VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Jakub Srp

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

SIMULÁTOR PRŮMYSLOVÉ KOMUNIKACE STANDARDU IEC 61850
IEC 61850 INDUSTRIAL COMMUNICATION SIMULATOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jakub Srp**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Blažek**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

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

Ústav telekomunikací

Student: Bc. Jakub Srp ***ID:*** 177783

Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Simulátor průmyslové komunikace standardu IEC 61850

POKYNY PRO VYPRACOVÁNÍ:

Práce je zaměřena na realizování pracoviště, které bude simulovat síťovou infrastrukturu standardu IEC 61850. Student provede analýzu komunikačního standardu IEC 61850. Dále bude zprovozněna ICS/SCADA síť s implementovanými protokoly ze standardu IEC 61850. Výstupem práce bude pracoviště, které bude realizovat komunikaci protokolů MMS (Manufacturing Message Specification), GOOSE (Generic Object Oriented Substation Events), SMV (Sampled Measured Values) a SNTP (Simple Network Time Protocol) mezi zařízeními dle standardu. Dále budou implementovány zařízení (alespoň tři typy), které budou simulovat koncové prvky v průmyslovém odvětví (relé, elektroměry a další) dle specifikací reálných zařízení. Dílčím cílem práce bude vytvoření generátoru dat zmíněných protokolů se záznamovým (logovacím) systémem pro zpětnou analýzu.

DOPORUČENÁ LITERATURA:

[1] W. Huang, "Learn IEC 61850 configuration in 30 minutes," 2018 71st Annual Conference for Protective Relay Engineers (CPRE), College Station, TX, 2018, pp. 1-5.doi: 10.1109/CPRE.2018.8349803

[2] Libiec61850: Open source library for IEC 61850, 2016, [online] Available: http://libiec61850.com/libiec61850/.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: Ing. Petr Blažek

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Práce se zabývá komunikačními protokoly podle standardu IEC 61850. Konkrétně se jedná o protokoly SMV (Sampled Measured Values), GOOSE (Generic Object Oriented Substation Events) a MMS (Manufacturing Message Specification) a jejich využití v systémech SCADA. V práci je navržen simulátor, který na zařízení Raspberry Pi simuluje komunikaci pomocí zmíněných protokolů, dle standardu IEC 61850. Je simulována ochrana přepětí, podpětí, jistič, odpojovač, HMI. Pro ovládání stanice je využit protokol MMS. Simulace je ovlivnitelná uživatelem z konfiguračních textových souborů.

KLÍČOVÁ SLOVA

IEC 61850, GOOSE, SMV, MMS, SNTP, SCADA, IED, simulátor, síťová komunikace, Raspberry Pi

ABSTRACT

This thesis is focused on IEC 61850 communication protocols SMV, GOOSE and MMS and their implementation in SCADA systems. There is a simulator, run on Raspberry Pi, that generates data according to IEC 61850 and transmits the data using protocols in question. The simulation consist of various virtual devices e.g. surge protection, undervoltage protection, circuit breaker, disconnector, HMI. The MMS protocol is used for station control. Simulation can be user-defined from textual configuration file.

KEYWORDS

IEC 61850, GOOSE, SMV, MMS, SNTP, SCADA, IED, simulator, network communication, Raspberry Pi

SRP, Jakub. *Simulátor průmyslové komunikace standardu IEC 61850*. Brno, Rok, 92 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Petr Blažek

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Simulátor průmyslové komunikace standardu IEC 61850“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Petru Blažkovi. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Již dávno není pravdou, že telekomunikační sítě jsou využívány výhradně k telefonické komunikaci mezi dvěma klienty. V současné době jsou tyto sítě využívány například také pro přenos dat nebo pro monitoring a vzdálenou správu průmyslových zařízení. Jeden ze systémů, který to umožňuje, je systém SCADA (z anglického "Supervisory Control And Data Acquisition"). Systémem SCADA se obvykle rozumí software, který monitoruje daná průmyslová zařízení a následně předává data pro jejich další zpracování. Pomocí systému SCADA se dají zařízení také ovládat. V praxi se může jednat například o vyčítání hodnot měřicích přístrojů, případně vypnutí, či zapnutí zařízení při dosažení určité hraniční hodnoty. Jedním ze standardů, který je v energetice využíván pro komunikaci mezi zařízeními, je soubor norem IEC 61850, který je popsán v této práci.

Tato práce ve své teoretické části popisuje právě soubor norem IEC 61850 a několika jejich komunikačních protokolů. Konkrétně se jedná o protokoly GOOSE (Generic Object Oriented Substation Events), SMV (Sampled Measured Values), MMS (Manufacturing Message Specification) a SNTP (Simple Network Time Protocol). V praktické části je na zařízeních Raspberry Pi realizována infrastruktura, která simuluje prvky tak, aby byla zastoupena komunikace těchto protokolů. Každý z těchto protokolů zastává svůj jedinečný úkol v rámci topologie.

