VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE PROTOKOLŮ PROFINET, ETHERNET/IP A MODBUS V PROGRAMOVACÍM JAZYCE JAVA

JAVA IMPLEMENTATION OF PROFINET, ETHERNET/IP AND MODBUS PROTOCOLS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Marek Almáši

AUTHOR

VEDOUCÍ PRÁCE — Ing. Kryštof Zeman, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Marek Almáši ***ID:*** 233307

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

NÁZEV TÉMATU:

Implementace protokolů Profinet, Ethernet/IP a Modbus v programovacím jazyce Java

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce bude seznámení se s protokoly Profinet, Ethernet a Modbus a jejich implementace do systému OpenMUC. V teoretické části práce bude proveden stručný popis historie protokolů a detailní popis jejich struktury. Dále zde bude popsán systém OpenMUC a způsoby přidávání vlastních ovladačů do tohoto frameworku. V praktické části budou poté samotné protokoly implementovány v programovacím jazyce Java. Nejdříve jako samostatně běžící programy a následně budou začleněny do systému OpenMUC. Výsledkem práce tedy budou dvě samostatně běžící aplikace.

DOPORUČENÁ LITERATURA:

[1] Modbus [online]. 2020 [cit. 2020-09-14]. Dostupné z: https://www.modbus.org/.

[2] OpenMUC [online]. 2020 [cit. 2020-09-14]. Dostupné z: https://www.openmuc.org/.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 19.5.2023

Vedoucí práce: Ing. Kryštof Zeman, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Cieľom práce bolo oboznámenie sa s vlastnosťami, štruktúrou a využitím priemyslových protokolov a ich následná implementácia. V rámci implementácie bolo nutné dodržiavať všetky pravidlá a štruktúry samotných protokolov modbus, profinet a ethernet/IP, ako aj protokolov do ktorých sú enkapsulované v rámci OSI modelu. Implementácia bola vykonaná v jazyku java.

KĽÚČOVÉ SLOVÁ

Priemyslový protokol, modbus, profinet, ethenet/ip, priemyslové iot, m2m

ABSTRACT

The aim of the work was to become familiar with the properties, structure and use of industrial protocols and their subsequent implementation. As part of the implementation, it was necessary to observe all the rules and structures of the modbus, profinet and ethernet/IP protocols themselves, as well as the protocols in which they are encapsulated within the OSI model. The implementation was done in java.

KEYWORDS

Industrial protocol, modbus, profinet, ethenet/ip, industrial iot, m2m

ALMÁŠI, Marek. *IMPLEMENTACE PROTOKOLŮ PROFINET, ETHERNET/IP A MODBUS V JAZYCE JAVA*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 71 s. Diplomová práca. Vedúci práce: Ing. Kryštof Zeman, Ph.D.

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Bc. Marek Almáši
VUT ID autora:	233307
Typ práce:	Diplomová práca
Akademický rok:	2022/23
Téma závěrečnej práce:	IMPLEMENTACE PROTOKOLŮ PROFINET, ETHERNET/IP A MODBUS V JAZYCE JAVA

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomové práce, pánu Ing. Kryštofovi Zemanovi Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Diplomová práca má za cieľ analýzu a implementáciu priemyslových protokolov modbus, profinet a ethernet/IP.

V rámci teoretickej časti budú predstavené základné pojmy a rozdelenie komunikácie medzi zariadeniami, konkrétne M2M, H2H, M2H. Ďalej budú popísané systémy SCADA a priemyselné protokoly ako také. Cieľom tejto časti je uviesť problém priemyselnej komunikácie a nachádzať jej úskalí, spoločne s definíciou protokolov, ktoré budú ďalej podrobne rozobrané a položia sa tak základy pre správnu implementáciu.

V praktickej časti sa práca bude sa zaoberať implementáciou troch najčastejších protokolov. Bude obsahovat ich prvotnú implementáciu ako samostatné aplikácie bez akýchkoľvek dodatočných systémov.

Nasledovať bude úvod do systému OpenMUC a implementácia týchto protokolov do tohto systému.

Na záver bude k dispozícii návod na nastavenie systému OpenMUC ako aj výsledky z testovania implementovaných protokolov.

1 Machine-to-Machine komunikácia

Komunikácia M2M (Machine-to-Machine) má bohatú históriu, ktorá siaha až do počiatkov výpočtovej techniky a telekomunikácií. Korene M2M komunikácie možno nájsť vo vývoji telemetrických systémov, ktoré vznikli v polovici 20. storočia[1]. Telemetria zahŕňala bezdrôtový prenos údajov zo vzdialených senzorov alebo zariadení do centrálnej monitorovacej stanice, čo umožňuje zber informácií bez ľudského zásahu[2]. Tieto skoré aplikácie komunikácie M2M sa používali predovšetkým v oblastiach, ako je monitorovanie počasia, priemyselná automatizácia a vedecký výskum.

V 70. a 80. rokoch 20. storočia, s pokrokom v počítačových sieťach, komunikácia M2M nabrala ďalšiu dynamiku. Vznik protokolov ako Modbus a Profibus umožnil zariadeniam a systémom od rôznych výrobcov bezproblémovú komunikáciu a zdieľanie údajov. Tieto protokoly sa stali široko používanými v priemyselnej automatizácii, čo umožňuje strojom vymieňať si informácie a efektívnejšie spolupracovať. Počas tohto obdobia sa komunikácia M2M točila prevažne okolo káblových pripojení a proprietárnych protokolov.

Rýchla expanzia bezdrôtových technológií na konci 20. storočia pripravila pôdu pre významný pokrok v komunikácii M2M. Rozvoj mobilných sietí a internetu priniesol nové možnosti prepojenia strojov a zariadení na veľké vzdialenosti. Zavedenie bezdrôtových protokolov ako GSM, GPRS a neskôr 3G a 4G umožnilo vzdialené monitorovanie, ovládanie a výmenu dát v celosvetovom meradle[1]. Komunikácia M2M našla uplatnenie v rôznych oblastiach vrátane verejných služieb, dopravy, zdravotníctva a inteligentných miest.

21. storočie bolo svedkom dramatického nárastu M2M komunikácie so vznikom internetu vecí (IoT). IoT rozšíril rozsah M2M komunikácie nad rámec tradičných priemyselných prostredí, aby zahŕňal každodenné predmety a spotrebiteľské zariadenia. Šírenie nízkoenergetických bezdrôtových technológií, ako sú Wi-Fi, Bluetooth, umožnilo bezproblémové pripojenie a komunikáciu medzi obrovským množstvom zariadení. To viedlo k vývoju inteligentných domácností, pripojených áut, nositeľných zariadení a mnohých ďalších aplikácií internetu vecí, ktoré sa spoliehajú na komunikáciu M2M pri výmene údajov, automatizácii a rozšírených funkciách[1].

V súčasnosti sa komunikácia M2M naďalej vyvíja s príchodom sietí 5G a pokrokmi v oblasti edge computingu a cloudových technológií. Tieto vylepšenia umožňujú rýchlejšie a spoľahlivejšie pripojenie, spracovanie údajov v reálnom čase a väčšiu škálovateľnosť. Komunikácia M2M je teraz kritickým prostriedkom pre nové technológie, ako sú autonómne vozidlá, priemyselný internet vecí, inteligentné siete a systémy monitorovania zdravotnej starostlivosti.

História M2M komunikácie ukazuje vývoj technológií a rastúci význam vzájomne prepojených zariadení a systémov. Od počiatkov telemetrie až po modernú éru in-

ternetu vecí M2M komunikácia zmenila odvetvia, zvýšila efektivitu a otvorila nové možnosti automatizácie, monitorovania a kontroly. Keďže technológia neustále napreduje, komunikácia M2M je pripravená hrať ešte významnejšiu úlohu pri formovaní nášho prepojeného sveta.

1.1 M2M aplikácie a príklady

Hlavným účelom technológie stroj-stroj je čerpať údaje zo senzorov a prenášať ich do siete. Systémy M2M často využívajú verejné siete a prístupové metódy – napríklad bezdrôtové alebo cez ethernet – aby boli nákladovo efektívnejšie.

Medzi hlavné komponenty systému M2M patria senzory, Wi-Fi zariadenia alebo mobilné komunikačné spojenie a autonómny výpočtový softvér naprogramovaný tak, aby pomohol sieťovému zariadeniu interpretovať dáta a robiť rozhodnutia[2]. Tieto aplikácie M2M prekladajú dáta, ktoré môžu spúšťať predprogramované, automatizované akcie.

Okrem možnosti vzdialene monitorovať zariadenia a systémy patria medzi hlavné výhody M2M[1]:

- Znížené náklady minimalizáciou údržby a prestojov zariadení;
- Zvýšenie výnosov odhalením nových obchodných príležitostí pre servis produktov v teréne.
- Zlepšený zákaznícky servis proaktívnym monitorovaním a servisom zariadenia pred jeho zlyhaním alebo len vtedy, keď je to potrebné.

1.2 Rozdiely medzi M2M a H2H

M2M (Machine-to-Machine), H2M (Human-to-Machine) a H2H (Human-to-Human) sú odlišné formy komunikácie, ktoré slúžia na rôzne účely v rôznych doménach.

Komunikácia M2M sa týka výmeny údajov a informácií medzi strojmi alebo zariadeniami bez priamej účasti človeka. Umožňuje automatizáciu a diaľkové ovládanie procesov, čo umožňuje zariadeniam bezproblémovú interakciu a spoluprácu. Komunikácia M2M často využíva protokoly a technológie prispôsobené interakciám so strojmi, ako sú protokoly IoT (napr. MQTT, CoAP) a protokoly priemyselnej automatizácie (napr. Modbus, PROFIBUS). Komunikácia M2M nachádza rozsiahle uplatnenie v sektoroch, ako je výroba, logistika a inteligentná infraštruktúra, kde zariadenia a senzory komunikujú a koordinujú sa, aby optimalizovali operácie a zvýšili efektivitu.

Na druhej strane komunikácia H2M zahŕňa interakciu medzi ľuďmi a strojmi. Vzťahuje sa na tok informácií od ľudí k strojom, zvyčajne cez používateľské rozhra-

nia alebo vstupné zariadenia. Komunikácia H2M umožňuje ľuďom ovládať a monitorovať stroje a poskytovať pokyny, príkazy alebo otázky. Táto forma komunikácie je rozšírená v každodenných technológiách, ako sú počítačové rozhrania, mobilné aplikácie a systémy domácej automatizácie. Komunikačné rozhrania H2M sú navrhnuté tak, aby boli užívateľsky prívetivé, intuitívne a pohotové, čo jednotlivcom umožňuje efektívne komunikovať so strojmi a dosahovať požadované výsledky.

H2H komunikácia predstavuje tradičnú komunikáciu človeka s človekom, kde dochádza k výmene informácií priamo medzi jednotlivcami. Táto forma komunikácie je životne dôležitá pre spoluprácu, koordináciu a sociálne interakcie. H2H komunikácia zahŕňa širokú škálu médií vrátane osobných rozhovorov, telefónnych hovorov, e-mailov, videokonferencií a platforiem sociálnych médií. Na rozdiel od komunikácie M2M a H2M, komunikácia H2H zahŕňa zložitý ľudský jazyk, neverbálne podnety, emócie a kontext. Hrá kľúčovú úlohu v rôznych aspektoch spoločnosti, ako je podnikanie, vzdelávanie, zdravotníctvo a osobné vzťahy.

Kľúčový rozdiel medzi týmito typmi komunikácie spočíva v zapojených entitách a povahe interakcií. Komunikácia M2M sa zameriava na umožnenie bezproblémových interakcií medzi strojmi, uľahčenie automatizácie a optimalizácie procesov. Komunikácia H2M umožňuje ľuďom komunikovať so strojmi a poskytuje možnosti kontroly a monitorovania. H2H komunikácia na druhej strane zahŕňa výmenu informácií, myšlienok a emócií medzi jednotlivcami, podporuje spoluprácu, porozumenie a sociálne prepojenie.

Pochopenie týchto rozdielov je nevyhnutné pre navrhovanie efektívnych komunikačných systémov a rozhraní prispôsobených špecifickým kontextom a požiadavkám. Či už ide o autonómne spolupracujúce stroje, interakciu ľudí so strojmi alebo jednotlivcov zapojených do medziľudskej komunikácie, každá forma slúži odlišným účelom a vyžaduje si rôzne prístupy na dosiahnutie úspešných a zmysluplných interakcií.

1.3 Priemyslové protokoly

Komunikačný protokol je kombináciou pravidiel používaných v procese výmeny údajov rôznych sieťových zariadení a softvérových systémov. Komunikačné protokoly bežne obsahujú zoznam všetkých formálnych požiadaviek a noriem vrátane syntaxe, obmedzení, procedúr, obnovy chýb a synchronizácie komunikácie[4].

Priemyselné protokoly sú základnou súčasťou moderných systémov priemyselnej automatizácie[5]. Tieto protokoly definujú pravidlá a štandardy pre komunikáciu a výmenu dát medzi zariadeniami a systémami v priemyselných prostrediach. Zabezpečujú bezproblémovú interoperabilitu, spoľahlivosť a efektivitu a umožňujú efektívnu komunikáciu rôznych zariadení a komponentov od rôznych výrobcov. Priemyselné protokoly vykazujú vlastnosti, ako je determinizmus, schopnosti v reálnom

Tab. 1.1: Rozdiely medzi H2H a M2M [3]

	M2M	H2H
Rozsah oneskorenia	10ms niekoľko minút	250ms (hlas) niekoľko hodín až dní (e-mail)
Typ zariadení / prenosu	GSM/UMTS/LTE/Wifi moduly, ethernet, profinet, Modbus...	GSM/UMTS/LTE/Wifi, ethernet, USB / flash úložiská, hovorený hlas
Mobilita	cca 90% statické zariadenia	cca 51% mobilné smartphony
Dĺžka spojenia	často veľmi krátka, pri prenose konfiguračných dát môže byť dlhšia	často veľmi dlhá, predovšetkým pri prenose multimédií a hovorenej reči
Frekvencia spojenia	veľmi častá (napríklad hello správy)	relatívne zriedkavá
Veľkosť správy	typicky veľmi krátka (väčšina správ je len réžia spojenia), rádovo do 1000 byteov	dlhá (správy vždy obsahujú dáta), jedna správa vo viacerých TCP packetoch (65535 bytov jeden)
Množstvo zariadení	vo veľkých sieťach 100000+	typicky 2, zriedkavo viac 100
Kľúčové metriky	latencia, energetická efektivita	oneskorenie, stratovosť dát

čase, robustnosť a škálovateľnosť, ktoré sú rozhodujúce pre časovo citlivé a kritické priemyselné procesy. Príklady široko používaných priemyselných protokolov zahŕňajú Modbus, PROFIBUS, PROFINET, EtherCAT a Ethernet/IP, pričom každý je prispôsobený špecifickým aplikačným doménam a požiadavkám. Tieto protokoly umožňujú integráciu zariadení, ako sú programovateľné logické riadiace jednotky (PLC), rozhrania človek-stroj (HMI), senzory, akčné členy a poľné zariadenia, čo uľahčuje monitorovanie, riadenie a výmenu údajov v reálnom čase. Tým, že priemyselné protokoly poskytujú štandardizovaný a interoperabilný rámec, zvyšujú flexibilitu, škálovateľnosť a efektívnosť systémov priemyselnej automatizácie, čo umožňuje priemyselným odvetviam optimalizovať ich procesy, zvyšovať produktivitu a prispôsobovať sa vyvíjajúcemu sa technologickému pokroku.

1.3.1 Modbus

Modbus je široko používaný priemyselný protokol, ktorý už niekoľko desaťročí pomáha pri komunikácii a výmene dát v systémoch priemyselnej automatizácie. Bol vyvinutý spoločnosťou Modicon (dnes súčasť Schneider Electric) koncom 70. rokov minulého storočia a odvtedy sa stal de facto štandardom pre pripojenie rôznych zariadení a zariadení v priemyselných prostrediach[5].

