úloha je realizována na procesorovém modulu PFC200 doplněném o požadované vstupní a výstupní moduly. Konkrétně se jedná o variantu 2. generace s dvěma ethernetovými porty a portem RS-232/-485. V procesorovém modulu je nainstalován nový firmware verze 23, který podporuje programování vývojovým prostředím Codesys 3.

Realizovanou úlohou je část systému MaR (měření a regulace) haly odstavů tramvají a jejího administrativně technického zázemí pro DPMO (Dopravní podnik města Olomouce). Konkrétně se úloha zabývá měřením a regulaci spotřeby elektrické energie v závislosti na povoleném čtvrthodinovém maximu. Součástí je úlohy je integrace do stávajícího systému MaR.

V celém systému MaR se nachází dva řídící systémy PFC200 od firmy Wago, pojmenované RMAR1 a RMAR3. Praktická část této práce se věnuje pouze RMAR1 a jeho komunikaci s RMAR3 a nadřazeným systémem MaR.

Tuto úlohu jsem rozdělil do dvou větších částí. První částí je programová část realizována v prostředí Codesys a druhou částí je vizualizace, ke které jsem využil aplikaci Quick Control 4. Tato aplikace bude popsaná v kapitole 5.5.

5.1 Popis činnosti

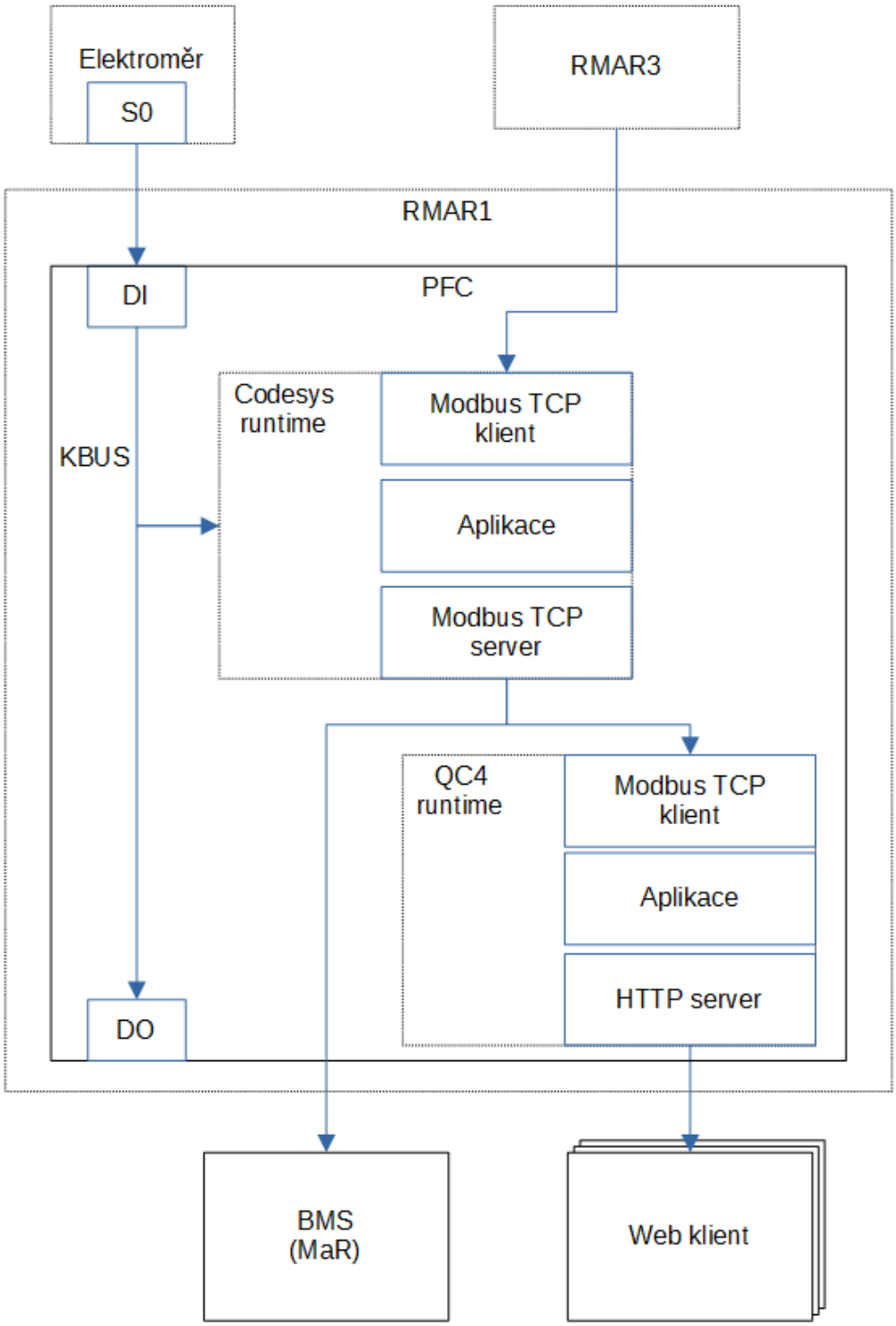
Řídicí systém bude měřit spotřebu elektrické energie a počítat predikci čtvrthodinové spotřeby. Na základě vypočítané predikce a nastaveného maxima bude regulovat spotřebu. Regulace bude probíhat odpojováním spotřebičů podle stanovených priorit s možností nastavení výkonů každého stupně. Aplikaci bude možné přepnout do manuálního režimu, kde bude možné blokovat chod spotřebičů ručním ovládáním.

Tab. 5.1: Seznam digitálních vstupů a výstupů

Vstup	**Popis**	**Výstup**	**Popis**
DI 1	Signál čtvrthodiny	D0 1	Blokování spotřebičů 1. stupně
DI 2	Činná energie dodávka	D0 2	Blokování spotřebičů 2. stupně
DI 3	Činná energie odběr	D0 3	Blokování spotřebičů 3. stupně
DI 4	Jalová energie kapacitní	D0 4	Blokování spotřebičů . stupně
DI 5	Jalová energie induktivní		

Spotřeba bude měřena pomocí impulsních výstupů elektroměru, které budou přivedeny na digitální vstupy. Celkem bude z elektroměru přivedeno pět signálů. Popisy jednotlivých vstupů jsou uvedeny v tabulce 5.1. Ze systému budou vyvedeny

čtyři výstupní signály, jeden pro každý regulační stupeň. Všechny analogové vstupy a výstupy budou v této verzi nezapojené, jedná se pouze o rezervu pro plánované rozšíření aplikace.



Obr. 5.1: Schéma realizované aplikace

Dále bude k aplikaci zpracována vizualizace, zobrazující aktuální měřené a celkové hodnoty. Vizualizace bude obsahovat deník událostí, přehled alarmů, trendy i možnost nastavení sjednaných kapacit, výkonů jednotlivých odpojovaných stupňů spotřebičů a stavů elektroměru.

Vizualizace bude zobrazována na operátorských panelech na dveřích rozvaděčů. Z tohoto důvodu je důležité zajistit různá uživatelská oprávnění proti neoprávněnému přístupu. Výchozí oprávnění budou umožňovat pouze prohlížení. Pro ovládání a možnost nastavování se bude nutné přihlásit.

Integrace do stávajícího systému MaR bude realizována komunikací pomocí protokolu Modbus TCP. Řídicí systém RMAR1 bude komunikovat s nadřazeným systémem MaR jako server a druhým řídicím systémem bude komunikovat jako klient.

5.2 Měření spotřeby

Spotřeba elektrické energie je měřena pomocí impulsních výstupů elektroměru. Měření probíhá v programu *Measurement*. Protože se jedná o program, který reaguje na změny na digitálních vstupech, je nutné, aby perioda volání byla dostatečně malá a systém tak zachytil všechny pulsy z elektroměru. V úloze *IOTask*, která volá program *Measurement* je sice možnost nastavit úlohu jako *freewheeling*, ale perioda spouštění je zbytečně malá a navíc tak častý běh programu zabírá větší množství výpočetního výkonu. Z tohoto důvodu jsem se rozhodl spouštění úlohy nastavit jako *External* a událostí, která spouští tuto úlohu, je perioda lokální sběrnice KBUS, pomocí které komunikuje procesorový modul s vstupními a výstupními moduly.

Samotný přepočet pulsů na fyzikální veličiny, jako jo v tomto případě činný a jalový výkon, se provádí ve funkčním bloku *pulse2Count*. Tento blok obsahuje dva vstupní parametry, kterými jsou digitální vstup, na který je přiveden impulsní výstup elektroměru a váhu jednoho pulsu. Tuto hodnotu je nutné zjistit z elektroměru. Funkční blok obsahuje ještě dva vstupně-výstupní parametry. Prvním je struktura *EnergyCount*, která obsahuje všechny hodnoty potřebné pro výpočet aktuální hodnoty výkonu a spotřeby resp. dodávky elektrické energie. Všechny hodnoty jsou zobrazeny a popsány na následujícím výpisu.

Druhým vstupně-výstupním parametrem je hodnota typu *REAL* pro změnu uložené hodnoty celkové spotřeby. Protože čtvrthodinová spotřeba se počítá z aktuální hodnoty celkové spotřebované energie a odečtu provedeného na začátku čtvrthodiny, je nutné provést kompenzaci změny uložené spotřeby, aby čtvrthodinová spotřeba odpovídala skutečnosti a nebyla ovlivněna provedenou změnou. Proto se rozdíl mezi zapisovanou a uloženou hodnotou přičte k aktuální hodnotě položky *writeDiff*, která se později využívá pro kompenzaci výpočtu čtvrthodinové spotřeby. Vynulování *writeDiff* proběhne vždy na začátku čtvrthodiny.

Výpis 5.1: Struktura *EnergyCount*

```
TYPE EnergyCount :
STRUCT
    pulseCount: ULINT;              // Počet naměřených pulsů
    lastPulseTimestamp: ULINT;      // Časová značka posledního
   pulsu
    currentValue: Real2Words;       // Aktuální hodnota výkonu (č
   inný/jalový)
    cumulativeValue: Real2Words;    // Celková hodnota spotřebované
   /dodané el. energie
    writeDiff: REAL;                // Rozdíl pro správný výpočet č
   tvrthodinové spotřeby při zápisu stavu elektroměru
END_STRUCT
END_TYPE
```

Typ *Real2Words* je opět vlastní struktura, vytvořena pro přenos dat protokolem Modbus TCP a bude dále popsaná v kapitole 5.4. Všechny proměnné typu *EnergyCount* jsou definovány v PVL (Persistent Variable List), jedná se o perzistentní proměnné, které se nevymažou při výpadku napájení.

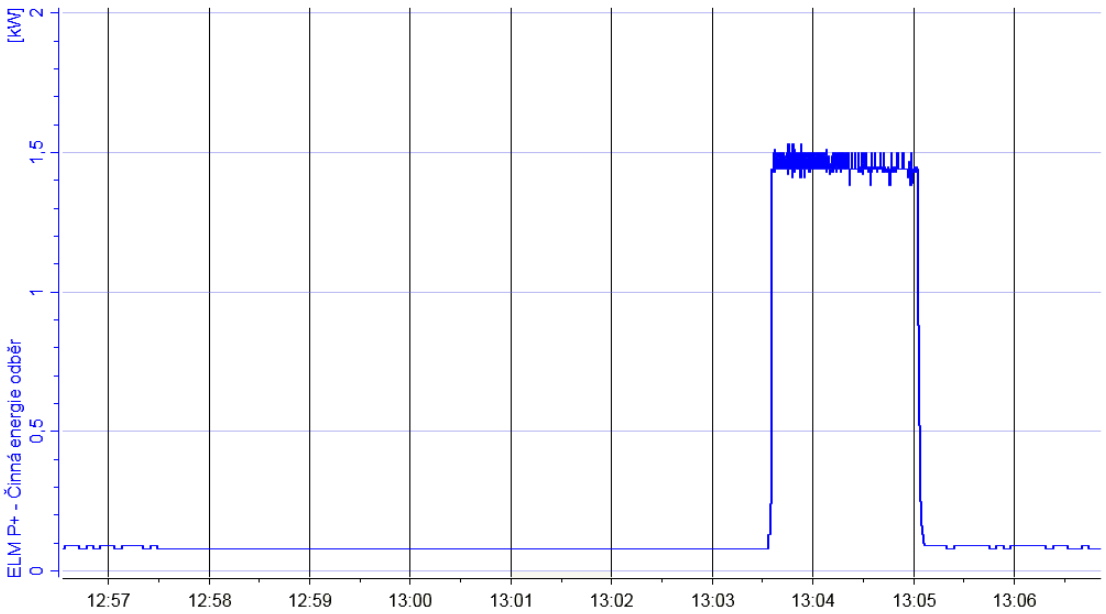
Při prvním přijatém pulsu, se uloží časová značka, kdy byl puls přijat. Po příchodu druhého a všech dalších pulsů se spočítá interval mezi aktuálním a předchozím pulsem. Z tohoto intervalu se vypočítá teoretický počet pulsů za hodinu a vynásobením váhou pulsu se zjistí aktuální hodnota.

$$P = \frac{T_h}{T_i} \cdot Imp_{val} = \frac{60 \cdot 60 \cdot 1000}{4000} \cdot 0,0001 = 0,09 \qquad [\text{kW}] \qquad (5.1)$$

Na výše uvedené rovnici je příklad výpočtu činného výkonu, kde T_h je počet ms v jedné hodině, T_i je interval mezi pulsy v ms a Imp_{val} je váha pulsu. V tomto případě je elektroměr nastaven na 10 000 pulsů na kWh a váha pulsu je rovna převrácené hodnotě. Jedná se o nastavení elektroměru, který byl využit pro testovací účely.

Hodnota měřeného výkonu se počítá pouze v době příchodu pulsu nebo pokud čas mezi pulsy přesáhl dobu předchozího intervalu. To znamená, že při prodloužení času mezi pulsy, to je snížení výkonu, se vypočítaný výkon začne postupně snižovat až po uplynutí času mezi posledními dvěma pulsy.

Na obrázku 5.2 je zobrazen průběh aktuálního odběru získaný funkčním blokem *pulse2Count* s využitím testovacího elektroměru. Při měření byla k elektroměru připojena televize zajišťující konstantní odběr přibližně 100 W a mikrovlnná trouba, která způsobila zvýšení odběru. V grafu jde vidět, že měření s konstantním odběrem je stabilní, ale při zvýšení odběru hodnota kmitá v rozmezí od 1 400 W do 1 500 W. Hlavní podíl na těchto kmitech má nepřesnost měření času mezi pulsy, kde hraje roli perioda komunikace lokální sběrnice, kterou komunikuje procesorový modul s vstupními a výstupními moduly. Perioda lokální sběrnice KBUS je nastavena 10 ms.