Protokol MMS je využit k pravidelnému monitorování stanice, či sepnutí nebo naopak vypnutí měření veličin. Dále jsou implementovány zařízení, které simulují koncové prvky v průmyslovém odvětví (relé, jistič, či odpojovač) dle specifikací reálných zařízení. Tyto zařízení primárně používají protokol GOOSE, který je implementován pro hlášení stavů zařízení a odesílání časově kritických zpráv v případě problému na síti, způsobeného zkratem, přepětím či podpětím. Odesílaná data jsou generována a následně odesílána pomocí protokolu SMV.

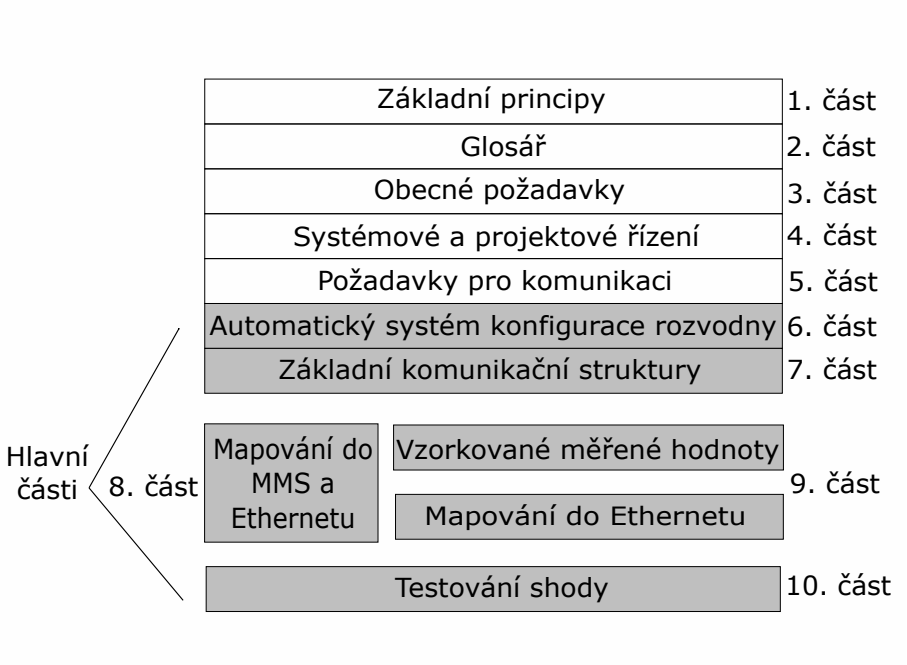
Uživatel je schopen definovat a ovlivňovat simulaci pomocí vstupních textových konfiguračních souborů a tlačítkových ovládacích prvků. Každá stanice taktéž loguje veškerý provoz, který je ukládán do textového souboru. Výstupem práce je univerzální, jednoduše šiřitelný program, který po drobných úpravách bude schopen monitoringu a komunikace podle standardu IEC 61850.

Diplomová práce je zaměřena na co nejrealističtější simulaci komunikace, snadnou rozšiřitelnost a přenositelnost knihovny standardu IEC 61850.

1 Standard IEC 61850

V energetice se nachází velké množství komunikačních protokolů, které mezi sebou nejsou kompatibilní. Jak uvádí [1], pro komunikaci je proto nutné implementovat postupy, které nejsou jednoduché a ekonomicky výhodné. Standard IEC 61850 je soubor norem a předpisů, které definují komunikaci v oblasti energetiky a elektrizačních soustav. Jeho českým ekvivalentem je norma ČSN EN 61850. Cílem této normy nebo také standardu IEC 61850, je sjednotit postupy komunikace a definovat tak neměnný, celosvětově uznávaný standard, který je vhodný pro současné, ale i budoucí potřeby průmyslové komunikace. Primárně je tento standard používán pro monitoring a ovládání elektrických rozvoden.[2]