Modbus funguje na architektúre master-slave, kde nadradené zariadenie iniciuje komunikáciu a riadi výmenu dát s jedným alebo viacerými podriadenými zariadeniami. Využíva jednoduchú a ľahkú štruktúru správ, čo uľahčuje implementáciu a nasadenie. Modbus podporuje sériovú komunikáciu (Modbus RTU a Modbus ASCII) aj komunikáciu založenú na Ethernete (Modbus TCP/IP), čo ponúka flexibilitu pri pripájaní zariadení cez rôzne fyzické médiá[5].

Jednou z významných výhod Modbusu je jeho otvorenosť a interoperabilita. Získal široké uplatnenie v rôznych odvetviach, čo umožňuje bezproblémovú integráciu zariadení od rôznych výrobcov. Modbus podporuje viacero typov údajov a ponúka všestranné mechanizmy prístupu k údajom, ktoré umožňujú operácie čítania a zápisu do jednotlivých údajových bodov alebo skupín údajov. Vďaka tejto flexibilite je vhodný pre širokú škálu aplikácií, od malých systémov s niekoľkými zariadeniami až po rozsiahle priemyselné siete s tisíckami zariadení[5].

Modbus je známy svojou jednoduchosťou, čo z neho robí efektívny a spoľahlivý protokol pre komunikáciu v priemyselných prostrediach. Poskytuje robustné mechanizmy kontroly chýb a obnovy, ktoré zaisťujú integritu údajov a odolnosť voči chybám[5]. Modbus tiež ponúka deterministickú komunikáciu, ktorá umožňuje presné načasovanie a synchronizáciu výmeny dát, čo je rozhodujúce pre časovo kritické procesy.

Okrem toho Modbus poskytuje diagnostické a monitorovacie funkcie, ktoré používateľom umožňujú diagnostikovať problémy s komunikáciou, sledovať stav zariadenia a riešiť problémy. Ponúka kódy diagnostických funkcií a chybové kódy, ktoré uľahčujú identifikáciu a riešenie problémov s komunikáciou, čím zlepšujú celkovú údržbu a správu systému.

V priebehu rokov sa Modbus vyvíjal a prispôsoboval novým technológiám a požiadavkám. S rastúcim využívaním Ethernetu a TCP/IP v priemyselných sieťach sa Modbus TCP/IP stal preferovaným variantom, ktorý ponúka výhody rýchlejších dátových tokov, väčších adresných priestorov a kompatibility s existujúcou ethernetovou infraštruktúrou.

Na záver, Modbus obstál v skúške časom ako spoľahlivý a široko používaný priemyselný protokol. Jeho jednoduchosť, interoperabilita, determinizmus a všestrannosť z neho urobili základnú voľbu pre pripojenie zariadení a umožnenie komunikácie v systémoch priemyselnej automatizácie. Keďže priemyselné odvetvia sa neustále vyvíjajú a prijímajú nové technológie, Modbus zostáva základným nástrojom na dosiahnutie bezproblémovej výmeny údajov, riadenia a monitorovania v rôznych priemyselných aplikáciách.

V nasledujúcej tabuľke je popísaný zoznam najtypyckejších správ prenášaných protokolom Modbus 1.2.

Tab. 1.2: Správy protokolu Modbus [5]

Označenie funkcie	Typ registru
1	Prečítať hodnotu cievky
2	Prečítať diskrétny vstup
3	Prečítať holding register
4	Prečítať vstupný register
5	Zápis jednej cievky
6	Zápis jedného holding registra
15	Zápis viacerých cievok
16	Zápis viacerých holding registrov

1.3.2 Profinet

Profinet je široko používaný priemyselný protokol, ktorý sa stal základným kameňom modernej priemyselnej automatizácie. Profinet, vyvinutý spoločnosťami Profibus a Profinet International (PI), kombinuje výhody ethernetovej technológie s deterministickými schopnosťami a schopnosťami v reálnom čase vyžadovanými v priemyselných

prostrediach. Profinet umožňuje bezproblémovú komunikáciu a výmenu dát medzi rôznymi zariadeniami, vrátane PLC, ovládačov pohybu a senzorov v širokej škále priemyselných aplikácií.

Jednou z kľúčových vlastností Profinet je jeho flexibilita a škálovateľnosť. Podporuje jednoduchú komunikáciu medzi zariadeniami a komplexné distribuované riadiace architektúry. Profinet umožňuje integráciu zariadení od rôznych výrobcov, podporuje interoperabilitu a znižuje zložitosť implementácie. Profinet navyše podporuje komunikáciu v reálnom čase, čo umožňuje presnú synchronizáciu a rýchlu výmenu dát, čo je rozhodujúce pre časovo citlivé priemyselné procesy[4].

PROFINET funguje na štandardnej ethernetovej infraštruktúre a využíva svoju rýchlosť, šírku pásma a širokú dostupnosť. Využíva protokoly priemyselného Ethernetu a využíva rôzne komunikačné režimy vrátane TCP/IP na konfiguráciu a diagnostiku a protokolov UDP/IP a Real-Time (RT) na prenos údajov v reálnom čase. Použitie štandardnej ethernetovej technológie zjednodušuje integráciu Profinetu s existujúcimi ethernetovými sieťami[4], čím zabezpečuje kompatibilitu a jednoduché nasadenie.

Profinet tiež poskytuje pokročilé diagnostické a monitorovacie schopnosti, ktoré umožňujú efektívne riešenie problémov, údržbu a optimalizáciu systému. Podporuje komplexnú diagnostiku na rôznych úrovniach vrátane diagnostiky zariadení, diagnostiky siete a diagnostiky celého systému. To umožňuje rýchlu identifikáciu porúch, úzkych miest v sieti alebo problémov s výkonom, čo vedie k zlepšeniu prevádzkyschopnosti a zníženiu prestojov v priemyselných prostrediac[4].

Celkovo Profinet predstavuje výkonný priemyselný protokol, ktorý kombinuje výhody technológie Ethernet so špecifickými požiadavkami priemyselnej automatizácie. Jeho flexibilita, možnosti v reálnom čase, škálovateľnosť a možnosti integrácie z neho robia ideálnu voľbu pre širokú škálu priemyselných aplikácií, čím umožňujú odvetviam dosahovať efektívnu a spoľahlivú komunikáciu, zvýšenú produktivitu a vylepšenú prevádzkovú dokonalosť[4].

Profinet musí zabezpečiť, aby sa správy doručovali s primeranou rýchlosťou a determinizmom v závislosti od úlohy. Nie všetky aplikácie vyžadujú rovnaký výkon. Napríklad načítanie konfiguračných údajov pre procesný nástroj môže trvať niekoľko minút bez ovplyvnenia výroby. Na druhej strane oneskorenie komunikácie len niekoľko milisekúnd medzi PLC a vysokorýchlostným VFD môže výrazne ovplyvniť proces.

V nasledujúcich tabuľkách je popísaný zoznam služieb protokolu profinet 1.3, nasledovaný ich detailným popisom 1.4. Jendotlivé hodnoty Service ID, Option a Suboption sa nachádzajú v každej správe, na základe čoho je typ požiadavky, ktorú táto správa prenáša identifikovaný.

Tab. 1.3: Typy služieb protokolu Profinet [4]

Service ID	Popis
3	Get
4	Set
5	Identify
6	Hello

Tab. 1.4: Detailný popis služieb protokolu profinet [4]

Service ID	Option	Suboption	Popis
3	1	1	MAC adresa
3,4,5,6	1	2	IP parameter
3,4,5,6	1	3	Všetky IP parametre
3,5	2	1	Typ stanice
3,4,5,6	2	2	Názov stanice
3,5,6	2	3	ID zariadenia
3,5	2	4	Rola zariadenia
3,5	2	5	Možnosti zariadenia
4	5	1	Štart transakcie
4	5	2	Koniec transakcie
4	5	3	Signál (zablikaj LED)
4	5	4	Factory reset
5	255	255	Všetky hodnoty
6	6	1	Po spustení pošle Hello správu

1.3.3 EtherNet/IP

Ethernet/IP je populárny priemyselný protokol široko používaný v systémoch priemyselnej automatizácie pre bezproblémovú komunikáciu a interoperabilitu[6]. Je postavený na základoch technológie Ethernet, využíva možnosti vysokorýchlostného prenosu dát, škálovateľnosť a širokú dostupnosť. Ethernet/IP umožňuje zariadeniam od rôznych výrobcov komunikovať a vymieňať si dáta v reálnom čase, čo umožňuje efektívne riadenie, monitorovanie a koordináciu priemyselných procesov[6].

Jednou z kľúčových predností Ethernetu/IP je jeho flexibilita a otvorenosť. Podporuje širokú škálu zariadení vrátane programovateľných logických ovládačov PLC, rozhraní človek-stroj HMI, pohonov, senzorov a akčných členov. Dodržiavaním štandardných ethernetových protokolov, ako sú TCP a UDP, zabezpečuje Ethernet/IP

kompatibilitu a jednoduchú integráciu do existujúcich ethernetových sietí. To uľahčuje bezproblémovú konektivitu naprieč viacerými vrstvami automatizačnej hierarchie, od dielne až po podnikovú úroveň[6].

Ethernet/IP využíva spoločný priemyselný protokol (CIP), všestranný komunikačný rámec, ktorý poskytuje jednotné rozhranie pre rôzne priemyselné zariadenia. CIP podporuje rôzne komunikačné služby, vrátane explicitných správ pre komunikáciu z bodu do bodu a implicitných správ pre aplikácie distribuovaného riadenia. To umožňuje zariadeniam vymieňať si dáta, informácie o stave a príkazy štandardizovaným a efektívnym spôsobom, pričom podporuje aplikácie v reálnom čase a časovo kritické aplikácie[6].

Ďalšou významnou výhodou Ethernet/IP je jeho schopnosť spracovať veľké množstvo dát. Vďaka svojim vysokorýchlostným prenosovým rýchlostiam a veľkému adresnému priestoru môže Ethernet/IP vyhovieť zložitým automatizačným systémom, ktoré vyžadujú rozsiahlu výmenu a spracovanie údajov. Táto schopnosť je obzvlášť cenná v aplikáciách zahŕňajúcich riadenie pohybu, robotiku a pokročilé systémy strojového videnia[6].

Ethernet/IP tiež podporuje funkcie diagnostiky a správy siete, čo umožňuje efektívne riešenie problémov, monitorovanie a údržbu priemyselných sietí. Ponúka funkcie, ako je zisťovanie siete, konfigurácia zariadenia a komplexná diagnostika, čo umožňuje rýchlu identifikáciu a riešenie problémov so sieťou. To prispieva k zlepšeniu spoľahlivosti systému, zníženiu prestojov a zjednodušeniu postupov údržby[6].

Ako sa Ethernet/IP neustále vyvíja, zahŕňa vznikajúce technológie a štandardy, aby vyhovovali vyvíjajúcim sa potrebám priemyselnej automatizácie. Integrácia Time-Sensitive Networking (TSN) ďalej zlepšuje možnosti Ethernet/IP poskytovaním determinizmu a synchronizácie v časovo kritických aplikáciách.

Protokol sa používa v širokej škále zariadení, ako sú roboty, osobné počítače, programovateľné logické automaty, sálové počítače, vstupno/výstupné adaptéry a iné podobné zariadenia.

V nasledujúcej tabuľke je popísaný zoznam najtypyckejších správ prenášaných protokolom Ethernet/IP 1.5.

1.4 Systémy SCADA

SCADA je skratka pre dozornú kontrolu a získavanie údajov, alebo po anglicky „Supervisory Control and Data Acquisition“.

SCADA je počítačový systém na zhromažďovanie a analýzu údajov v reálnom čase, na monitorovanie a riadenie zariadení, ktoré sa zaoberajú kritickými a časovo citlivými materiálmi alebo udalosťami. Systémy SCADA boli prvýkrát použité v 60.

Tab. 1.5: Správy protokolu Ethernet/IP [6]

Funkcia	Typ registru
0x00	nop (prázdna správa)
0x01	zoznam cieľov
0x04	zoznam služieb
0x63	zoznam identít
0x64	zoznam rozhraní
0x65	registrovanie relácie
0x66	odregistrovanie relácie
0x70	poslanie dát

rokoch 20. storočia a v súčasnosti sú neoddeliteľnou súčasťou prakticky všetkých priemyselných závodov a výrobných zariadení[7].

SCADA systém umožňuje organizáciám[8]:

- Riadiť procesy lokálne alebo na vzdialených miestach
- Získavať, analyzovať a zobrazovať dáta v reálnom čase
- Priamou interakciou s priemyselnými zariadeniami, ako sú senzory, ventily, čerpadlá a motory
- Zaznamenávajte a archivujte udalosti pre budúce použitie alebo vytváranie správ

Hardvér, ako sú vzdialené terminálové jednotky (RTU) a programovateľné logické riadiace jednotky (PLC), slúžia ako lokálne zberné miesta na získavanie informácií zo senzorov. Tento hardvér v modernom systéme SCADA často spúšťa akcie pripojeného zariadenia prostredníctvom naprogramovanej logiky. V systéme SCADA sú zozbierané údaje zo senzorov zhromažďované počítačom bežne známym ako brána, anglicky „gateway“.

1.4.1 Príklad aplikácie SCADA

Typický SCADA systém môže byť nastavený tak, aby monitoroval kritický únik na potrubí. Keď sa únik zistí, môže vykonať reťaz príkazov pomocou strojov, aby buď upozornil na signál úniku a/alebo okamžite zatvoril ventil. Takto sa minimalizujú alebo eliminujú nebezpečné podmienky, strata príjmu alebo výroby. Každý SCADA systém je možné prispôsobiť tak, aby presne vyhovoval konkrétnej aplikácii[8]. Môže to byť relatívne jednoduché – malá administratívna budova až po veľmi zložité – jadrová elektráreň.

1.4.2 Význam SCADA systémov

Významom SCADA systémov je automatizácia. Umožňuje organizácii starostlivo študovať a predvídať optimálnu reakciu na namerané podmienky a tieto reakcie zakaždým automaticky vykonávať. Spoliehanie sa na presné riadenie stroja pri monitorovaní zariadení a procesov prakticky eliminuje ľudskú chybu. Ešte dôležitejšie je, že automatizuje bežné, únavné, rutinné úlohy, ktoré musí opakovane vykonávať človek, čo ďalej zvyšuje produktivitu[7]. Okrem toho sú systémy SCADA potrebné na monitorovanie a riadenie situácií, kde organizácia nemusí mať dostatok pracovnej sily na ich pokrytie. Spoľahlivá komunikácia a prevádzkyschopnosť týchto situácií je rozhodujúca pre ich ziskovosť.

1.4.3 Komponenty SCADA systémov

SCADA systémy využívajú[7]:

- systémy riadenia distribúcie (DCS)
- systémy riadenia procesov (PCS)
- programovateľné logické ovládače (PLC)
- vzdialené terminálové jednotky (RTU)

2 Prvotná implementácia protokolov a príprava na implementáciu do OpenMUC

Samotná implementácia protokolov modbus, ethernet/IP a profinet bola vykonaná v programovacom jazyku java v prostredí Eclipse. Všetky protokoly sú schopné pracovať v stave klient alebo server (resp. „master“ a „slave“). Nasledujúce kapitoly popisujú implementáciu samostatných aplikácií.

2.1 Modbus

Protokol modbus je zo spomínanej trojice protokolov pravdepodobne najjednoduchší na implementáciu. Protkol modbus v implementovanej verzii využíva na jeho prenos protokol TCP. Protokol TCP je implementovaný v štandardnej knižnici javy java.net.* . To znamená že je možné sa sústrediť na implementáciu samotného modbus-u, a nie je nutné riešiť ostatné vrstvy OSI.