Obr. 5.2: Průběh výstupu z funkčního bloku *pulse2Count*

Měření bylo provedeno na elektroměru, který je nastaven na 10 000 pulsů na kWh, jak již bylo zmíněno. Pro výslednou hodnotu odběru 1,5 kWh to znamená, že perioda pulsů z elektroměru je 240 ms. Jestliže impuls přijde těsně po skončení jednoho cyklu lokální sběrnice, detekce pulsu programem se tak může opozdit až o zmíněných 10 ms a výsledný vypočítaný odběr bude ovlivněn chybou. Po dosazení hodnot časů 240 ms a 250 ms za T_i do rovnice 5.1, obdržím následující hodnoty.

$$P = \frac{3\ 600\ 000}{240} \cdot 0,0001 = 1,5 \qquad [\text{kW}] \qquad (5.2)$$

$$P = \frac{3\ 600\ 000}{250} \cdot 0,0001 = 1,44 \qquad [\text{kW}] \qquad (5.3)$$

Vypočítané hodnoty přibližně odpovídají minimální a maximální hodnotě kmitů při zvýšeném odběru na obrázku 5.2.

5.3 Predikce a regulace spotřeby

Na rozdíl od počítání pulsů a aktuálního odběru není výpočet predikce čtvrthodinové spotřeby časově kritický a není nutné ho provádět s co nejmenší periodou. Z tohoto důvodu jsem vytvořil úlohu *Task100ms*, která je spouštěná periodicky každých 100 *ms*. Tato úloha volá program pro výpočet predikce čtvrthodinové spotřeby a program pro její regulaci. Oba zmíněné programy budou popsány v pokračování této kapitoly.

Před výpočtem predikce si musím určit období, za které se bude predikce počítat. Tento interval počítám pomocí čtvrthodinového pulsu z elektroměru. S tímto pulsem pracuje program *T15start*, který je volán z úlohy *IOTask*, která jak bylo zmíněno je spouštěná s periodou lokální sběrnice.

Po příchodu zmíněného pulsu si program uloží několik hodnot, které jsou důležité pro následující výpočty nebo pro nadřazené systémy. Jedná se hlavně o časovou značku příchodu čtvrthodinového pulsu, časovou značku posledního pulsu činného výkonu před začátkem čtvrthodiny a celkovou spotřebu v době začátku čtvrthodiny.

Pokud z jakéhokoliv důvodu nepřijde impuls pro následující čtvrthodinu, je v programu počítán čas od začátku čtvrthodiny. V případě, že počítaný čas přesáhne dobu čtvrthodiny a deseti vteřin je automaticky spuštěna nová čtvrthodina.

Důležitou hodnotou pro výpočet predikce je průměrný odběr v aktuální čtvrthodině. Tento výpočet jsem nejdřív počítal s přesností času v sekundách, ale to se ukázalo jako špatné rozhodnutí, protože výsledná hodnota byla značně nestabilní. Z toho důvodu jsem začal měřit čas v milisekundách. Na obrázku 5.3 je zobrazeno porovnání průběhů získaných výpočtem s různou časovou přesností pro první dvě minuty v čtvrthodině. Modrou barvou je označen průběh průměrného odběru s přesnosti měření času na vteřiny a oranžovou barvou je znázorněn průběh počítaný s přesností času v milisekundách. V grafu lze vidět, že při počítání s přesnosti času v sekundách, je hodnota průměrného odběru na začátku čtvrthodiny nestabilní a každý příchozí impuls vytvoří špičku. Hodnota průměru se ustálí až po více než dvou minutách.

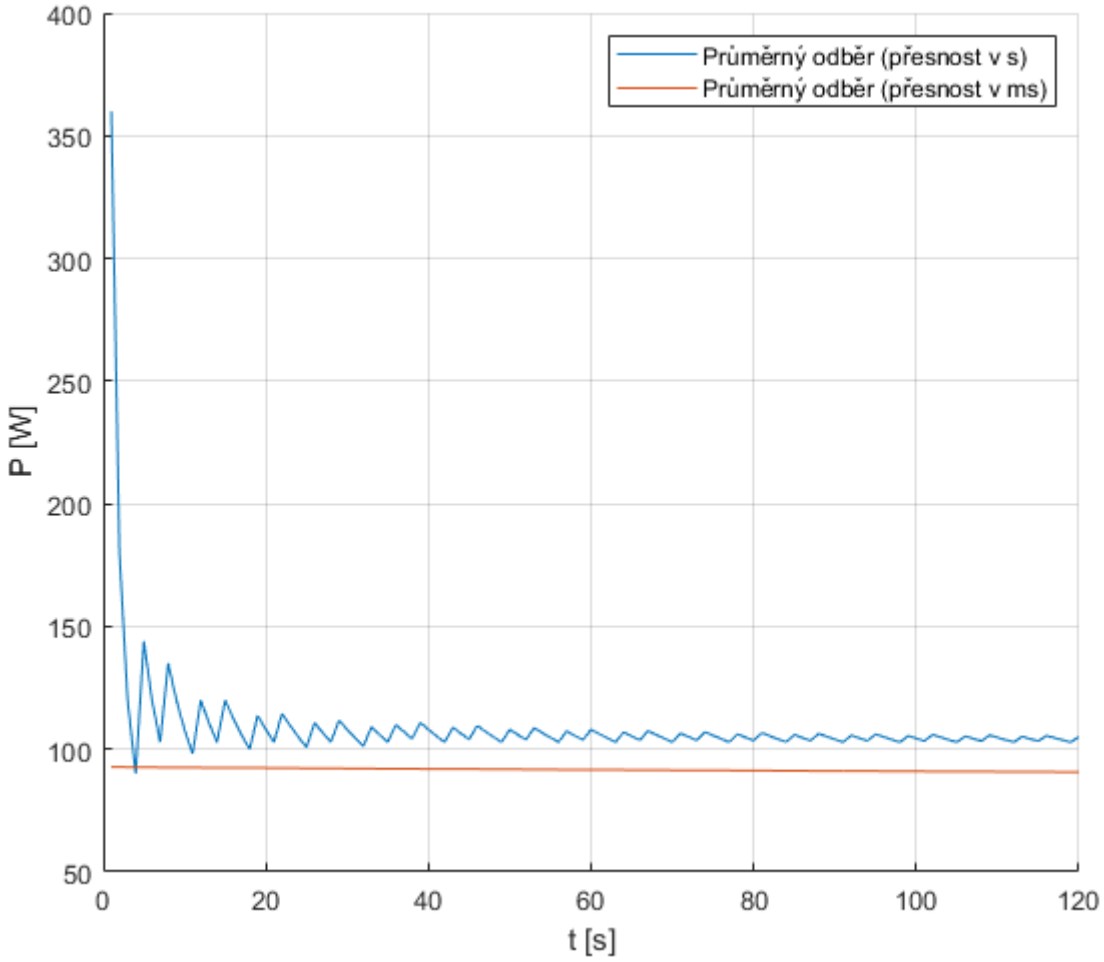
Měření bylo provedeno při konstantním odběru, ale u obou měření byl odběr různý. Pro první měření (modrý průběh) byl odběr přibližně 103 W a při druhém měření 81 W.

Výpočet predikce se provádí v programu *Estimation* a jedná se o predikci spotřeby za aktuální čtvrthodinu. Čas začátku a konce čtvrthodiny je řízen pulsem z elektroměru. Výpočet predikce je rozdělen do tří částí podle času v aktuální čtvrthodině.

V prvních pěti minutách čtvrthodinového intervalu je očekávaná hodnota spotřeby za zbývající čas vypočítaná podle rovnice 5.4.

$$P_r = P_{avgLast} \cdot (1 - t) + P_{avg} \cdot t \tag{5.4}$$

Kde P_r je očekávaná hodnota spotřeby za zbývající čas, $P_{avgLast}$ je průměrný odběr za minulou čtvrthodinu, P_{avg} je průměrný odběr v aktuální čtvrthodině a t je hodnota reprezentující čas v aktuálních pěti minutách a nabývá hodnot od 0 na začátku čtvrthodiny do 1 čase pěti minut od začátku čtvrthodiny.



Obr. 5.3: Porovnání průběhů průměrného odběru s přesností času v sekundách a milisekundách

V druhém pětiminutovém intervalu, je hodnota P_r rovná průměrnému odběru v aktuální čtvrthodině. V posledních pěti minutách čtvrthodiny je výpočet prováděn podle následující rovnice.

$$P_r = P_{avg} \cdot (1 - t) + P \cdot t \tag{5.5}$$

Kde P_r je očekávaná hodnota spotřeby za zbývající čas, P_{avg} je průměrný odběr v aktuální čtvrthodině, P je aktuální odběr a t je hodnota, která reprezentuje čas v aktuálních pěti minutách a nabývá hodnot od 0 do 1.

Z vypočítané hodnoty P_r a průměrného odběru za aktuální čtvrthodinu (P_{avg}) je následně vypočítaná predikce spotřeby na konci aktuální čtvrthodiny podle následující rovnice.

$$P_{est} = P_r \cdot (1 - t_q) + P_{avg} \cdot t_q \tag{5.6}$$

Kde P_{est} je predikovaná hodnota spotřeby, P_r je hodnota spotřeby za čas zbývající do konce čtvrthodiny, P_{avg} je průměrný odběr v aktuální čtvrthodině a t_q je hodnota od 0 do 1 reprezentující čas ve čtvrthodině.

Regulace spotřeby se provádí blokováním chodu spotřebičů na základě rozdílu vypočítané predikované spotřeby a sjednaného maxima s dodavatelem elektrické energie. Existují čtyři stupně priority blokování spotřebičů.

1. Blokování elektrického dohřevu TUV a napájení Sahar
2. Blokování chodu bivalentního zdroje tepelných čerpadel
3. Blokování chodu tepelných čerpadel
4. Blokování ohřevu vzduchu pro ventilaci montážních jam

Pokud překročí predikovaná hodnota spotřeby sjednanou maximální hodnotu, spočítá se spotřeba každého odpojovaného stupně za čas zbývající do konce čtvrthodiny. Podle tohoto výkonu se blokuje chod spotřebičů tak, aby predikovaná hodnota byla menší než sjednané maximum.

5.4 Modbus TCP

Nedílnou součástí většiny řídicích systému jsou různé komunikační protokoly pro komunikaci s různými dalšími systémy. U této aplikace se požaduje pouze komunikace pomocí protokolu Modbus TCP s již existujícím nadřazeným systémem MaR a s druhým řídicím systémem. Pro komunikaci s druhým PLC jsem zvolil komunikaci jako klient, kdy si požadované data bude systém vyčítat podle potřeby. Data z druhé stanice a vlastní data bude poskytovat nadřazenému systému a QC4 aplikaci jako server.

Mapovat registry pro komunikaci pomocí Modbusu lze pouze jako datový typ *WORD*, což je 16 bitů. Většina hodnot, které potřebuji komunikovat, je datového typu *REAL*, který má délku 32 bitů. Pro mapování hodnoty typu *REAL* do 16 bitových registrů jsem vytvořil datový typ *Real2Words*. Na následujícím výpisu je zobrazena deklarace toho typu.

Výpis 5.2: Deklarace datový typu *Real2Words*

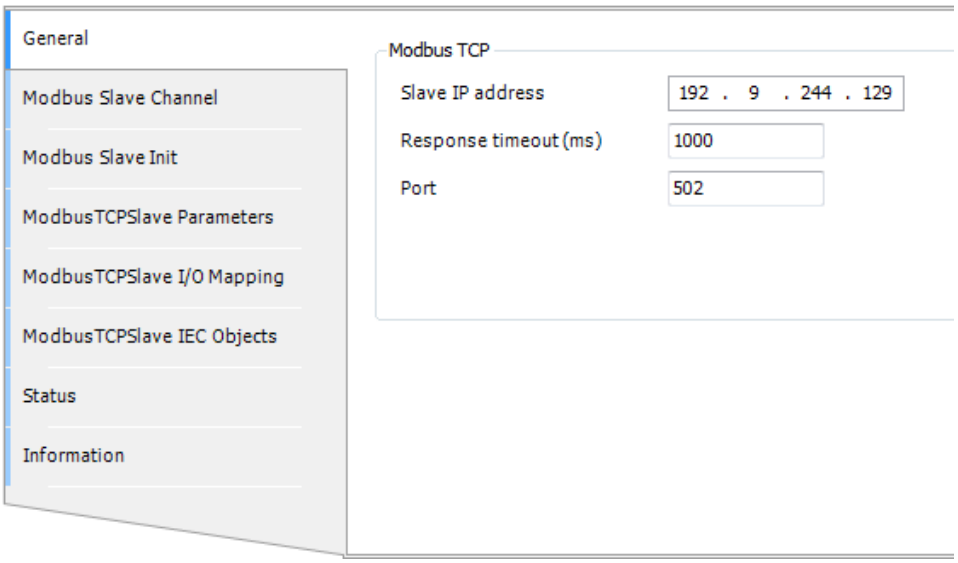
```
1 TYPE Real2Words :
2 UNION
3   value: REAL;
4   w: ARRAY [0..1] OF WORD;
5 END_UNION
6 END_TYPE
```

Tento typ není *STRUCT* jako typ *EnergyCount*, ale jedná se o *UNION*. To znamená, že všechny definované položky, v mém případě se jedná o *value* a *w*,

mají stejnou počáteční adresu v paměti. V praxi to znamená, že můžu v programu přistupovat k hodnotě, která je typu *REAL* a tuto hodnotu můžu bez problému namapovat do 16 bitových registrů.

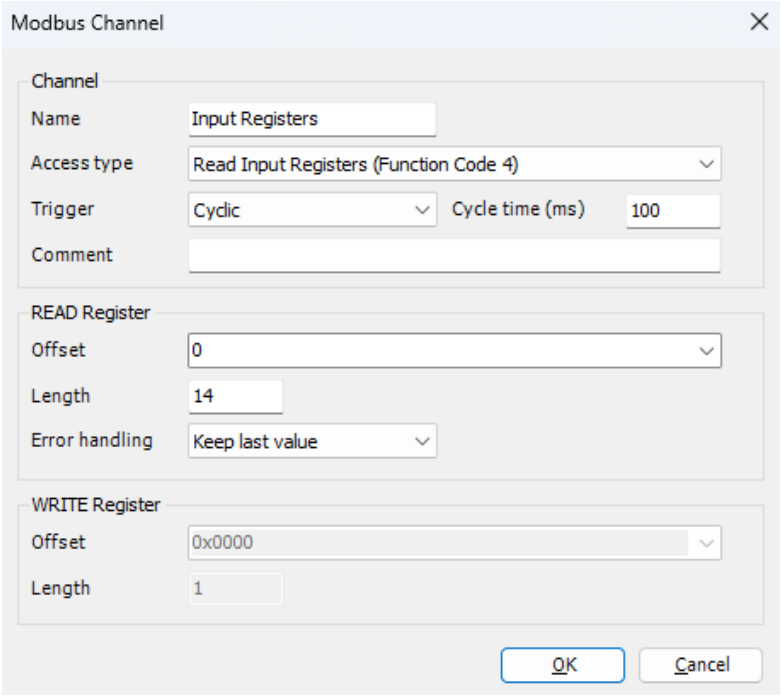
5.4.1 Modbus TCP Master

Pro čtení analogových hodnot z druhého řídicího systému je využit komunikační protokol Modbus TCP v roli klienta. Před konfigurací komunikace se musí přidat všechny potřebné zařízení do projektu. Jedná se o Ethernet adaptér a následně *Modbus TCP Master* do procesorového modulu PFC200 ve stromečku *devices* (většinou k nalezení na levé straně Codesys aplikace). Následně lze do takto vytvořeného *Master* zařízení přidat potřebný počet *Slave* zařízení, s kterými chci komunikovat. V případě této aplikace se jedná pouze jedno zařízení.



