TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

NÁSTROJ PRO TESTOVÁNÍ PROTOKOLU IEC-104

Bakalářská práce

Studijní program: B2646 – Informační technologie

Studijní obor: 1802R007 – Informační technologie

Autor práce: **Michael Sucharda**

Vedoucí práce: Ing. Jan Kraus, Ph.D.

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michael Sucharda**
Osobní číslo:	**M12000183**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Nástroj pro testování protokolu IEC-104**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se podrobně s protokolem IEC 60870-5-104 a s vybranými implementacemi v konkrétních zařízeních.
2. Na existujících aplikacích otestujte a zanalyzujte základní funkce a vlastnosti tohoto protokolu.
3. Vyberte vhodné softwarové řešení pro implementaci vlastního klienta, který prostřednictvím tohoto protokolu získá a jednoduše vizualizuje měřená data.
4. Diskutujte výhody a nevýhody navrženého řešení.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Manuál jednotky ELVAC RTU 7.4 a implementace protokolu IEC 104, manuál aplikace EIC Master.**

[2] **IEC Server project documentation dostupné z: http://jkl.kilu.de/IECServer/IECS-SERVER-DOC.html**

[3] **Dokumentace aplikace QTester 104 dostupné z: http://sourceforge.net/projects/qtester104/**

Vedoucí bakalářské práce: **Ing. Jan Kraus, Ph.D.**
Ústav mechatroniky a technické informatiky

Konzultant bakalářské práce: **Ing. Viktor Bubla**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2014**

Termín odevzdání bakalářské práce: **15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2014

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15. 5. 2015

Podpis:

Anotace

V této práci se řeší seznámení s protokolem ISO/IEC 60870, hlavně pak s jeho částí ISO/IEC 60870-5-104. Dále jsou zde uvedeny a popsány některé vybrané implementace tohoto protokolu. Zejména se zde řeší aplikace implementující klientskou část komunikace pomocí protokolu. Tedy aplikace schopné přijímat data pomocí toho protokolu. Následně je zde popsáno vlastní řešení aplikace přijímající data a demonstrace tohoto řešení. Nakonec je vlastní řešení porovnáno s vybranými již existujícími aplikacemi.

Abstract

This paper consists of learning about protocol ISO/IEC 60870, mainly about its part ISO/IEC 60870-5-104. There are presented and described some selected implementation of this protocol. Especially implementation dealing with client side of communication using this protocol. Then there is described my own solution of client aplication and demonstration of this solution. Finally my own solution is compared with selected existing applications.

Klíčová slova

ISO/IEC 60870, ISO/IEC 60870-5-104, protokol klientská aplikace

Key Words

ISO/IEC 60870, ISO/IEC 60870-5-104, protocol, client application

Obsah

Seznam obrázků

1 Úvod

Mnou vybrané téma bakalářské práce se týká testování části 5-104 protokolu ISO/IEC 60870. Ten se zabývá definicí norem pro dálkové řízení SCADA systému a to zejména v elektrických sítích. SCADA systémy, z anglického supervisory control and data acquisition, neboli „dohled a sběr dat“, jsou systémy používané pro dálkové řízení územně rozsáhlých systému. To zahrnuje prakticky cokoliv od průmyslového závodu, až po národní rozvodní sítě. Běžnými součástmi těchto systému jsou vzdáleně řiditelné jednotky (RTU), nebo programovatelné logické kontroléry (PLC). Většina kontrolních akcí je prováděna těmito zařízeními. Například PLC může řídit průtok v chlazení v části výrobního procesu. Ovšem SCADA systém umožňuje operátorovi nastavit hodnoty pro tento průtok a určit zobrazení nebo zaznamenání výstražných podmínek, jako ztráta průtoku nebo vysoká teplota. Sběr data pak začíná v RTU nebo PLC a zahrnuje přečtení těchto dat z měřidel a informací o stavu měřícího zařízení, které jsou skombinovány tak aby operátor v řídící místnosti mohl rozhodnout o úpravě nebo přepsání normálních funkci RTU/PLC.

Pro účely komunikace přenosu dat a řízení v těchto systémech existují různé komunikační protokoly, které definují standardy, pro podobu datových rámců, definují metody přenosu dat, komunikační sítě pro přenos těchto dat, metody řízení a přenosu řídících příkazu. Ve většině případu existují v těchto systémech dva typy stanic, stanice řídící a stanice řízené. Komunikace tak funguje buďto na principu master-slave nebo klient-server. Já se ve své práci zabývám protokolem definovaným v normě ISO/IEC 60780-5.

V teoretické části práce se zabývám popisem protokolu, zejména pak jeho pro mojí práci nejdůležitějších části 5-101 a 5-104. Popisuji zde zejména podobu přenášených dat definovanou v tomto protokolu. Seznámení se s tímto protokolem bylo jedním z cílů mé práce a odrazovým můstkem pro splnění dalších části. Také jsou zde uvedeny některé nejznámější podobné komunikační protokoly a uveden důvod proč ve své práci používám právě ISO/IEC 60870-5-104 Dále je zde také popis dalších použitých prostředků k seznámení s problematikou. Tedy nástroje umožňující analýzu a rozbor datových paketu přenášených sítí Ethernet a uspořádaných podle definic protokolu. Nebo o aplikace umožňující praktické navázání komunikace pomocí tohoto protokolu a vysílání řídích signálu.

V praktické části se pak zabývám nastavením konkrétních zařízení RTU od firmy ELVAC a.s., které jsem měl k dispozici. Tato zařízení umožňují sběr dat a příjem řídích signálů pomocí více různých protokolu a datových sběrnic. Jedním z nich je i ISO/IEC 60870-5-104, měl jsem je tak k dispozici jako profesionální řešení dané problematiky. Dále pak zhodnocením vyhledáváním informací o protokolu a hledáním existujících dostupných řešení protokolu 5-104. Následuje popis průběhu mého seznamování se se strukturou data odesílaných pomocí protokolu 5-104 za pomoci vybraných existujících implementací tohoto protokolu.

Další část práce tvoří popis mého vlastního řešení jednoduché klientské aplikace umožňující příjem a jednoduché zobrazení dat, a vyslat některé pokyny řízení definované pomocí protokolu ISO/IEC. Popis prostředků použitých k dosažení vlastního řešení a odůvodnění proč jsem k vlastnímu řešení zvolil právě tyto prostředky.

Následuje část, ve které porovnávám svou mnou vytvořenou aplikaci s některými již existujícími. A to pak zejména hlediska jejich funkčnosti a ovladatelnosti. Na základě tohoto srovnání pak diskutuji výhody a nevýhody vlastní implementace.

2 Teoretická část

V této části se práce je obecně představen protokol ISO/IEC 60870, podrobněji rozebrány jeho části 60870-5-101 a 60870-5-104, které jsou zejména důležitá pro tuto práci a dále obecně představeny další prostředky, použité dále v práci. Jedná se hlavně o již hotové aplikace implementující komunikaci prostřednictví protokolu ISO/IEC 60870, které byli použity k nastudování a pochopení základních funkci protokolu.

2.1 Podobné komunikační protokoly

Pro stejné nebo velmi podobné účely jako IEC 60780-5-104 existují v praxi i jiné komunikační protokoly. Jedním z nich je DNP3, který je používán pro zejména v eklektických a vodohospodářských rozvodných sítích. Tento protokol byl vytvořen pro komunikaci mezi různými typy přístroju pro kontrolu a získávání dat. Hraje velkou roli ve SCADA systémech, kde je používán pro komunikaci mezi master stanicí (kontrolním centrem), vzdáleně řiditelnými jednotkami (RTU) a jednotkami pro kontrolu elektrických sítí. Tento protokol byl vytvořen v době kdy IEC 60870-5 byl stále ve vývoji a bylo třeba zajistit interoperabilitu mezi zařízeními od různých výrobců. V roce 1993 tak GE-Harris Kanada použili částečně dokončený protokol IEC 60870-5 jako základ pro otevřený a okamžitě implementovatelný protokol, který se specificky zaměřil na potřeby Severní Ameriky. Nejrozšířenější je protokol DNP3 právě na tomto území. [10]

Modbus je otevřený protokol pro vzájemnou komunikaci typu klient-server mezi různými zařízeními a po různých typech sítí a sběrnic. Byl vytvořen v roce 1979 firmou MODICON od dubna 2004 ho pak spravuje Modbus Organization. Je široce používán v průmyslu zejména pro komunikaci mezí řídící stanici a řízenou stanicí ve SCADA systémech. Při komuniakci po sériové lince může příkazy vysílat pouze jednotka označená jako master, v komunikaci přes Ethernet pak může Modbus command vyslat kterékoliv zařízení.[12]

Profibus je standart pro real-time distribuovanou kontrolu. Cílem při vytváření bylo implementovat, zavést a rozšířit používání bitově-sériové sběrnice. Profibus je ve své funkčnosti zaměřený hlavně na zpracovatelský průmysl. [13]

V zásadě lze říci, že v praxi existuje několik různých implementací komunikačních protokolu, v minulosti pak byli těchto soupeřících protokolu až desítky, jejichž funkčnost se do značné míry překrývá, ovšem nikdy neřeší danou problematiku

v úplném rozsahu. Tato skutečnost v prvé řadě znesnadňovala vzájemnou komunikaci mezi přístroji od různých výrobců. O řešení tohoto problému se snaží norma ISO/IEC 61850, která definuje jednotný protokol pro definici základních služeb potřebných k přenosu dat, podporu pro co největší interoperability mezi systémy různých výrobců, společný formát pro ukládání kompletních dat. Abstraktní datové modely popsané v ISO/IEC61850 mohou být mapovány na řadu protokolů. Protokoly definované v této normě mohou komunikovat přes TCP/IP a pomocí vysokorychlostního Ethernetu dosáhnout požadovaných dob odezvy.[11]

Ve své práci se zabývám protokolem ISO/IEC 60870-5-104, protože je dnes široce používaný pro kontrolu elektrických rozvodných sítí v České republice.

2.2 Obecné představení normy ISO/IEC 60870

Norma ISO/IEC 60870 (u nás zavedená jako ČSN EN 60870) definuje systém pro dálkovou kontrolu, supervisory control and data acquisition, neboli SCADA systémy. Takové systémy jsou používány pro kontrolu elektrických přenosových sítí a jiné geograficky rozsáhlé systémy. Na specifikaci protokolů pracovala technická skupina TG 03 technického výboru TC 57 Mezinárodní elektrotechnické komise IEC, přičemž práce na ní začala v roce 1988. ISO/IEC 60870 má 6 části.

První dvě části ISO/IEC 60870-1 a ISO/IEC 60870-2 se zabývají obecnými ustanoveními (všeobecné zásady, výklad zvláštních výrazů) a provozními podmínkami (napájení a elektromagnetická kompatibilita, klimatické a další neelektrické vlivy).

Třetí část, ISO/IEC 60870-3, popisuje elektrické charakteristiky rozhraní, a to znamená podmínky pro rozhraní, které musí být splněny, aby mohli různé prvky spojené navzájem vytvořit funkční síť.

Čtvrtá část ISO/IEC 60870-4 se zabývá charakteristikami, které mají vliv na provoz systémů dálkového ovládání a souvisejí s vlastnostmi aplikace a funkcemi zpracování dat. Dále stanovuje soubor pravidel pro hodnocení a specifikaci požadavků na výkon.

Pro mojí práci nejdůležitější je pátá část ISO/IEC 60870-5 specifikující přenosové protokoly pro dálkové ovládání zařízení a soustav, které jsou založeny na sériovém

přenosu binárně kódovaných dat a jsou určeny pro dohled a ovládání geograficky rozlehlých procesů.

Normy ISO/IEC 60870-5 vycházejí z modelu master-slave a specifikují funkce pro systéme dálkového ovládání. Dvě nejdůležitější z nich jsou, Report By Exception a mechanizmus přiřazovaní časových značek.

V komunikačních systémech s modelem master-slave je jedna řídící jednotka master, které odesílá požadavky všem svým podřízeným jednotkám, slaves. Každá podřízená jednotka reaguje na ty požadavky, které jsou jí určeny.

Pro jednotku master je často velmi důležité dozvědět co nejrychleji o situaci kdy na jednotce slave došlo k nějaké mimořádné události, změnila se hodnota příslušné proměnné. K tomu je určena funkce Report By Exception (RBE). Ta umožňuje jednotce slave požádat o komunikaci s jednotkou master.