Implementácia protokolu modbus umožňuje komunikáciu medzi dvoma inštanciami naprogramovanej aplikácie na rôznych zariadeniach, kde jedna predstavuje klient a druhá server. Možná je aj komunikácia so simulačnými nástrojmi, ako napríklad CodeSys. Takýto typ komunikácie bol v rámci semestrálnej práce otestovaný.

Pre testovacie účely v rámci semestrálnej práce bolo vytvorené grafické rozhranie. Nepredpokladá sa, že toto rozhranie bude ďalej využité pre účely diplomovej práce, preto počas jeho vytvárania neboli využité takmer žiadne pravidlá UX/UI dizajnu. GUI je vytvorené v angličtine.

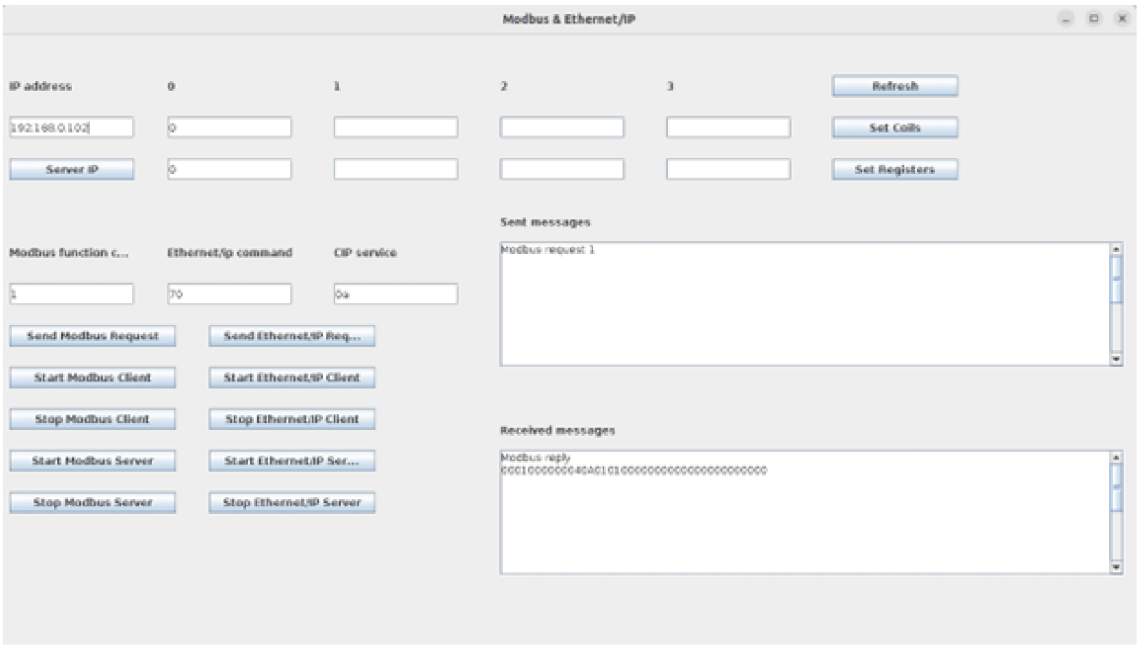
V rámci grafického rozhrania je možné spustiť modbus klient / server, je možné ho aj zastaviť. V prípade klienta je nutné zadať IP adresu serveru, na ktorý sa má pripojiť. Všetky ostatné hodnoty, ako MAC adresa a TCP port sú určené samotnou aplikáciou, prípadne java.net.* knižnicou.

Spojenie je nutné inicializovať manuálne, poslaním prvej správy zo strany klienta. Typ správy je možné určiť podľa zadanej hodnoty v príslušnom textovom poli. V prípade, že bola zadaná korektná hodnota, server na túto správu korektne odpovie.

Hodnoty jednotlivých cievok a registrov je možné takisto zadať a odoslané správy berú ohľad na tieto hodnoty. V prípade „Get“ správy sa hodnoty z odpovede serveru premietnu do textových polí na strane klienta. V prípade „Write“ správy sa hodnoty na strane klienta zapíšu na strane serveru.

Grafické rozhranie síce umožňuje zápis len štyroch registrov / cievok z dôvodu prehľadnosti grafického rozhrania, ale samotný protokol je implementovaný robustnejšie. Je možné prenášať oveľa väčšie množstvo hodnôt bez potreby úpravy implementácie. Tieto hodnoty ale nie je možné zadať pomocou grafického rozhrania.

Na nasledujúcich obrázkoch je možné vidieť grafické rozhranie pre modbus klient 2.1, ktoré sa aktivuje určením IP adresy a kliknutím na tlačidlo "Server IP", a následným kliknutím na tlačidlo "Start Modbus Client". Grafické rozhranie pre modbus server 2.2 je rovnaké, aktivuje sa kliknutím na tlačidlo "Start Modbus Server". Je nutné, aby bol server aktivovaný ako prvý. Na ďalších obrázkoch 2.3 2.4 je možné vidieť zachytenú komunikáciu, ktorá sa vykoná kliknutím na tlačidlo "Send Modbus Request"na strane klienta.

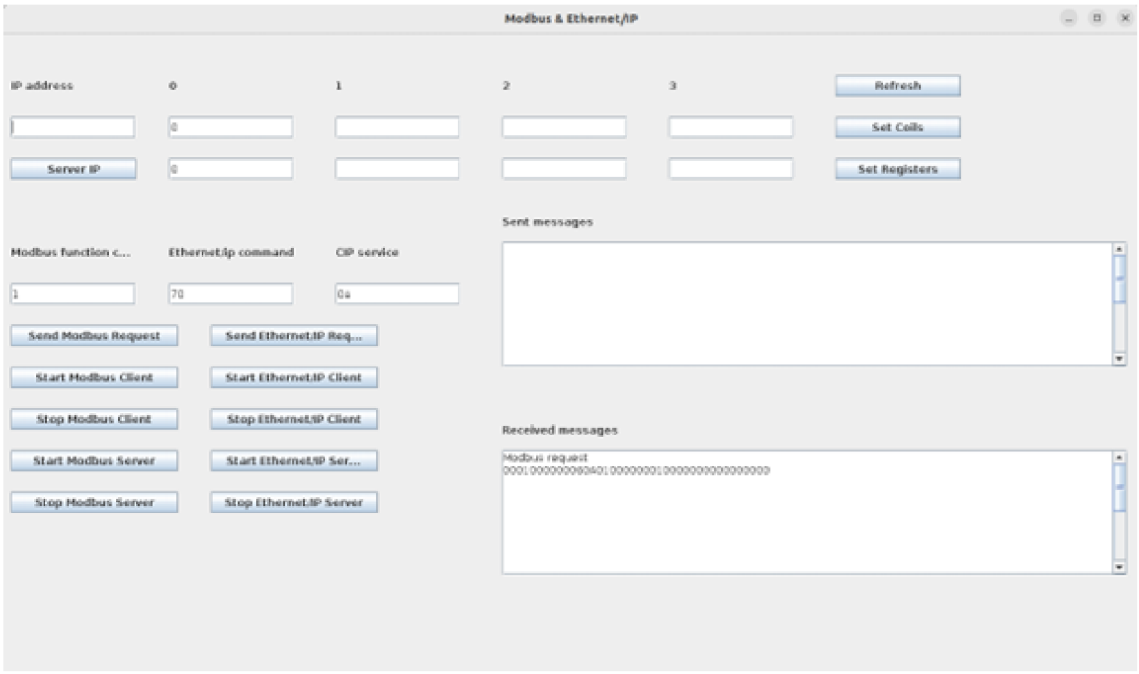


Obr. 2.1: Grafické rozhranie pre Modbus Klient

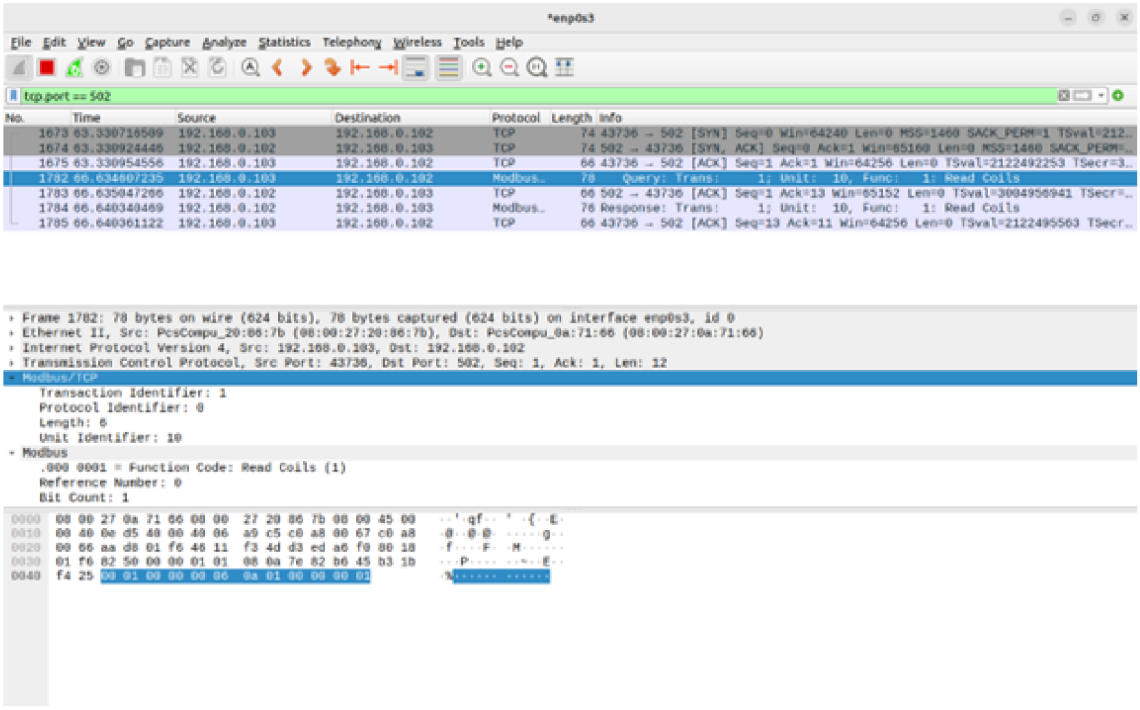
2.2 Profinet

Štruktúra protokolu profinet je tiež relatívne jednoduchá, avšak protokol v jeho najviac používanej forme (profinet_dcp) využíva len protokol ethernet na jeho smerovanie medzi zariadeniami. Toto predstavuje problém, nakoľko štandardné knižnice jazyku java neumožňujú ísť takto nízko v OSI modeli.

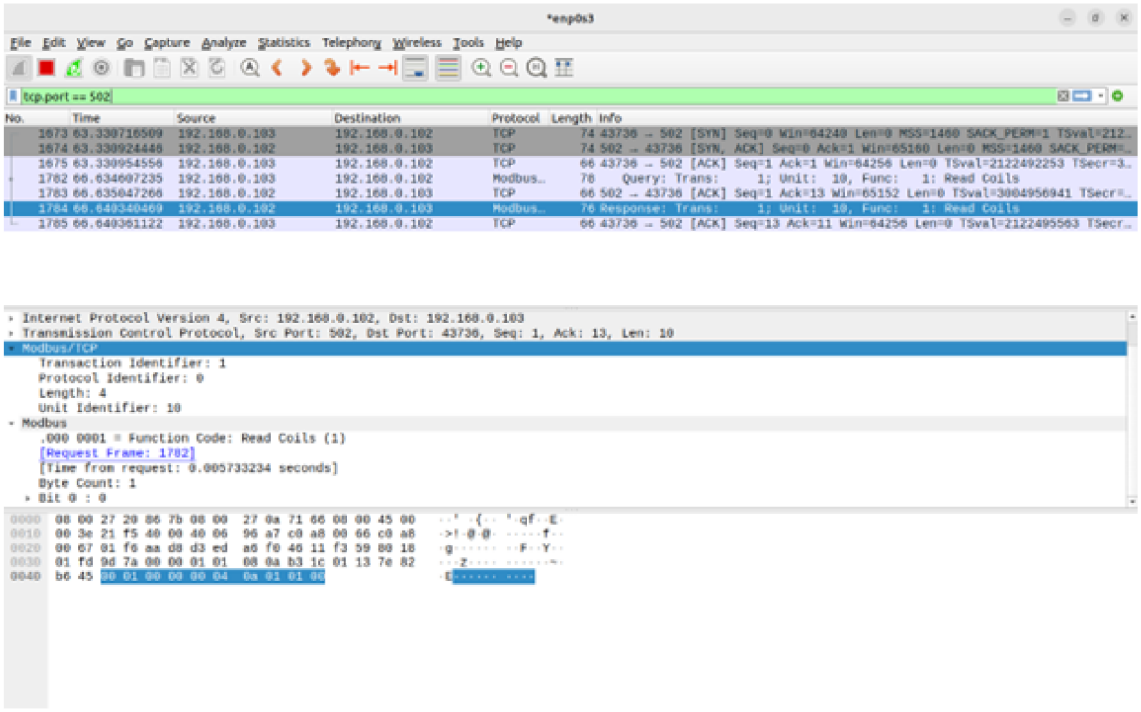
Na vytvorenie štruktúry a spracovanie dát, prenášaných pomocou protokolu profinet, boli vytvorené triedy a logika, ktorá je viac-menej zhodná s implementáciou protokolu modbus. Avšak na jeho prenos medzi zariadeniami bola využitá externá knižnica pcap4j. Tento prístup je relatívne pomalý a nie je isté, či bude zachovaný v rámci diplomovej práce a či bude knižnica pcap4j využitá v rámci implementácie protokolu do prostredia OpenMUC.



Obr. 2.2: Grafické rozhranie pre Modbus Server



Obr. 2.3: Zachytená komunikácia cez program wireshark, Modbus Query



Obr. 2.4: Zachytená komunikácia cez program wireshark, Modbus Response

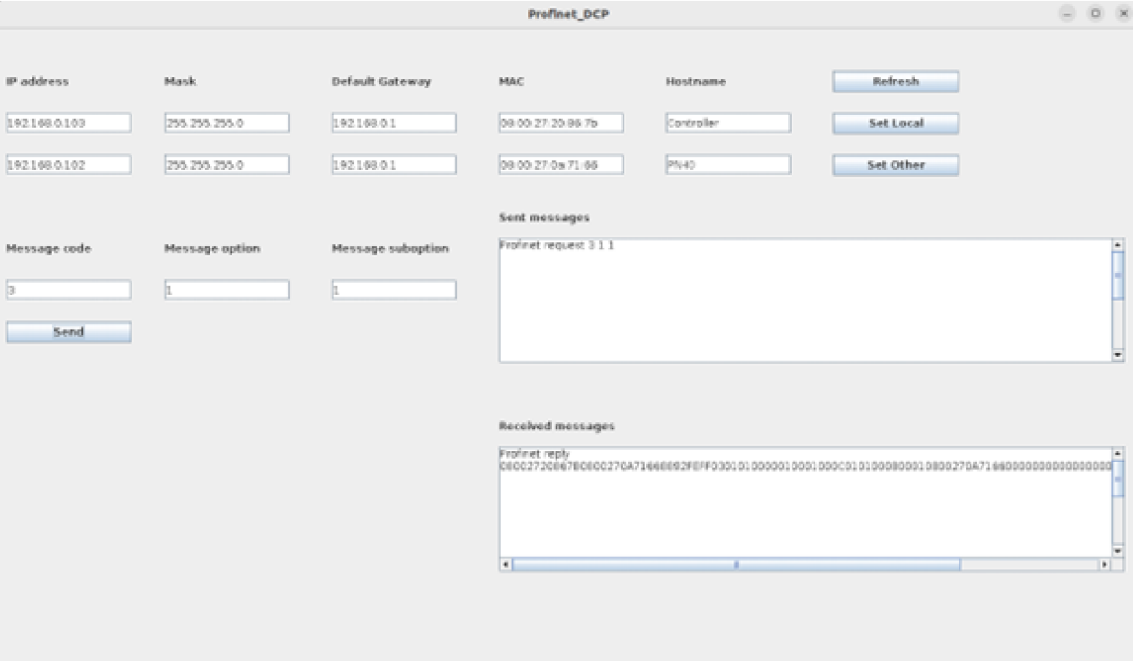
Komunikácia funguje na veľmi podobnom princípe, avšak keďže nie je nutné vytvárať žiadne kontrolované spojenie, ako v prípade protokolu modbus a jeho prenosu cez protokol TCP, nie je nutné štartovať žiaden profinet klient alebo server. Na komunikáciu medzi aplikáciami stačí odoslať profinet správu smerovanú na druhé zariadenie, ktoré na túto správu odpovie. Tento spôsob komunikácie bol overený medzi dvoma inštanciami tej istej aplikácie bežiacimi na dvoch rôznych zariadeniach, ako aj medzi aplikáciou a industriálnym PLC.