Obr. 5.4: Modbus TCP Master - Nastavení spojení se *Slave* zařízením

Do další části konfigurace komunikace patří nastavení registrů případně bitů, které se mají číst nebo zapisovat. To se provádí v záložce *Modbus Slave Channel*. Každý definovaný kanál (Modbus Channel) je jeden příkaz protokolu Modbus TCP. Definice kanálu je zobrazena na obrázku 5.5. Mezi důležité parametry patří *Name* (název kanálu, podle kterého následně probíhá mapování), *Access* type (Modbus funkce), *Offset* (počáteční adresa), *Length* (počet registrů/bitů, které se budou číst nebo zapisovat). Parametr Trigger slouží pro spouštění dané funkce. Použil jsem periodické spouštění s periodou 100 *ms*. Další možností je aplikační spouštění, kdy se funkce spouští v programu a perioda záleží na periodě programu, z kterého je volána.



Obr. 5.5: Modbus TCP Master - Definice Modbus Channel pro čtení vstupních registrů

Pro tuto aplikaci jsem definoval dva kanály, jeden pro čtení 7 analogových měření z druhého systému a 7 bitů označujících chyby měření pro každou hodnotu.

Poslední důležitou části konfigurace je samotné mapování registrů. Na obrázku 5.6 je zobrazeno mapování obou čtených kanálů definovaných v předchozím kroku. Důležitý je sloupec *Channel*, ve kterém je název, který jsem si definoval při vytváření kanálu. V tomto případě je to jednoduché, ale kdybych měl definováno více kanálu, které vyčítají vstupní registry, musel bych zvolit různá jména, jinak bych nebyl schopen tyto registry v této části rozlišit.

Do sloupečku *Variable* se zapíše jméno proměnné, do které chci aby se čtená hodnota zapsala, případně můžu vytvořit novou proměnnou.

5.4.2 Modbus TCP Slave

Nastavení komunikace Modbus TCP v roli *Slave* je výrazně jednodušší. Po přidání zařízení *Modbus TCP Slave device* do Ethernet adaptéru ve stromečku zařízení, je při konfiguraci důležité zaškrtnout položku *Watchdog*, bez které se samotná komunikace nespustí.

Následně jsem definoval počet Holding a Input registrů. Ve výchozím nastavení jsou Input registry určené pouze výstup z řídicího systému a Holding registry jako

Variable	Mapping	Channel	Address	Type
		Input Registers	%IW10	ARRAY [0..13] OF WORD
Application.GVL.T3131.value.w[1]		Input Registers[0]	~~%IW9~~	WORD
Application.GVL.T3131.value.w[0]		Input Registers[1]	~~%IW10~~	WORD
Application.GVL.M3131.value.w[1]		Input Registers[2]	~~%IW11~~	WORD
Application.GVL.M3131.value.w[0]		Input Registers[3]	~~%IW12~~	WORD
Application.GVL.T3151.value.w[1]		Input Registers[4]	~~%IW13~~	WORD
Application.GVL.T3151.value.w[0]		Input Registers[5]	~~%IW14~~	WORD
Application.GVL.T3171.value.w[1]		Input Registers[6]	~~%IW15~~	WORD
Application.GVL.T3171.value.w[0]		Input Registers[7]	~~%IW16~~	WORD
Application.GVL.M3171.value.w[1]		Input Registers[8]	~~%IW17~~	WORD
Application.GVL.M3171.value.w[0]		Input Registers[9]	~~%IW18~~	WORD
Application.GVL.T3181.value.w[1]		Input Registers[10]	~~%IW19~~	WORD
Application.GVL.T3181.value.w[0]		Input Registers[11]	~~%IW20~~	WORD
Application.GVL.T3191.value.w[1]		Input Registers[12]	~~%IW21~~	WORD
Application.GVL.T3191.value.w[0]		Input Registers[13]	~~%IW23~~	WORD
		Discrete Inputs	%IB48	ARRAY [0..0] OF BYTE
		Discrete Inputs[0]	%IB48	BYTE
Application.GVL.T3131.error		Bit0	~~%IX46.0~~	BOOL
Application.GVL.M3131.error		Bit1	~~%IX46.1~~	BOOL
Application.GVL.T3151.error		Bit2	~~%IX46.2~~	BOOL
Application.GVL.T3171.error		Bit3	~~%IX46.3~~	BOOL
Application.GVL.M3171.error		Bit4	~~%IX46.4~~	BOOL
Application.GVL.T3181.error		Bit5	~~%IX46.5~~	BOOL
Application.GVL.T3191.error		Bit6	~~%IX46.6~~	BOOL
		Bit7	%IX48.7	BOOL

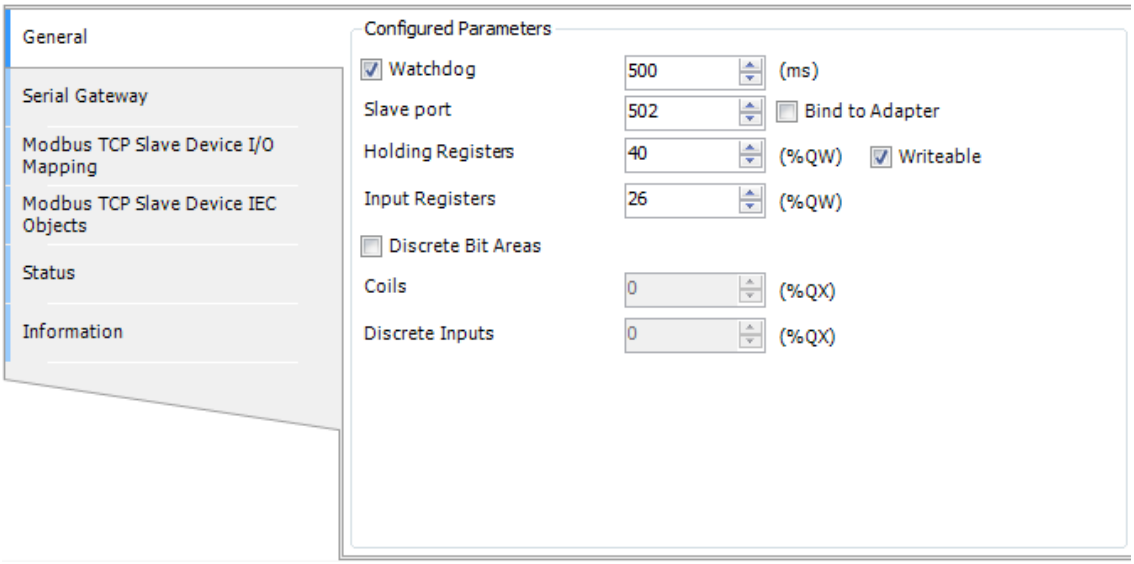
Obr. 5.6: Příklad mapovaní registrů pro komunikaci protokolem Modbus TCP s využitím datového typu *Real2Words*

vstup do systému. U Holding registrů je možno pomocí zaškrtávacího políčka *Writeable* umožnit čtení i zápis.

Posledním krokem je nastavení mapování, to je obdobné jako v předchozím případě. V obou případech existuje možnost mapovaní registrů importovat/exportovat z/do souboru s příponou **.csv*. Využití tohoto způsobu je vhodné zejména v případě, že systémy, které spolu komunikují, jsou vyvíjené různými programátory a je potřeba zpracovat komunikační tabulky. Tyto tabulky jsou většinou zpracovány v tabulkovém procesoru, z kterého lze snadno vygenerovat soubor **.csv* pro import mapování.

5.5 Quick Control 4

Následující část se věnuje softwaru QC4 (Quick Control 4), v kterém je zpracována vizualizace praktické částí. Tento software je vyvíjen společnosti Asyc a lze ho využít pro monitorování a řízení různých technologických procesů, které vyžadují rychlou odezvu systému a snadnou následnou údržbu. Konfigurace je založená na souborech



Obr. 5.7: Modbus TCP Slave - Nastavení komunikace

typu XML s vizualizací v podobě webové aplikace. QC4 je podporován na operačních systémech Linux i Windows. Mezi nejdůležitější vlastnosti patří [18]:

- Rekonfigurace bez nutností zastavení aplikace
- Archivace hodnot se zobrazením pomocí grafu
- Archivace a prohlížení událostí
- Přehled poruch a výstrah
- Možnost práce s věrohodností hodnoty
- Podpora protokolů:
  - SAIA S-Bus
  - Modbus TCP i RTU
  - IEC 62056
  - IEC 60870-5-104
- Komunikace s EPS (Elektrická požární signalizace), EZS (Elektrická zabezpečovací signalizace)

5.5.1 Instalace

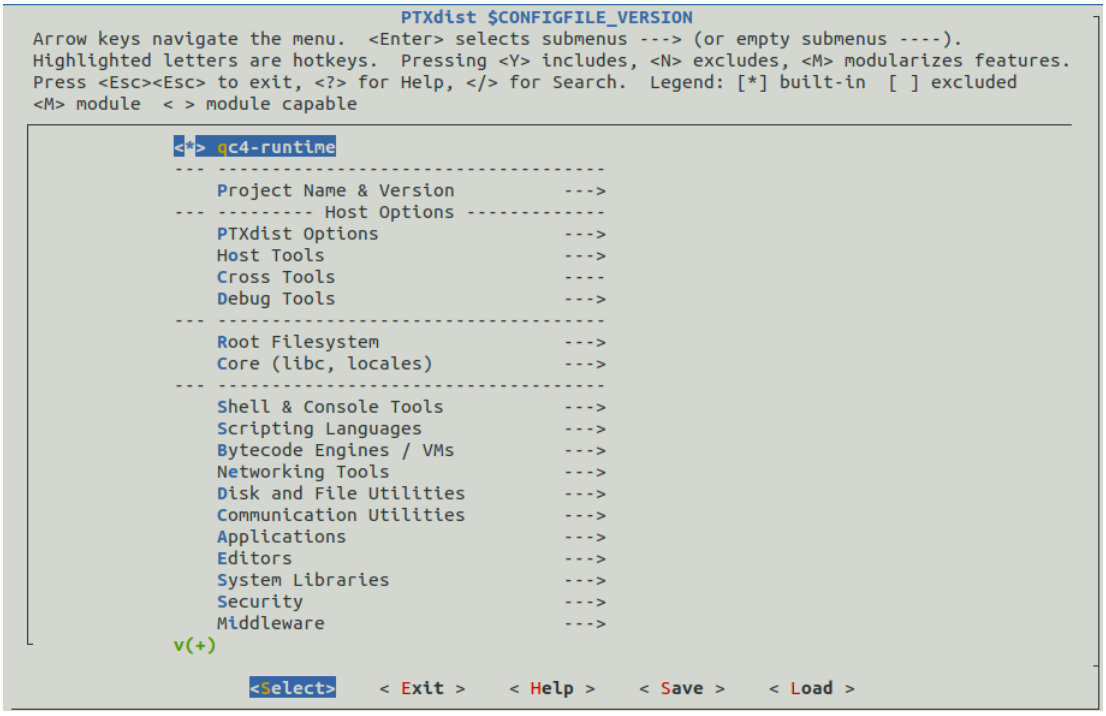
Před instalací softwaru Quick Control 4, je nutné nainstalovat všechny potřebné balíčky, které tento program vyžaduje pro správnou funkčnost. Hlavní chybějící částí je aplikační rámec Qt.

K vytvoření instalačních balíčku pro procesorové moduly PFC200 druhé generace nabízí firma Wago vlastní SDK. Toto SDK je volně k dispozici na GitHubu společnosti Wago [19]. Nově existuje verze pro kompilaci firmwaru verze 24 s vyu-

žitím softwaru Docker. V této aplikaci je použit firmware verze 23 a pro kompilaci byl použit systém s operačním systémem Ubuntu, jak je doporučeno v návodě. Další postup vychází z návodu dostupného z [19].

Na zvoleném systému je třeba nainstalovat Git a rozšíření pro podporu velkých souborů. Dalším krokem je instalace nástroje cross toolchainu pro vytvoření souborů spustitelných na jiné platformě než je prováděná kompilace. Dále se musí stáhnout, nakonfigurovat, zkompilovat a nainstalovat nástroj PTXdist.

Následuje konfigurace prostředí projektu, to je výběr softwarového nastavení, hardwarové platformy podle návodu. V dalším kroku se příkazem `ptxdist menuconfig` z adresáře `~/wago/ptxproj` spustí konfigurační menu pro výběr komponent, které se mají zkompilovat. Položka *qc4-runtime* je doplněná pomocí souborů *qc4-runtime.in* a *qc4-runtime.make* umístěných v adresáři `~wago/ptxproj/rules`. Tyto soubory definují i všechny softwarové závislosti, které mají být zkompilovány. Soubory jsem obdržel již plně funkční, protože vytvoření těchto souborů je mimo rozsah této práce.



Obr. 5.8: Config Menu nástroje PTXdist

Výběr položky *qc4-runtime* přidá do kompilace i všechny závislosti, které jsou potřeba pro instalaci QC4. Po zaškrtnutí této položky není potřeba v menu měnit nic jiného a je možné spustit kompilaci.

Po dokončení kompilace je v adresáři `/wago/ptxproj/platform-wago-pfcXXX/images/` obraz firmwaru s požadovaným softwarem, určeným pro instalaci pomocí SD karty.