Norma ISO/IEC 60870-5 má několik částí, z nichž prvních pět je:

- ISO/IEC 60870-5-1 *Formáty přenosového rámce* – asynchronní přenos dat s linkovými protokoly *half-duplex* a *full-duplex,* standardy pro kódování, formátování a synchronizování datových rámců s hodnotami proměnných a metody zajištění integrity dat.
- ISO/IEC 60870-5-2 *Procedury spojového přenosu* – procedury pro sériový přenos kódovaných digitálních dat.
- ISO/IEC 60870-5-3 *Obecná struktura aplikačních dat* – pravidla pro strukturování jednotek aplikačních dat v přenosových rámcích.
- ISO/IEC 60870-5-4 *Definice a kódování aplikačních informačních prvků* – pravidla pro definování informačních prvků, zvláště digitálních a analogových procesních proměnných, často používaných v systémech dálkového řízení.
- ISO/IEC 60870-5-5 *Základní aplikační funkce* – standardy pro zajištění interoperability různých zařízení elektrizační soustavy.

Část ISO/IEC 60870-5-101 je Společná norma pro úkoly dálkového ovládání. Jejím cílem je umožnit funkční interoperabilitu mezi kompatibilními systémy.

Publikována v roce 1995, a doplněna v roce 1998, tak aby definovala také přenos úplné časové značky.

Části ISO/IEC 60870-5-102 a ISO/IEC 60870-5-103 specifikují další společné standarty první pro přenos integrovaných součtů hodnot v elektrizačních soustavách. Druhá pak informační rozhraní ochran.

Část ISO/IEC 60870-5-104 s názvem Síťový přístup pro IEC 60870-5-101 používají normalizované transportní profily. Zjednodušeně řečeno, zatímco část 5-101 specifikuje mechanismy přenosu dat, část 5-104 stanovuje (nebo v některých případech jen doporučuje) jejich použití v běžných komunikačních sítích; z nich nejpopulárnější je Ethernet s TCP/IP spojením.

Část 6 skupiny standardů ISO/IEC 60780 se týká protokolů pro dálkové ovládání kompatibilních se standardy ISO a doporučeními ITU-T. Od části 5 se liší zejména tím, že vychází z modelu klient-server a je využívána pro větší systémy zpracování dat a počítačové sítě. Jde především o přenos informací mezi jednotlivými dispečinky elektrizační soustavy navzájem a mezi nimi a dispečinky elektráren.

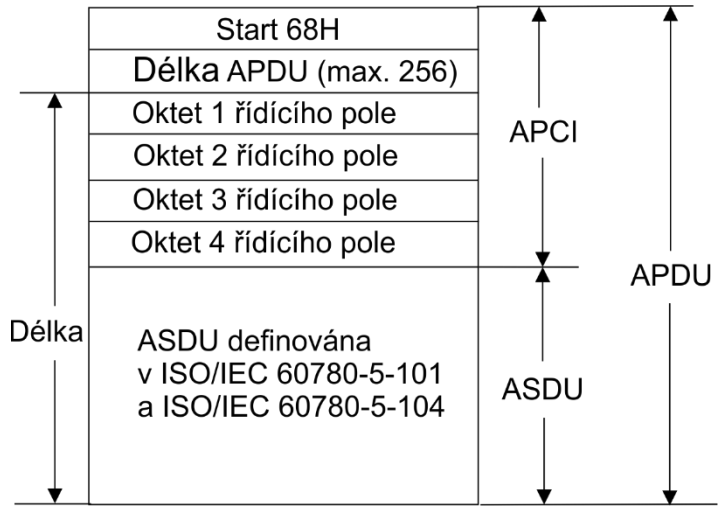
Zde bych rád upozornil na skutečnost, že řídící jednoty případně aplikace master, jsou při používání těchto norem označovány jako klienti. Pod pojmem server pak vystupují jednotky řízené, čili ty které se starají o sběr a posílání dat.

2.3 Podrobnější představení částí 5-101 a 5-104 protokolu ISO/IEC 60870

Jak již bylo zmíněno výše část 5-101 definuje metody přenosu dat mezi část 5-104 použití těchto metod v běžných komunikačních sítích. ASDU, jednotka dat aplikační služby (z anglického Application Service Data Unit), definované podle části 5-101 tak například neobsahuje žádný stop a start mechanizmus nebo mechanizmus pro ochranu před ztrátou nebo duplikací dat. K tomuto účelu definuje část 5-104 APCI, řídící informace aplikačního protokolu (z anglického Application Protol Control Information). ASDU a APCI pak dohromady tvoří celek označovaný APDU, jednotka dat aplikačního protokolu (z anglického Application Protocol Data Unit).

2.3.1 Část 5-104

Pro stanovení začátku ASDU obsahuje každá APCI následující vymezující prvky: start znak, stanovení délky ASDU a řídící pole (viz obrázek).



Obrázek 1: APDU definované podle normy

Start 68H definuje začátek toku dat. Délka APDU definuje délku hlavní části APDU, které zahrnuje čtyři oktety řídícího pole v APCI a celé ASDU. První počítaným oktetem je první oktet řídícího pole a posledním počítaná mým oktetem je poslední oktet ASDU. Maximální délka ASDU je tak 249, neboť maximální hodnota délky pole APDU je 253 (255 – oktet start a oktet délky) a délka řídícího pole je 4 oktety. Řídící pole definuje informaci pro ochranu před ztrátou a duplikací zpráv, začátek a konec přenosu a kontrolu přenosového spojení. Používají se 3 typy formátu řídicího pole.[1]

Formát pro provádění číslovaných informačních přenosů (I formát). První bit oktetu 1 řídícího pole nastaven na 0, definuje I formát. APDU s I formátem vždy obsahuje ASDU[1]. Řídící informace i formátu je uvedena na obrázku.

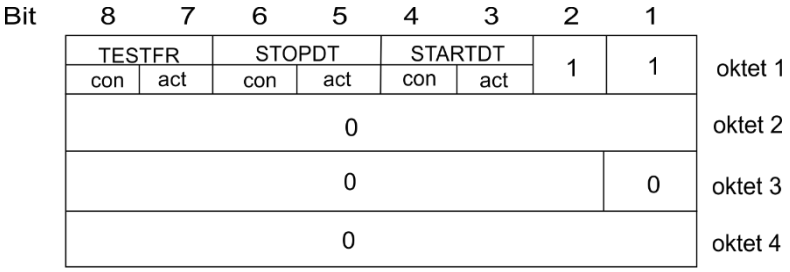
Bit	8	7	6	5	4	3	2	1	
	Pořadové číslo vysílání N(S)						LSB	0	oktet 1
	MSB	Pořadové číslo vysílání N(S)							oktet 2
	Pořadové číslo příjmu N(R)						LSB	0	oktet 3
	MSB	Pořadové číslo příjmu N(R)							oktet 4

Obrázek 2: Řídící pole typu I formát

Formát pro provádění číslované kontrolní funkce (S formát). První bit oktetu 1 řídícího pole nastaven na 1 a druhý bit nastaven na 0 definuje S formát. APDU s S formátem obsahují pouze APCI[1].

Obrázek 3: Řídící pole typu S formát

Formát pro provádění nečíslované řídící funkce (U formát). První bit oktetu 1 řídícího pole nastaven na 1 a druhý bit nastaven na 1 definuje S formát. APDU s U formátem obsahuje pouze APCI. Řídící informace U formátu uvedena na obrázku. Kdykoliv může být aktivní pouze jedna funkce TESTFR (test rámce), STOPDT (ukončení přenosu dat) nebo STARTDT (zahájení přenosu dat).[1]



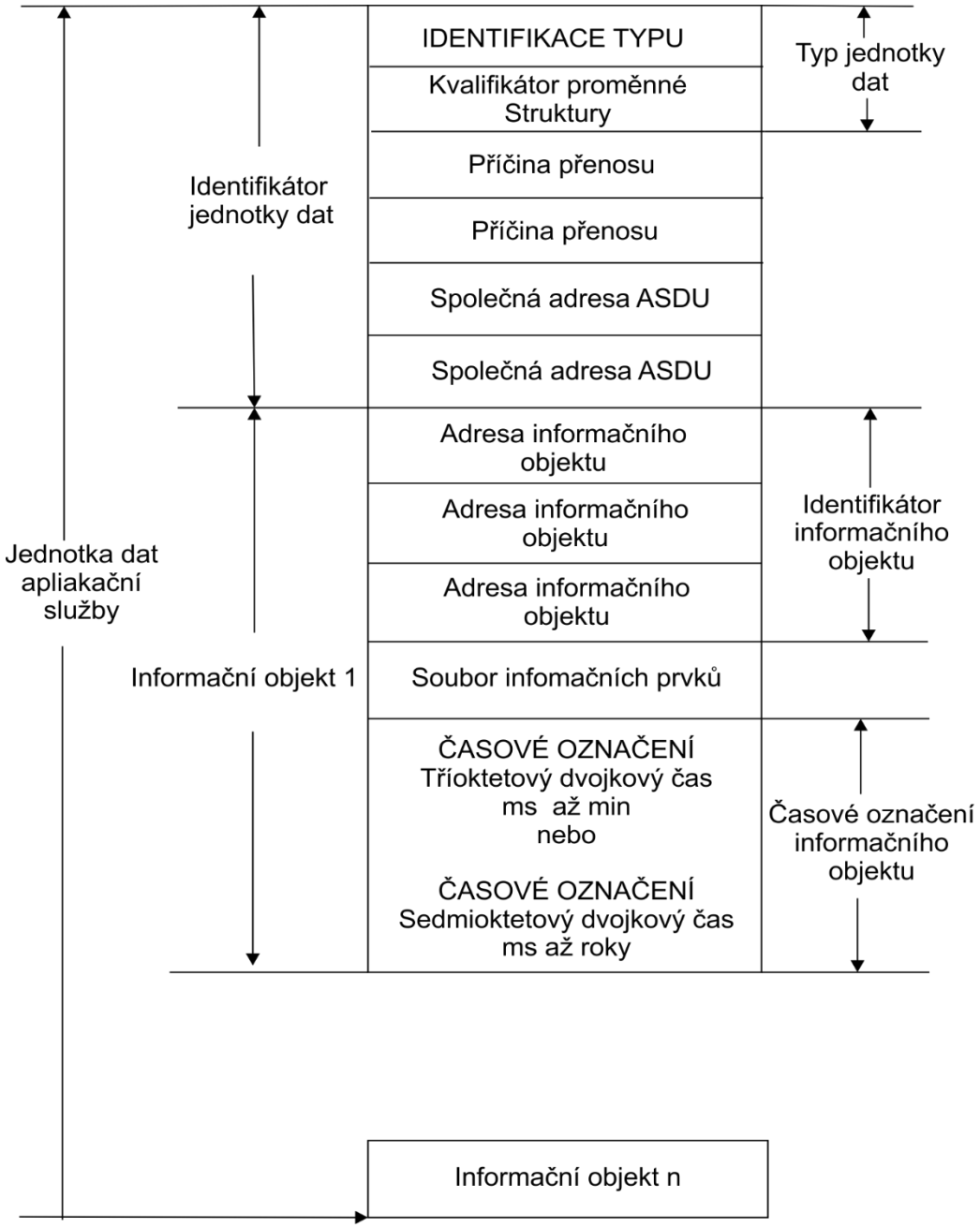
Obrázek 4: Řídící pole typu U formát

V případě, že spojení mezi řídící a řízenou stanicí je již zahájeno, avšak není používáno, je možné ho v obou směrech pravidelně testovat vysíláním zkušebních APDU, TESTFR = act, které přijímací stanice potvrzuje vysláním TESTFR = con. Obě stanice mohou zahájit zkušební proceduru po stanoveném časovém úseku, v němž nedošlo k přenosu dat.

Příkazy STARTDT, zahájení přenosu dat, a STOPDT, ukončení přenosu dat, používá řídící stanice k řízení přenosu dat z řízené stanice. Při vytvoření spojení mezi stanicemi není automaticky zahájen přenos dat. Výchozí nastavení je tedy STOPDT. V tomto stavu nevysílá řízená stanice žádná data s výjimkou nečíslovaných řídích funkcí a potvrzení takových funkcí. Řídící stanice musí aktivovat přenos uživatelských dat po daném spojení vysláním STARTDT act. Na tento příkaz řízená stanice odpoví STARTDT con. Pokud není STARTDT potvrzen dojde k ukončení daného spojení mezi stanicemi.

Veškerá neodbavená uživatelská data z řízené stanice jsou vyslána až po STARTDT con. V případě přepnutí aktivního spojení na jiné spojení vyšle řídící stanice STOPDT act, řízená stanice zastaví přenos dat a vyšle STOPDT con. Po přijetí STOPDT con může řídící stanice dané spojení ukončit.