Grafické rozhranie slúži len na testovacie účely, nepredpokladá sa že bude používané v rámci diplomovej práce. Avšak na rozdiel od grafického rozhrania pri protokole modbus, je nutné zadať všetky hodnoty. To znamená IP adresu, masku a MAC adresu pre obidve strany na obidvoch inštanciách aplikácie. Výhodou je, že táto komunikácia môže byť úplne nezávislá od nastaveniach na sieťovej karte daného počítača, kde daná aplikácia beží – zadané hodnoty sa nemusia zhodovať s hodnotami na sieťovej karte.

Pre inicializáciu komunikácie je nutné vybrať typ správy, pomocou hodnôt „message code“, „message option“ a „message suboption“. Napríklad kombinácia 3 1 1 slúži na získanie MAC adresy.

Na nasledujúcich obrázkoch je možné vidieť grafické rozhranie pre profinet 2.5 2.5. Aktivácia prebieha pomocou určení sieťových parametrov a kliknutí na tlačidlá "Set Local"pre stanicu, na ktorej táto aplikácia beží a "Set Other"pre stanicu, s

ktorou sa má komunikovať. Aj keď profinet ako taký pracuje formou kliet a server, tieto stanice budú rovnocenné a obidve z nich vedia vykonávať obidve roly. Klient je vždy tá strana, ktorá odošle správu kliknutím na tlačidlo "Send". Na ďalších obrázkoch 2.7 2.8 je možné vidieť zachytenú komunikáciu, ktorá sa vykoná kliknutím na tlačidlo "Send".



Obr. 2.5: Grafické rozhranie pre Profinet Klient

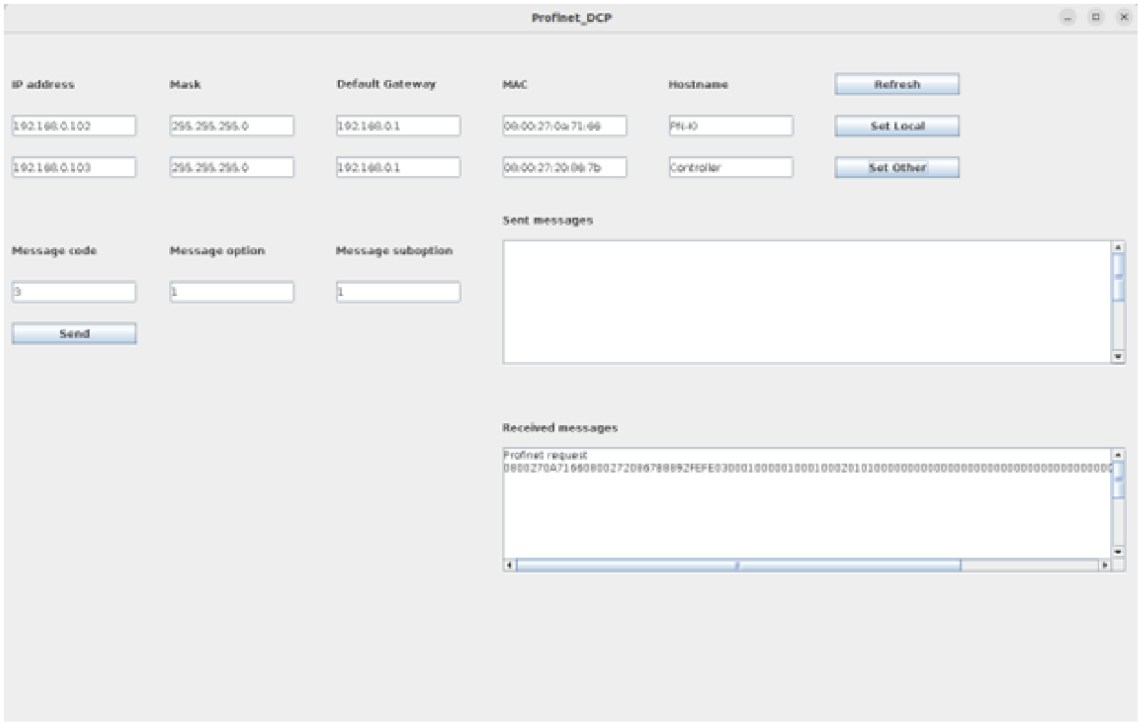
2.3 Ethertnet/IP

Protokol ethernet/IP je zo spomínaných protokolov pravdepodobne najkomplikovanejší na implementáciu. Rovnako ako protokol modbus využíva na jeho prenos protokol TCP.

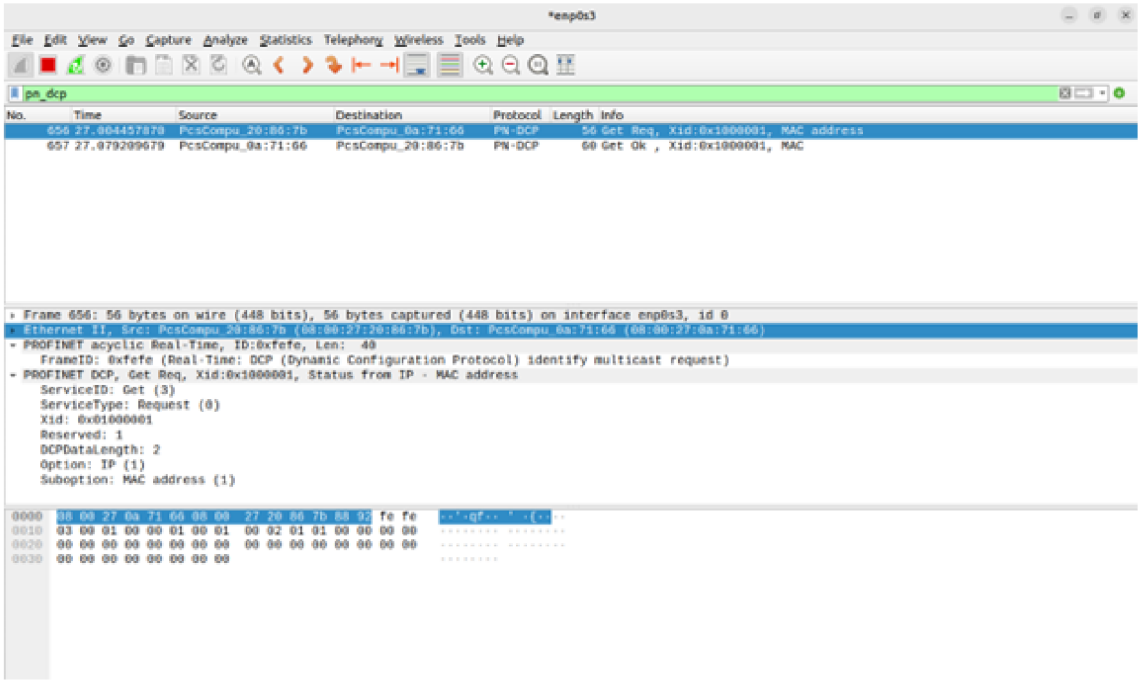
Protokol ethernet/IP veľmi úzko súvisí s protokolom CIP. Je zrejme zbytočné implementovať len jeden z týchto protokolov, nakoľko by jeho funkcionalita bola veľmi obmedzená.

Protokol CIP avšak počas semestrálnej práce implementovaný nebol. Z tohto dôvodu prenáša implementácia protokolu ethernet/IP len takzvané „dummy“ dáta, ktoré predstavujú dopredu vytvorenú štruktúru protokolu CIP, na ktorú je následne odpovedané inou podobnou štruktúrou, ktorá predstavuje korektnú odpoveď.

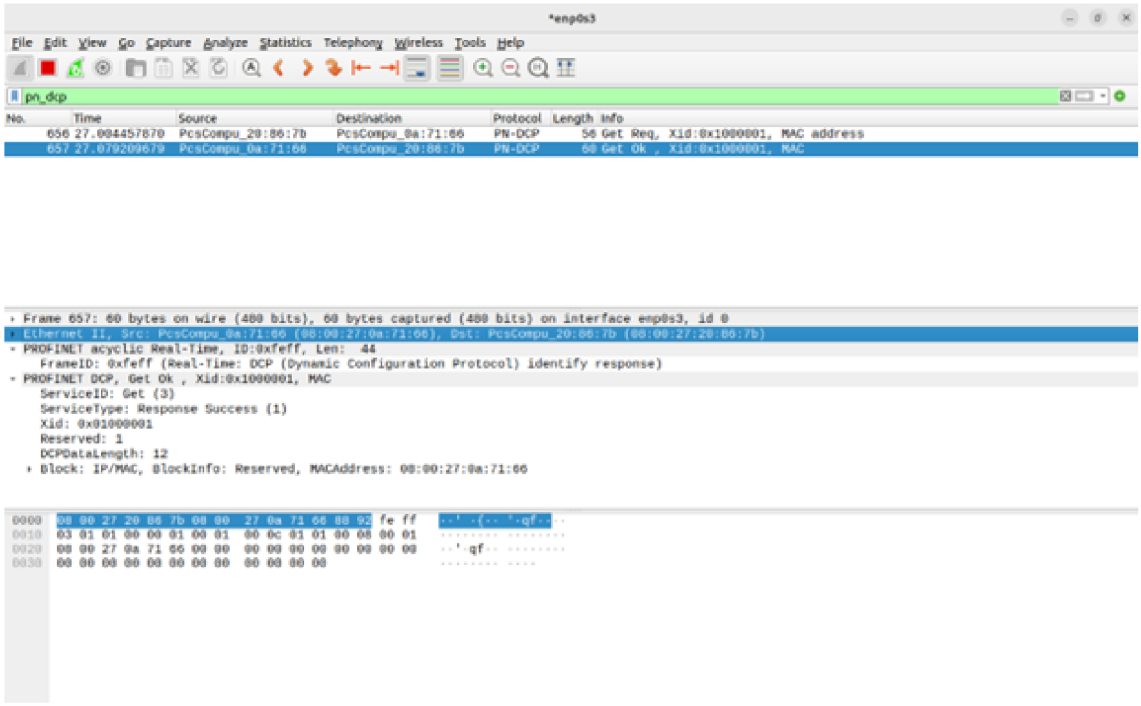
Nakoľko sú protokoly modbus a ethernet/IP implementované v rámci rovnakej aplikácie, obi dva fungujú v jednom grafickom rozhaní.