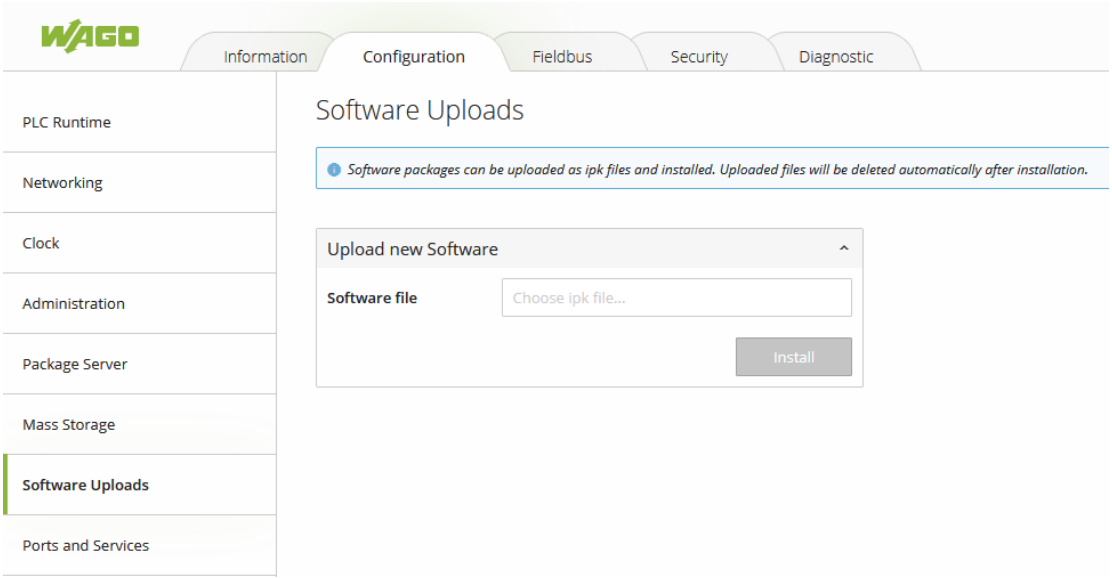
Druhou možností je nainstalovat pouze potřebné balíčky, které se nachází v adresáři `~/wago/ptxproj/platform-wago-pfcXXX/packages/`. Seznam balíčků pro instalaci QC4:

- qt5_5.15.0_armhf.ipk
- fontconfig_2.13.92_armhf.ipk
- freetype_2.10.1_armhf.ipk
- libdrm_2.4.110_armhf.ipk
- qc4-runtime_2.2.9_armhf.ipk

Protože jsem již dříve aktualizoval firmware, tak nyní mi postačuje nainstalovat pouze výše uvedené balíčky. To je možné dvěma způsoby:

- z internetového prohlížeče pomocí WBM (web-based management)
- připojením se k systému pomocí SSH a následnou instalací pomocí příkazové řádky

První varianta s využitím běžného internetového prohlížeče je určitě jednodušší. Po přihlášení do WBM v záložce *Configuration* a položce *Software Uploads* je možnost vybrat z počítače soubor následně nainstalovat. Vybrané soubory musí být ve formátu **.ipkg*.



Obr. 5.9: Instalace softwarových balíčku z WBM

V mém případě jsem zvolil druhou možnost a to připojení k systému pomocí SSH. Pro přenos potřebných balíčků do PFC200 jsem využil WinSCP klienta, ale lze využít i jiné SFTP klienty. Do systému jsem se připojil pomocí PuTTy SSH klienta a pomocí příkazu `opkg` nainstaloval připravené balíčky.

Po dokončení instalace vznikl v adresáři `/opt/QC4/` skript *qcservice*, který jsem zkopíroval do adresáře `/etc/init.d/`. Následně jsem v adresáři `/etc/rt.d` vytvořil

symbolický odkaz na zkopírovaný skript a pojmenoval ho **S98_qcservice**, aby se tato služba spustila po startu řídicího systému. Číslo 98 je shodné s číslem u skriptu, který spouští Codesys runtime a určuje pořadí spouštění skriptů po startu systému. Služba *qcservice* podporuje standardní příkazy *start, stop, restart* a *status*. Manuální spuštění služby je možné následujícím příkazem:

```
service qcservice start
```

Následně je možné nahrát konfiguraci QC4 aplikace do řídicího systému. Konfigurace, které se věnuje následující část práce, se nachází v adresáři `/home/qcservice/` a její adresářová struktura vypadá následovně:

```
/ ..... Kořenový adresář
└── home
    └── qcservice ..... Adresář aplikace QC4
        ├── app ..... Adresář obsahující konfiguraci QC4 aplikace
        │   ├── html ..... Adresář obsahující soubory pro vizualizaci
        │   │   ├── img ..... Adresář obsahující statické SVG soubory
        │   │   ├── *.js
        │   │   ├── *.css
        │   │   ├── *.xhtml
        │   │   ├── *.hmtl
        │   │   └── *.svg
        │   └── *.xml ..... XML konfigurační soubory
        ├── cache
        ├── data ..... Symbolický odkaz na úložiště SD karty (/media/sd/)
        │   └── qc4-data ..... Adresář s databází historických hodnot
        └── qcservice.xml ..... Základní konfigurační soubor runtime
```

Dále se v adresáři `/home/qcservice/` nachází různé soubory s ladicími výstupy, ale ty najdou uplatnění pouze při ladění chyb aplikace a nejsou pro provoz nijak důležité.

Vnitřní paměť systému má omezenou kapacitu a tak jsem se rozhodl ukládat historická data na SD kartu. Před použitím je nutné na kartě vytvořit souborový systém. To lze provést pomocí WBM v záložce *Configuration* se nachází položka *Mass Storage*. K vytvoření souborového systému slouží okno *Create new Filesystem on Memory Card*, které obsahuje dva parametry. Typ souborového systému, na výběr je z Ext4 a FAT, a označení SD karty, které je po úspěšném vytvoření souborového systému zobrazeno v okně *Device* jako Volume name.

5.5.2 Konfigurace

Základem každé QC4 aplikace je soubor runtime konfigurace. V tomto souboru jsou definovány všechny použité applety a jejich příslušné konfigurační soubory. Na následujícím výpise je runtime konfigurace realizované aplikace. V další částí této kapitoly budou popsány použité applety.

Výpis 5.3: Runtime kofigurace

```
<?xml version="1.0" encoding="UTF-8"?>
<runtime xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="urn:asyc:runtime"
    xsi:schemaLocation="urn:asyc:runtime qrc:/asyc/runtime.xsd"
    id="runtime" trace-level="3">
  <applet id="pfc200" plugin="QcModbus" class="QcModbus" config="
   pfc200.xml" trace-level="3"/>
  <applet id="asyc" plugin="QcCore" class="ExpressionServer" config
   ="asyc_expression.xml" trace-level="3"/>
  <applet id="http" plugin="QcHTTPServer" class="QcHTTPServer"
   config="http.xml" trace-level="3"/>
  <applet id="trendserver" plugin="QcTrendServer" class="
   TrendServer" config="trendserver.xml" trace-level="3"/>
  <applet id="perm" plugin="QcCore" class="ExpressionServer" config
   ="perm.xml" trace-level="3"/>
  <applet id="perm_admin" plugin="QcCore" class="ExpressionServer"
   config="perm_admin.xml" trace-level="3"/>
</runtime>
```

Applety se definují elementem *applet*. Pro každý applet se vytvoří DaServer s názvem definovaným atributem *id*. V tomto DaServeru budou všechny hodnoty definované v daném appletu. Pro testovací účely lze použít element *no-applet* a tím odstranit vybraný applet z konfigurace.

QcModbus

V současné době ještě neexistuje applet pro výměnu dat QC4 runtime a Codesys runtime. Protože jsem v Codesys aplikaci již implementoval komunikaci Modbus TCP, rozhodl jsem se pro výměnu dat mezi QC4 a Codesys s využitím tohoto protokolu.

Další zvažovanou možností výměny dat, byla výměna pomocí souboru. Do tohoto souboru bych zapisoval potřebné hodnoty i s časovou značkou. Pro čtení souborů existuje applet *FileParser*, který prochází zadaný soubor a pomocí regulárních výrazů hledá požadované hodnoty.

Applet *QcModbus* umožňuje komunikaci jak protokolem Modbus TCP tak i po sériové lince, kde je možnost komunikovat v režimu RTU i ASCII. Na následujícím výpise je použitá konfigurace s doplněným příkladem komunikace pomocí sériové linky. V jednom appletu může být pouze komunikace *TCP* nebo *serial*.

Výpis 5.4: Konfigurace *QcModbus* appletu

```
<?xml version="1.0" encoding="UTF-8"?>
<QcModbus
  xmlns="urn:asyc:QcModbus"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:asyc:QcModbus qrc:/asyc/QcModbus/QcModbus
   .xsd"
>
  <devices roundTime="3000" roundTimeItem="roundTime">
    <device daServer="pfc200" address="0" config="pfc200_data.xml"/
   >
  </devices>
  <connection failureItem="komPor">
    <controller storeItem="data:QcModbus/connectionBlocked"
   controlItem="connectionBlocked"/>
    <TCP ip="10.90.1.80" port="502"/>
  </connection>
</QcModbus>
```

Uvnitř elementu *devices* se elementem *device* definuje zařízení s kterým se bude komunikovat. Atributem *daServer* se definuje jméno pro DaServer vytvořený pro dané zařízení. V případě, že není tento atribut definován, sdílí všechna zařízení jeden DaServer a je vhodné jednotlivé zařízení rozlišit pomocí atributu *prefix*.

Zdánlivě nejdůležitějším atributem je *config*, který definuje cesty k souboru s definicí všech komunikovaných hodnot. Příklad této konfigurace je ve výpise 5.5. Komunikovat lze uchovávací registry (HR), vstupní registry (IR), diskrétní vstupy (DI) a cívky (COIL). Adresa u 32 bitových hodnot uchovávacích a vstupních registrů (IR) udává vždy adresu prvních 16 bitů. Dalších 16 bitů se čte/zapisuje z/do další postupné adresy. Atributem *access* se určuje možnost čtení, zápisu nebo čtení i zápisu do hodnoty.

Výpis 5.5: Příklad konfigurace hodnot komunikovaných *QcModbus* appletem

```
<?xml version="1.0" encoding="UTF-8"?>
<QcModbus_data
  xmlns="urn:asyc:QcModbus"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:asyc:QcModbus qrc:/asyc/QcModbus/
   QcModbus_devices.xsd"
>
  <comFailure id="komPor"/>
  <COIL id="PregAut" address="0" access="R/W"/>
  <HR type="float" id="P15M01" address="0" access="R/W"/>
  <IR type="float" id="P+" address="0"/>
</QcModbus_data>
```

Expression server

Applet *ExpressionServer* umožňuje provádět různé přepočty hodnot. V rámci této aplikace je využit pro uchování dat v rámci jednoho DaServeru, výpočet hodnot, které nejsou komunikovány protokolem Modbus a pro zápis hodnot z vizualizace.

Důležitým atributem při definování je *saveDir*, který definuje cestu pro uložení perzistentních dat. Pokud atribut není definován, data nejsou uložená.

Výpis 5.6: Příklad konfigurace položek *ExpressionServeru*

```
<?xml version="1.0" encoding="utf-8"?>
<expressions xmlns="urn:asyc:QcCore:ExpressionServer"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:asyc:QcCore:ExpressionServer qrc:/asyc/
   QcCore/ExpressionServer.xsd"
 saveDir="data" addTs="true">
  <alarm id="komPor" source="pfc200:komPor"/>
  <alarmNC id="komPor/NC"/>
  <expression id="ELM/P+" source="pfc200:P+" type="float"/>
  <expression id="ELM/P15M01" source="pfc200:P15M01" target="pfc200
   :P15M01" type="float"/>
  <expression id="ELM/T15" source="pfc200:T15" expr="arg1/1000"
   type="int"/>
  <variable id="ELM/var" type="string"/>
</expressions>
```

Pomocí appletu *ExpressionServer* lze definovat několik typů hodnot, mezi ty nejdůležitější patří *expression*, který čte hodnotu z položky definovanou atributem *source*. Zápis zdrojové položky se provádí ve formátu <DaServer>:<id>. Při zápisu do této položky, například z vizualizace, zapíše tuto hodnotu do položky definované atributem *target*. Pokud však není tento atribut definován je zápis ignorován. Atribut *type* určuje datový typ položky a pomocí *expr* se čtená hodnota přepočítá podle uvedeného vzorce. Jako vzorec může být jakýkoliv výraz v jazyce *JavaScript*. K prvnímu zdrojovému prvku se přistupuje pomocí proměnné *arg1*, k druhému *arg2*, atd.

Dalším typem je *variable*, taková položka se chová jako normální proměnná, kterou je možné číst, zapisovat do ní nebo zobrazovat, nepodporuje atributy *source, target, expr*. Iniciální hodnotu určuje atribut *value*.

Element *alarm* vytvoří položku typu BOOL, která reprezentuje určitý typ alarmu. Tato hodnota je společně s hodnotou *alarmNC* následně použitá v appletu *AlarmController*. Položka *alarm* reprezentuje aktuální stav alarmu a *alarmNC* reprezentuje stav kvitace vybraného alarmu.

Pro vytvoření konstanty se používá element *const*, který má povinné atributy *value, type*. Zápis do této položky není možný.

TrendServer

Dalším využitým appletem je *TrendServer*, který slouží pro ukládání průběhů jednotlivých hodnot. Konfigurační soubor je v příloze A.

Prvním krokem při konfiguraci je nastavení počtu signálů a alokace paměti. K tomu slouží atributy *item-count, page-count* a *page-size*. *Item-count* označuje maximální počet položek, které může server zaznamenávat. Atribut *page-size* definuje velikost stránky pro sdílenou paměť i pro uložení do souboru, v rozmezí 64 B až 64 kB. Atributem *page-count* se následně definuje počet těchto stránek, které se mají alokovat.

Dalším důležitým elementem je *sql-connection* pro konfiguraci spojení s databází. Atribut *name* specifikuje jméno datového zdroje, *type* určuje typ databáze a posledním atributem je *database*, který určuje jméno vytvořené databáze. Uvnitř elementu *sql-connection* se ještě definuje elementem *tables* názvy tabulek pro uložení filtrů, meta dat, identifikátorů a tabulky s cestami k datům. Element *FiltersTableExt* definuje aplikační filtry, kterých hodnoty se následně definují pro jednotlivé signály a umožňují podle těchto hodnot signály filtrovat.

Konfigurace dočasného úložiště neboli cache se provádí elementem *cache*, který obsahuje několik vnitřních elementů. Prvním je *save-no-full-page* pro nastavení času, kdy se uloží i nekompletní stránka. Element *cache-interval* definuje minimální počet volných stránek a dobu, po které se mají přesunout do trvalého úložiště. Element *index-file* nastavuje ukládání indexu.

Datového úložiště má vždy určité limity a to hlavně na malých systémech jako PFC 200. Z tohoto důvodu je součástí konfigurace i nastavení mazání dat. Elementem *eraser* se toto mazání nastaví. Atribut *strart-delay-s* určuje zpoždění v sekundách pro jednodušší start serveru, *interval-ms* určuje dobu v milisekundách mezi pokusy o smazání souboru a poslední atribut je *control-interval-h*, který určuje interval kontrol datového adresáře. Elementem *storage* uvnitř *eraser* se definuje doba ukládání dat. Tento element obsahuje dva atributy. *Days* definuje počet dnů datového záznamů, které mají být uloženy a *delete-days* definuje kolik dnů záznamů má být smazáno najednou.

Poslední částí konfigurace je nastavení DaServerů a jejich položek, které se mají zaznamenávat. K tomu slouží element *recorders*, do kterého se vkládají elementy *recorder* pro každý DaServer, z kterého se budou zaznamenávat hodnoty. *Recorder* obsahuje dva atributy. Prvním je *server*, který udává jméno DaServeru a *config* udávající cestu k souboru s souboru s jednotlivými položkami, které se mají zaznamenávat. Příklad s jednou hodnotou je uveden na výpisu 5.7.