2.3.2 Část 5-101



Obrázek 5:Struktura ASDU: Počet oktetů příčiny přenosu je 1 nebo 2. Počet oktetů společné adresy ASDU je 1 nebo 2. Počet oktetů adresy informačního objektu je 1,2 nebo 3. Toto musí být v systému pevně nastaveno.

Identifikace typu definuje strukturu, typ a formát nedujících *Informačních objektů*. Je definován jako číslo od 1 do 255. *Informační objekty* jsou rozlišeny různými čísly *Identifikace typu*. Hodnota 0 se nepoužívá a v normě je definován rozsah hodnot 1 až 127. Hodnoty 128 až 255 definovány nejsou. Hodnoty v rozsahu 136 až 255 pak mohou nezávisle definovat uživatelé normy[2].

Konkrétně pak:

- Rozsah 0 až 44 slouží pro provozní informace ve směru sledování.
- Rozsah 45 až 69 slouží pro provozní informace ve směru ovládání.
- Rozsah 70 až 99 slouží pro Systémové informace ve směru sledování.
- Rozsah 100 až 109 složí pro Systémové informace ve směru ovládání.
- Rozsah 110 až 119 souží pro Parametr ve směru sledování.
- Rozsah 120 až 127 slouží pro přenos souborů.

Dále pak například:

- 9 označuje měřenou normalizovanou hodnotu
- 13 značí naměřenou hodnotu s pohyblivou řádovou čárkou
- 45 je jednoduchý povel
- 100 označuje dotazový povel
- 102 značí povel čtení
- 103 značí povel pro časovou synchronizaci
- 104 označuje zkušební povel

Kvalifikátor proměnné struktury určuje počet informačních objektů nebo prvků obsažených v ASDU na prvních 7 bitech (tedy 0 až 127). Osmý bit pak určuje adresování Informačních objektů a prvků.[2]

Příčina přenosu identifikuje na prvních 6 bitech příčinu přenosu. 7 bit slouží pro rozlišení kladného a záporného potvrzení aktivace. Na 8 bitu je definován zkušební bit u těch ASDU, které byli definovány během zkoušky. Následující oktet je nepovinný a slouží k udání adresy původce ASDU, přičemž hodnota 0 je standardní a slouží k definování provozních informací, které musí byt přeneseny na všechny části decentralizovaného systému, hodnoty 1 až 255 lze použít pro adresování specifické části systému, na kterou je vrácena příslušná informace. Příkladem konkrétních hodnot může

být hodnota 1 značící periodické přenosy, hodnota 3 značící spontánní přenosy nebo hodnoty 6 a 7 značící aktivaci, respektive potvrzení aktivace.

Společná adresa ASDU určují adresu konkrétní stanice, jedná se o parametr, který je pro systém pevně nastaven. Hodnota 0 se nepoužívá. Maximální hodnota, tedy 255 nebo 65535 (v závislosti na použití jednoho nebo obou oktetů) je globální adresa sloužící pro všechny stanice.

Adresa informačního objektu je parametr, který je v systému pevně nastaven. Ve směru ovládání se tato hodnota používá pro místo určení, ve směru sledování pak pro určení zdroje. ASDU s nedefinovanými adresami informačního objektu řídící stanice zruší. Třetí oktet se používá pouze v případe strukturování adresy.[2]

Dále pak jsou obsaženy informační prvky, které jsou strukturovány podle definic v ISO/IEC 60870-5-4. Sem spadá v zásadě ta nejpodstatnější část celého přenosu. Tedy přenášená hodnota. Druhy přenášených informací definované podle normy jsou různé. Jedná se například o jednobitovou nebo dvoubitovou informaci, normalizovanou hodnotu, hodnotu s měřítkem, krátké číslo s pohyblivou řádovou čárkou. Dále jsem, pak spadají povely např. jednoduchý povel, dvoj povel. Také různá časová označení informačního objektu.

Je zde zařazen i Kvalitativní deskriptor, což je informace skládající se z 5 bitů, které poskytují řídící stanici dodatečné informace o kvalitě informačního objektu. Tato informace se skládá z:

- Přeplněno/nepřeplněno – indikuje, zda hodnota informačního objektu nepřekročila stanovený rozsah hodnot.
- Blokováno/neblokováno – hodnota je pro přenos blokována
- Zaměněno/nezaměněno – hodnota určena automaticky nebo vstupem obsluhy
- Neaktuální/aktuální – hodnota je aktuální pokud její poslední aktualizace proběhla úspěšně
- Neplatné/platné - pokud je po zaznamenání hodnoty zjištěn nenormální stav zdroje této informace, je označena jako neplatná, hodnota informačního objektu pro tento stav není definovaná

2.4 Použité existující implementace ISO/IEC 60870

Pro lepší pochopení a vyzkoušení vlastností protokolu, následné testování a porovnání vlastních výsledků bylo využito několika existujících implementacích protokolu.

Přístroje RTU od firmy ELVAC a. s. Jedná se o přístroje určené ke vzdálenému řízení, měření a sběr dat v energetických sítích. Tyto zařízení poskytují širokou škálu komunikačních rozhraní, na kterých implementují různé komunikační protokoly včetně pro mne důležitého ISO/IEC 60780-5-104.

IEC Server je volně dostupná aplikace napsaná v jazyce Java, které simuluje serverovou stranu komunikace pomocí protokolu ISO/IEC 60780. Tato aplikace umožňuje simulovat posílání různých typu hodnot, zejména pak simulaci měřených hodnot. Také simulovat reakci na některé příkazy od řídící stanice definované podle protokolu.[4]

QTester104 je volně dostupná aplikace napsaná v jazyce C++ za pomoci frameworku QT, která umožňuje testování protokolu ISO/IEC 60780-5-104. Tato aplikace může být použita k navázání s pojení se zařízením, příjmem dat z tohoto zařízení a posílání některých příkazu zařízení.[6]

IEC Master je aplikace od společnosti ELVAC jejíž demo verzi jsem měl k dispozici. Tato aplikace umožnuje navázání spojení se zařízením, příjem dat a řízení zařízení pomocí příkazů.

Wireshark je aplikace umožňující odchytávání a analýzu paketů na síťových spojeních. Používá k monitorování a testování počítačových sítí a také k testování různých komunikačních protokolu. Tato aplikace umožňuje také filtrování komunikace pomocí protokolu ISO/IEC 60780-5-104. Dále je také schopna jednoduše zobrazit obsah a strukturu dat přenášených pomocí tohoto protokolu.[5]

Knihovna j60870 od organizace openmuc je knihovna pro programovací jazyk Java implementující funkce a metody potřebné pro vytváření aplikací, které komunikují protokolem ISO/IEC 60780-5-4. Tato knihovna byla použita k navržení a implementaci mé vlastní aplikace pro komunikaci protokolem. [3]

3 Cíl práce

Cílem této práce je seznámit se s protokolem pro řízení rozsáhlých systémů ISO/IEC60870 používaným zejména v elektrických rozvodných sítích. Podrobně se pak seznámit s jeho částmi ISO/IEC 60870-5-104 a ISO/IEC 60870, které definují datové rámce přenosu dat a jejich přenos pomocí nejpoužívanější komunikačních sítí zejména Ethernetu.

Dále pak seznámit s jednotkami RTU od firmy ELVAC a. s., které implementují mimo jiné tento komunikační protokol. Také se pak seznámit se s možnostmi nastavení těchto jednotek a následně je nastavit tak aby byli schopny přijmout naměřená data z měřících přístrojů a dále pak tyto data poslat pomocí protokolu 5-104 za použití Ethernetové sítě.

V další části je to pak mým úkolem vyhledat implementace tohoto komunikačního protokolu v již existujících aplikacích. S pomocí těchto aplikací pak analyzovat a lépe se seznámit se strukturou posílaných dat a další komunikace pomocí protokolu 5-104. A dále s pomocí těchto aplikací otestovat základní funkce pro příjem dat a pro účely řízení pomocí protokolu.

Následně pak vybrat vhodné softwarové řešení pro implementaci vlastní klientské aplikace, která bude umožňovat pomocí tohoto protokolu získat a jednoduše zobrazit měřená data. Dále by aplikace také měla být schopna vyslat alespoň nějaké příkazy řízení definované v protokolu, pro otestování této funkcionality.

Nakonec pak vlastní řešení implementace tohoto protokolu porovnat s vybranými již existujícími implementacemi. Na základě tohoto porovnání diskutovat výhody a nevýhody mého vlastního řešení. A zhodnotit celkový průběh řešení a jeho konečné výsledky.

4 Návrh řešení

V postupu řešení je prvním krokem seznámení se s přístroji RTU od firmy ELVAC a. s. Zejména pak s možnostmi jejich nastavením tak aby byli schopné měřit některé vybrané hodnoty, případně tyto hodnoty přebírat od jiných měřících zařízení a následně tyto hodnoty posílat pomocí protokolu ISO/IEC 60870-5-104. Tyto přístroje pak budou jednou z možností jak analyzovat základní funkce daného protokolu. Data z těchto přístrojů budou posléze použita k seznámení se se strukturou těchto dat.

Pro lepší pochopení struktury dat, existence některých řídích hodnot v těchto datech a lepší pochopení funkcí protokolu pak dojde k seznámení se samotnou normou ISO/IEC 60870 zejména pak jejími částmi ISO/IEC 60870-5-101 a ISO/IEC 60870-5-104. Kde je struktura dat a přenosu dat podrobně popsána.

Dále za tímto účel provedu analýzu těchto datových přenosu sítí ethernet pomocí programu Wireshark, jenž umožní zobrazení komunikace mezi klientskou a serverovou stranou komunikace.

Aby bylo možné, komunikaci mezi klientskou a serverovou částí komunikace analyzovat bude potřeba nejprve navázat komunikaci s daným zařízením. K tomu nejprve použiji demo verzi aplikace IEC Master od firmy ELVAC a. s. Později pak ke stejnému účelu použiji volně dostupnou aplikaci QTester104. Obě tyto aplikace by měli umožnit navázání komunikace a vyslání některých základních příkazů.

Pro testování odesílání dat a tím vlastně i jejich příjmu dále také použiji volně dostupnou aplikaci IEC Server, by měla umožnit nastavení alespoň některých parametrů komunikace a odesílání. A dále pak odesílání dat simulovat.

Nakonec se seznámím s knihovnou pro jazyk Java j60870 od organizace openmuc. S jejíž pomocí se pokusím o implementaci vlastního klientské aplikace. Která by následně měla navázat komunikaci se serverovou stranou a zajistit příjem a zobrazení dat.

5 Vlastní řešení

Ve vlastní řešení bylo nutné nejprve nastavit jednotky RTU, tak aby přijímaly naměřené hodnoty z připojených měřících přístrojů a posílaly je pomocí protokolu ISO/IEC 60870-5-104. Následně se zabývám analýzou protokolu pomocí programu Wiraeshark. Dále pak pomocí Java knihovny j60870 od organizace openmuc naprogramovat vlastní jednoduchou aplikaci, která zajistí připojení k přístroji a příjem data zaslaných protokolem ISO/IEC 60870. Nakonec své řešení zhodnocuji a diskutuji jeho výhody a nevýhody.