Obr. 2.6: Grafické rozhranie pre Profinet Server

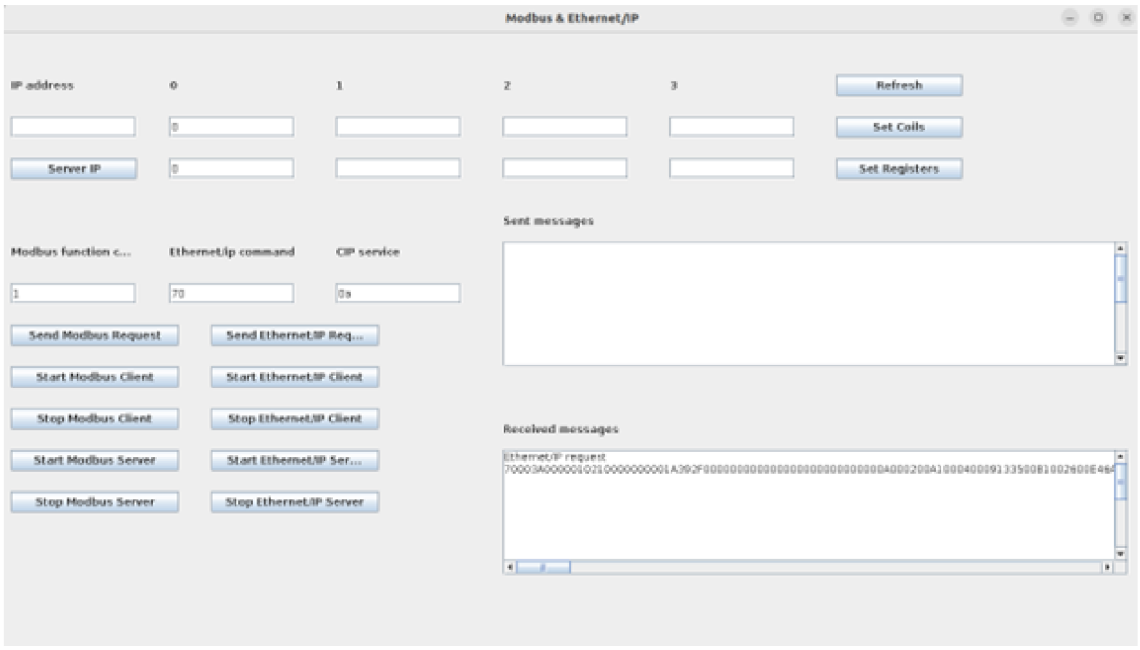


Obr. 2.7: Zachytená komunikácia cez program wireshark, Profinet Request

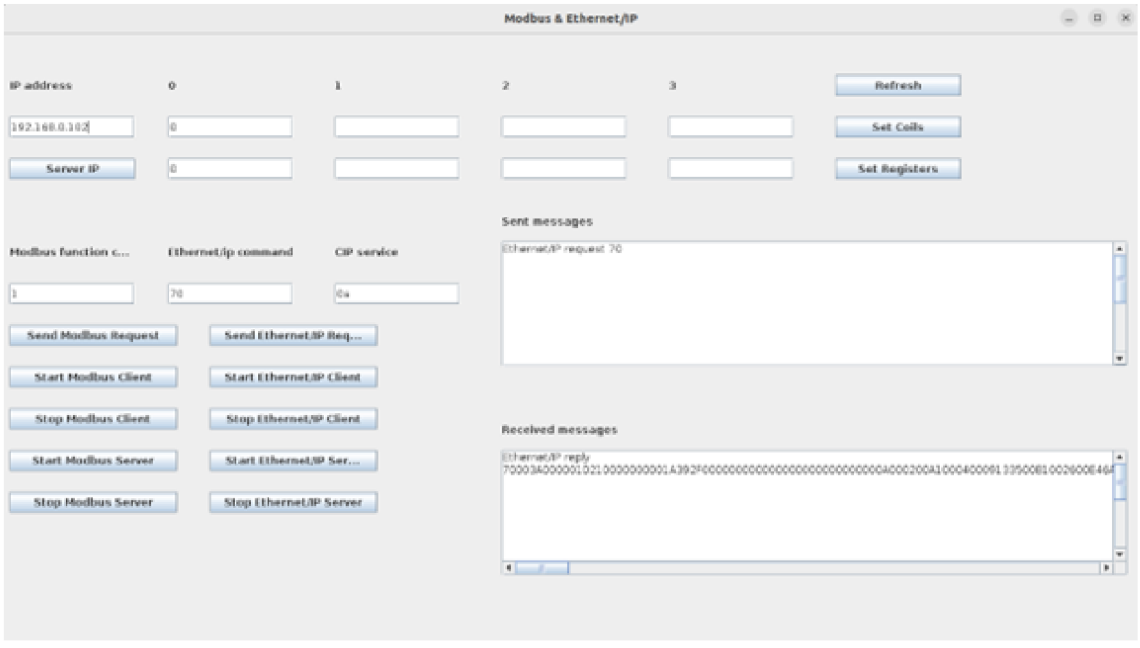


Obr. 2.8: Zachytená komunikácia cez program wireshark, Profinet Response

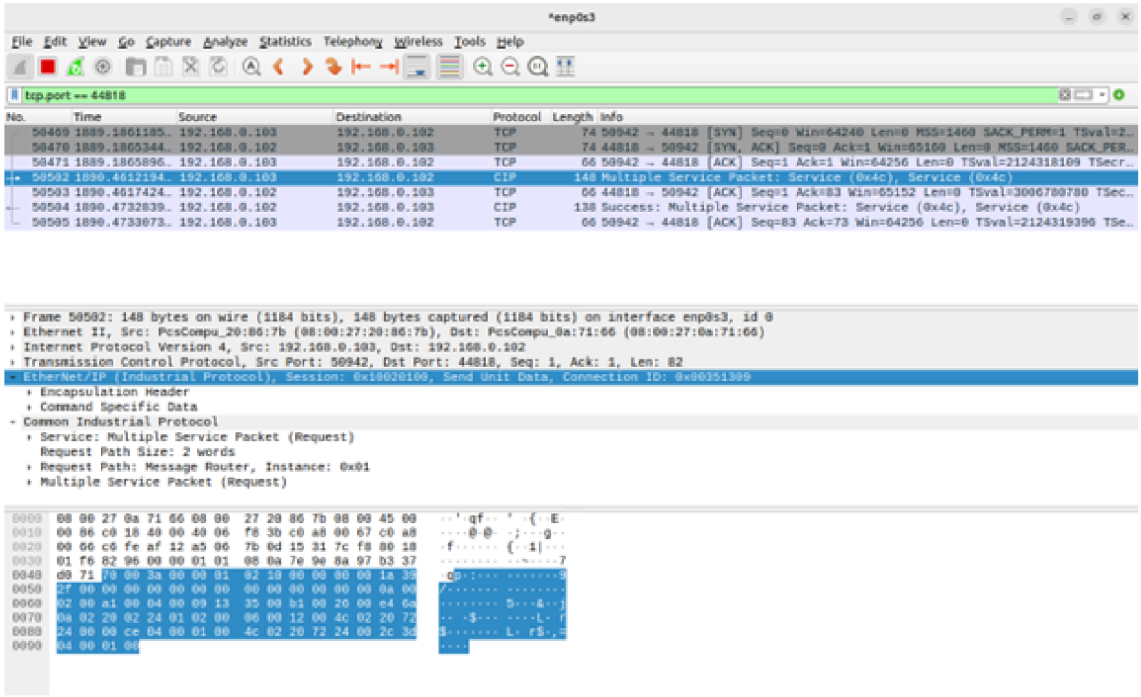
Na nasledujúcich obrázkoch je možné vidieť grafické rozhranie pre ethernet/IP klient 2.9, ktoré sa rovnako ako pri protokole modbus aktivuje určením IP adresy a kliknutím na tlačidlo "Server IP", a následným kliknutím na tlačidlo "Start Ethernet/IP Client". Grafické rozhranie pre ethernet/IP server 2.10 je rovnaké, aktivuje sa kliknutím na tlačidlo "Start Ethernet/IP Server". Znovu je nutné, aby bol server aktivovaný ako prvý. Aj keď je v rámci grafického rozhrania určovať aj typ správy pre protokol CIP, zatiaľ existuje len jedna (0x0a) a aj tú nie je možné ďalej upravovať. Na ďalších obrázkoch 2.11 2.12 je možné vidieť zachytenú komunikáciu, ktorá sa vykoná kliknutím na tlačidlo "Send Ethernet/IP Request"na strane klienta.



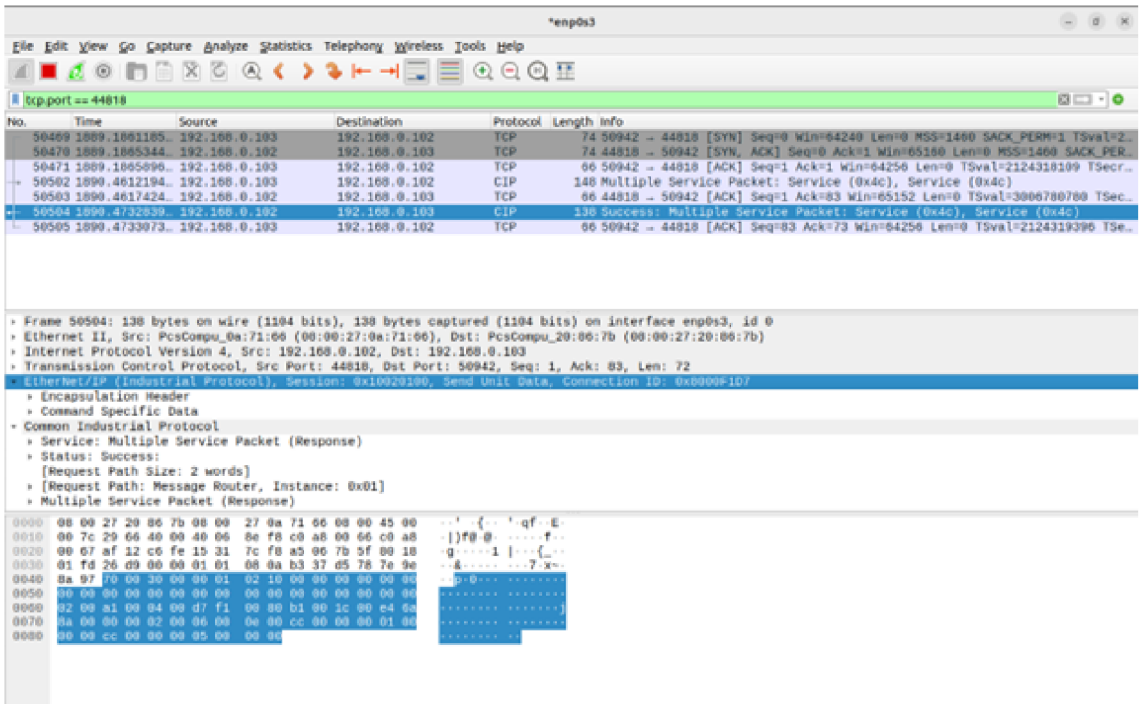
Obr. 2.9: Grafické rozhranie pre Ethernet/IP Klient



Obr. 2.10: Grafické rozhranie pre Ethernet/IP Server



Obr. 2.11: Zachytená komunikácia cez program wireshark, Ethernet/IP Request



Obr. 2.12: Zachytená komunikácia cez program wireshark, Ethernet/IP Response

3 Systém OpenMUC

V posledných rokoch viedol rýchly rast internetu vecí (IoT) k rastúcemu dopytu po efektívnych a škálovateľných monitorovacích a riadiacich systémoch. OpenMUC (Open Metering and Control), open-source framework, sa ukázal ako výkonné riešenie pre vývoj flexibilných a prispôsobiteľných IoT/SCADA systémov. OpenMUC poskytuje modulárnu a rozšíriteľnú platformu na získavanie, spracovanie a vizualizáciu dát, vďaka čomu je ideálnou voľbou pre rôzne aplikácie vrátane správy inteligentných sietí, automatizácie budov a riadenia priemyselných procesov.

3.1 Vlastnosti a možnosti OpenMUC

OpenMUC ponúka celý rad schopností, ktoré prispievajú k jeho efektívnosti a flexibilite.

3.1.1 Zber dát

OpenMUC podporuje získavanie údajov z rôznych zdrojov, vrátane senzorov, meračov a akčných členov, poskytovaním širokej škály možností pripojenia. Ponúka množstvo protokolov ako Modbus, OPC UA, MQTT a REST, čo umožňuje bezproblémovú integráciu s rôznymi zariadeniami a systémami. Modulárny dizajn OpenMUC umožňuje pridanie vlastných ovládačov pre špecifický hardvér, čím sa zabezpečí kompatibilita s rôznymi zdrojmi údajov[9].

3.1.2 Spracovanie dát

OpenMUC uľahčuje spracovanie údajov v reálnom čase prostredníctvom výkonného dátového potrubia. Framework ponúka kolekciu komponentov transformácie a analýzy údajov vrátane filtrovania, agregácie a matematických operácií. Tieto komponenty možno jednoducho konfigurovať a spájať tak, aby tvorili komplexné spracovateľské reťazce umožňujúce manipuláciu a obohatenie získaných údajov. Okrem toho OpenMUC podporuje vykonávanie užívateľom definovaných skriptov a algoritmov, čo umožňuje vývojárom implementovať vlastnú logiku spracovania[9].

3.1.3 Ukladanie dát

OpenMUC poskytuje flexibilné možnosti pre ukladanie a pretrvávanie údajov. Podporuje rôzne databázy, ako PostgreSQL, MySQL, InfluxDB a Apache Cassandra, čo umožňuje používateľom vybrať si najvhodnejšie riešenie pre ich špecifické požiadavky. Táto všestrannosť umožňuje efektívne narábanie s veľkými objemami údajov

generovaných systémami internetu vecí. OpenMUC navyše podporuje kompresiu a šifrovanie údajov, čím zaisťuje integritu a bezpečnosť údajov[9].

3.1.4 Vizualizácia a používateľské rozhranie

OpenMUC ponúka komplexné nástroje na vizualizáciu dát a vývoj používateľského rozhrania. Framework podporuje rôzne vizualizačné knižnice a widgety, ktoré umožňujú vytváranie interaktívnych grafov, grafov a tabuliek. Okrem toho OpenMUC poskytuje API a knižnice na vývoj samostatných aplikácií a integráciu externých systémov s platformou OpenMUC[9].

3.1.5 Rozšíriteľnosť a integrácia

Modulárna architektúra OpenMUC a rozsiahly systém zásuvných modulov umožňujú jednoduchú rozšíriteľnosť a integráciu s externými systémami. Framework poskytuje dobre definované API a dokumentáciu pre vývoj zásuvných modulov, čo umožňuje používateľom pridávať vlastné funkcie do základného systému. OpenMUC tiež podporuje integráciu s externým softvérom, ako sú systémy SCADA (Supervisory Control and Data Acquisition) a systémy plánovania podnikových zdrojov (ERP), prostredníctvom štandardizovaných protokolov a API[9].

3.2 Architektúra OpenMUC

Jadrom OpenMUC je modulárna architektúra, ktorá umožňuje používateľom prispôsobiť systém ich špecifickým potrebám. Platforma pozostáva z množstva modulov, z ktorých každý zodpovedá za špecifickú funkciu, ako je zber údajov, spracovanie údajov alebo komunikácia s externými systémami. Moduly sú prepojené cez systém správ, ktorý umožňuje komunikáciu medzi nimi.

Jednou z kľúčových vlastností OpenMUC je jeho podpora pre širokú škálu komunikačných protokolov vrátane Modbus a MQTT. To umožňuje používateľom komunikovať so širokou škálou zariadení a systémov bez ohľadu na ich komunikačné protokoly.

3.3 Implementácia protokolov do prostredia OpenMUC

Integrácia nového protokolu do rámca OpenMUC rozširuje jeho možnosti a umožňuje bezproblémovú komunikáciu so širším spektrom zariadení a systémov. Implementácia nového protokolu zahŕňa niekoľko kľúčových krokov na zabezpečenie úspešnej

integrácie a kompatibility s existujúcim rámcom.

3.3.1 Analýza a špecifikácia protokolu

Prvým krokom pri implementácii nového protokolu v rámci OpenMUC je vykonanie komplexnej analýzy špecifikácií a požiadaviek protokolu. Pochopenie štruktúry správ protokolu, dátových formátov a komunikačných vzorov je rozhodujúce pre návrh efektívnej integračnej stratégie.

3.3.2 Vývoj modulu protokolu

Po dokončení analýzy protokolu je ďalším krokom vývoj modulu protokolu v rámci OpenMUC. Modul by mal byť navrhnutý tak, aby zvládal úlohy špecifické pre protokol, ako je analýza správ, kódovanie a dekódovanie.

3.3.3 Integrácia s architektúrou OpenMUC

Integrácia nového modulu protokolu do architektúry OpenMUC vyžaduje starostlivé zváženie modulárnej štruktúry rámca. Modul by mal byť bezproblémovo integrovaný s existujúcimi modulmi komunikácie a spracovania dát OpenMUC. To zahŕňa definovanie vhodných rozhraní a komunikačných kanálov na uľahčenie interakcie medzi modulom protokolu a inými komponentmi.

3.3.4 Konfigurácia a registrácia protokolu

Keď je modul protokolu vyvinutý a integrovaný, mal by byť konfigurovateľný v rámci OpenMUC. To zahŕňa poskytovanie možností na nastavenie parametrov špecifických pre protokol, ako sú nastavenia pripojenia, formáty údajov a stratégie spracovania správ. Okrem toho musí byť protokolový modul zaregistrovaný v OpenMUC, čo umožňuje užívateľom jednoducho vybrať a nakonfigurovať požadovaný protokol pre ich monitorovacie a riadiace aplikácie.

3.3.5 Testovanie a overovanie

Na zabezpečenie správneho fungovania integrácie nového protokolu je nevyhnutné dôkladné testovanie a validácia. To zahŕňa vytváranie testovacích scenárov, ktoré pokrývajú rôzne funkcie protokolu a okrajové prípady. Modul by mal byť prísne testovaný na integritu údajov, spoľahlivosť a výkon, aby bola zaručená jeho kompatibilita s rámcom OpenMUC a schopnosť efektívne zvládnuť scenáre v reálnom svete.

Nasledovaním týchto krokov, implementácia nového protokolu do rámca OpenMUC zvyšuje jeho všestrannosť a rozširuje možnosti pripojenia. Používatelia môžu využiť novo pridaný protokol na prepojenie so širším rozsahom zariadení, systémov a komunikačných štandardov, čím sa otvárajú možnosti pre rôzne monitorovacie a riadiace aplikácie v rámci ekosystému OpenMUC.

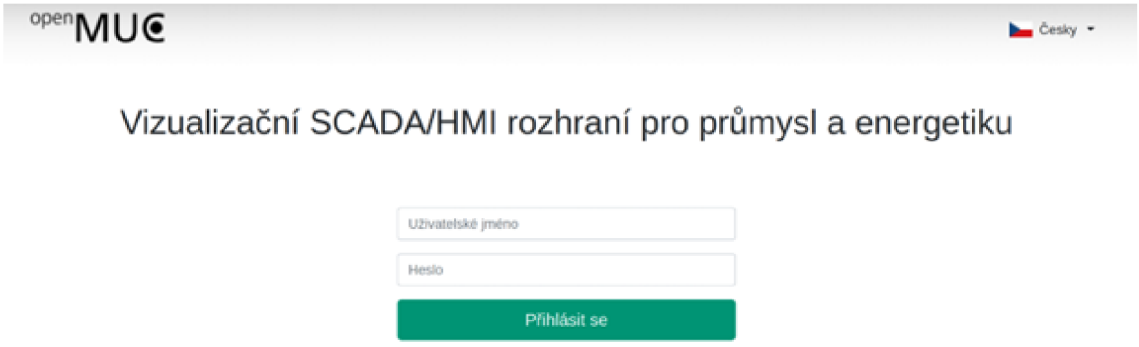
3.3.6 Zhrnutie

OpenMUC je výkonná a flexibilná platforma pre budovanie distribuovaných monitorovacích a riadiacich systémov. Jeho modulárna architektúra a podpora širokej škály komunikačných protokolov z neho robia atraktívnu možnosť pre vývojárov, ktorí chcú vytvoriť prispôsobené riešenia monitorovania a riadenia. Vďaka svojej rastúcej komunite vývojárov a používateľov bude OpenMUC pravdepodobne aj naďalej dôležitým nástrojom na budovanie inteligentných systémov v rôznych aplikáciách.