Výpis 5.7: Konfigurace zaznamenávaných položek v *TrendServeru*

```
<?xml version="1.0" encoding="utf-8"?>
<TrendRecorder
 xmlns="urn:asyc:history:trendrecorder"
 xmlns:f="urn:asyc:history:filter"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:asyc:history:trendrecorder qrc:/asyc/
   history/trendrecorder.xsd"
>
  <ITEM id="ELM/P+" f:label="ELM P+" f:desc="Činná energie odběr"
   type="float" format="##.#" min="0" max="10" units="kW"/>
</TrendRecorder>
```

Každá položka je definována elementem *ITEM*. Identifikátor hodnoty v rámci DaServeru je definován atributem *id*. Atributy s prefixem *f:* jsou hodnoty pro aplikační filtry, zbylé atributy se používají při zobrazování hodnot pro popis signálu a správné nastavení os.

Procesorový modul PFC200 obsahuje pouze 4096 MB interní paměti, proto jsem se rozhodl využít možnost rozšíření paměti SD kartou a historická data ukládat právě tam. Maximální podporovaná kapacita SD karty je 32 GB. Protože SD karty této kapacity jsou cenově velmi dobře dostupné, rozhodl jsem se maximální podporovanou velikost využít.

LogServer

Dalším appletem pro ukládaní dat je *LogServer*. Applet sleduje změny nakonfigurovaných položek a tyto změny zapisuje do databáze. Konfigurace je velice podobná konfiguraci *TrendServeru*.

Na rozdíl od *TrendServeru* není třeba nastavovat alokaci paměti, ale pouze spojení s databází elementem *sql-connection*. V tomto elementu je možné definovat i počet dnů, po který mají být uchovávány data v databázi a to atributem *max-log-days*. Atributem *detele-at* je možné nastavit v kolik hodin proběhne smazání starých dat. Tyto atributy jsou nepovinné pokud není zapotřebí měnit výchozí hodnoty, které jsou 365 dní a čas 3:00:00.

Aplikační filtry se nedefinují uvnitř *sql-connection*, ale mimo a to elementem *filters* uvnitř, kterého se vytváří elementy *column* s jediným atributem *name*.

Elementy *item-run, item-err, item-lost* slouží pro konfiguraci zaznamenávání systémových událostí *LogServeru*.

Poslední částí konfigurace je opět definování všech DaServerů, z kterých se budou zaznamenávat změny signálů. K tomu slouží opět *recorders* a jeho vnitřní element *recorder* pro každý DaServer.

Definování jednotlivých položek je obdobné jako ve výpisu 5.7, ale navíc je možné uvést atributy *logtype* (typ zaznamenávaného signály při změně stavu) a *logtypeW* (typ zaznamenávaného signálu při povelu) a neuvádí se atributy *min, max, units*.

AlarmSum

Následujícím použitým appletem je *AlarmSum*, který má za úkol vytvořit agregované stavy alarmů pro klienta a současně respektovat aktuální oprávnění přihlášeného uživatele. Tento applet je nutné nakonfigurovat pro každou skupinu oprávnění, která je definována jak identifikátorem appletu v aplikační konfiguraci tak atributem *perm* elementu *AlarmClient*, který musí mít hodnotu **perm_<název oprávnění>**. Identifikátor appletu se tvoří obdobným způsobem pouze s předponou **alarm_**.

Výpis 5.8: Konfigurace appletu *AlarmSum*

```
<?xml version="1.0" encoding="utf-8"?>
<AlarmClient
  xmlns="urn:asyc:history:alarmclient"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:asyc:history:alarmclient qrc:/asyc/
   history/alarmclient.xsd"
  perm="perm"
>
  <alarms>
    <items server="asyc" config="asyc_alarm.xml"/>
  </alarms>
</AlarmClient>
```

Konfigurace appletu obsahuje pouze element *alarms* s elementy *items* pro každý DaServer, v kterém se nachází hodnoty, které chci reprezentovat jako alarm. Atribut *server* označuje jméno DaServeru a *config* cestu k souboru s konfigurací jednotlivých skupin alarmů, jako je tomu na výpisu 5.9.

Výpis 5.9: Konfigurace alarmů a alarmových skupin appletu *AlarmSum*

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="urn:asyc:history:alarmdata" xmlns:f="urn:asyc:history:
   filter"
 xsi:schemaLocation="urn:asyc:history:alarmdata qrc:/asyc/history/
   alarmdata.xsd"
>
<GROUP id="SUMA">
  <ITEM id="komPor" f:label="komPor" f:desc="Porucha komunikace
   Modbus" logtype="porucha"/>
```

```
8   <ITEM id="ELM/P15M/H" f:label="ELM P15M H" f:desc="Predikce spotř
      eby v aktuální čtvrthodině - překročení rezervované kapacity"
      logtype="vystraha"/>
9   </GROUP>
10  </data>
```

Protože v realizované aplikaci jsou pouze dva alarmy, vytvořil jsem pouze jedinou skupinu, ale je možné vytvořit i více skupin. Atributy začínající **f:** můžou být použity pro vlastní sloupce a je ještě nutné definovat je v ovladači *AlarmController* jako elementy *column*. Tento ovladač je součástí appletu *QcHTTPServer*, který bude popsán v další části. To neplatí o atributech *f:read* a *f:write*, které určují oprávnění pro čtení resp. zápis.

QcHTTPServer

Applet *QcHTTPServer* obshauje HTTP server, který zpracovává celou QC4 aplikaci a následně je možné tuto aplikaci otevřít v běžném webovém prohlížeči. Konfigurace appletu je v příloze B.

Element *listener* konfiguruje možností pro příchozí spojení. Hlavním atributem je *port*, kde je spuštěná aplikace. Dalšími atributy jsou:

- readTimeout - čas v *ms*, kde musí dojít k přečtení celého požadavku
- minThreads, maxThreads - minimální resp. maximální počet vláken vytvořených pro obsluhu požadavků
- cleanupInterval - interval v *ms*, kdy dochází k pokusu o úklid nepoužívaných vláken
- enableCompression - nastavení komprese odpovědi

Element *log* nastavuje záznam požadavků a jeho atributy lze nastavit cestu souboru, do kterého se budou záznamy ukládat a maximální velikost souboru. Vnitřní elementy *path* nastavují filtry, pro záznam požadavků. Ukládají se ty požadavky, které splní minimálně jeden z vypsaných filtrů.

Důležitou částí je konfigurace použitých ovladačů, která definuje jaké funkce bude obsahovat výsledná vizualizace. K definování ovladačů se používá element *map*, obsahující následující atributy:

- path - požadavky začínající uvedenou cestou jsou zpracovány příslušným ovladačem
- id - identifikátor ovladače
- plugin - plugin z kterého se má ovladač nahrát
- class - název ovladače
- config - cesta ke konfiguračnímu souboru ovladače

Ve vytvořené aplikaci jsou použity následující ovladače:

- DebugController - zpřístupňuje stavy appletů a umožňuje se přihlásit k dalším ladícím nástrojům
- LogController - zpřístupňuje databázi deníku událostí pro klienta
- AlarmController - zpřístupňuje stavy alarmů, jejich kvitaci a maskování, zapisuje hodnoty přímo do zdrojových dat v *ExpressionServeru* (položky *alarm* a *alarmNC*)
- TrendController, TableController - zpřístupňují historická data signálu z databází pro klienta, pro správnou funkčnost jsou potřeba oba ovladače
- AuthController - slouží pro správu oprávnění ve vizualizaci
- AppController - zpřístupňuje soubory na disku

Konfigurace jednotlivých ovladačů je výrazně jednodušší než konfigurace appletů a nachází se v příloze C.

DebugController obsahuje pouze hlavní element *debugcontroller* s atributy *password*, který obsahuje heslo pro přístup a *maxLogSize*, který udává maximální počet záznamů změn jedné hodnoty.

Konfigurace *LogControlleru* obsahuje pouze nastavení spojení s databázi deníku. Nastavení se provádí elementem *sql-connection* a je shodné s nastavením appletu *LogServer*. Podobně jsou na tom ovladače *TrendController* a *TableController*, které mají v konfiguraci pouze nastavení spojení s databází. V obou případech je nastavení shodné s appletem *TrendServer*.

AlarmController využívá obdobnou konfiguraci jako applet *AlarmSum*, ale navíc může obsahovat elementy *column* uvnitř elementu *columns*. Tyto elementy slouží pro definici vlastních hodnot, podle kterých lze ve vizualizaci filtrovat alarmy a taky je doplnit do zobrazení. Každý element *column* musí obsahovat atribut *name* definující jeho jméno, podle kterého je následně použit v konfiguraci položek appletu *AlarmSum*, jak je tomu ve výpisu 5.9.

Výpis 5.10 zobrazuje konfigurace ovladače *AuthController*. Hlavní element *authcontroller* obsahuje dva důležité atributy. Prvním je *permServerBaseId*, který definuje předponu názvu appletů s oprávněními. Druhým atributem je *sessionContextServers*, který definuje předponu názvů appletů *AlarmSum*. Uživatelé jsou definování elementem *login*, který obsahuje atributy:

- user - přihlašovací jméno uživatele
- password - heslo uživatele
- permGroup - oprávnění uživatele po přihlášení

Skupiny oprávnění jsou definovány v runtime konfiguraci, která je zobrazená na výpisu 5.3, jako applety *ValueServer*.

Výpis 5.10: Příklad konfigurace ovladače *AuthController*

```
<?xml version="1.0" encoding="utf-8"?>
<authcontroller
  xmlns="urn:asyc:QcHTTPServer:authcontroller"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:asyc:QcHTTPServer:authcontroller qrc:/
   asyc/QcHTTPServer/authcontroller.xsd"
  permServerBaseId="perm"
  sessionContextServers="alarm"
>
  <autologin host="192.168.1.0/24" permGroup="servis"/>
  <login user="servis" password="servis" permGroup="servis"/>
  <login user="obsluha" password="1234" permGroup="ovl"/>
</authcontroller>
```

Ovladač umožňuje i možnost automatického přihlášení z definovaných adres. Automatické přihlášení se vytváří elementem *autologin* s atributy *host*, určující IP adresu uživatele a *permGroup*, který jako v předchozím případě určuje skupinu oprávnění přidělenou po přihlášení. Do atributu *host* lze zadat jednu adresu nebo rozsah adres. V případě, že adresa se shoduje s více položkami *autologin* prioritu má dříve nadefinovaná.

Posledním využitým ovladačem je *AppController*, který zpřístupňuje soubory na disku pro webového klienta. Do souborů **.xhtml* a **.svg* před odesláním dosadí živá data a dále zpracovává dotazy pro změnu živých dat (getData) a nastavování hodnot (setData) v DaServerech. Důležitým atributem, který obsahuje hlavní element *appcontroller*, je *dataPath*. Jedná se o cestu v souborovém systému, kterou je nahrazena cesta definovaná při konfiguraci ovladače v HTTP dotazu. Atribut *directoryIndex* udává adresu, která je vrácená pokud ovladač přijme dotaz na kořenový adresář ("/").

Výpis 5.11: Konfigurace ovladače *AppController*

```
<?xml version="1.0" encoding="utf-8"?>
<appcontroller
  xmlns="urn:asyc:QcHTTPServer:appcontroller"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:asyc:QcHTTPServer:appcontroller qrc:/asyc
   /QcHTTPServer/appcontroller.xsd"
  dataPath="html" directoryIndex="index.xhtml"
>
  <permissions file="" use="false"/>
  <script-engine>
    <script>
      function qualityIsGood(argq1){
        return (argq1 &amp; (QUALITY_LAST | QUALITY_INIT |
```

```
    QUALITY_BAD))==0;
13      }
14    </script>
15  </script-engine>
16 </appcontroller>
```

Uvnitř elementu *appcontroller* je možné elementem *permissions* definovat oprávnění pro jednotlivé položky všech DaServerů. Tuto možnost jsem, ale v této aplikaci nevyužil. Dále je možnost elementem *script-engine* definovat vlastní funkce používané ve výrazech v živých obrazovkách.

5.5.3 Vizualizace

Vizualizace je realizována formou webové aplikace, která se připojí na HTTP server vytvořený appletem *QcHTTPServer* a výchozí stránkou aplikace je soubor *index.xhtml*. Tento server umožňuje v obrazovkách použití tří elementů z jmenného prostoru qc:

- **qc:display** - umožňuje skrývat některé elementy podle aktuálního oprávnění, atribut *accessList* určuje seznam oprávnění, které musí mít přihlášený uživatel, vnitřní elementy obsahují atribut *qc:access* s hodnotou TRUE nebo FALSE, který určuje logiku, kdy budou tyto elementy zobrazeny, použití *qc:display* elementu pro skrývání tlačítka je zobrazeno ve výpisu 5.12
- **qc:value** - slouží k výpočtu za základě hodnot DaServerů, obsahuje následující atributy:
  - target - atribut rodičovského elementu, kde se zapíše výsledek
  - source - zdrojové hodnoty pro výpočet `<DaServer>:<Id>`, při zápisu více hodnot se zdroje oddělují čárkou
  - expression - výraz v jazyce JavaScript pro úpravu zdrojových hodnot
  - format - formátovací řetězec

  Příklad použití elementu *qc:value* pro aktualizaci zobrazované hodnoty analogu je ve výpise 5.13.
- **qc:select** - element podobný *qc:value*, ale je zpracováván na klientu a ne na serveru, atribut *source* se neodkazuje na položky z DaServerů, ale položky v Session Storage, oproti *qc:value* může navíc obsahovat:
  - targetId - identifikátor cílového elementu
  - startsWith - pokud zdrojová hodnota začíná tímto řetězcem je cílovému prvku přiřazena třída v atributu *class*
  - class - CSS třída přiřazená cílovému prvku

  Výpis 5.12 zobrazuje příklad použití *qc:select*, pro barvení tlačítka přepínajícího obrazovky, pokud je zobrazována obrazovka, na kterou tlačítko přepíná.

Výpis 5.12: Použití elemetů qc:display a qc:select

```
<qc:display accessList="ovladani">
  <button qc:access="true" class="menuButton" id="ovladani" onclick
   ="qc.setVariable('app:w2','ovladani.svg');">
       Ovládání
       <qc:select source="app:w2" startsWith="ovladani.svg" class=
   "selectedButton"/>
    </button>
</qc:display>
```

Výpis 5.13: Použití elemetu qc:value