5.1 Přednastavení ELVAC RTU

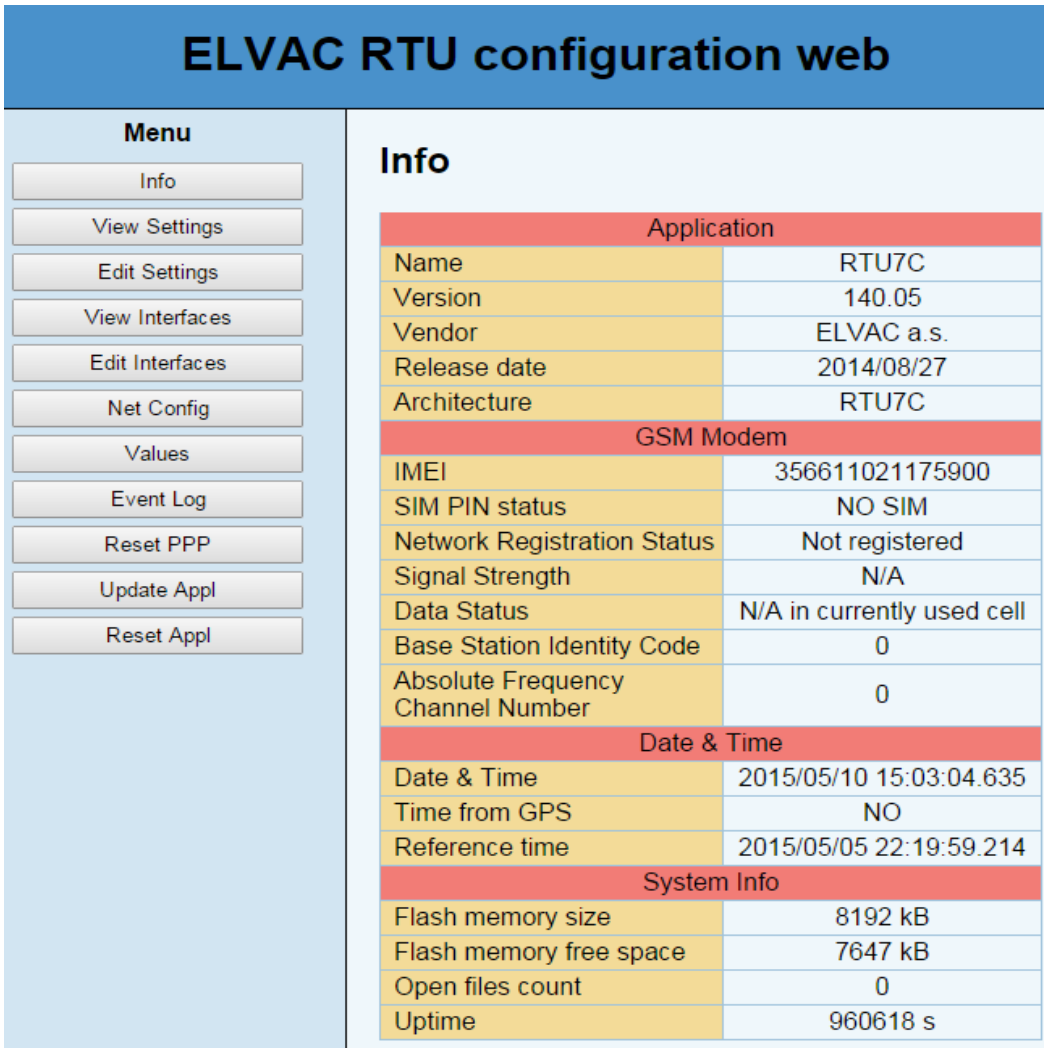
Před samotným nastavením bylo třeba naistalovat programy k tomu potřebné. Konkrétně se jednalo o Microsoft SQL 2008 databázi, ELVAC Správa SQL pro RTU, RTU Komunikátor a ELVAC RTU Uživatelské centrum.

V programu ELVAC Správa pro RTU bylo třeba vybrat konkrétní databázi a poté již program databázi vytvoří automaticky. V programu RTU komunikátor nastavit databází, se kterou bude komunikovat. Stejně tak bylo třeba zvolit databázi v programu ELVAC RTU Uživatelské centrum.

Nové jednotce je poté třeba přes webové rozhraní nastavit několik hodnot, tak aby správě komunikovala s Uživatelským centrem pro další nastavení. V části Edit Settings je třeba nastavit UDP-API Port na hodnotu 1720 a PPP Configuration Enabled na YES.

V části Edit Interfaces je třeba nastavit jednotlivé parametry položky HioCom2 Slave na hodnoty: TYPE: UDP, Source port: 9999, Destiantion IP: nastavit IP adresu počítače ze kterého bude zařízení RTU nastavováno, Destination port: 9999, Pernament: YES.

Pro uložení všech těchto nastavení je poté třeba pomocí položky Reset Appl zařízení restartovat. Poté je možné přistoupit k samotnému nastavení RTU pomocí uživatelského centra.

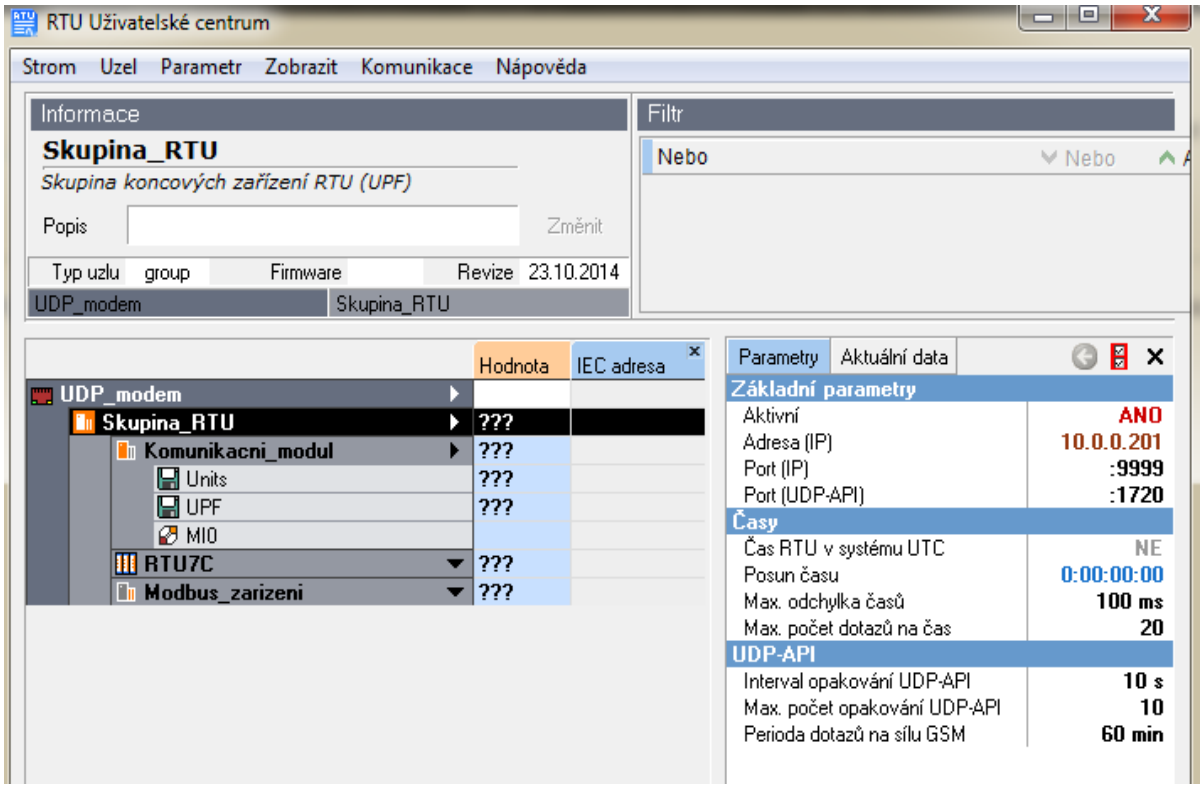


Obrázek 6: Ukázka webového konfiguračního rozhraní jednotky ELVAC RTU

5.2 Nastavení ELVAC RTU

V RTU uživatelském centru je třeba nejprve vytvořit nový strom nastavení pro RTU zařízení. V horním menu pomocí položky Strom -> Přidat rozhraní -> UDP modem přidat nové rozhraní. Poté kliknout na položku UDP_modem v hlavním okně a pomocí klávesy Insert vyvolat okno s možností přidání skupiny zařízení, zde pro můj případ zvolit RTU_MainGroup a potvrdit.

V hlavním okně pod položkou UDP_modem přibude položka Skupna_RTU. Po kliknutí na tuto položku jsem v menu parametrů, napravo v okně, musel nastavit IP adresu zařízení, které budu nastavovat. Se stále označenou položkou Skupina_RTU kliknutím na klávesu Insert vybrat konkrétní typ jednotky RTU, kterou nastavuji, v mém případě RTU7C_UPF, potvrdit a v následujícím okně zvolit vstupy a výstupy které budu chtít nakonfigurovat.



Obrázek 7: RTU uživatelského centra

Opět v hlavním okně přibude nová položka tentokrát s názvem RTU7C. Označíme tuto položku a v menu parametrů nastavit Sériové číslo, jedná se sériové číslo, konkrétního přístroje, které je uvedené přímo na jeho těle.

Poté všechny položky aktivovat, označením a stisknutím mezerníku. Položku Skupina_RTU parametrizovat, stiskem kláves ctrl + P. A resetovat položku Komunikační_modul, pravý klik -> Příkazy -> Reset.

Protože naměřené hodnoty přebírá v mém případě RTU z jiných zařízení připojených přes sériovou linku RS-485 a využívající Modbus protokol je potřeba v RTU ještě nastavit tato zřízení.

Označením položky Skupina_RTU kliknutím na klávesu Insert opět vyvolat okno pro přidání konkrétních zařízení, tentokrát bylo třeba zvolit položku Modbus_zařízení a potvrdit. Položka Modbus_zařízení přibude i v hlavním okně.

Poté bylo třeba nastavit parametry modbus zařízení. Seriové číslo musí být nastaveno na číslo v rozsahu 501 až 510, v případě že je připojeno modbus zařízení více musejí mít nastavené rozdílná sériová čísla. Správně bylo také třeba nastavit komunikační

rozhraní, a to sice Číslo COM portu na COM1, Komunikační rychlost na 115200 baud, a Adresu modbus serveru nastavit podle modbus adresy přístroje.

Poté označením Modbus_zařízení a kliknutím na Insert vyvolat okno s nabídkou vstupu a výstupů, která umožňuje přidat vstupy a výstupy pro modbus zařízení. V mém případě AI7_MB, neboli analogový vstup. V hlavním okně přibude položka znázorňující tento vstup.

Označením opět vyvoláme menu parametrů, zde je bylo třeba nastavit položku adresa na číslo modbus registru kde se bude nacházet daná hodnota, položku Kanál, v případě více vstupů/výstupů musejí mít toto nastavení jedinečné, položku Funkce nastavit na „4 - čti vstupní registr“ a položku Typ vstupní hodnoty na „reálné číslo 3“.

Po nastavení všech požadovaných vstupů je potřeba opět aktivovat všechny položky provést parametrizaci Skupiny_RTU a resetovat Komunikační_modul.

5.3 Nastavení protokolu ISO/IEC 60870-5-104 v zařízení RTU

Dále bylo potřeba v zařízení RTU nastavit komunikaci zařízení pomocí protokolu ISO/IEC 60870-5-104. K tomuto cíli vedl následující postup.

Po kliknutí pravým tlačítkem myši na Komunikační_modul zvolit položku „Vybrat volitelné kanály“. Objeví se okno s výběrem kanálů, v části Modul rozhraní vyberat MI0 a potvrdit. V hlavním okně se pod položkou Komunikační_modul objeví nová položka MI0. Této položce bylo v příslušném menu, pro správný chod, nutno nastavit několik parametrů.

Typ rozhraní nastavit na TCP server. Zdrojovou IP adresu na localhost, tedy 127.0.0.1. Nastavením cílové adresy na konkrétní hodnotu lze omezit přístup pomocí protokolu jen z konkrétní IP. Zdrojový port nastavit na 2404, i přesto že lze nastavit jakýkoliv port 2404 je v protokolu definován jako standardní pro komunikaci pomocí 104 na jednotkách slave a měl být tedy vždy být použit tento. Komunikační protokol nastavit na IEC104. A do Společné adresy ASDU zadat 16 bitovou adresu stanice. Poté už položku MI0 aktivovat označením a stiskem mezerníku.

Následně by se u každého vstupního a výstupního registru měla objevit nová položka s názvem IEC adresa. Jedná se o 24 bitovou adresu konkrétní hodnoty

přenášenou v ASDU. Pokud je tato hodnota nastavena na 0, nebude daný registr viditelný pomocí protokolu 104. Jinak musí mít všechny registr tuto adresu jedinečnou. Poté už je opět jen potřeba parametrizovat položku Skupina_RTU a resetovat Komunikační_modul.

5.4 Vyhledávání informací a dostupných aplikací

Následně jsem se pustil do vyhledávání konkrétních informací o protokolu, v začátcích svého snažení jsem totiž neměl k dispozici samotnou normu. Musel jsem se tedy pokusit získat informace o struktuře dat v protokolu z jiných zdrojů. Tento úkol se ukázal být celkem obtížný, při zběžném vyhledání jsem narazil spíše na nabídky jednotlivých profesionálních řešení než na informace o samotném protokolu. Po delším hledání se mi podařilo nalézt několik zdrojů, ovšem většina byla hodně obecná a některé se, jak jsem později zjistil, úplně neshodovali s daným popisem přímo v normě. Přesto mi tyto popisy pomohli získat alespoň základní představu o dané problematice.