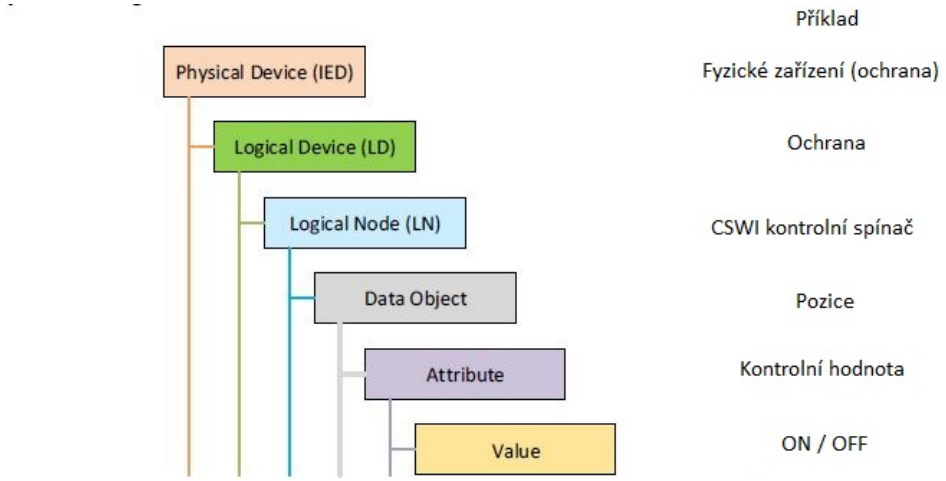
Standard IEC 61850 pracuje se všemi vrstvami referenčního modelu ISO/OSI. Nicméně kritická data jsou směrována ze sedmé, aplikační, vrstvy, přímo do druhé, linkové, vrstvy. Standard definuje, až sedm základních typů zpráv, které mají určitou úroveň a dělí se ještě na pod zprávy. Samotný protokol GOOSE má dvě úrovně, které jsou buď 4 ms nebo 10 ms.[1] Celý standard IEC 61850 je rozdělen na 10 hlavních částí, přičemž nejdůležitější části jsou části 6 – 10. Tak, jak je vyobrazeno na obr. 1.1.



Obr. 1.1: Struktura standardu IEC 61850 [3].

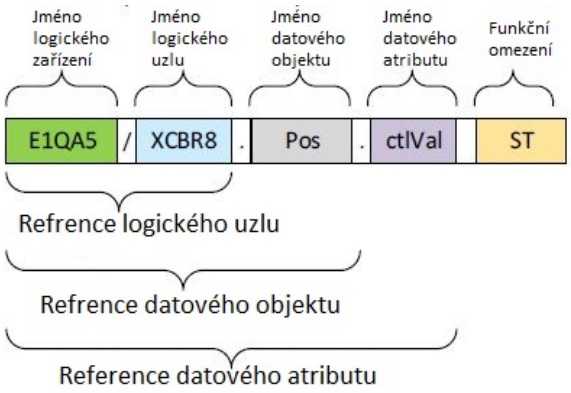
1.1 Datový model standardu IEC 61850

Standard IEC 61850 má jasně definovanou syntaxi, která vychází z objektů, které popisuje. Standard je objektově orientovaný, což přispívá k trvalému zachování datového modelu. Mimo jiné, je také tímto způsobem zajištěna kompatibilita a vzájemná interoperabilita mezi zařízeními od různých výrobců. [1] Datový model je složen z prvků, které jsou vyobrazeny na obr. 1.2. Datový model by se dal svojí strukturou přirovnat k referenčnímu modelu ISO/OSI, kde každá vrstva zajišťuje specifickou funkci v identifikaci dat jednotlivých protokolů. Podrobný popis jednotlivých prvků je níže.



Obr. 1.2: Datový model standardu IEC 61850 [3].

Model je rozdělen do jednotlivých částí, které v kombinaci vytvoří adresu objektu, která má i sama o sobě určitou výpovědní hodnotu. Reálný příklad je uveden na obr. 1.3.



Obr. 1.3: Reálný příklad adresace objektu [3].

1.1.1 Fyzické zařízení (IED)

Zkratka PD vychází z anglického slovního spojení Physical Device. Může být také nazýváno jako IED (Intelligent Electronical Device). Tímto fyzickým zařízením se rozumí například relé nebo jiné fyzické zařízení v rozvodně např. ochrana, jistič, SAS apod. Zařízení je připojeno do sítě, tím pádem musí být definováno IP adresou a svým unikátním názvem, který může být o délce maximálně 10 znaků. [1][3]

1.1.2 Logické zařízení (LD)

Zkratka LD vychází z anglického slovního spojení Logical device. V rámci jednoho fyzického zařízení, se může nacházet více logických zařízení. Toto zařízení agreguje data z jednotlivých logických zařízení do nadřazeného fyzického zařízení. Každé logické zařízení obsahuje jeden nebo více logických uzlů.[3]

1.1.3 Logický uzel (LN)

Zkratka LN vychází z anglického slovního spojení Logical Node. Každá funkce rozvodny nebo zařízení, komunikující podle standardu IEC 61850 je virtuálně reprezentována logickým uzlem. Jedná se o seskupení dat a služeb, které souvisí s určitou funkcí rozvodny. Tím pádem všechna data, která jsou rozvodnou vygenerována, mohou být přiřazena k určitému logickému uzlu. Ve standardu IEC 61850 je právě Logický uzel tou nejmenší entitou, která je schopna výměny dat.[3]