```
<text id="ELM#P+_val" x="290" y="10" style="fill:#00008B; font-size
   :11pt;"font-weight="bold">
 0.0 kW
  <qc:value target="innerHTML" source="exps:ELM/P+" format="##.# kW
   "/>
</text>
```

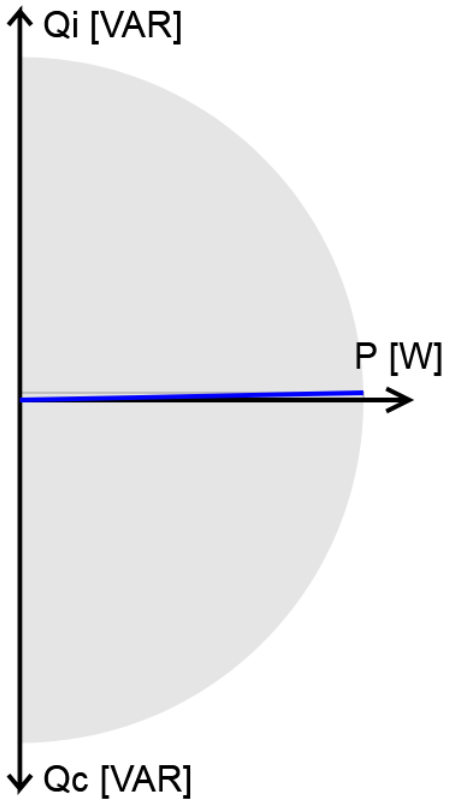
Vizualizace realizované aplikace je složená z celkem osmi obrazovek. Mezi obrazovkami lze přepínat pomocí menu v levé částí obrazovky. K vizualizaci lze přistoupit z jakéhokoliv počítače ve stejné síti, pomocí internetového prohlížeče zadáním IP adresy a nakonfigurovaného portu *QcHTTPServeru*.

Úvodní obrazovkou je Spotřeba, která zobrazuje všechny měřené i počítané hodnoty. Na všechny výstupní analogové hodnoty je možné kliknout a zobrazit tak okno s průběhem hodnoty za poslední den. Aktuální odběr je zobrazen jak číselně tak i v grafické podobě (obr. 5.10), z které lze lehce vyčíst poměr mezi činným a jalovým výkonem. Data na tomto obrázku zobrazují naměřený odběr mikrovlnné trouby.

Druhou obrazovkou je obrazovka zobrazující průběh průměrné čtvrthodinové spotřeby a predikovanou hodnotu spotřeby za čtvrthodinu. Tento graf je zobrazen na obrázku 5.11. Modrý průběh zobrazuje průměrný odběr a šedý signál znázorňuje predikovanou hodnotu odběru za aktuální čtvrthodinu. Horní mez osy y je nastavena podle aktuálního sjednaného maxima odběru. Hodnoty maxim lze nastavit na obrazovce Nastavení.

Další obrazovkou je Ovládání, kde je možné přepínat mezi automatickým a ručním režimem odpojování spotřebičů a taky odpojit požadované spotřebiče. K přístupu a ovládání je nutné mít dostatečné oprávnění, to lze získat přihlášením z obrazovky Přihlášení.

Zvýšenou úroveň oprávnění vyžaduje i obrazovka Nastavení. Tato obrazovka je přístupná všem uživatelům, ale možnost úprav je pouze pro uživatelé s oprávněním pro zápis. Na obrazovce lze nastavit maximální čtvrthodinový odběr pro každý

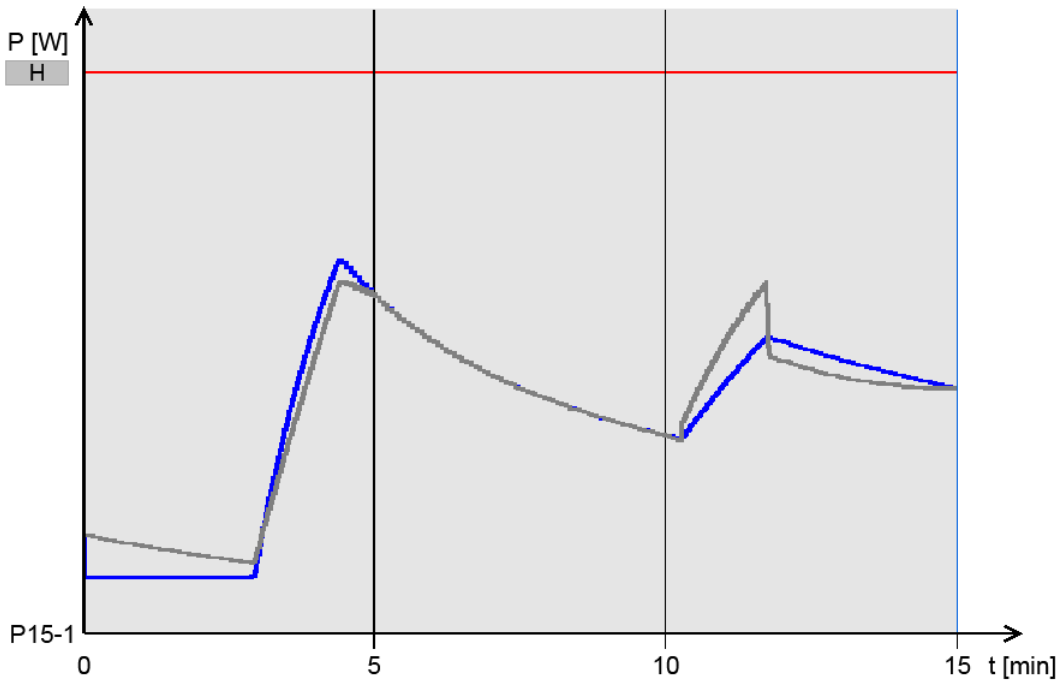


Obr. 5.10: Grafické znázornění aktuálního odběru

měsíc, příkon jednotlivých stupňů odpojovaných spotřebičů a celkové stavy odběru a dodávky elektrické energie.

Následující obrazovkou jsou Trendy, zobrazující průběh vybraných hodnot v čase. Ve výchozím nastaví je zobrazen průběh aktuálního odběru a spotřeby v aktuální čtvrthodině. Zobrazené signály je možné měnit tlačítkem "Přidat signál" nebo dvojitým kliknutím do grafu. Signálům, které mají stejnou jednotku, lze nastavit společnou osu y. Kliknutím pravým tlačítkem myši na vybraný signál v tabulce ve spodní částí obrazovky a výběru *Synchronizovat rozsah* a následně stačí vybrat signál podle, kterého má být osa synchronizovaná.

Následuje obrazovka alarmů, která zobrazuje záznam všech uplynulých alarmů. Ve výchozím zobrazení se u jednotlivých alarmů zobrazuje čas vzniku, informace jestli alarm aktivní a kvitovaný, čas kvitace a typ alarmu. Proto jsem v konfiguraci ovladače *AlarmController* definoval vlastní sloupce, které výchozí zobrazení doplní o důležité informace.



Obr. 5.11: Graf vývoje predikované spotřeby (šedý průběh) a průměrné spotřeby (modrý průběh) v aktuální čtvrthodině

Výpis 5.14: Nastavení vlastích sloupců v souboru *alarm.html*

```
alarmConfig = {
    ...
    additionalColumns: [
      { field: "label", insertBefore: "active", textFilter: true },
      { field: "desc", insertBefore: "active", textFilter: true }
    ],
    ...
}
```

Doplnění těchto vlastních sloupců do tabulky s alarmy se provádí v souboru *alarm.html*. Příklad je zobrazen na výpisu 5.14. Objekt *alarmConfig* obsahuje vlastnost *additionalColumns*, která definuje vlastní sloupce. Tato vlastnost obsahuje pole objektů, kde každý objekt tvoří jeden sloupec. Objekt sloupce obsahuje následující vlastnosti:

- field - identifikátor sloupce, musí být shodný s názvem v *AlarmControlleru*
- label - označení sloupce, není povinný
- insertBefore - identifikátor již existujícího sloupce, před který bude nový sloupec vložen, pokud není vyplněn je vložen na konec
- textFilter - pokud je TRUE je sloupec použit pro textové vyhledávání

Poslední obrazovkou s historickými daty je Deník, který formou tabulky zobra-

zuje změny hodnot nakonfigurovaných v appletu *LogServer*. V tomto appletu jsem opět nadefinoval vlastní sloupce a ty je potřeba přidat do vizualizace. Postup je obdobný jako u vlastních sloupečků u alarmů. V souboru *denik.html* do objektu *logConfig* a jeho vlastnosti *additionalColumns* jsem doplnil pole s objekty pro každý sloupec, stejně jako ve výpisu 5.14.

Poslední obrazovkou je přihlašovací obrazovka, na které je možné se přihlásit do účtu s vyšším oprávněním a provádět tak operace, které jsou jinak blokovány.

Obr. 5.12: Přihlašovací dialog

Aktuální úroveň oprávnění je zobrazená nad tlačítky pro přihlášení/odhlášení. Existují tři úrovně oprávnění:

1. Host - nejnižší úroveň oprávnění, umožňuje pouze prohlížení
2. Ovládání - střední úroveň oprávnění umožňující ovládání režimu odpojování spotřebičů
3. Servis - nejvyšší úroveň oprávnění, je povolen zápis v obrazovce nastavení

Všechny výše popsané obrazovky se nachází v příloze D.

6 Zhodnocení výsledků

V rámci diplomové práce jsem provedl výběr vhodné open-source platformy jako alternativa ke klasickým PLC systémům. Do výběru jsem zařadil řídicí sytémy PFC200 od firmy Wago, řadu Unipi Patron od firmy Unipi a RevolutionPi od firmy Kunbus.

Jako nejvýhodnější platforma podle požadavků praktické úlohy, nákladového hodnocení a vývojového prostředí byla vybrána řada řídicích systému Unipi Patron. Hlavním důvodem je nízká pořizovací cena a software dodávaný ke každému zakoupenému systému, takže není potřeba kupovat další licence.

Cílem praktické úlohy bylo měřením spotřeby elektrické energie, predikce čtvrthodinové spotřeby a následná regulace, která je prováděná blokováním chodu spotřebičů. Úloha byla realizována na řídicích systémech od firmy Wago, které jsem měl již k dispozici. Velkou výhodou této platformy je široká nabídka rozšiřujících modulů a snadno tak lze vytvořit hardware na míru požadované aplikaci. K měření spotřeby jsem využil digitální vstupy, na které jsou připojené impulsní výstupy elektroměru. Procesorový modul komunikuje s rozšiřujícími moduly pomocí vnitřní sběrnice KBUS. Důsledkem toho vzniká při měření intervalu mezi pulsy významná chyba, která se přenese do výpočtu okamžitého odběru (viz obrázek 5.2). Jedním možným řešením je od určité frekvence pulsů počítat průměrný interval několika pulsů místo jednoho intervalu. Dalším řešením je snížit nastavenou periodu sběrnice KBUS a tím snížit chybu měření času, ale snížený čas při větším počtu vstupních a výstupních modulů nemusí být dodržován.

Experimentálně jsem k měření spotřeby elektrické energie využil i externí digitální vstupy v režimu čítače, přípojené k řídicímu systému rozhraním Ethernet protokolem Modbus TCP. Nepřesnost v měření času mezi pulsy, způsobena komunikací protokolem Modbus TCP, znemožnila použít toto řešení ve finální verzi aplikace.

Dále jsem zpracoval program pro výpočet predikované čtvrthodinové spotřeby a následné odpínání spotřebičů, který je podrobně popsán v kapitole 5.3.

Vizualizace je zpracována v programu QC4, kterému se věnuje kapitola 5.5. Původně jsem zamýšlel pro vizualizaci využít WebVisu z prostředí Codesys. Při testování této možnosti jsem zjistil, že výsledná vizualizace není příliš responzivní. Například při změně obrazovky se změna provedla až po několika vteřinách od stisknutí tlačítka. Tento problém jsem se snažil odstranit lepším nastavením úlohy, která volá WebVisu, ale bohužel neúspěšně. Velká odezva je pravděpodobně způsobená nedostatečným výpočetním výkonem. Při spuštěném Codesys runtimu, který obsahuje relativně malou aplikaci, využívá tento proces až 65 % výkonu procesoru.

QC4 aplikace se konfiguruje pomocí XML souborů, použitá konfigurace je popsána v kapitole 5.5.2 a celá konfigurace je k dispozici v přílohách diplomové práce na přiloženém paměťovém médiu.

Mezi nedostatky bych zařadil bezpečnost přihlašování do vizualizace, protože data jsou odesílaná nezabezpečeným protokolem HTTP a hrozí nebezpečí odposlechu a navíc jsou přihlašovací údaje uživatelů v konfiguračním souboru jako prostý text. Bohužel se jedná a systémovou záležitost, kterou nemůžu ovlivnit. Neoprávněný přístup do aktuální vizualizace může mít za následek pouze neoprávněnou změnu několika hodnot, ale v případě rozšíření aplikace o další částí by následky mohly být horší.

Jako další nedostatek lze uvést omezenou podporu mobilních zařízení. Vizualizaci je sice možné v mobilním telefonu jednoduše otevřít, ale nepřizpůsobí se zařízením s menší obrazovkou.

Výhodou využití systému s operačním systémem Linux je možnost nahradit aplikaci QC4 i Codesys runtime jiným dostupným softwarem. Použití jiných softwarů závisí na dostupnosti binárních balíčků a taky je třeba zohlednit výpočetní výkon platformy.

Závěr

Tato diplomová práce se věnuje výběrů nejvhodnější open source automatizační platformy a následné realizaci automatizační úlohy. Součástí úlohy bylo vytvoření programu pro měření spotřeby elektrické energie a výpočet predikované spotřeby, podle které probíhá regulace a taky vytvoření vizualizace pro prezentaci dat s možnosti manuálního ovládání.

V prvních třech kapitolách byly popsány vybrané řídicí systémy, do kterých patří produktová řada PFC200 od firmy Wago, dále řídicí systémy Unipi Patron a modulární systémy Revolution Pi firmy Kunbus.

V další kapitole je srovnání vybraných platforem z hlediska pořizovacích nákladů, komunikačních možností a vývojového prostředí a následný výběr nejvhodnější platformy na základě stanovených požadavků. Realizace proběhla na systémech PFC200, protože již byl k dispozici.

Kapitola 5 se věnuje popisu a realizaci praktické úlohy. Konkrétně se jedná o tvorbu programu pro měření spotřeby elektrické energie z impulsních výstupů elektroměru, výpočtem predikované čtvrthodinové spotřeby a její regulací spotřeby. Dále je popsáno nastavení komunikace protokolem Modbus TCP, který je používáno pro komunikaci s druhou řídicí stanicí a nadřazeným systémem.

Další část této kapitoly se věnuje programu Quick Control 4, v kterém jsem zpracoval vizualizaci pro realizovanou aplikace. Jsou zde popsán způsob instalace programu a všech potřebných nástrojů. Následně je zde popsána konfigurace QC4 aplikace a použité applety. Popisem vytvořené vizualizace se zabývá poslední část kapitoly.

Poslední kapitola se zabývá zhodnocením výsledků dosažených v rámci práce, je zde popsán i problém s přesností měření času pří výpočtu spotřeby z impulsního výstupu. Dále jsou v kapitole zmíněny nedostatky a výhody vybraného řešení.

Výsledkem práce je tedy aplikace pro měření spotřeby, výpočet čtvrthodinové predikce spotřeby, regulaci a vizualizaci dat s možností manuálního ovládání. Aplikace sice nebyla realizována na vybrané platformě, ale zkušenosti získány tvorbou této aplikace jsou přenositelné i na jiné platformy.

Literatura

[1] Řídicí systémy. *Wago* [online]. © 2022 WAGO [cit. 20.12.2022]. Dostupné z URL: <https://www.wago.com/cz/automatizacni-technika/poznejte-plc>.

[2] Procesorový modul PFC200. *Wago* [online]. © WAGO [cit. 20.12.2022] Dostupné z URL: <https://www.wago.com/cz/automatizacni-technika/poznejte-plc/pfc200>.

[3] System Manual WAGO I/O System 750/753. *Wago* [online]. Version 3.1.0 © 2022 WAGO GmbH & Co. KG [cit. 21.12.2022] <https://www.wago.com/cz/procesorov%C3%BD-modul/procesorov%C3%BD-modul-pfc100/p/750-8101#downloads>.

[4] Procesorový modul PFC100. *Wago* [online]. © WAGO [cit. 21.12.2022] Dostupné z URL: <https://www.wago.com/cz/automatizacni-technika/poznejte-plc/pfc100>.

[5] VOJÁČEK, A. Programovací režimy pro PLC dle IEC 61131-3 (CoDeSys). In: *automatizace.hw.cz* [online]. © 2011, 3. března 2011 21:41 [cit. 21.12.2022] Dostupné z URL: <https://automatizace.hw.cz/programovaci-rezimy-pro-plc-dle-iec-611313-codes

[6] Unipi Patron *Unipi.technology Knowledge Base* [online]. © UniPi.technology 2018 [cit. 27.12.2023] Dostupné z URL: <https://kb.unipi.technology/cs:hw:007-patron>.

[7] Unipi Extension *Unipi.technology Knowledge Base* [online]. © UniPi.technology 2018 [cit. 27.12.2022] Dostupné z URL: <https://kb.unipi.technology/cs:hw:04-extensions>.

[8] Vývoj hardware *Unipi.technology* [online]. © UniPi.technology 2014 [cit. 28.12.2022] Dostupné z URL: <https://www.unipi.technology/cs/zakazkovy-vyvoj/vyvoj-hardware-417>.

[9] Jak vybrat správný software *Unipi.technology Knowledge Base* [online]. © UniPi.technology 2018 [cit. 28.12.2023] Dostupné z URL: <https://kb.unipi.technology/cs:sw:00-start>.

[10] Mervis *Unipi.technology Knowledge Base* [online]. © UniPi.technology 2018 [cit. 28.12.2023] Dostupné z URL: <https://kb.unipi.technology/cs:sw:01-mervis>.

[11] Meet the Revolution Pi products. *Industrial Raspberry Pi - The Revolution Pi* [online]. © KUNBUS GmbH [cit. 29.12.2022]. Dostupné z URL: <https://revolutionpi.com/revolution-pi-series/>.

[12] RevPi Connect base module *Industrial Raspberry Pi - The Revolution Pi* [online]. © KUNBUS GmbH [cit. 29.12.2022]. Dostupné z URL: <https://revolutionpi.com/revpi-connect/>.

[13] Overview - RevPi Connect. *Industrial Raspberry Pi - The Revolution Pi* [online]. © KUNBUS GmbH [cit. 30.12.2022]. Dostupné z URL: <https://revolutionpi.com/tutorials/uebersicht-revpi-connect/>.

[14] Overview - RevPi Core. *Industrial Raspberry Pi - The Revolution Pi* [online]. © KUNBUS GmbH [cit. 30.12.2022]. Dostupné z URL: <https://revolutionpi.com/tutorials/overview-revpi-core/>.

[15] About PiCtory. *Industrial Raspberry Pi - The Revolution Pi* [online]. © KUNBUS GmbH [cit. 02.01.2023]. Dostupné z URL: <https://revolutionpi.com/tutorials/was-ist-pictory-2/>.

[16] Overview RevPi Con CAN *Industrial Raspberry Pi - The Revolution Pi* [online]. Copyright © KUNBUS GmbH [cit. 02.01.2023]. Dostupné z URL: <https://revolutionpi.com/tutorials/uebersicht-revpi-con-can/>.

[17] Industrial Raspberry Pi with CODESYS. *Industrial Raspberry Pi - The Revolution Pi* [online]. KUNBUS GmbH [cit. 02.01.2023]. Dostupné z URL: <https://revolutionpi.com/revpi-connect-codesys/>.

[18] QuickControl. *ASYC, s.r.o* [online]. Copyright © 2011 ASYC, s.r.o. [cit. 02.05.2023]. Dostupné z URL: <https://asyc.cz/index.php?a=24&lang=cs&id=qc>.

[19] WAGO/pfc-firmware-sdk-G2 at FW23-V04.01.10. *GitHub* [online]. Copyright © 2023 GitHub, Inc. [cit. 08.05.2023]. Dostupné z URL: <https://github.com/WAGO/pfc-firmware-sdk-G2/tree/FW23-V04.01.10>.

Seznam symbolů a zkratek

API	Application Programming Interface
CAN	Controller Area Network
DMPO	Dopravní podnik města Olomouced
eMMC	embedded Multi Media Card
MaR	Měření a regulace
PLC	Programmable Logic Controller
RAM	Random Access Memory
SCADA	Supervisory Control and Data Acquisition
SDK	Software development kit
SFP	Small Form-factor Pluggable
SO-DIMM	Small Outline Dual In-line Memory Module
SSL	Secure Socket Layer
SSH	Secure Shell
TLS	Transport Layer Security
VPN	Virtual Private Network

Seznam příloh

A Konfigurační soubor TrendServeru

Výpis A.1: Konfigurace TrendServeru