Také jsem se pustil do hledání a prozkoumávání již existujících implementací umožňujících komunikaci protokolem 5-104. Pro získání představy jsem měl sice k dispozici demo verzi aplikace IEC Master, ale pátral jsem po více možnostech k porovnání. Také jsem pátral po knihovnách, které by usnadnili řešení mé vlastní aplikace.

Průběh toho snažení byl velice podobný hledání informací o protokolu. Valná většina řešení, které jsem nalezl, byly nabídky profesionálních řešení, které pouze zřídka nabízeli demoverze. Pokud už byli demoverze k dispozici bylo jejich použiti časově omezeno na několik dnů a pro delší práci se tak nehodili. Nelezl jsem i několik volně dostupných řešení. Většina z nich ale měla velmi špatnou nebo vůbec žádnou funkčnost. Navíc se tyto projekty zdáli mrtvé, a nedalo se tedy očekávat žádné zlepšení situace. Aplikace sim-104[14] například.

Dobře vypadající nálezy tak byli vlastně jen čtyři. Aplikace IEC server, kterou jsem při své práci využil, ale která implementuje serverovou stranu komunikace. Poté knihovna pro Javu j60870, kterou jsem nakonec použil pro implementaci vlastního řešení. Program Wireshark, který umožňuje testování protokolu.

A Aplikace QTester104, která se jevila jako dobré volně dostupné řešení implementující klientskou část komunikace, a která slibovala i možnost využití svých

knihoven k vlastnímu řešení. Bohužel jsem zde hned v počátku narazil na problém s funkčností.

Aplikaci je možné stáhnout buďto v podobě již zkompilovaného binárního souboru, nebo lze stáhnout zdrojové kódy a ty si zkompilovat sám. Nejprve jsem pro jednoduchost volil první možnost, ovšem binární soubor se mi nedařilo spustit. Vždy jsem dostal jen chybovou hlášku o QT pluginu pro platformu windows. Při zjišťování co může danou chybu způsobovat, jsem narazil na několik možností.

Většina se týkala toho, že chybí nainstalována některá knihovna frameworku QT a jedy potřeba jí doinstalovat, jiné se týkali špatného použití knihoven přímo v kódu aplikace. Zkusil jsem tedy nejprve doinstalovat některé knihovny. To v mém případě ale nepomohlo. Tím pádem se zdálo, že problém bude přímo ve zdrojových kódech aplikace. Což se mi potvrdilo i pokusem o vlastní kompilaci stažených zdrojových kódů.

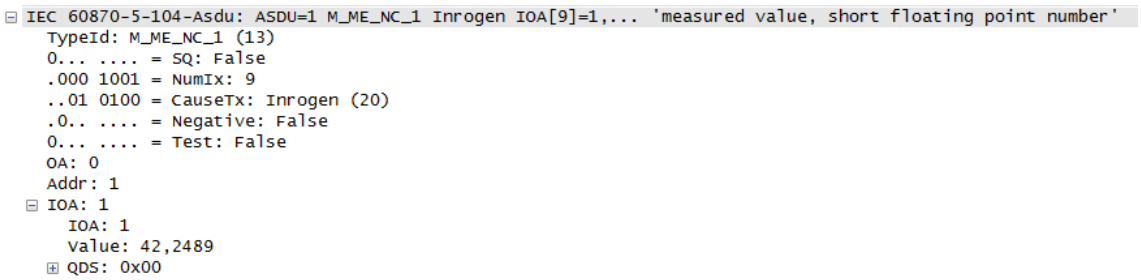
Práci na tomto problému jsem tedy omezil a pokračoval v řešení s dostupnými funkčními prostředky. Průběžně jsem se pak ke QTesteru104 vracel a nakonec se mi podařilo aplikaci zkompilovat tak aby byla funkční. Tyto problémy pak byly jedním z hlavních důvodu, proč jsem k realizaci vlastního řešení rozhodl použít knihovnu j60870.

5.5 Testování existujícími aplikacemi

Dále jsem tedy pustil do samotného praktického seznamování s protokolem 5-104 a testování jeho možností. Pro navázání komunikace jsem jako serverovou část používal jednotky ELVAC RTU pokud byli dostupné a jindy také aplikaci IEC Server. Jako klientskou část jsem používal demo verzi aplikace IEC Master. Ta mi umožnila navázat spojení, přijímat data a testovat některé příkazy, ovšem vzhledem k tomu, že se jedná o demo verzi, běží aplikace pouze omezenou dobu. Jedná se zhruba o 2 až 3 minuty. Pote se aplikace sama ukončí a je třeba jí znovu spustit a znovu navazovat spojení se serverem. Řešením toho by QTester104 problémy s ním jsem ovšem již popsal výše.

Vypínání aplikace IEC Master sice bylo poměrně nepříjemné, díky programu Wireshark však mnou práci seznamování se přímo se strukturou dat, a vlastně i průběhem samotné komunikace nějak podstatně neomezilo. Wireshark se ve skutečnost ukázal jako úplně výborný pomocník pro tyto účely. Pomocí tohoto programu bylo možné zachytit veškerou komunikaci probíhající na mé síťové kartě. Dokonce pak Wireshark umožňuje filtrovat a zobrazovat pouze APCI a ASDU komunikace pomocí protokolu 5-104. Dokáže

pak také zobrazit data obsažená v ASDU. A to tak jak jsou strukturovaná po bitech a zobrazuje i význam daných parametru, tak jak je definován v samotné normě. To mi samozřejmě významně pomohlo v pochopení celé struktury dat v době, kdy jsem ještě neměl k dispozici samotnou normu definující tento protokol. Rozsáhlejší ukázky z programu Wireshark, sloužící jako ukázka prováděné analýzy jsou přiloženy v příloze této práce.

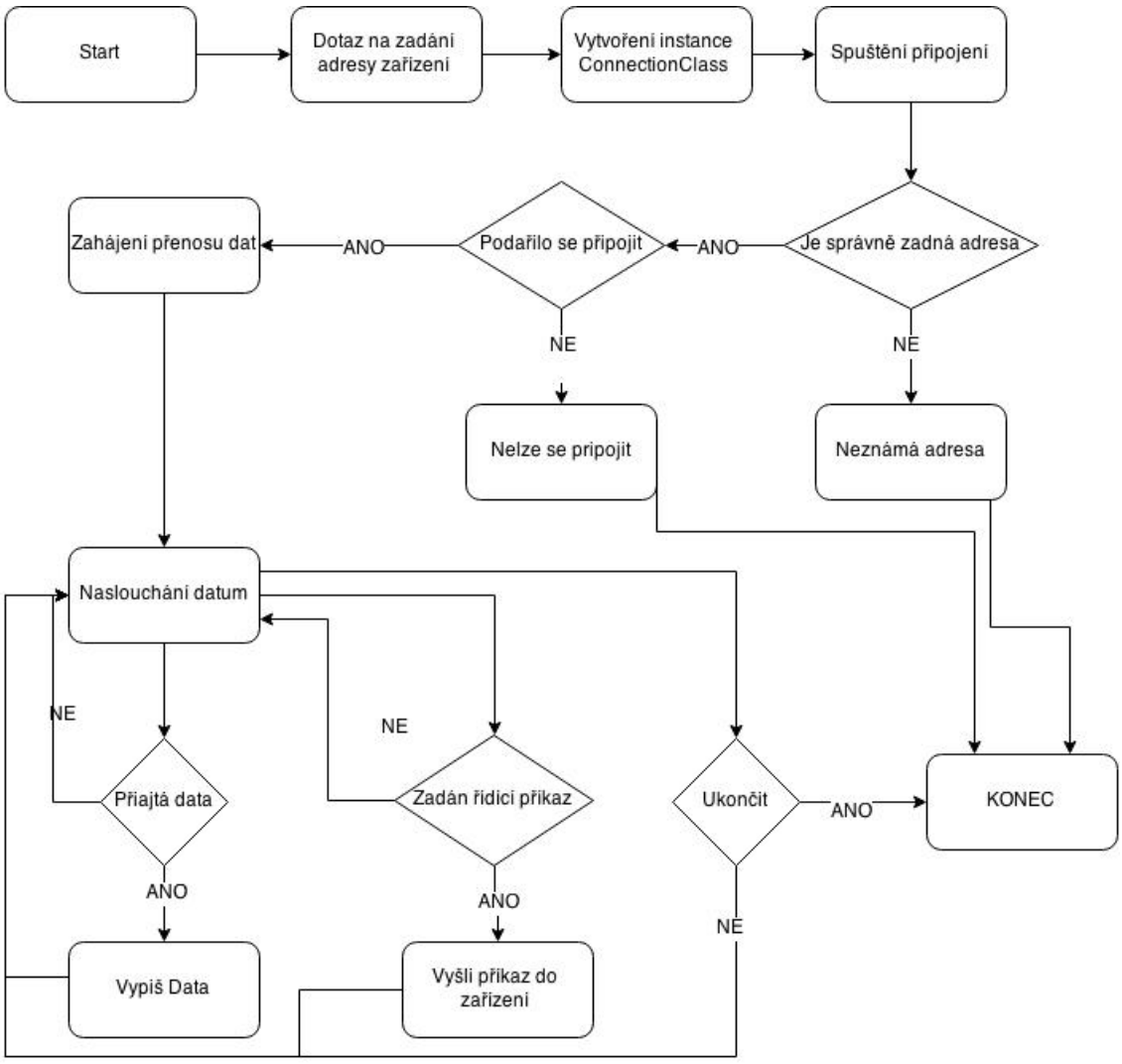


Obrázek 8:Ukazka zobrazení ASDU v programu Wireshark

5.6 Vlastní aplikace

Jak již bylo zmíněno, pro vytvoření vlastní aplikace jsem použil knihovnu j60780 pro jazyk Java od organizace openmuc. Tato knihovna obsahuje funkce, které zajišťují možnost připojení k řízené stanici, a také umožňují podrobnější nastavení tohoto připojení. Dále knihovna obsahuje funkce pro naslouchání po aktivním spojení a správný příjem a rozparsování dat bloku ASDU popsaném v jedné z předchozích kapitol. Knihovna také obsahuje funkce pro vyslání řídích příkazu zařízením. Pustil jsem se tedy do studia dokumentace knihovny[3] a seznámil se s potřebnými funkcemi k vytvoření vlastní implementace.

Program tedy nejprve požádá o zadání adresy zařízení, ke kterému se chceme připojit. Poté se vytvoří instance třídy ConnectionClass, což je mnou vytvořená třída, která se stará o vytvoření spojení a dále pak jeho ovládání, tedy vysílání příkazů. Při spuštění této třídy dojde nejprve vytvoření instance třídy ClientSap, tato třída je implementována přímo v použité knihovně a slouží k nastavení parametrů spojení a přímému připojení k ISO/IEC 60870-5-104 serveru.



Obrázek 8: Diagram naznačují průběh mé vlastního programu

Poté dojde k nastavení, ověření správnosti a případně překladu, adresy ze vstupu zadaného uživatelem za pomoci funkcí standardní třídy jazyka Java InetAddress. Pokud nelze rozpoznat adresu je vyvolána výjimka UnknownHostException a program je ukončen.

Pokud rozpoznávání adresy proběhlo v pořádku, dojde k pokusu o připojení ke stanici, k tomuto účelu slouží funkce connect instance třídy ClientSap. Tato funkci se zadávají dva parametry, a sice adresa získaná v předchozím kroku a port, na který se má připojit. Port je pro řízené stanice pomocí protokolu ISO/IEC 60870-5-104 je definován přímo v normě na hodnotu 2404. V případě že se připojení je vyvolána výjimka, která ohlásí nemožnost připojení se a program se ukončí.

V případě, že pokus o připojení proběhl úspěšně, je inicializována instance třídy Connection Implementované v knihovně. Tato třída reprezentuje spojení mezi klientem a serverem. Nově vytvořené spojení úspěšně navázalo TCP/IP spojení se serverem. Spojení je tedy nyní aktivní a již ho možné ho ovládat. K zahájení přenosu dat použijeme funkci startDataTransfer tohoto aktivního spojení. Této funkci předáme jako parametr instanci třídy ConnectionListener. Tato třída implementuje ConnectionEventListener z použité knihovny j60870. A stará se o příjem a zobrazení příchozích ASDU.