1.1.4 Datový objekt (DO)

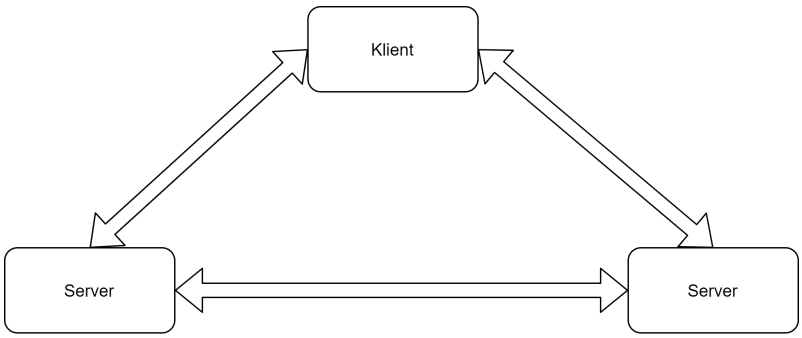
Zkratka DO vychází z anglického slovního spojení Data Object. Logické uzly se skládají z datových objektů. Tyto datové objekty mohou být předávány mezi jednotlivými logickými uzly. Datové objekty tvoří základ objektově orientovanému standardu IEC 61850.[1]

1.1.5 Datový atribut

Tato část datového modelu je nejmenší. Funkcí datového atributu je uchovávat určitou hodnotu. U datového atributu musí být vždy určen jak datový typ, tak název. Může se jednat například o jednoduchý typ *Boolean* nebo o složený typ *Quality*.[4]

1.2 Komunikace podle standardu IEC 61850

Při komunikaci podle standardu IEC 61850 je rozlišována komunikace mezi IED, které jsou topologicky na stejné úrovni a těmi, které jsou topologicky nadřazené. Komunikace mezi jednotkami stejné úrovni se nazývá horizontální. Naopak přenos zpráv mezi řídícím střediskem a jemu podřízenými jednotkami se nazývá vertikální. Pro představu může sloužit jednoduchá topologie, která je vyobrazena na obr.1.4. Primární rozdíl mezi vertikální a horizontální komunikací je v principu, na kterém funguje. Vertikální komunikaci standard IEC 61850 definuje jako typ klient – server. Tento typ komunikace využívá protokol SMV. Oproti tomu protokol GOOSE využívá horizontální komunikace. Pro optimální fungování komunikace je nutné zajistit také časovou synchronizaci jednotlivých uzlů. [4]



Obr. 1.4: Příklad jednoduché topologie [4].

1.2.1 Vertikální komunikace

Vertikální komunikace je využívána například pro ovládání stanic člověkem nebo jinak operované řídící stanice SCADA. Při této komunikaci jednotlivá IEC 61850 pracují jako server, který shromažďuje data a je připraven odpovídat na žádosti od klienta, v tomto případě na žádost systému SCADA. Sytém SCADA, působící jako klient, iniciuje komunikaci a zasílá požadavky na vyčítání dat a kontrolních příkazů. [8]

Označení vertikální vyplývá z organizační hierarchie jednotlivých stanic. Při této komunikaci dochází k přenosu zpráv mezi stanicí, která je umístěna výše a stanicí, která je umístěna níže. Dalo by se tedy říci, že komunikace probíhá vertikálně. Tento typ komunikace využíván například pro přenos souborů, ovládacích příkazů nebo podávání hlášení. Jako komunikační protokol je využíván protokol MMS (Manufacturing Message Specification). [5]

Pro vysílání surových vzorků měření ostatním IED v rozvodně je využit SMV (Sampled Measured Values) protokol. Komunikace probíhá vždy mezi klientem a serverem. Jde o protokol, který je čistě pro zasílání velkého počtu vzorků v krátkém čase. Neřeší se, že například jeden vzorek nedorazil, proto je použit typ komunikace vydavatel – odběratel.

1.2.2 Horizontální komunikace

Při horizontální komunikaci jsou si z topologického hlediska zařízení rovny. Tato komunikace je typu vydavatel – odběratel. Toto označení pochází z anglického termínu publisher – subscriber. V obou případech se ovšem jedná o totožnou komunikaci. Rozdíl je pouze z hlediska jazykového.

Horizontální komunikace se využívá pro komunikaci mezi jednotlivými prvky rozvodny na stejné topologické úrovni. Dříve bylo nutné jednotlivé piny všech zařízení fyzicky propojit měděnými vodiči, pro zajištění jejich komunikace. Tento princip může být do jisté míry zachován i v současné době, nicméně provedení je mnohem ekonomičtější a efektivnější. Mohou být použity například kroucené dvoulinky, čili také metalické kabely, které jsou přímo spojovány pin na pin pomocí konektorů RJ-45, ovšem za podpory technologie Ethernet.