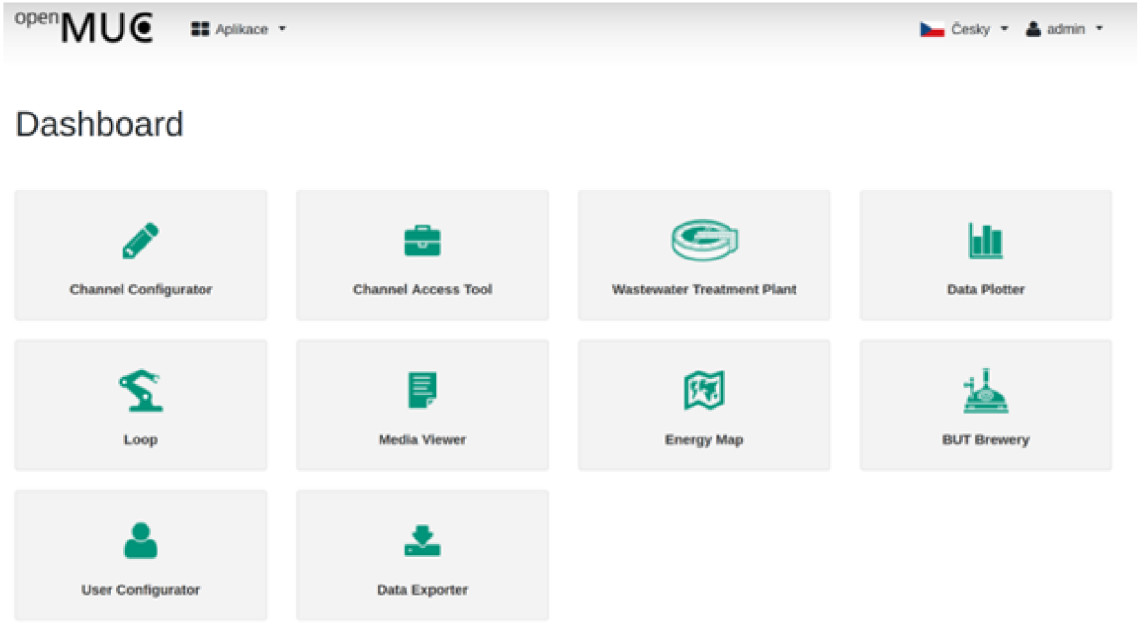
4 Implementácia

Za predpokladu, že inicializácia systému prebehla bez problémov, inštancia openMUC sa spustí po spustení projektu v prostredí Intellij Idea. Je možné spustiť systém aj cez iné prostredia, avšak Intellij Idea umožňuje build a debug kódu bez volaní z konzole. Používatelia k nemu môžu pristupovať zadaním adresy Local Loopback na porte 8888 (127.0.0.1:8888) v ľubovoľnom webovom prehliadači.

Po zadaní tejto url budú používatelia presmerovaní do okna. Ako prihlasovacie údaje je možné použiť kombináciu mena a hesla: admin/admin Prihlasovací formulár je možné vidieť na obrázku 4.1. Samotné prostredie je možné vidieť na obrázku 4.2.



Obr. 4.1: Prihlasovanie sa do prostredia OpenMUC



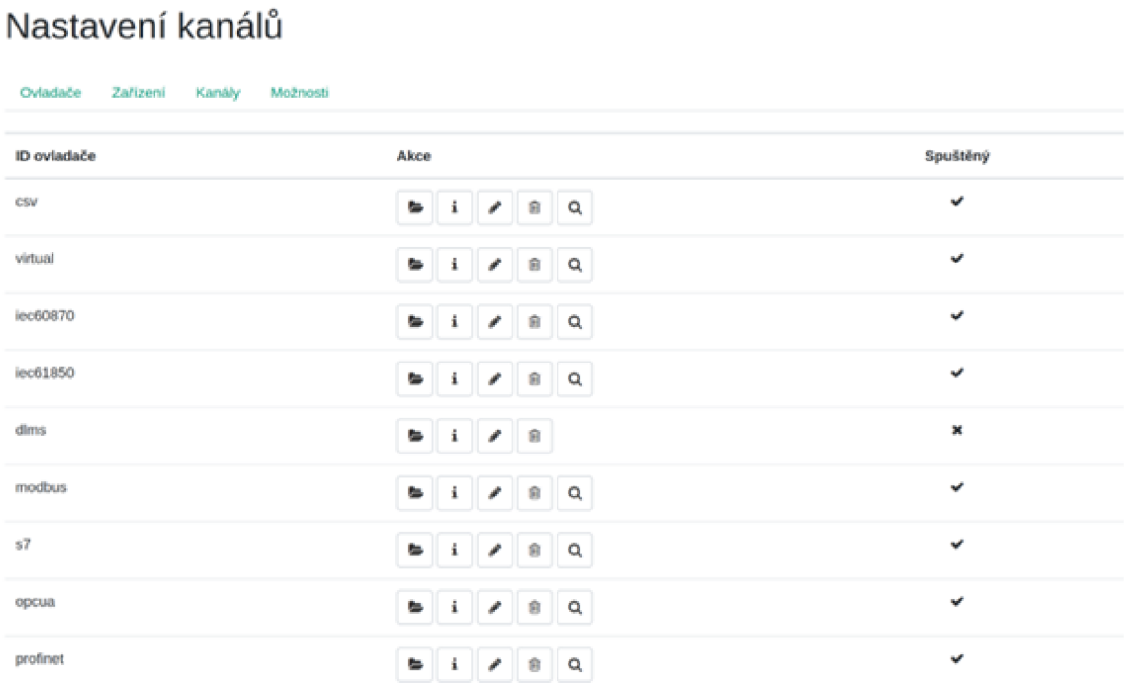
Obr. 4.2: Prostredie OpenMUC

4.1 Štruktúra komponentov

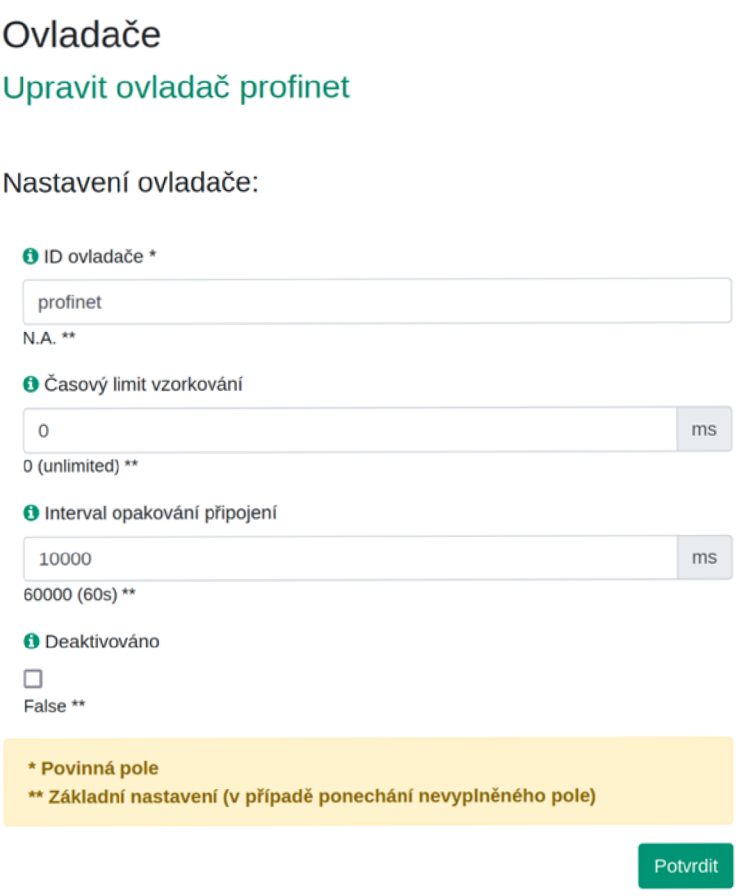
OpenMUC ako taký sa striktne drží štruktúry ovládač -> zariadenie -> kanál (po anglicky: driver -> device -> channel). To znamená, že na začiatok komunikácie je nutné nastaviť prvky v tomto poradí. Nastavenia sa vykonávajú pod tlačidlom „Channel Configurator“.

4.2 Ovládač (Driver)

Ovládač ako taký predstavuje samotný protokol, pomocou ktorého sa následne bude komunikovať. Väčšina funkcionality ako aj jeho nastavení je vykonaná priamo v kóde prostredia OpenMUC. Je len nutné sa uistiť, že nie je vypnutý. Na obrázku 4.3 je možné vidieť zoznam nainštalovaných ovládačov, a na obrázku 4.4 nastavenie ovádača pre protokol profinet.



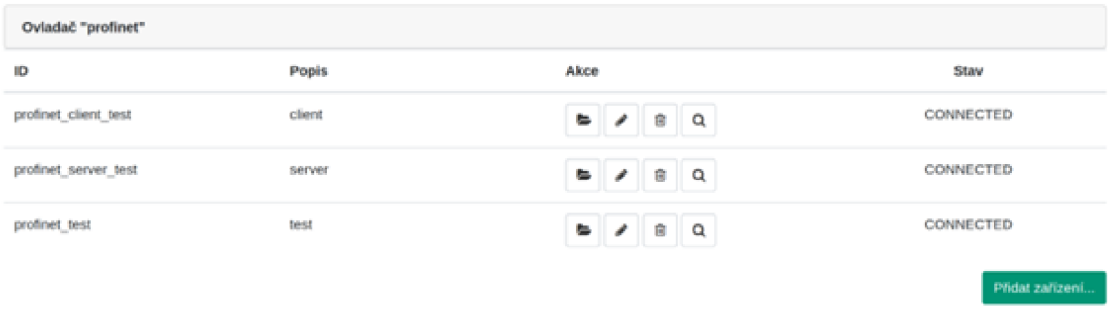
Obr. 4.3: Zoznam ovádačov



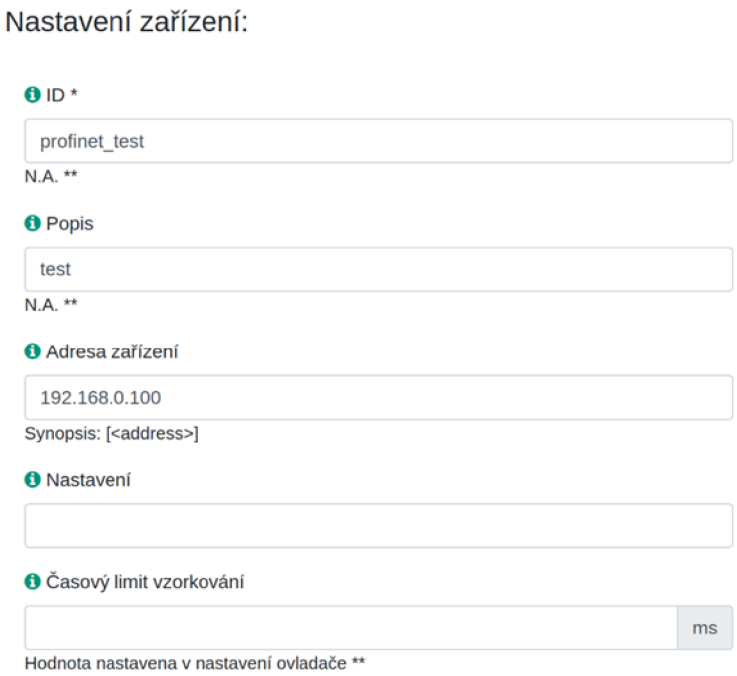
Obr. 4.4: Nastavenie ovládača pre profinet

4.3 Zariadenie (Device)

Zariadenie predstavuje pripojené zariadenie, s ktorým bude následne OpenMUC cez daný protokol komunikovať. Zariadenia sa vytvárajú pre každý ovládač samostatne, pričom ich je možné mať vytvorených naraz viacero. OpenMUC ukáže aj stav pripojenia pre jednotlivé zariadenia. Na obrázku 4.5 je možné vidieť zoznam vytvorených zariadení, pre protokol profinet. Na obrázku 4.6 je nastavenie jedného z týchto zariadení. IP adresa je v tomto prípade dobrovoľná.



Obr. 4.5: Zoznam zariadení používajúcich profinet

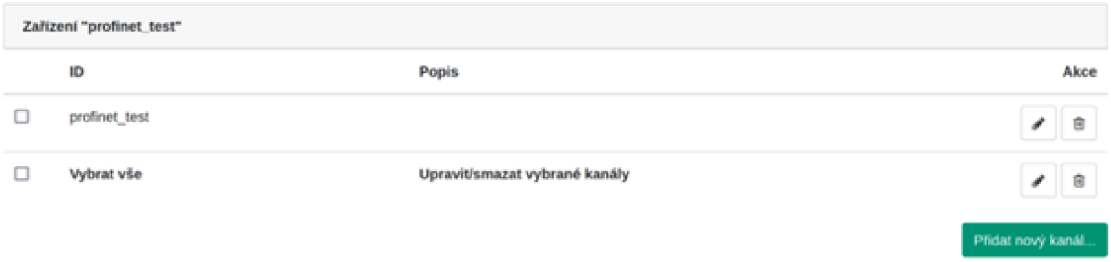


Obr. 4.6: Nastavenie zariadenia

4.4 Kanál (Channel)

Zariadenie môže mať definovaných niekoľko kanálov. Ich správanie je definované v rámci samotného protokolu, ale väčšinou tieto kanály vytvoria spojenie s daným zariadením a následne periodicky posielajú určitý typ správy, ktorý je v rámci daného kanálu definovaný. Jeho nastavenie je už veľmi špecifické, nakoľko každý protokol potrebuje iné parametre.

Každé nastavenie môže byť v rámci implementácie daného protokolu definované ako „povinné“. V prípade, ak takéto nastavenie chýba, kanál alebo zariadenie nebude vôbec fungovať. Z hľadiska grafického rozhrania je jednen kanál rovný jednej vyčítanej hodnote. Na obrázku 4.7 môžeme vidieť výber zariadenia, pre ktoré chceme nastavovať jednotlivé kanály a na obrázku 4.8 nastavenie samotného kanálu.



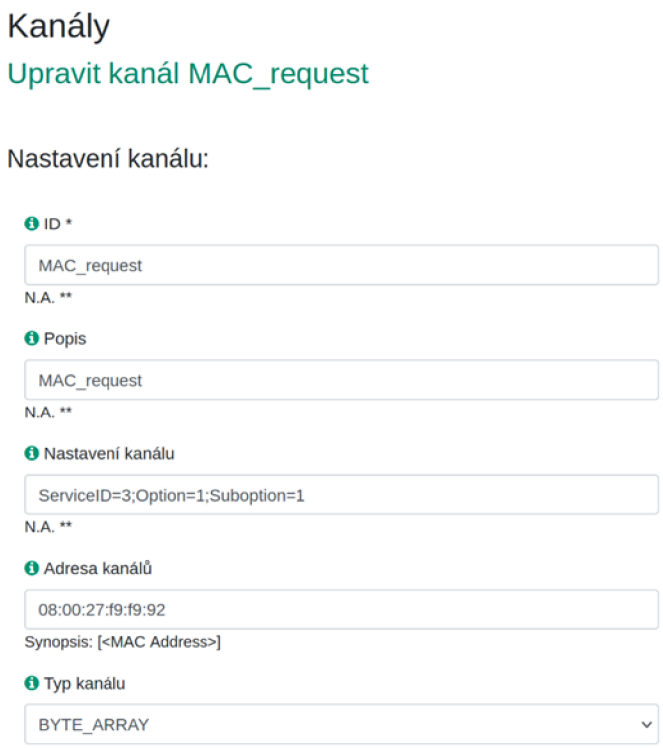
Obr. 4.7: Výber zariadenia

4.5 Implementácia protokolu Modbus

Protokol modbus už implementovaný v rámci prostredia OpenMUC bol. Ide o implementáciu postavenú na knižnici j2mod. Nakoľko nemá veľmi zmysel implementovať ten istý protokol znovu, bolo v rámci diplomovej práce vykonané testovanie tejto implementácie a následné opravy prípadných chýb.