Všechny příchozí ASDU po daném spojení jsou do ConnectionEventLister přesměrována, jeho implementace pro třídu, která data přijímá je tedy nutná. Třída implementujíc ConnectionEventListener pak musí definovat funkci newASDU, která přijímá ASDU po spojení a jejich zobrazení. Definovat musí také funkci conectionClosed, která je použita, pokud bylo spojení mezi klientem a serverem nečekaně ukončeno. V takovém případě je oznámen důvod tohoto ukončení, pokud je znám, a program je ukončen.

Pro ovládání aktivního spojení jsem ve svém programu pro ukázku naimplementoval dva příkazy. Jejich použití je zadáním příkazu do okna programu. Zadáním příkazu *timeSync* je stanici vyslán příkaz k synchronizování času s klientem, parametrem funkce je aktuální systémový čas. Druhý příkaz je spuštěn zadáním *read.* Tento příkaz odešle stanici dotaz na čtení jejich hodnot přístupných pomocí protokolu. Stanice na tento příkaz odpoví odesláním ASDU pro každou z těchto hodnot.

Další možností ovládání mého programu je příkaz *help,* který zobrazí nápovědu pro ovládání programu. A příkaz *exit,* po zadání tohoto příkazu dojde nejprve k ukončení aktivního spojení a poté k ukončení celého programu.

Na obrázku ukázky programu můžeme vidět navázání spojení pomocí localhost s aplikací IEC Server, popsanou v předchozí kapitole. Lze zde vidět přijetí automaticky odeslané ASDU zprávy s naměřenou hodnotou, poté dvě ASDU přijatá v reakci na příkaz *read*, a poté opět jedno automaticky odeslané ASDU.

Lze vidět, že program pro jednotlivá ASDU vypisuje Typ přenášených dat, jak pomocí hodnoty, tak pomocí identifikačního řetězce tak jak je popsán v normě (např.: M_ME_NC_1 na obrázku). Dále také vypisuje důvod přenosu, adresu identifikačního objektu (hodnota označená IOA) a samotnou hodnotu. Vypisuje také další informace definované v ASDU a popsané v protokolu ISO/IEC 60870-5-101.

```
"C:\Program ...
You can use following commands to control connection:

Input "exit" to end this program.
Input "timeSync" to send time synchronization command.
Input "read" to receive data from device.
Input "help" to show this help for commands.


Please input address of device to connect to:
192.168.1.248

Connected!

Recieved ASDU:
Type ID: 13, M_ME_NC_1, Measured value, short floating point number
Cause of transmission: SPONTANEOUS, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 1
Short float value: 11.1
Quality, overflow: false, blocked: false, substituted: false, not topical: false, invalid: false

read
Recieved ASDU:
Type ID: 13, M_ME_NC_1, Measured value, short floating point number
Cause of transmission: REQUESTED_BY_GROUP_1_COUNTER, test: false, negative con: true
Originator address: 0, Common address: 1
IOA: 1
Short float value: 11.1
Quality, overflow: false, blocked: false, substituted: false, not topical: false, invalid: false

Recieved ASDU:
Type ID: 100, C_IC_NA_1, Interrogation command
Cause of transmission: ACTIVATION_TERMINATION, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 0
Qualifier of interrogation: 102

Recieved ASDU:
Type ID: 13, M_ME_NC_1, Measured value, short floating point number
Cause of transmission: SPONTANEOUS, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 1
Short float value: 22.2
Quality, overflow: false, blocked: false, substituted: false, not topical: false, invalid: false
```

Obrázek 9: Ukázka běhu mého vlastního programu. Vidět lze oznámení o připojení, přijetí spontánně odeslaných dat a reakci na příkaz pro zaslání dat, a další přijetí spontánně odeslaných dat.

6 Zhodnocení výsledků

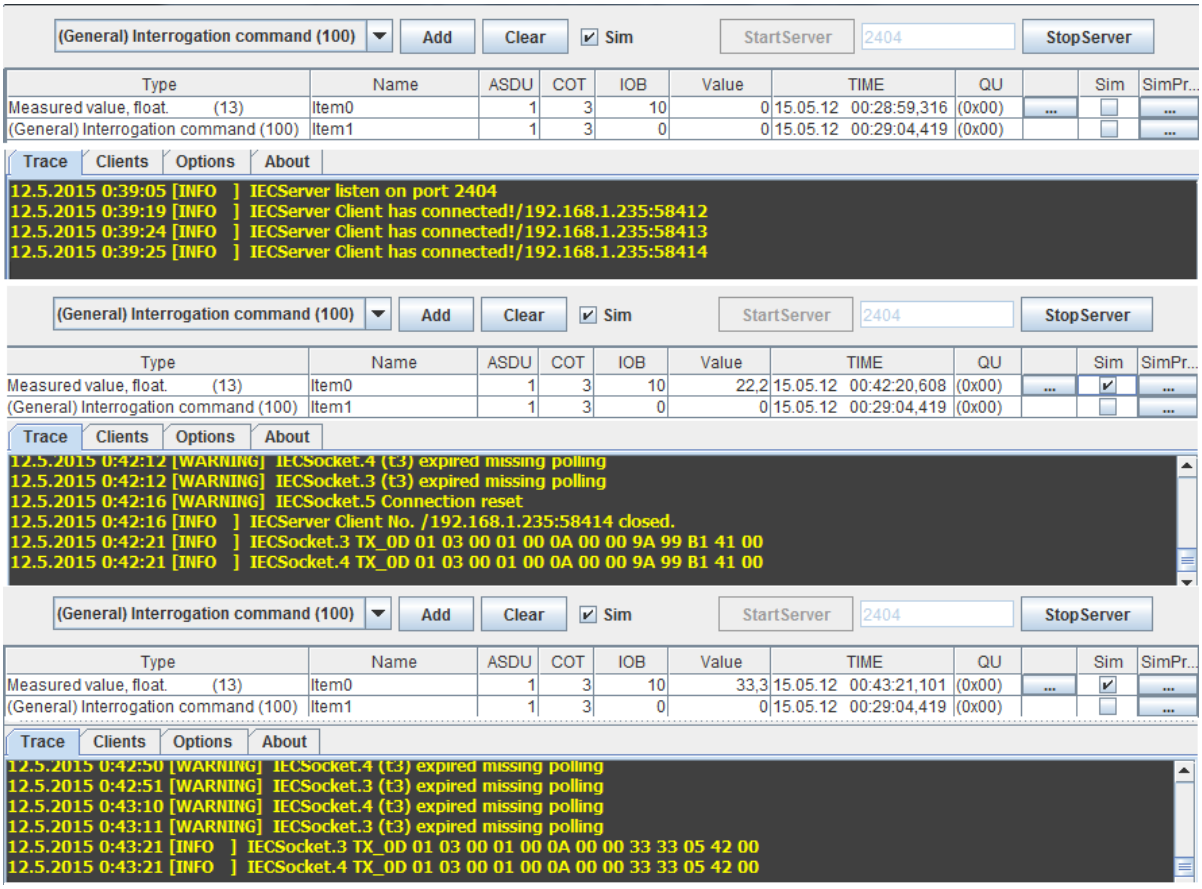
Mojí vlastní aplikaci se mi tedy podařilo vytvořit. Aplikace je schopná přijmout a zobrazit data. Funkční je také vysílání příkazů.

Co se porovnání s aplikacemi IEC Master a QTester104 týče je moje aplikace značně jednoduší. Moje aplikace funguje pouze jako command line aplikace, zatímco ostatní dvě mají plně grafické rozhraní. To je pro ovládání vhodnější, zejména v případě aplikací, které umožňují širší možnosti ovládání zařízení. V mém případě s pouze dvěma příkazy je ovládání programu ještě bez větších problému zvladatelné, v případě rozšiřování možností programu by ovšem bylo zřejmě potřeba vhodné grafické rozhraní do aplikace přidat.

Zobrazení sledování jednotlivých ASDU a hodnot v nich obsažených je v aplikacích QTester104 a IEC master poměrně jasné a přehledné, opět zejména díky grafickému rozhraní. Ovšem moje aplikace v tomto, dle mého názoru, nějak výrazně nezaostává. Sice vypisuje jednotlivá ASDU pod sebe, ovšem vzhledem k tomu že jsou všechny informace obsažené v ASDU jasně vypsány na několika řádcích není jejich sledování nějak obtížné. Obě zmíněné aplikace také hodnoty obsažené v ASDU aktualizují v zobrazení grafického rozhraní. V mojí aplikaci tak lze lépe sledovat postupný vývoj přijatých hodnot. Některé s parametru obsažených v ASDU dokonce nejsou v porovnávaných aplikacích zobrazeny vůbec a to zejména v případě QTesteru104. Jedná se například o parametry indikující zkoušku nebo kladné a záporné potvrzení, případně kvalitativní deskriptor.

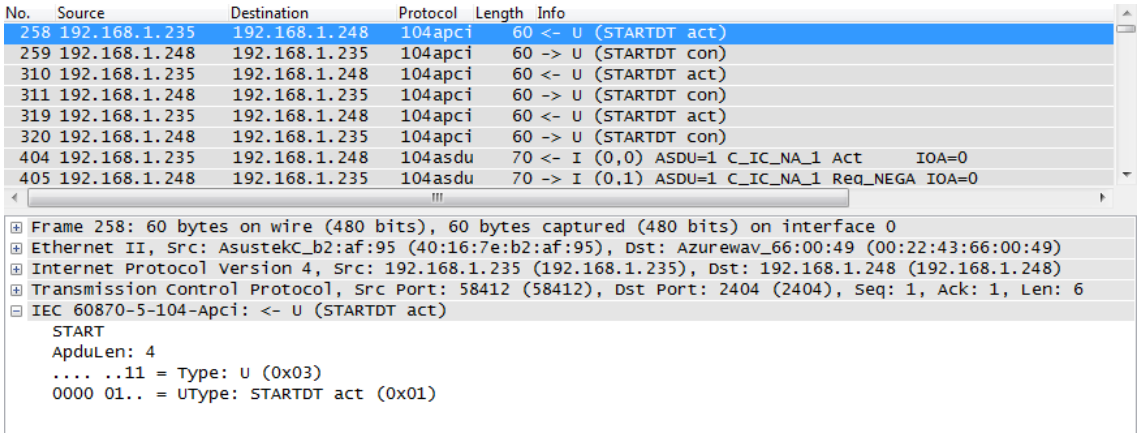
Lze tedy říci, že moje vlastní řešení za ostatními zaostává, zejména co se pohodlí ovládání týče. Ovšem vzhledem k problémům se zprovozněním aplikace QTester104 a časovým omezením práce s demo verzí IEC Master. By se po drobných úpravách mohlo jednat o vhodné řešení pro úplně základní testování komunikace protokolem ISO/IEC 60870-5-104.

Na následující obrázcích je ukázáno, navázaní spojení s aplikací IEC Server, a příjem stejné sady dat, ve stejném čase, z této aplikace, pomocí mé vlastní aplikace, aplikací IEC master a QTester104 a také ukázka těchto dat v programu Wireshark Tyto obrázky vhodně ukáží rozdíl v jednotlivých programech a v práci s daty a ovládání protokolu v nich.