Při tomto řešení, oproti původní verzi, je možné komunikaci přenášet po jediném kabelu a data distribuovat do jednotlivých zařízení v síti pomocí přepínačů, bez nutnosti jejich fyzického propojení. To znamená, že není nutné vytvářet metalický kabelový propoj pro všechny stanice zvlášť, nýbrž stačí jeden centrální kabel, který dokáže přenášet ethernetové rámce. Nové řešení využívající Ethernet proto přináší úsporu z hlediska nutné kabeláže, tím pádem i rychlejší implementaci, která zabere méně prostoru.

Konkrétní využití horizontální komunikace spočívá například v zasílání povelu, který přichází z vývodního do přívodního pole monitorovaných prvků. Dalším možným použitím může být blokáda mezi přívodními poli a příčnou spojkou přípojnic u dvousekčních a více sekčních rozvoden. Příklad rozvodny využívající standard IEC 61850 je uveden v [17].

Při horizontální komunikaci jsou zprávy přenášeny pomocí multicastu. Za multicasty jsou v síťové komunikaci označovány zprávy, které mají více od jednoho zdroje více příjemců. Pro časově kritický a spolehlivý přenos standard IEC 61850 využívá GOOSE (Generic Object Oriented Substation Events) protokol. [6][7]

1.2.3 Časová synchronizace

Pro zajištění optimálního výkonu každé sítě, a správného fungování síťových protokolů, je nutné zajistit časovou synchronizaci jednotlivých uzlů v síti. Pro správnou synchronizaci, je nutné, aby jeden uzel sítě zastupoval funkci serveru a tím určoval čas pro ostatní uzly sítě.

Ostatní uzly, které v komunikaci fungují jako klient, vyšlou žádost serveru, který do paketu s odpovědí přidá svůj lokální čas. Následně klient vypočítá rozdíl mezi svojí časovou značkou a značkou, kterou obdržel od serveru. Po dokončení výpočtu, klient přenastaví své interní hodiny právě o tuto hodnotu, aby v konečném výsledku byly zařízení ve fázi.

Přesnost synchronizace závisí především na použití komunikačního protokolu a hardwaru. Nutno podotknout, že při přenosu také vznikají určité chyby a nepřesnosti. Většina těchto chyb je zapříčiněna zpožděním, mezi vytvořením časové značky serverem a následným zpracováním paketu s odpovědí ze strany klienta. Jako nejpřesnější a nejflexibilnější protokol pro časovou synchronizaci přes internet, nebo v rámci LAN sítě, je označován protokol SNTP. [9]

SNTP protokol je využíván například pro značkování rámců. Díky časové značce je možné identifikovat přijaté rámce a porovnávat je s odeslanými. V případě, že při komunikaci dojde k problému, je možné identifikovat čas vzniku problému a tím usnadnit diagnostiku. Využití časových značek taktéž přináší větší přehlednost a výpovědní hodnotu do logovacích souborů.

Standard IEC 61850 rozlišuje několik úrovní přesnosti časové synchronizace. Jedná se celkem o pět tříd, které jsou rozděleny podle důležitosti z hlediska stupně ochrany. Na časové synchronizaci je závislé odesílání například kritických zpráv protokolu GOOSE. Jednotlivé třídy přesnosti, spolu s povolenými odchylkami se nachází v tabulce 1.1.

Tab. 1.1: Časové třídy přesnosti [9].

IEC Třída T1	±1 ms
IEC Třída T2	± 0.1 ms
IEC Třída T3	± 25 µs
IEC Třída T4	± 4 µs
IEC Třída T5	± 1 µs

2 Komunikační protokoly podle standardu IEC 61850

V kapitole 1.2 jsou rozebrány určité typy komunikace dle standardu IEC 61850. U každého komunikačního typu je také uveden příklad protokolu, který je využíván. Současná kapitola rozebírá jednotlivé protokoly u zobrazuje jejich praktické využití. V práci jsou rozebrány protokoly GOOSE, SMV, MMS, SNTP a jejich aplikace v reálném prostředí.