4.6 Implementácia protokolu Profinet

Implementácia protokolu profinet sa ukázala ako značne náročná. Existujúca dokumentácia tohto protokolu je relatívne obmedzená. Nakoľko ale ide o protokol fungujúci predovšetkým nad druhou vrstvou OSI, nie je nutné na jeho fungovanie výrazné množstvo logiky alebo algoritmov. Zoznam existujúcich správ sa ukázal ako postačujúci. Práca na druhej úrovni OSI sa rovnako ukázala veľmi problematická, nakoľko ide o niečo, čo samotná java neumožňuje. V rámci testovacej aplikácie bol na toto



Obr. 4.8: Nastavenie kanálu

použitý systém pcap pomocou knižnice pcap4j. Tento prístup sa ukázal príliš pomalý na implementáciu v prostredí OpenMUC.

Preto bolo využité takzvané java native interface (jni), kde bol kód vykonávajúci funkcionalitu spojení na druhej vrstve OSI napísaný v jazyku C, a java v prostredí OpenMUC len následne volala potrebné funkcie. Táto C knižnica je súčasťou práce a je poskytnutá v prílohách.

Ďalším problémom bolo to, že k nižším vrstvám OSI a takzvaným RAW Socketom má len root, preto na vykonávanie tohto kódu je nutné mať root práva. V rámci testovania bol OpenMUC spustený ako root. Avšak, nakoľko je všetok kód, ktorý tieto práva potrebuje odizolovaný do knižnice napísanej v jazyku C, OpenMUC ako taký bude aj po implementácii protokolu profinet stále fungovať aj bez týchto práv.

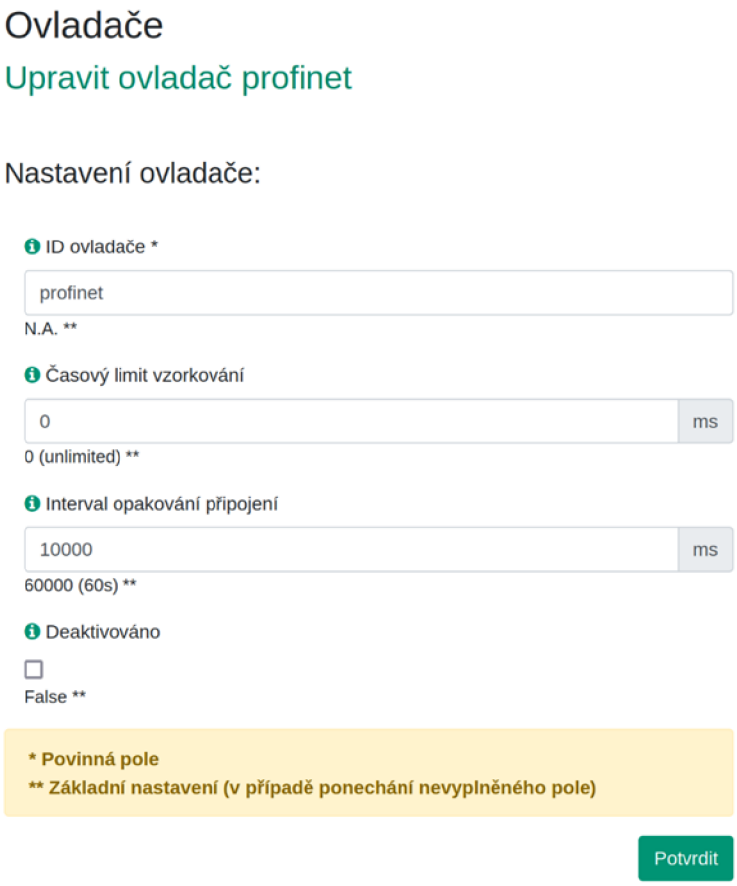
Po jeho spustení ako bežný používateľ budú všetky doterajšie protokoly fungovať rovnako ako doteraz, ale nebude schopný posielať a čítať profinetové správy.

4.6.1 Nastavenie

Nastavenie pre protokol profinet prebieha rovnako, ako to bolo popísané v predošlých kapitolách pre iné protokoly.

4.6.2 Ovládač (Driver)

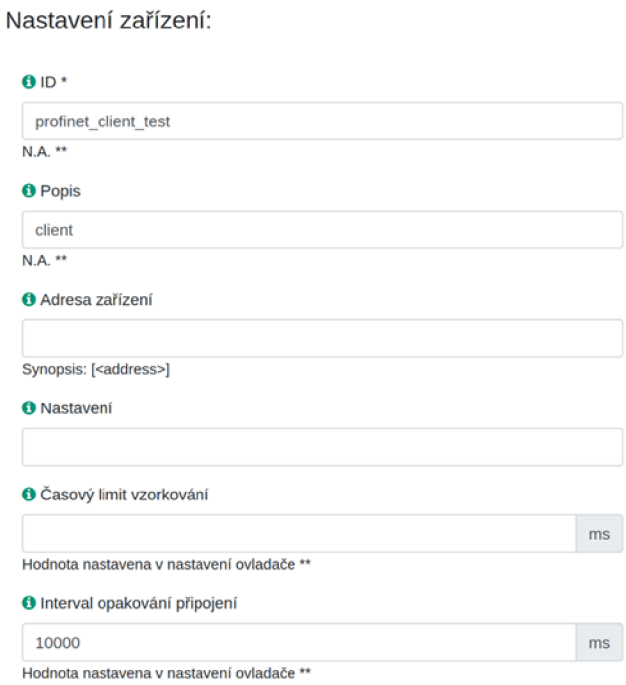
Ovládač nie je nutné nijako upravovať. Je len nutné sa uistiť, že nie je vypnutý. V rámci testovania je ale výhodné nastaviť interval opakovania pripojenia na nižšiu hodnotu, nakoľko nechceme aby bolo nutné v prípade problémov čakať 60 sekúnd na obnovu. Nastavenie vidíme na obrázku 4.9.



Obr. 4.9: Nastavenie ovládača pre protokol profinet

4.6.3 Zariadenie (Device)

Zariadenie rovnako nepotrebuje takmer žiadne nastavenia. Dokonca nie je nutná ani IP adresa, nakoľko komunikácia prebieha na druhej vrstve OSI pomocou MAC adries. Samozrejme, je výhodné nastaviť si jednoznačné ID a Popis, nech sa toto zariadenie dá v systéme jednoducho detekovať. Takéto nastavenie vidíme na obrázku 4.10.



Obr. 4.10: Nastavenie zariadenia pre protokol profinet

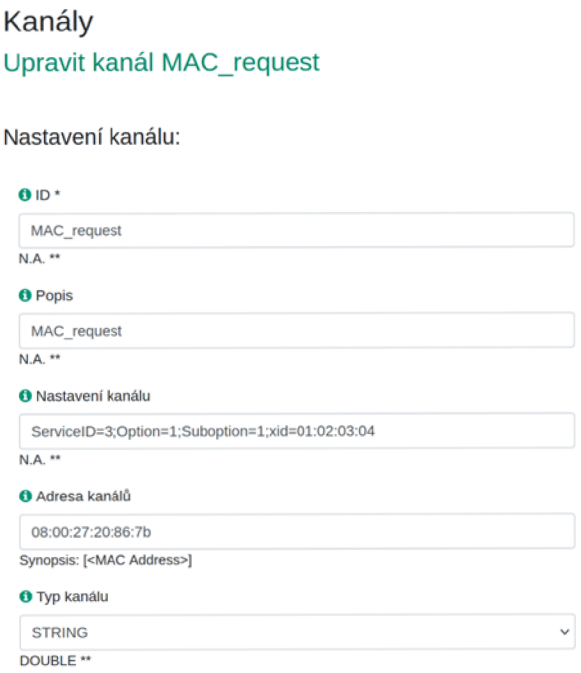
4.6.4 Kanál (Channel)

Nastavenie kanálu určuje takmer všetko. Povinná je MAC adresa v hexadecimálnom formáte oddelenom dvojbodkami. Samozrejme je nutné určiť typ správy, ktorú má kanál posielať. Ako príklad je uvedená správa „GET Mac address“, kde Service ID = 3, Option = 1 a Subobtion = 1. V rámci profinetu sa správy typu GET (Service ID = 3) zvyknú posielať cyklicky. To znamená že server posiela tú istú správu opakovane – pýta sa daného zariadenia neustále to isté. Toto je niečo, na čo sa OpenMUC veľmi dobre hodí. Iné typy správ sa takýmto spôsobom väčšinou neposielajú. Avšak implementácia protokolu profinet umožňuje posielanie akýchkoľvek správ, vrátane napríklad správy Service ID = 4, Option = 5 a Subobtion = 6, ktorá predstavuje úplné resetovanie daného zariadenia (anglicky factory reset). Pokojne aj cyklicky. Nastavenie kanálu je na obrázku 4.11.

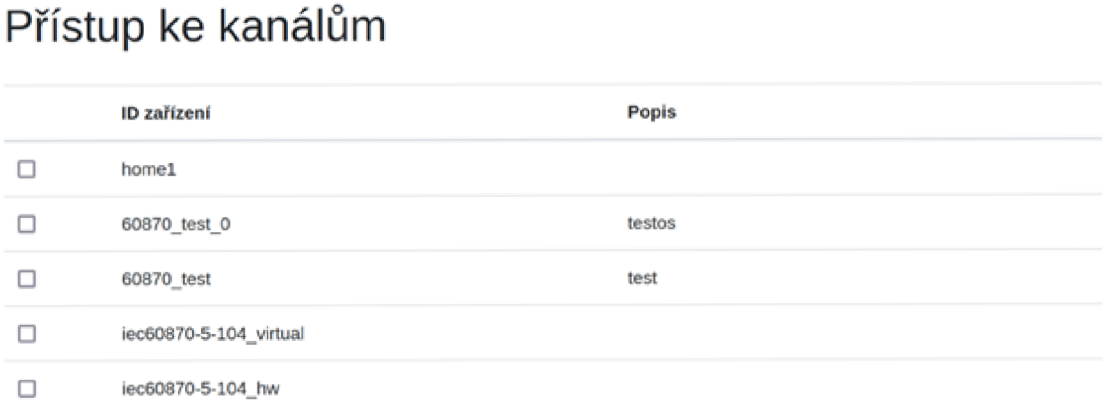
4.6.5 Prístup ku kanálom

Tlačidlo „Channel Access Tool“ umožňuje prístup ku kanálom a prípadne sledovanie ich správania. V zozname stačí nájsť zariadenie, ktoré chceme sledovať. Tento zoznam je možné vidieť na obrázkoch 4.12 a 4.13.

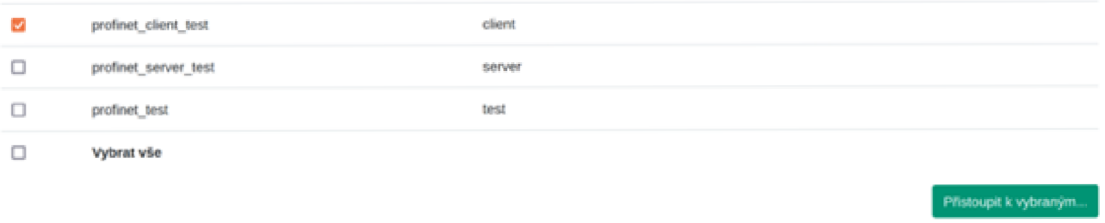
Je možné pripojiť sa k viacerým zariadeniam naraz, dokonca aj k zariadeniam pre rôzne protokoly.



Obr. 4.11: Nastavenie kanálu pre protokol profinet



Obr. 4.12: Výber zariadenia



Obr. 4.13: Samotný výber

4.7 Programové riešenie

OpenMUC vyžaduje štruktúru, ktorej je nutné sa striktne držať. Aj napriek tomu, implementácia jednotlivých protokolov v rámci prostredia OpenMuc je veľmi rozdielna. Čo je nutné, je implementácia Ovládača, Zariadenia a Kanála. Tohoto sa dosiahne pomocou rozšírenia už existujúcich tried "DriverService", "DriverDevice", "DriverChannel", prípadne "DriverActivator".

Veľmi dôležitými sú aj samotné premenné a metódy. Pristupuje sa k nim pomocou anotácií. Údaje v grafickom rozhraní vieme určiť, prípadne ich odtiaľ prečítať napríklad pomocou anotácie @Component a @Driver.

Dôležité sú predovšetkým metódy anotované ako @Connect, @Disconnect, @Read a @Write. Príklad takéhoto kódu môžme vidieť na obrázku 4.14.

```
@Component
@Driver(id = ProfinetDriver.ID,
        name = ProfinetDriver.NAME,
        /*description = ProfinetDriver.DESCRIPTION,*/
        device =  ProfinetConnection.class)
public class ProfinetDriver extends DriverActivator implements DriverService {

    1 usage
    public static final String ID = "profinet";
    1 usage
    public static final String NAME = "profinet";
    //public static final String DESCRIPTION = "profinet protocol";

}
```

Obr. 4.14: @Component a @Driver

4.7.1 Nadviazanie spojenia (@Connect)

Táto metóda slúži na nadviazanie spojenia. Väčšina iných protokolov používa túto metódu na vytvorenie TCP alebo UDP "socket-u", ktorý sa následne používa na ďalšiu komunikáciu. Nakoľko v rámci profinet-u nič také neexistuje, komunikácia sa inicializuje zaslaním správy Ïdentify all", alebo Service ID = 4, Option = 255 a Subobtion = 255. Pokiaľ toto vykonané nebolo, niektoré PLC na profinet komunikáciu ďalej neodpovedali.

4.7.2 Ukončenie spojenia (@Disconnect)

Ukončenie spojenia je často využívané na uvoľnenie socket-u vytvoreného počas nadviazania spojenia, avšak pri profinete nič podobné neexistuje. Táto metóda je však kvôli kompatibilite so zvyškom systému napriek tomu implementovaná.

4.7.3 Čítanie (@Read)

Čítanie dát z prijatých správ tiež väčšinou prebieha pomocou socket-u vytvoreného počas nadviazania spojenia. V prípade profinetu je situácia trochu iná. Je možné poslať akúkoľvek správu kedykoľvek v podstate bez pravidiel, a zariadenie, ktorému je táto správa smerovaná by malo odpovedať. Toto uľahčuje situáciu s posielaním správ, ale komplikuje ich čítanie. Je totižto nutné detekovať správu, ktorá bola poslaná ako odpoveď, a neexistuje žiaden systém, ktorý by toto robil za nás.

Preto bola vytvorená pomocná trieda "ProfinetListener", ktorá má v rámci každého kanálu za úlohu detekovať prichádzajúce správy. Táto trieda beží ako samostatné vlákno a je zabezpečená aj jej synchronizácia.