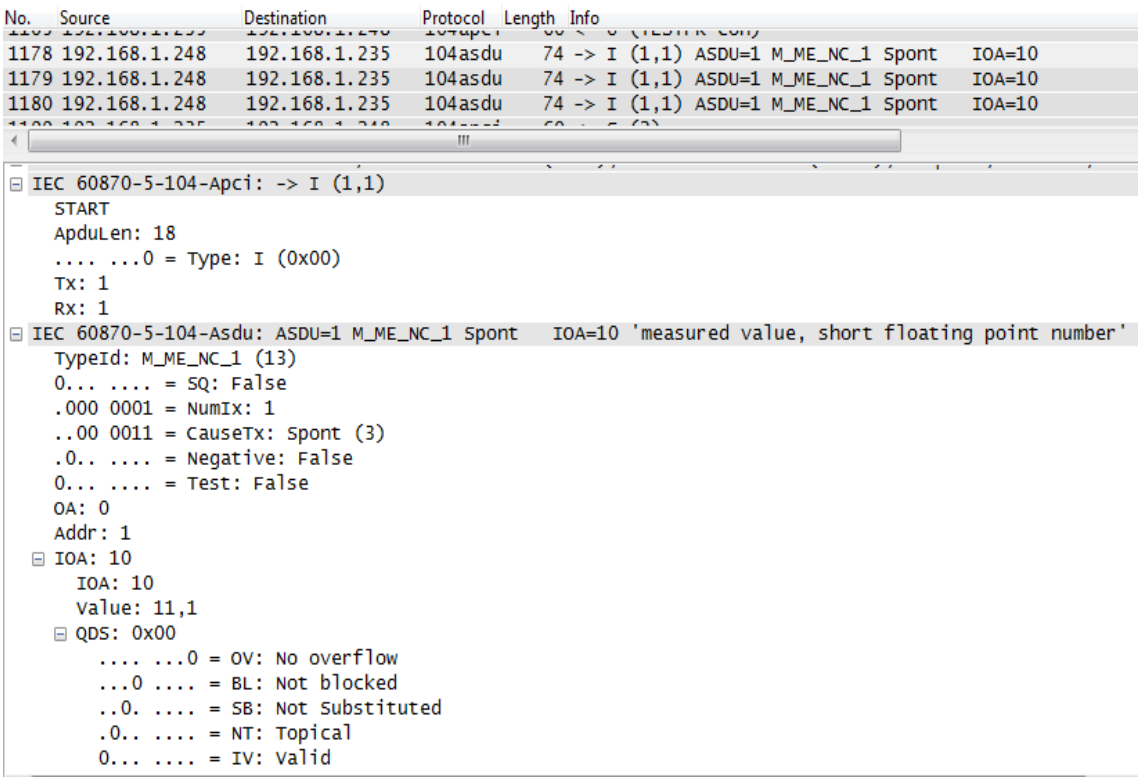
Obrázek 10: Ukázka komunikace programu IEC server

Na obrázku 11 je ukázán průběh komunikace v aplikaci IEC Server, podstatné jsou hlavně řádky textu začínající „[INFO]“. V první části obrázku vidíme informaci o navázání spojení se třemi klientskými aplikacemi Konkrétně s aplikací IEC Master, QTester104 a mojí vlastní aplikací. Navázání spojení je ukázán i ve Wiresharku (obrázek 11) Dále je zde oznámení o uzavření spojení s jedním klientem, to bylo způsobeno automatickým koncem aplikace IEC Master.



Obrázek 11: Ukázka navázání komunikace klient-server ve Wiresharku

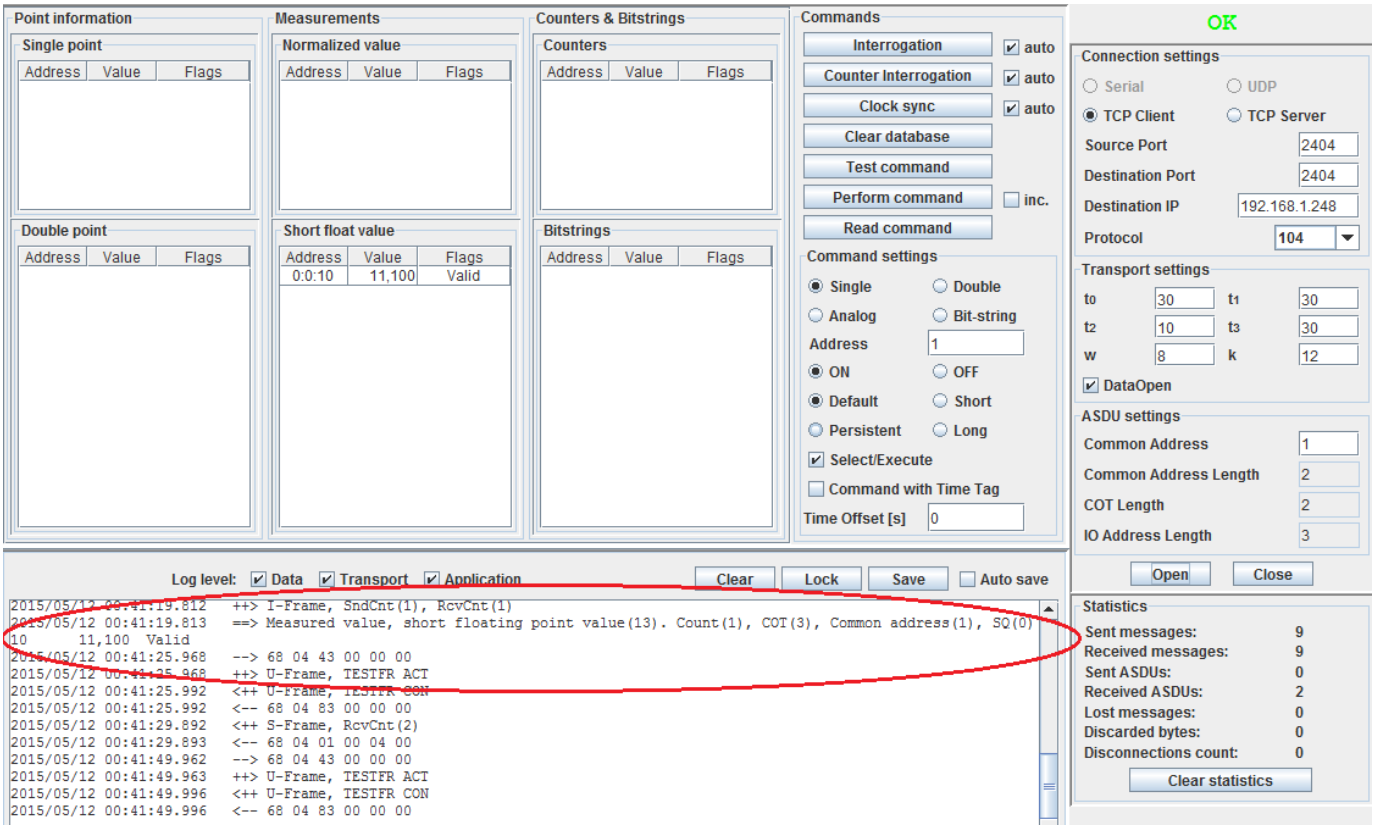
Na dalších řádcích je pak již zobrazena informace o odeslání ASDU zbylým dvěma klientům. Posloupnost hexadecimálních číslic následujících za řetězcem TX je vlastně reprezentace dat obsažených v odeslaném ASDU. Hexadecimální hodnota 0D neboli 13 decimálně označuje typ přenášené hodnoty. Lepe je tato struktura vidět v rozboru z programu Wireshark (obrázek 12).



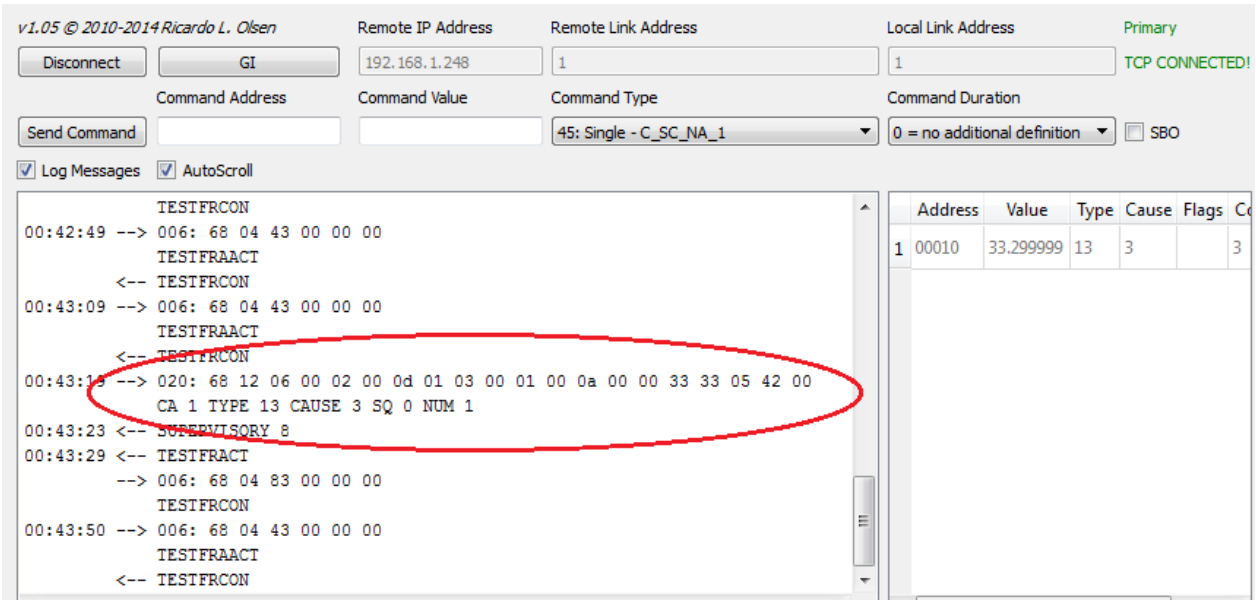
Obrázek 12: Ukázka rozboru ASDU v programu Wireshark

Na tomto obrázku můžeme lépe vidět pořadí daných parametrů, tak jak je definováno v normě a význam těchto přenášených hodnot tak jak je definovaný v protokolu 5-101. A jak jsem jej pospal již v teoretické částí této práce.

Na dalších obrázcích (13, 14 a 15) je pak ukázka programů IEC Master, Qtester104 a v mojí vlastní aplikaci. Je zde také ukázán příjem stejných dat v těchto aplikacích. Ve zvýrazněné části obrázku 13 vidíme informaci o příjmu ASDU v aplikaci IEC Master. Jsou zde zobrazeny základní informace o ASDU ovšem například informace obdržené v kvalitativní deskriptoru chybí. Dále program zobrazuje aktuální hodnoty a v pravém dolním rohu je statistika o přijatých a odeslaných zprávám a ASDU.



Obrázek 13: Ukázka příjmu dat v programu IEC Master



Obrázek 14: Ukázka příjmu dat v programu Qtester104

```
"C:\Program ...
You can use following commands to control connection:

Input "exit" to end this program.
Input "timeSync" to send time synchronization command.
Input "read" to receive data from device.
Input "help" to show this help for commands.


Please input address of device to connect to:
192.168.1.248

Connected!

Recieved ASDU:
Type ID: 100, C_IC_NA_1, Interrogation command
Cause of transmission: REQUEST, test: false, negative con: true
Originator address: 0, Common address: 1
IOA: 0
Qualifier of interrogation: 20

Recieved ASDU:
Type ID: 13, M_ME_NC_1, Measured value, short floating point number
Cause of transmission: SPONTANEOUS, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 10
Short float value: 11.1
Quality, overflow: false, blocked: false, substituted: false, not topical: false, invalid: false

Recieved ASDU:
Type ID: 13, M_ME_NC_1, Measured value, short floating point number
Cause of transmission: SPONTANEOUS, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 10
Short float value: 22.2
Quality, overflow: false, blocked: false, substituted: false, not topical: false, invalid: false

Recieved ASDU:
Type ID: 13, M_ME_NC_1, Measured value, short floating point number
Cause of transmission: SPONTANEOUS, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 10
Short float value: 33.3
Quality, overflow: false, blocked: false, substituted: false, not topical: false, invalid: false
```

Obrázek 15: Ukázka přijmu dat v mé vlastní aplikaci

Na obrázku 14 pak vidíme příjem dat v aplikaci Qtester104. Ve zvýrazněné části je ukázána informace o přijatém ASDU a jeho obsahu. Tato informace je zde zobrazena stejným způsobem jako v IEC serveru (ukázáno výše v textu), který bohužel není moc dobře čitelný. Aktuální hodnota je pak zobrazena vpravo v okně a to i s příčinou přenosu a informací kolikrát byla tato hodnota přijata.

Na obrázku 15 je pak ukázka stejné komunikace jako v předchozích případech v mé vlastní aplikaci. Můžeme zde vidět, že moje aplikace zobrazí všechny informace v ASDU obsažené, podobným způsobem jako program Wireshark. Zobrazení těchto informací je tak čitelnější než v předchozím případě.

7 Závěr

V práci jsem se seznámil se základní s podobou datových rámců vysílaných pomocí protokolu ISO/IEC 60780-5-104 a blíže pak definovaných v ISO/IEC 60870-5-101. Dále jsem se seznámil s tím co jednotlivé části těchto dat znamení a k čemu daná informace slouží. Hlouběji jsem pak v praktickém použití, pro tento účel se jako výborný pomocník ukázal program Wireshark.

K účelu získání dat správně naformátovaných podle protokolu ISO/IEC 60870-5-104 mi posloužili přístroje RTU od firmy ELVAC a. s. Tyto přístroje jsou po správném nastavení schopny poskytovat naměřená data v potřebném formátu.

Následně jsem se po seznámení s existujícími implementacemi protokolu rozhodl pro vlastní řešení zvolit knihovnu pro jazyk Java j60870 od organizace openmuc. Ta mi umožnila realizovat vlastní jednoduché řešení klientské aplikace.