```
<?xml version="1.0" encoding="utf-8"?>
<TrendServer
  xmlns="urn:asyc:history:trendserver"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:asyc:history:trendserver qrc:/asyc/
   history/trendserver.xsd"
  page-count="12000"
  page-size="512"
  item-count="6000"
>
  <sql-connection name="TrendServer connection" type="SQLITE"
   database="data/trend.sqlite">
    <tables filters="qc_trend_filters" meta-data="qc_trend_ids" ids
   ="qc_trend_ids" path="qc_trend_path"/>
    <FiltersTableExt>
      <column name="label"/>
      <column name="desc"/>
      <column name="read"/>
    </FiltersTableExt>
  </sql-connection>
  <buffer time="10"/>
  <cache>
    <save-no-full-page max-interval="3600"/>
    <index-file skip-pages="10" max-no-save-hours="720"/>
    <cache-interval interval="600" minimum-free-pages="6000"/>
    <max-time-difference time="60"/>
  </cache>
  <eraser start-delay-s="120" interval-ms="10000" control-interval-
   h="24">
    <storage days="766" delete-days="7" />
  </eraser>
  <recorders>
    <recorder server="asyc" config="asyc_trend.xml"/>
    <recorder server="runtime" config="runtime_trend.xml"/>
  </recorders>
</TrendServer>
```

B Konfigurační soubor QcHTTPServeru

Výpis B.1: Konfigurace QcHTTPServeru

```
<?xml version="1.0" encoding="UTF-8"?>
<QcHTTPServer
  xmlns="urn:asyc:QcHTTPServer"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:asyc:QcHTTPServer qrc:/asyc/QcHTTPServer/
   qchttpserver.xsd"
>
  <listener port="8080" readTimeout="60000" minThreads="1"
   maxThreads="100" cleanupInterval="1000" enableCompression="true"
   >
    <request maxSize="16000" maxMultiPartSize="1000000" />
    <log file="access.log" maxSize="100000000">
        <path end=".html"/>
        <path end=".xhtml"/>
        <path end=".svg"/>
        <path begin="/setdata"/>
    </log>
  </listener>
  <sessions expiration="3600000">
    <cookie  path="/" comment="" domain="" localName="QcHTTPID"
   externName="PHPSESSID" sameSite="Lax"/>
  </sessions>

  <map path="/debug/" id="debug" plugin="QcHTTPServer" class="
   DebugController" config="html_debug.xml"/>
  <map path="/log/" id="log" plugin="QcLogServer" class="
   LogController" config="html_log.xml"/>
  <map path="/alarm/" id="alarm" plugin="QcAlarmServer" class="
   AlarmController" config="html_alarm.xml"/>
  <map path="/trend/data/" id="trend" plugin="QcTrendServer" class=
   "TrendController" config="html_trend.xml"/>
  <map path="/trend/" id="trend_table" plugin="QcHTTPServer" class=
   "TableController" config="html_trend.xml"/>
  <map path="/perm/" id="perm" plugin="QcHTTPServer" class="
   AuthController" config="html_perm.xml"/>
  <map path="/" id="html" plugin="QcHTTPServer" class="
   AppController" config="html.xml"/>
</QcHTTPServer>
```

C Konfigurace ovladačů QcHTTPServeru

Výpis C.1: Konfigurace ovladače DebugController

```
<?xml version="1.0" encoding="utf-8"?>
<debugcontroller xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" maxLogSize="100" password="">
</debugcontroller>
```

Výpis C.2: Konfigurace ovladače LogController

```
<?xml version="1.0" encoding="utf-8"?>
<logcontroller
  xmlns="urn:asyc:QcHTTPServer:logcontroller"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:asyc:QcHTTPServer:logcontroller qrc:/asyc/QcHTTPServer/logcontroller.xsd"
>
  <sql-connection name="logcontroller_connection" type="SQLITE" database="data/log.sqlite" table="log"/>
</logcontroller>
```

Výpis C.3: Konfigurace ovladače AppController

```
<?xml version="1.0" encoding="utf-8"?>
<appcontroller
  xmlns="urn:asyc:QcHTTPServer:appcontroller"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:asyc:QcHTTPServer:appcontroller qrc:/asyc/QcHTTPServer/appcontroller.xsd"
  perm="perm" dataPath="html" directoryIndex="index.xhtml" maxAge="1"
>
  <permissions file="" use="false"/>
  <script-engine>
    <script>
      function qualityIsGood(argq1){
        return (argq1 &amp;amp; (QUALITY_LAST | QUALITY_INIT | QUALITY_BAD))==0;
      }
    </script>
  </script-engine>
</appcontroller>
```

Výpis C.4: Konfigurace ovladače TrendController a TableController

```
<?xml version="1.0" encoding="utf-8"?>
<trendcontroller
  xmlns="urn:asyc:QcHTTPServer:trendcontroller"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:asyc:QcHTTPServer:trendcontroller qrc:/
   asyc/QcHTTPServer/trendcontroller.xsd"
>
  <sql-connection name="trendcontroller_connection" type="SQLITE"
   database="data/trend.sqlite"/>
</trendcontroller>
```

Výpis C.5: Konfigurace ovladače AuthController

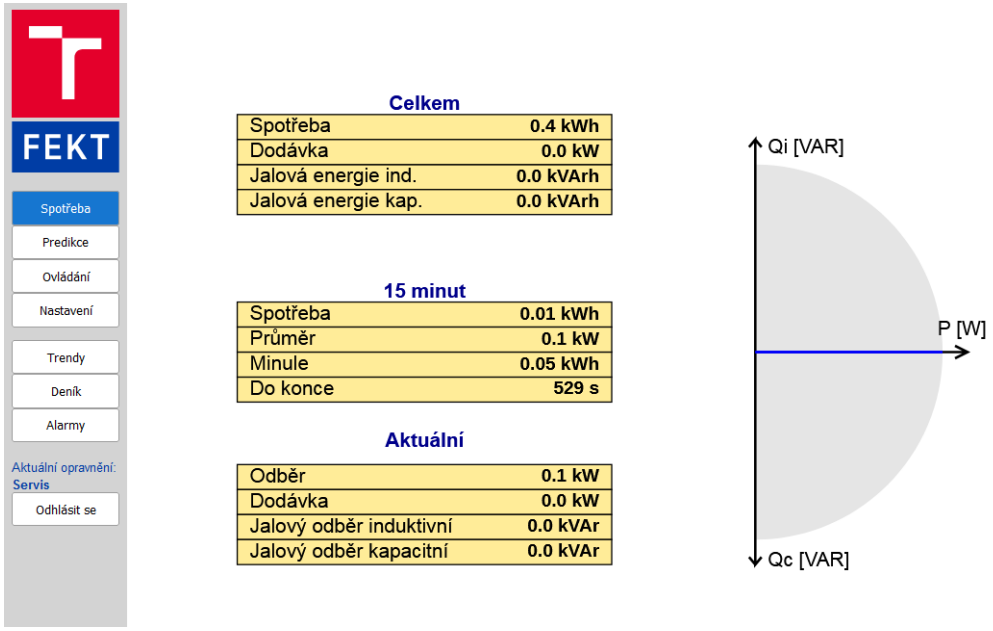
```
<?xml version="1.0" encoding="utf-8"?>
<authcontroller
  xmlns="urn:asyc:QcHTTPServer:authcontroller"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:asyc:QcHTTPServer:authcontroller qrc:/
   asyc/QcHTTPServer/authcontroller.xsd"
  permServerBaseId="perm"
  sessionContextServers="alarm"
>
  <autologin host="10.90.1.120" permGroup="servis"/>
  <login user="servis" password="servis" permGroup="servis"/>
  <login user="obsluha" password="1234" permGroup="ovl"/>
</authcontroller>
```

Výpis C.6: Konfigurace ovladače AlarmController

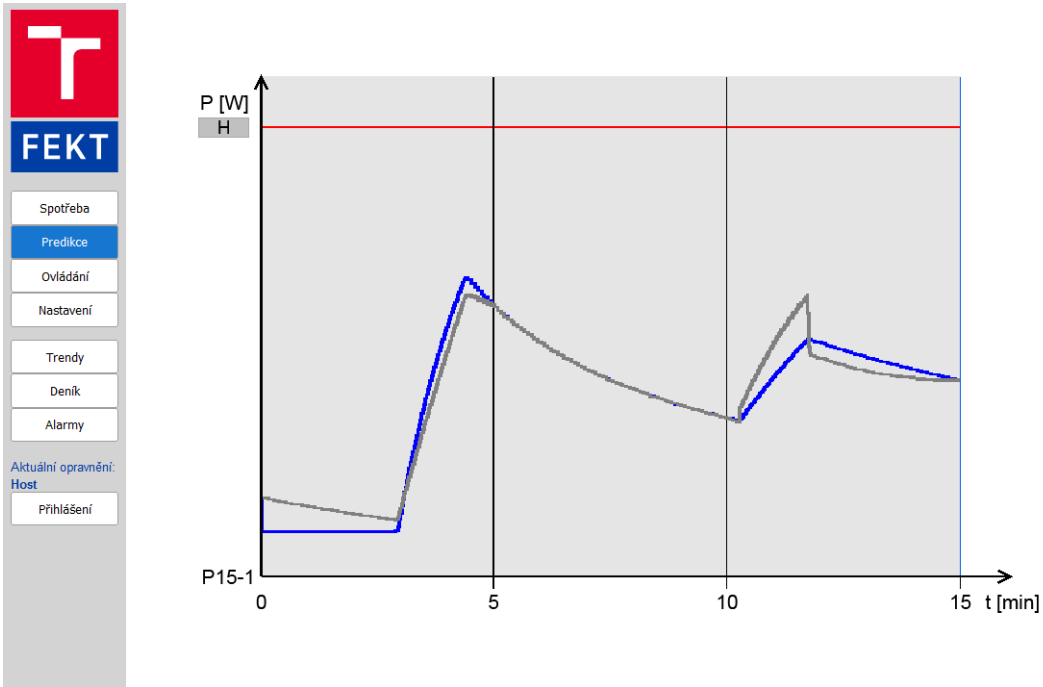
```
<?xml version="1.0" encoding="utf-8"?>
<AlarmClient
  xmlns="urn:asyc:history:alarmclient"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:asyc:history:alarmclient qrc:/asyc/
   history/alarmclient.xsd"
  perm="unused"
>
  <columns>
      <column name="label"/>
    <column name="desc"/>
  </columns>