2.1 GOOSE protokol

Zkratka GOOSE vznikla z anglického názvu Generic Object Oriented Substation Events. Jak již bylo zmíněno v kapitole 1.2, GOOSE protokol je využíván ve standardu IEC 61850 k horizontální komunikaci typu vydavatel – odběratel. Tento model byl zvolen kvůli zajištění vysokých přenosových rychlostí a spolehlivosti, která je od GOOSE protokolu vyžadována. GOOSE zprávy jsou využívány pro stavových a řídících informací mezi jednotlivými IED. [9]

Protokol GOOSE je možné použít jak pro horizontální, tak pro vertikální typ komunikace, ovšem většinou je využíván pro komunikaci horizontální. Detailní popis a rozdíl mezi horizontální a vertikální komunikací je popsán v kapitole číslo 1.2. Horizontální komunikace probíhá mezi dvěma prvky na stejné topologické úrovni. většinou se jedná o ochrany, které periodicky zasílají data o stavu dané jednotky k jednotce sousední.[16]

2.1.1 Struktura komunikace

I přesto, že je protokol GOOSE možné využít jak pro vertikální, tak pro horizontální komunikaci, je tento protokol ve standardu IEC 61850 využíván především pro komunikaci horizontální. Jedná se o komunikaci na stejné topologické či hierarchické úrovni. Většinou se jedná o ochrany, které periodicky zasílají data o stavu jedné jednotky k jednotce sousední. Tato komunikace je založena na principu odesílatel – příjemce, což znamená, že neprobíhá žádné navazování komunikace.[15]

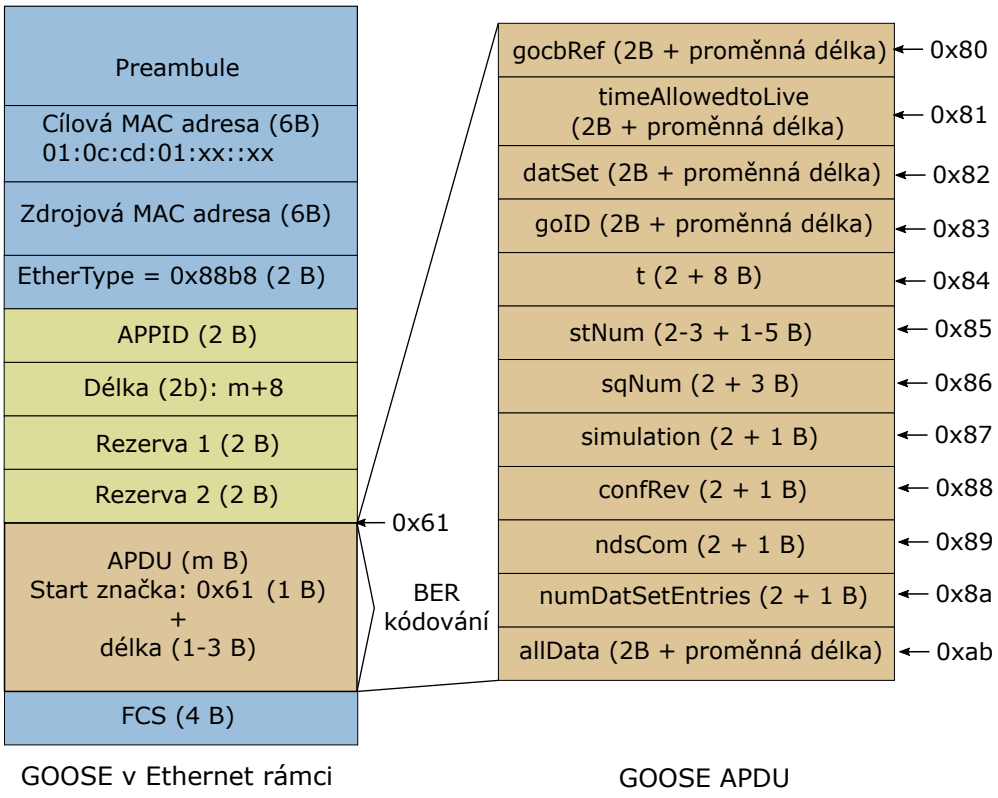
V rámci referenčního modelu ISO/OSI jsou GOOSE zprávy spojovány se třemi vrstvami. konkrétně se jedná o první, fyzickou vrstvu, druhou, linkovou vrstvu a sedmou, aplikační vrstvu. Na druhé vrstvě jsou zapouzdřeny do Ethernetového rámce podle standardu 802.3. [3] IED, která vysílá GOOSE zprávu, ji odešle všem ostatním IED v dané síti. GOOSE zprávě je přiřazen příznak vysoké priority, čímž se dostane na začátek fronty, který se nachází na portu ethernetového přepínače. V tu chvíli jsou tato odeslána prioritně a dostávají se před zbylou komunikaci. [9]

Standard IEC 61850 definuje, že je nutné, aby v případě nějaké nové události, IED obrdžely GOOSE zprávu během 3 milisekund. Aby byl protokol schopen splnit tyto podmínky, zasílá nelineární rychlostí zprávy s rostoucími pořadovými čísly a tím zajistí doručení alespoň jedné ze zpráv. Dále jsou zasílány zprávy v daném intervalu. Díky tomuto systému, jsou přijímací IED schopny detekovat výpadek či poruchu protistrany. [9]