Mé vlastní řešení jsem porovnal s některými již existujícími. A i když je značně jednoduší než řešení, se kterými jsem srovnával pro základní testování příjmu dat, a odeslání příkazů poslouží.

Možnosti budoucího pokračování mé práce jsou poměrně rozsáhle. V první řadě by se jistě jednalo o vylepšení mé klientské aplikace, aby byla lepé ovladatelná, umožňoval lepší přehlednější zobrazení přijatých dat a lepší možnosti odesílání příkazů. Dalším krokem by mohlo být vytvoření vlastní aplikace pro serverovou stranu komunikace promoci protokolu ISO/IEC 60870-5-104.

Seznam použité literatury

[1] *ČSN EN 6087-5-104*. *Systémy a zařízení pro dálkové ovládání - ČÁST 5-104: Přenosové protokoly - Síťový přístup pro IEC 60870-5-101 používající normalizované transportní profily*. 2007. Praha: Český normalizační institut.

[2] *ČSN EN 60870-5-101*. *Systémy a zařízení pro dálkové ovládání - Část 5-101: Přenosové protokoly - Společná norma pro základní úkoly dálkového řízení*. 2005. Praha: Český normalizační institut.

[3] *J60870 0.9 API* [online]. 2014. [cit. 2015-05-11]. Dostupné z: http://www.openmuc.org/projects/j60870/javadoc/

[4] *IEC Server manual* [online]. 2013. [cit. 2015-05-11]. Dostupné z: http://jkl.kilu.de/IECServer/IECS-SERVER-DOC.html

[5] *Wireshark* [online]. [cit. 2015-05-11]. Dostupné z: https://www.wireshark.org/

[6] Sourceforge. 2014. *Qtester104* [online]. [cit. 2015-05-11]. Dostupné z: http://sourceforge.net/projects/qtester104/

[7] CONTROL + S. 2013. *A free GPL'd IEC 60870-5-104 Protocol Tester* [online]. [cit. 2015-05-11]. Dostupné z: https://ricolsen1supervc.wordpress.com/2013/05/21/an-iec-60870-5-104-protocol-tester/

[8] IEC 60870. 2001-. Wikipedia: the free encyclopedia [online]. [cit. 2015-05-11]. Dostupné z: http://en.wikipedia.org/wiki/IEC_60870

[9] IEC 60870-5. 2001-. Wikipedia: the free encyclopedia [online]. [cit. 2015-05-11]. Dostupné z: http://en.wikipedia.org/wiki/IEC_60870-5

[10] DNP3. 2001-. Wikipedia: the free encyclopedia [online]. [cit. 2015-05-11]. Dostupné z: http://en.wikipedia.org/wiki/DNP3

[11] IEC 61850. 2001-. Wikipedia: the free encyclopedia [online]. [cit. 2015-05-11]. Dostupné z: http://en.wikipedia.org/wiki/IEC_61850

[12] Modbus. 2001-. Wikipedia: the free encyclopedia [online]. [cit. 2015-05-11]. Dostupné z: http://en.wikipedia.org/wiki/Modbus

[13] Profibus. 2001-. Wikipedia: the free encyclopedia [online]. [cit. 2015-05-11]. Dostupné z: http://en.wikipedia.org/wiki/Profibus

[14] Sim104. Sim104 [online]. [cit. 2015-05-11]. Dostupné z: https://code.google.com/p/sim104/downloads/detail?name=sim104.jar&can=2&q=

Příloha A – Ukázka průběhu komunikace klient server v programu Wireshark

Source	Destination	Protocol	Length	Info
192.168.1.235	192.168.1.248	104apci	60	<- U (STARTDT act)
192.168.1.248	192.168.1.235	104apci	60	-> U (STARTDT con)
192.168.1.248	192.168.1.235	104asdu	74	-> I (0,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104asdu	70	<- I (0,1) ASDU=1 C_IC_NA_1 Act IOA=0
192.168.1.248	192.168.1.235	104asdu	70	-> I (1,1) ASDU=1 C_IC_NA_1 Req_NEGA IOA=0
192.168.1.235	192.168.1.248	104apci	60	<- S (2)
192.168.1.248	192.168.1.235	104asdu	74	-> I (2,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (3)
192.168.1.248	192.168.1.235	104apci	60	-> S (1)
192.168.1.248	192.168.1.235	104asdu	74	-> I (3,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (4)
192.168.1.248	192.168.1.235	104asdu	74	-> I (4,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (5)
192.168.1.248	192.168.1.235	104asdu	74	-> I (5,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (6)
192.168.1.248	192.168.1.235	104asdu	74	-> I (6,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (7)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR act)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104asdu	74	-> I (7,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (8)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR act)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104asdu	74	-> I (8,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (9)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR act)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104asdu	74	-> I (9,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (10)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR act)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104asdu	74	-> I (10,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (11)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR act)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104asdu	74	-> I (11,1) ASDU=1 M_ME_NC_1 Spont IOA=1

Příloha B – Ukázka komunikace ve Wiresharku při vysílání řídících příkazů

Source	Destination	Protocol	Length	Info
192.168.1.235	147.230.77.98	104apci	60	<- U (STARTDT act)
147.230.77.98	192.168.1.235	104apci	60	-> U (STARTDT con)
147.230.77.98	192.168.1.235	104asdu	190	-> I (0,0) ASDU=1 C_IC_NA_1 ActCon IOA=0 | -> I (1,0) ASDU=1 M_SP_NA_1 Inrogen IOA[
192.168.1.235	147.230.77.98	104asdu	70	<- I (0,4) ASDU=1 C_CI_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	86	-> I (4,1) ASDU=1 C_CI_NA_1 ActCon IOA=0 | -> I (5,1) ASDU=1 C_CI_NA_1 ActTerm IOA=
192.168.1.235	147.230.77.98	104asdu	76	<- I (1,6) ASDU=1 C_CS_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	76	-> I (6,2) ASDU=1 C_CS_NA_1 ActCon IOA=0
192.168.1.235	147.230.77.98	104apci	60	<- S (7)
192.168.1.235	147.230.77.98	104asdu	70	<- I (2,7) ASDU=1 C_IC_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	190	-> I (7,3) ASDU=1 C_IC_NA_1 ActCon IOA=0 | -> I (8,3) ASDU=1 M_SP_NA_1 Inrogen IOA[
192.168.1.235	147.230.77.98	104asdu	69	<- I (3,11) ASDU=1 C_RD_NA_1 Req IOA=1
147.230.77.98	192.168.1.235	104apci	60	-> S (4)
192.168.1.235	147.230.77.98	104asdu	78	<- I (4,11) ASDU=1 C_TS_TA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	78	-> I (11,5) ASDU=1 C_TS_TA_1 ActCon IOA=0
192.168.1.235	147.230.77.98	104asdu	69	<- I (5,12) ASDU=1 C_RD_NA_1 Req IOA=1
192.168.1.235	147.230.77.98	104asdu	70	<- I (6,12) ASDU=1 C_IC_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	190	-> I (12,7) ASDU=1 C_IC_NA_1 ActCon IOA=0 | -> I (13,7) ASDU=1 M_SP_NA_1 Inrogen IO
192.168.1.235	147.230.77.98	104apci	60	<- S (16)
192.168.1.235	147.230.77.98	104asdu	70	<- I (7,16) ASDU=1 C_IC_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	190	-> I (16,8) ASDU=1 C_IC_NA_1 ActCon IOA=0 | -> I (17,8) ASDU=1 M_SP_NA_1 Inrogen IO
192.168.1.235	147.230.77.98	104asdu	69	<- I (8,20) ASDU=1 C_RD_NA_1 Req IOA=1
192.168.1.235	147.230.77.98	104asdu	78	<- I (9,20) ASDU=1 C_TS_TA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	78	-> I (20,10) ASDU=1 C_TS_TA_1 ActCon IOA=0
192.168.1.235	147.230.77.98	104apci	60	<- S (21)
192.168.1.235	147.230.77.98	104asdu	70	<- I (10,21) ASDU=1 C_IC_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	190	-> I (21,11) ASDU=1 C_IC_NA_1 ActCon IOA=0 | -> I (22,11) ASDU=1 M_SP_NA_1 Inrogen
192.168.1.235	147.230.77.98	104asdu	69	<- I (11,25) ASDU=1 C_RD_NA_1 Req IOA=1
147.230.77.98	192.168.1.235	104apci	60	-> S (12)
192.168.1.235	147.230.77.98	104asdu	69	<- I (12,25) ASDU=1 C_RD_NA_1 Req IOA=1
147.230.77.98	192.168.1.235	104apci	60	-> S (13)
192.168.1.235	147.230.77.98	104apci	60	<- U (TESTFR act)
147.230.77.98	192.168.1.235	104apci	60	-> U (TESTFR con)
192.168.1.235	147.230.77.98	104asdu	69	<- I (13,25) ASDU=1 C_RD_NA_1 Req IOA=1
192.168.1.235	147.230.77.98	104asdu	70	<- I (14,25) ASDU=1 C_IC_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	190	-> I (25,15) ASDU=1 C_IC_NA_1 ActCon IOA=0 | -> I (26,15) ASDU=1 M_SP_NA_1 Inrogen
192.168.1.235	147.230.77.98	104apci	60	<- S (29)

Příloha C – Ukázka rozboru ASDU ve Wiresharku

```
⊞ IEC 60870-5-104-Apci: -> I (1,0)
⊟ IEC 60870-5-104-Asdu: ASDU=1 M_SP_NA_1 Inrogen IOA[2]=2,... 'single-point information'
    TypeId: M_SP_NA_1 (1)
    0... .... = SQ: False
    .000 0010 = NumIx: 2
    ..01 0100 = CauseTx: Inrogen (20)
    .0.. .... = Negative: False
    0... .... = Test: False
    OA: 0
    Addr: 1
  ⊞ IOA: 2
  ⊞ IOA: 4
⊟ IEC 60870-5-104-Apci: -> I (2,0)
    START
    ApduLen: 82
    .... ...0 = Type: I (0x00)
    Tx: 2
    Rx: 0
⊟ IEC 60870-5-104-Asdu: ASDU=1 M_ME_NC_1 Inrogen IOA[9]=1,... 'measured value, short floating point number'
    TypeId: M_ME_NC_1 (13)
    0... .... = SQ: False
    .000 1001 = NumIx: 9
    ..01 0100 = CauseTx: Inrogen (20)
    .0.. .... = Negative: False
    0... .... = Test: False
    OA: 0
    Addr: 1
  ⊟ IOA: 1
      IOA: 1
      Value: 42,2489
    ⊟ QDS: 0x00
        .... ...0 = OV: No overflow
        ...0 .... = BL: Not blocked
        ..0. .... = SB: Not Substituted
        .0.. .... = NT: Topical
        0... .... = IV: Valid

Transmission Control Protocol, Src Port: 2404 (2404), Dst Port: 54749 (54749), Seq: 82, Ack: 62, Len: 36
IEC 60870-5-104-Apci: -> I (3,2)
  START
  ApduLen: 18
  .... ...0 = Type: I (0x00)
  Tx: 3
  Rx: 2
IEC 60870-5-104-Asdu: ASDU=1 M_ME_NC_1 Inrogen IOA=1 'measured value, short floating point number'
  TypeId: M_ME_NC_1 (13)
  0... .... = SQ: False
  .000 0001 = NumIx: 1
  ..01 0100 = CauseTx: Inrogen (20)
  .0.. .... = Negative: False
  0... .... = Test: False
  OA: 0
  Addr: 1
⊟ IOA: 1
    IOA: 1
    Value: 22,2
  ⊟ QDS: 0x00
      .... ...0 = OV: No overflow
      ...0 .... = BL: Not blocked
      ..0. .... = SB: Not Substituted
      .0.. .... = NT: Topical
      0... .... = IV: Valid
IEC 60870-5-104-Apci: -> I (4,2)
  START
  ApduLen: 14
  .... ...0 = Type: I (0x00)
  Tx: 4
  Rx: 2
IEC 60870-5-104-Asdu: ASDU=1 C_IC_NA_1 ActTerm IOA=0 'interrogation command'
  TypeId: C_IC_NA_1 (100)
  0... .... = SQ: False
  .000 0001 = NumIx: 1
  ..00 1010 = CauseTx: ActTerm (10)
  .0.. .... = Negative: False
  0... .... = Test: False
  OA: 0
  Addr: 1
⊟ IOA: 0
    IOA: 0
    QOI: Station interrogation (global) (20)
```