4.7.4 Zápis (@Write)

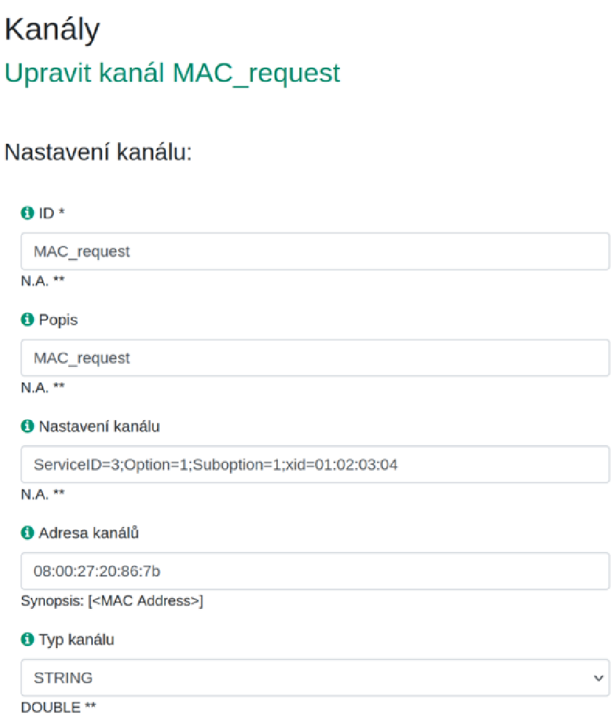
Zapisovanie dát sa používa v prípade, ak sú potrebné zmeny v už existujúcom kanály. Táto funkcia funguje len v prípade, ak bol výber správy v danom kanály nastavený na Service ID = 3, čo znamená správa typu GET. Funkcia zápisu potom pošle ekvivalentnú správu (ak taká existuje) typu SET, t. j. Service ID = 4. Tento zápis nielen zmení údaje v danom kanály, ale aj na zariadení s ktorým sa v rámci daného kanálu komunikuje.

4.8 Testovanie implementácie

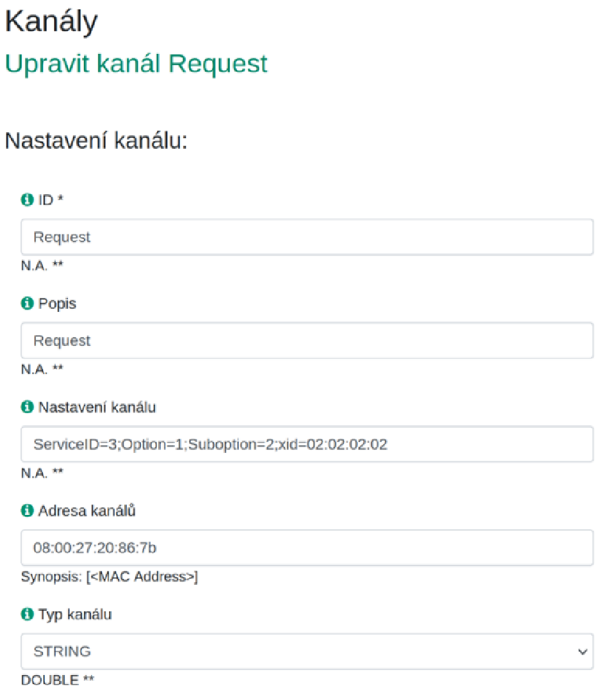
Nastavenie ovládača a zariadenia prebehlo v predošlých kapitolách. V rámci testu je toto nastavenie možné využiť. Avšak nastavenie kanálov bude potrebovať určité úpravy, nakoľko existujúce nastavenie by neposkytlo dostatočný test.

Na obrázkoch 4.15 a 4.16 je možné vidieť nastavenie kanálov pre správy Service ID = 3, Option = 1 a Subobtion = 1, a Service ID = 3, Option = 1 a Subobtion = 2. To znamená správy pre získanie MAC adresy a IP adresy. Každý z kanálov má samostatné xid, kvôli filtrovaniu týchto správ.

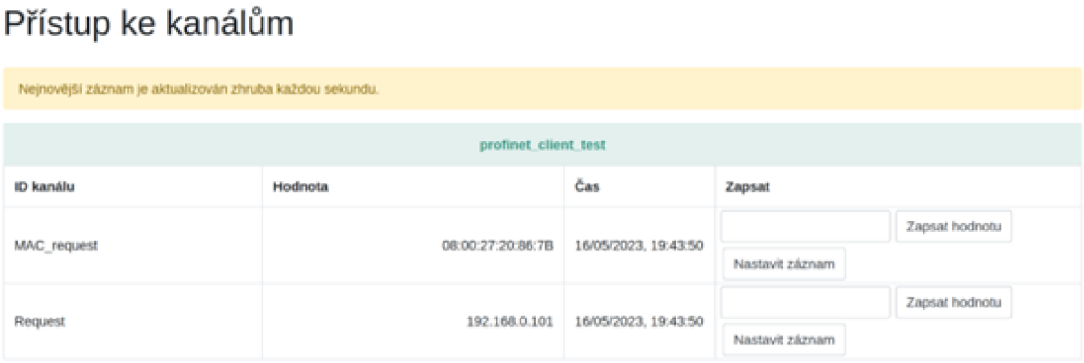
Na nasledujúcom obrázku 4.17 je možné vidieť kanály v prevádzke, a návratovú hodnotu týchto kanálov od pripojeného PLC vo formáte String.



Obr. 4.15: GET MAC address

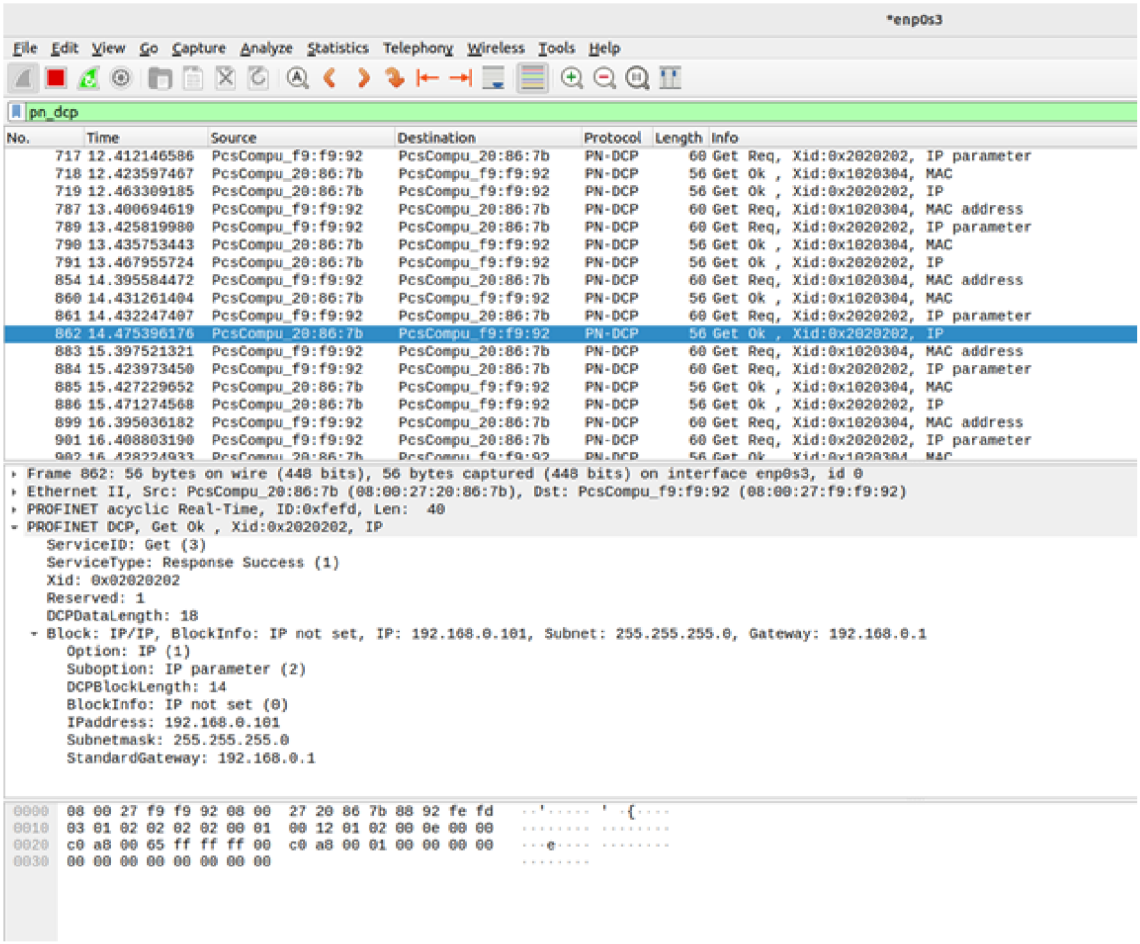


Obr. 4.16: GET IP address



Obr. 4.17: Spustené kanály

Priebeh celej komunikácie je možné vidieť na výpise z programu wireshark na obrázku 4.18. Vo výpise je možné vidieť, že každá správa využíva nastavené xid na základe čoho ich openMUC filtruje do jednotlivých kanálov.



Obr. 4.18: Správy v programe wireshark

Záver

Diplomová práca mala za cieľ predovšetkým implementovať jednotlivé protokoly do prostredia OpenMUC. OpenMUC korektne komunikuje pomocou protokolov modbus a profinet spolu s testovacími aplikáciami, ako aj s inými zariadeniami, ktoré tieto protokoly používajú. Správnosť komunikácie je dokázaná aj pomocou programu wireshark, kde je vidieť korektne detekovaná komunikácia pomocou jednotlivých protokolov a ich nadväznosť.

V prvej kapitole práce sú popísané spôsoby komunikácie, rozdiely medzi komunikáciou medzi ľuďmi a medzi strojmi, požiadavky a podmienky rôznych typov komunikácie.

Následne sú popísané priemyselné protokoly ako také, ako aj popis jednotlivých protokolov určených na implementáciu v rámci tejto práce. Predovšetkým pri popise jednotlivých správ je možné vidieť rozdiely a prípady využitia týchto protokolov.

V rámci testovacích aplikácií nebol žiaden z protokolov implementovaný kompletne. Nakoľko všetky tri obsahujú veľmi veľké množstvo typov jednotlivých správ, vždy bola implementovaná len ich podmnožina na testovanie korektnej komunikácie medzi zariadeniami.

Prostredie OpenMUC umožňovalo implementáciu všetkých protokolov.

Protokol modbus už implementovaný bol, preto bolo v rámci práce len vykonané testovanie a následne boli opravené nedostatky tejto implementácie.

Protokol profinet bol naimplementovaný úplne, aj keď jeho štruktúra je v kombinácii s prostredím OpenMUC dosť nešťastná. Nakoľko bolo nutné pristupovať k OSI vrstve 2 (ethernet), OpenMUC potreboval root práva. Java ako taká takisto takýto prístup neumožňuje. Našťastie bol celý tento prístup odizolovaný do knižnice, napísanej v jazyku C, ku ktorej OpenMUC pristupuje pomocou java natívneho rozhrania. OpenMUC síce stále potrebuje root práva na fuknčnosť profinetu, ale je plne spustiteľný aj bez nich a všetky protokoly (okrem profinetu) budú bez root práv fungovať. Protokol profinet bol testovaný voči virtuálnym zariadeniam, ako aj voči reálnym profinet PLC.

Protokol ethernet/IP bol v rámci testovania implementovaný len veľmi jednoducho. Jeho úplnú funkčnosť sa v rámci OpenMUC nepodarilo dosiahnuť, predovšetkým kvôli jeho komplexite a úplnemu nedostatku dokumentácie. Toto je spôsobené predovšetkým tým, že ethernet/IP je vyvýjaný proprietárne a jeho vývojári nechcú zbytočnú konkurenciu.

Referencie

[1] KIM, Jaewoo; LEE, Jaiyong; KIM, Jaeho; YUN, Jaeseok. M2M service platforms: Survey, issues, and enabling technologies. *IEEE communications surveys & tutorials*. 2013, roč. 16, č. 1, s. 61–76.

[2] WU, Geng; TALWAR, Shilpa; JOHNSSON, Kerstin; HIMAYAT, Nageen; JOHNSON, Kevin D. M2M: From mobile to embedded internet. *IEEE Communications Magazine*. 2011, roč. 49, č. 4, s. 36–43.

[3] SONG, Qipeng; NUAYMI, Loutfi; LAGRANGE, Xavier. Survey of radio resource management issues and proposals for energy-efficient cellular networks that will cover billions of machines. *EURASIP Journal on Wireless Communications and Networking*. 2016, roč. 2016. Dostupné z DOI: 10.1186/s13638-016-0636-y.

[4] FELD, Joachim. PROFINET-scalable factory communication for all applications. In: *IEEE International Workshop on Factory Communication Systems, 2004. Proceedings.* IEEE, 2004, s. 33–38.

[5] SWALES, Andy et al. Open modbus/tcp specification. *Schneider Electric*. 1999, roč. 29, s. 3–19.

[6] BROOKS, Paul. Ethernet/IP-industrial protocol. In: *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No. 01TH8597)*. IEEE, 2001, zv. 2, s. 505–514.

[7] BAILEY, David; WRIGHT, Edwin. *Practical SCADA for industry*. Elsevier, 2003.

[8] DANEELS, Axel; SALTER, Wayne. What is SCADA? 1999.

[9] FEUERHAHN, S. OpenMUC-Monitor & Control. *Online. http://www. openmuc. org*. 2011.

Zoznam symbolov a skratiek

OSI Model OSI (Open Systems Interconnection) popisuje sedem vrstiev, ktoré počítačové systémy používajú na komunikáciu cez sieť. Bol to prvý štandardný model pre sieťovú komunikáciu, ktorý na začiatku 80. rokov prijali všetky veľké počítačové a telekomunikačné spoločnosti.

TCP Transmission Control Protocol (TCP) je štandard, ktorý definuje, ako vytvoriť a udržiavať sieťovú konverzáciu, pomocou ktorej si môžu aplikácie vymieňať dáta. TCP spolupracuje s internetovým protokolom (IP), ktorý definuje, ako si počítače navzájom posielajú pakety dát.

UDP User Datagram Protocol (UDP) je komunikačný protokol, ktorý sa primárne používa na vytvorenie pripojení s nízkou latenciou a toleranciou straty medzi aplikáciami na internete.

UX/UI UI označuje obrazovky, tlačidlá, prepínače, ikony a ďalšie vizuálne prvky, s ktorými sa interaguje pri používaní webovej lokality, aplikácie alebo iného elektronického zariadenia. UX sa vzťahuje na celú interakciu s produktom, vrátane pocitov pri interakcii.

PLC PLC je skratka pre Programmable Logic Controller. Sú to priemyselné počítače používané na riadenie rôznych elektromechanických procesov na použitie vo výrobe, závodoch alebo iných automatizačných prostrediach. PLC sa líšia veľkosťou a tvarovými faktormi.

VFD VFD je skratka pre Variable-frequency drive, Pohon s premenlivou frekvenciou. Najbežnejšie použitie VFD je na ovládanie ventilátorov, čerpadiel a kompresorov. Tieto aplikácie predstavujú 75% všetkých pohonov fungujúcich na celom svete.

CIP CIP, Common Industrial Protocol, je mechanizmus na organizáciu a zdieľanie údajov v priemyselných zariadeniach. CIP je základná technológia za EtherNet/IP. CIP poskytuje spoločnú organizáciu údajov a spoločné zasielanie správ na riešenie rôznych druhov problémov s výrobnými aplikáciami.

Zoznam príloh

A Zdrojový kód implementácie systému OpenMUC

Príloha obsauje celý projekt OpenMUC vrátane všetkých doterajších protokolov, ako aj všetkých zmien a dodatkov vykonaných v rámci tejto diplomovej práce.

B Zdrojový kód implementácie C knižnice na prístup k nižším vrstvám OSI

Príloha obsauje zdrojový kód knižnice napísanej v jazyku C, pomocou ktorej OpenMUC pristupuje k takzvaným "Raw Socketom"pomocou java natívneho rozhrania.