  <alarms>
    <items server="exps" config="exps_alarm.xml"/>
  </alarms>
</AlarmClient>
```

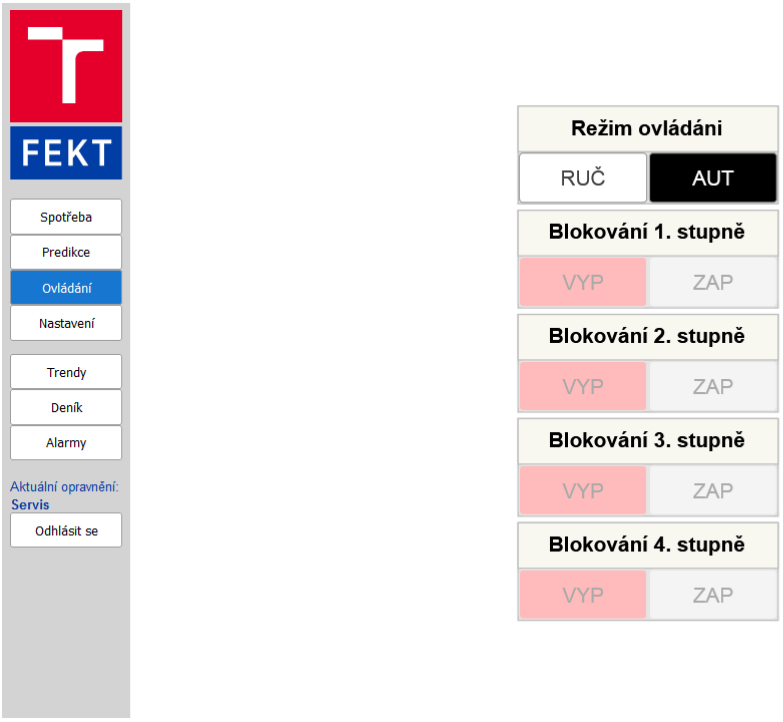
D Obrazovky vizualizace



Obr. D.1: Obrazovka Spotřeba

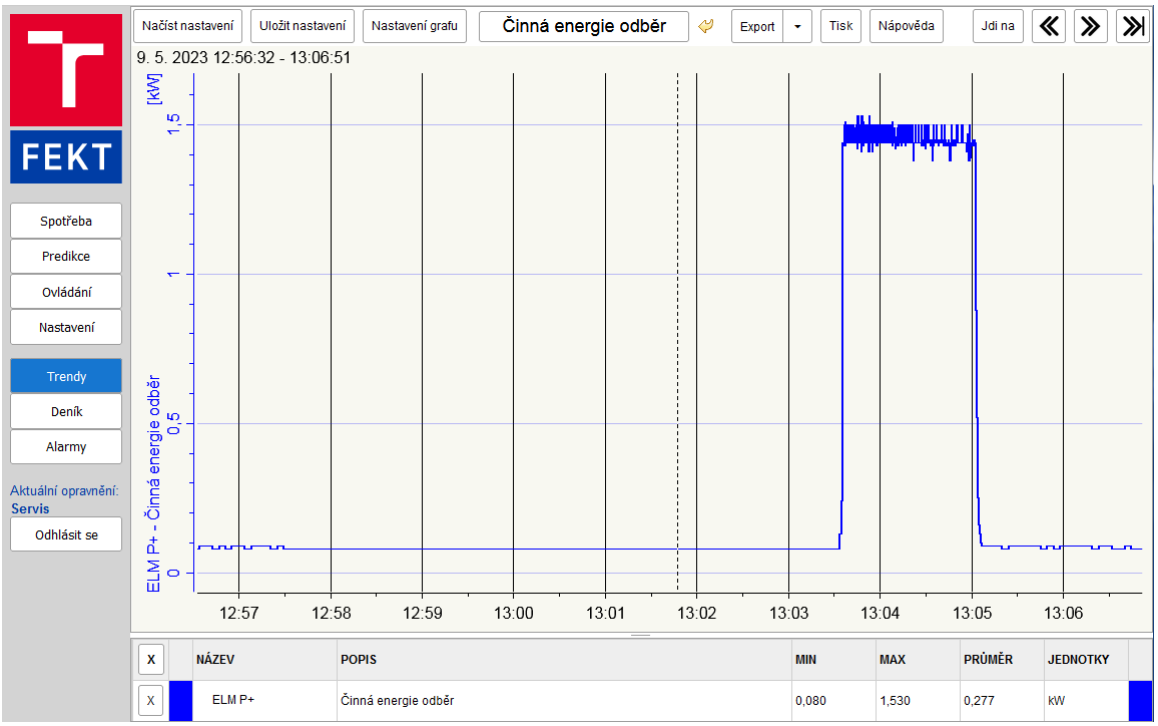


Obr. D.2: Obrazovka Predikce

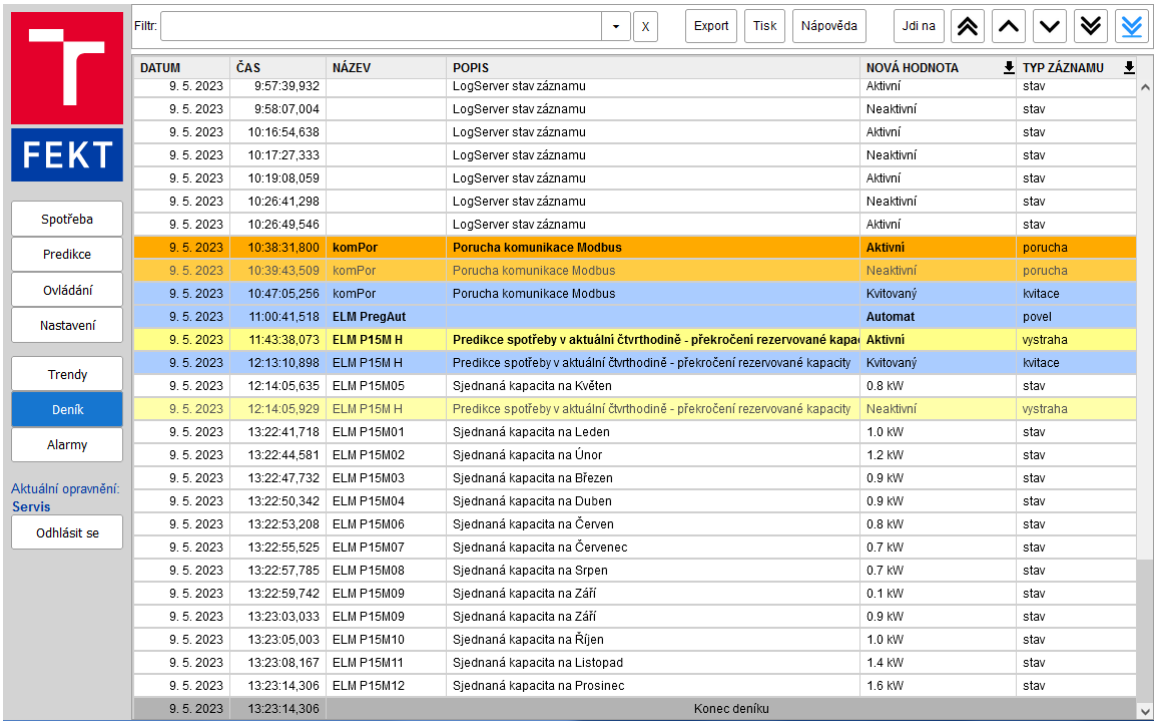


Obr. D.3: Obrazovka Ovládání

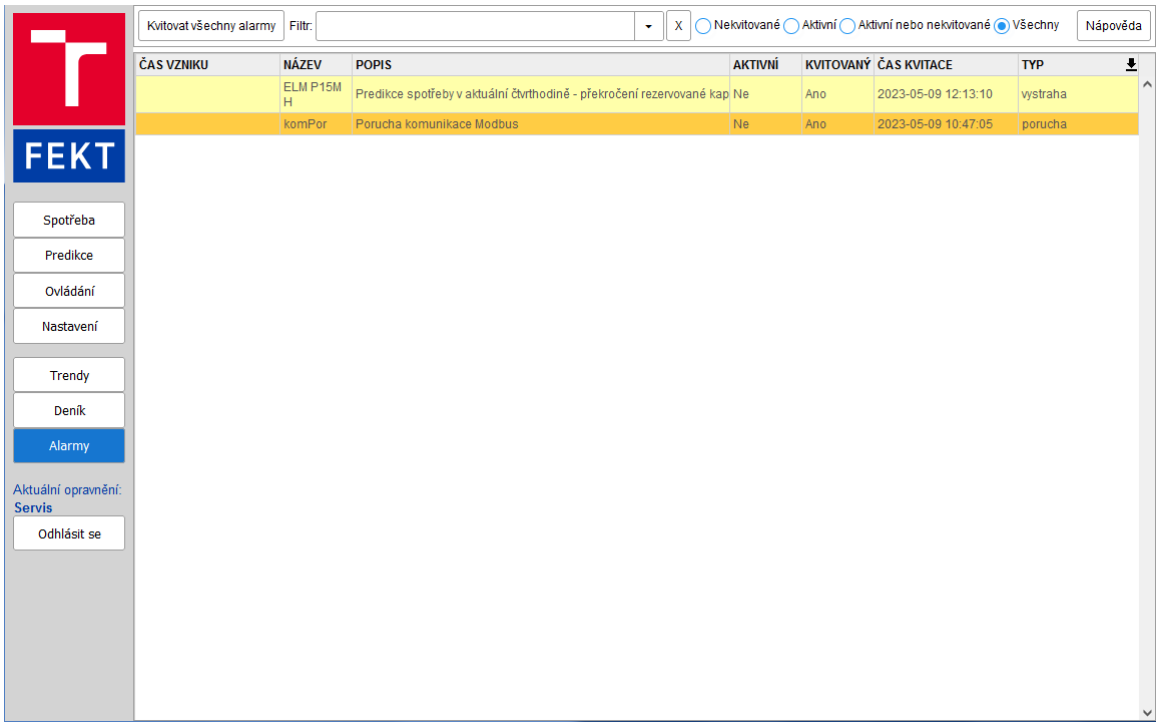
Obr. D.4: Obrazovka Nastavení



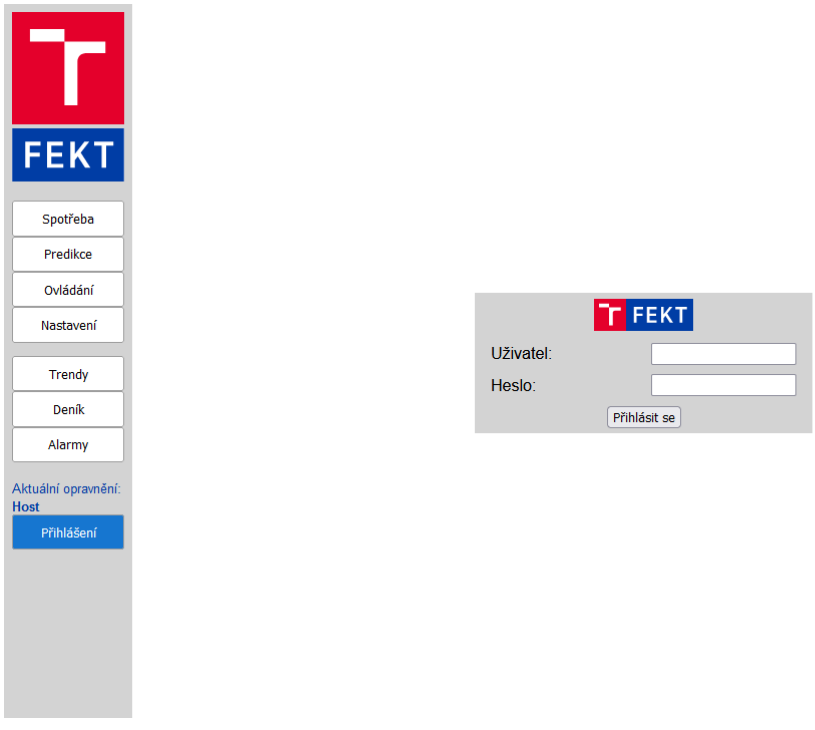
Obr. D.5: Obrazovka Trendy



Obr. D.6: Obrazovka Deník



Obr. D.7: Obrazovka Alarmy



Obr. D.8: Obrazovka Přihlášení

E Obsah elektronické přílohy

- `/` ..... kořenový adresář přiloženého archivu
  - `Codesys` ..... Adresář s projektem realizované úlohy v prostředí Codesys
    - `Jeremenkova MaR.projekt`
  - `QC4` ..... Adresář s QC4 aplikaci
    - `app` ..... Adresář obsahující konfiguraci a vizualizaci QC4 aplikace
      - `html` ..... Adresář obsahující soubory pro vizualizaci
        - `img` ..... Adresář obsahující statické SVG soubory
        - `qc4` ..... Knihovna *.js a *.css souborů pro klienta
        - `trend` ..... Trendové okna analogových hodnot
          - `*.xhtml`
        - `*.js`
        - `*.css`
        - `*.xhtml`
        - `*.hmtl`
        - `*.svg`
      - `*.xml` ..... XML konfigurační soubory
    - `qcservice.xml` ..... Základní konfigurační soubor runtime
  - `qc4_install` ..... Adresář s instalačními balíčky
    - `fontconfig_2.13.92_armhf.ipk`
    - `freetype_2.10.1_armhf.ipk`
    - `libdrm_2.4.110_armhf.ipk`
    - `qc4-runtime_2.2.9_armhf.ipk`
    - `qt5_5.15.0_armhf.ipk`
  - `xszyma00.pdf` ..... Elektronická verze Diplomové práce