Komunikace probíhá mezi dvěma stanicemi na stejné úrovni v infrastruktuře, na horizontální úrovni. Jeden z klientů je označen jako vydavatel a druhý je označen jako odběratel. Data jsou ukládána a vyčítána z lokálních vyrovnávacích pamětí obou klientů. Vydavatel zapíše potřebná data do své lokální vyrovnávací paměti, čili na straně odesílatele. Na druhé straně odběratel vyčítá hodnoty taktéž z lokální vyrovnávací paměti, nicméně se jedná o vyrovnávací paměť na straně příjemce. Komunikační systém má za úkol aktualizovat vyrovnávací paměti všech odběratelů. Vydavatel využívá obecnou třídu pro kontrolu událostí rozvodny (generic substation event control class), pro kontrolu celého procesu. [3]

2.1.2 GOOSE zpráva na linkové vrstvě

Jak již bylo uvedeno v kapitole 2.1.1, GOOSE zprávy je možné najít na druhé vrstvě referenčního modelu ISO/OSI. Pro komunikaci jsou tedy zapouzdřovány do ethernetových rámců podle standardu 802.3. Ethernetový rámec má standardní atributy, čili preambule, cílová adresa, zdrojová adresa, typ paketu, data rámce a kontrolní součet. Dále je rámec rozšířen o zapouzdření podle standardu IEEE 802.1Q, zajišťující tagování VLAN. Tato kapitola detailně popisuje strukturu zapouzdření a funkci jednotlivých polí v ethernetovém rámci. Samotná ethernetový rámec je vyobrazen na obr. 2.1. [15]



Obr. 2.1: Ethernetový rámec s GOOSE zprávou [3].

Na první pohled se může ethernetový rámec jako standardní rámec síťové komunikace, nicméně standard IEC 61850 definuje specifika, na jejichž základě je možné určit, že systém komunikuje podle daného standardu. Jak již bylo uvedeno, GOOSE zprávy jsou přenášeny po ethernetové síti pomocí multicastu. Technická komise standardu IEC 61850 vyčlenila konkrétní multicastovou MAC adresu, pro přenášení GOOSE zpráv. Konkrétně se jedná o následující adresu: *01:0c:cd:01:xx:xx*. [3] Zdrojová MAC se samozřejmě odvíjí vždy od konkrétního zařízení. [15]

Tab. 2.1: Hodnoty pole Typ Paketu [3].

Typ protokolu	Symbol
GOOSE Type 1	0x88b8
GSE Management	0x88b9
Sampled Values	0x88ba
GOOSE Type 1A	0x88b8

Nemusí se vždy jednat pouze o GOOSE protokol, může to být například i protokol SMV. Rozlišení jednotlivých protokolů v ethernetovém rámci se provádí v poli *Typ Paketu* nebo anglicky *EtherType*. Každý protokol má definovanou a standardizovanou hodnotu, která je registrována u autority IEEE. Dané hodnoty je možné vidět v tabulce 2.1 . Tyto hodnoty jsou vkládány do pole *Typ Paketu* a na jejich základě určeno, o jaký protokol standardu IEC 61850 se jedná. [3]

V běžném ethernetovém rámci je následující pole vyplněno přenášenými daty. Při GOOSE komunikaci tomu není jinak, avšak toto pole má jasně definovaná pravidla tak, aby vyhovovala standardu IEC 61850. [16] Právě v tomto poli najdeme samotný GOOSE protokol, který tak jako jiné komunikační protokoly, je rozdělen na několik podružných částí. V hlavičce protokolu se nachází dvoubajtové pole APPID. Tato zkratka je odvozeno z anglického Aplication ID a slouží k identifikaci přijímací aplikace. V podstatě se jedná o vytažení GOOSE zprávy z ethernetového rámce a rozlišení, kterí aplikaci náleží. Hodnota, kterou APPID nabývá, je kombinací APPID typu, což jsou nejvýznamnější bity dané hodnoty a jejího skutečného ID. Hodnota, které může dané pole nabývat například u GOOSE Typ 1 je: 0x0000 – 0x3FFF. [3][15]

Dalším dvoubajtovým pole je pole *Lenght* neboli délka. Tato hodnota definuje počet oktetů a to včetně hlavičky, čili od APPID, spolu s APDU. Zkratka APDU je odvozena z anglického Application Protocol Data Unit. Ze standardu je definováno, že pole *Lenght* bude nebývat hodnot 8+m, kde neznámá m reprezentuje délku samotného APDU, přičmž jeho délka nepřesáhne 1492. Rámce, které nebudou konzistentní nebo budou mít nevhodnou délku jsou automaticky zahozeny. [3]

Následují dvě, opět dvoubajtové pole, které nesou anglický název *Reserved 1* a *Reserved 2*. Každé z těchto polí má jiný význam. Nejedná se pouze o vyhrazený prostor v protokolu pro budoucí standardizaci, ale i tento fakt nebylo opomenuto.

Jak již bylo zmíněno, pole *Reserved 1* má dvoubajtovou strukturu. Obsahuje jeden S bit (z anglického simulated), který je mapován z parametru *Simulation* dané služby z zařízení odesílající GOOSE zprávu. 3 další bity jsou tak zvané R bity (z anglického reserved), a právě tyto bity jsou rezervovány pro budoucí standardizované aplikace. Zbylých 12 bitů je vyhrazeno pro zabezpečení GOOSE komunikace. Pokud není komunikace šifrována nebo jinak zabezpečena, bude tento parametr mít hodnotu 0.[16]

Pole *Reserved 2* je též využíváno pro bezpečnostní účely. Ve chvíli kdy pole není využíváno, je jeho hodnota také nastavena na 0.[3]

2.2 Sampled Values protokol

Sampled Values protokol, označován také jako Sampled Measured Values by se dal do češtiny volně přeložit jako protokol pro přenos navzorkovaných hodnot. SMV a GOOSE pracují velmi podobně. Hlavním rozdílem je typ komunikace a struktura zprávy jednotlivých protokolů. Zatímco GOOSE je primárně zaměřen na horizontální komunikaci, SMV slouží pouze pro vertikální komunikaci. Tato komunikace může probíhat například mezi měřící jednotkou a topologicky vyšším prvkem, což může být například HMI. Tato zkratka vychází z anglického Human Machine Interface, což je rozhraní mezi zařízením a člověkem, které představuje prostředky pro zobrazení a předání informace o stavu zařízení obsluze nebo operátorovi.

2.2.1 Struktura komunikace

Jak již bylo zmíněno, protokol SMV je velmi podobný protokolu GOOSE. Další společnou vlastností těchto protokolů je, že oba fungují na principu odesílatel – příjemce, neboli publisher – subscriber. To znamená, že protokoly nemají žádné navázání komunikace. Rozdíl je v tom, že SMV oproti GOOSE je určen výhradně pro vertikální komunikaci.

Koncept SMV komunikace spočívá v periodickém odesílání SMV zpráv v jasně definovaném časovém intervalu. Tento časový interval je definován dvěma faktory, ze kterých je následně vypočítán. Prvním faktorem, definujícím časový interval u protokolu SMV je naměřená frekvence signálu. Ta může mít například 50 Hz. Druhým definujícím faktorem pro výpočet časového intervalu je hodnota SPP. Zkratka SPP vychází z anglického Samples Per Periode. Volně přeloženo do češtiny se jedná o počet vzorků za určitou periodu. Sám název definuje i význam dané hodnoty. Ve standardu IEC 61850 může hodnota SPP nabývat dvou hodnot.[10] Konkrétně se jedná o hodnoty 80 a 256. Pokud by tedy naměřená frekvence signálu bylo doopravdy zmiňovaných 50 Hz a hodnota SPP například 80, výpočet časového intervalu by byl následovný:

$$1/50/80 = 0,00025s, \tag{2.1}$$

V případě, že by byl počet vzorků za danou periodu 256, výpočet by se změnil do následující podoby:

$$1/50/256 = 0,000078125s, \tag{2.2}$$

Je samozřejmé, že výsledek by se lišil i při jiné frekvenci signálu. Například při frekvenci 60 Hz by byl výpočet následovný:

$$1/60/80 = 0.00020833333s, \tag{2.3}$$

Při odesílání SMV zpráv, jsou všechny zprávy jsou rozřazeny do jednotlivých kategorií. Příjemce následně přijímá veškeré zprávy, které jednotliví odesílatelé poslali. Nepřijme je všechny, nýbrž je zanalyzuje a vyfiltruje si pouze zprávy z té kategorie, k jejímuž odběru se přihlásil. Vzhledem k tomu, že SMV protokol je typu odesílatel – příjemce, komunikace je možná pouze v rámci lokální sítě. [10]

2.2.2 SMV zpráva na linkové vrstvě

Již bylo zmíněno mnohokrát, že SMV protokol je velmi podobný protokolu GOOSE. Není tomu jinak ani při zapouzdřování a přenosu po druhé vrstvě referenčního modelu ISO/OSI. Protokol SMV využívá ke svému přenosu ethernetové rámce, do kterých je zapouzdřen.

Stejně jako u GOOSE protokolu je ethernetový rámec na první pohled naprosto standardní. Skládá se standardně z preambule, cílová adresy, zdrojová adresy, typu paketu, dat rámce a kontrolního součtu. Zobrazení Samotného ethernetového rámce, jehož obsahem je i protokol SMV, se nachází v příloze A.

Ze standardu IEC 61850 je definována cílová multicastová MAC adresa *01:0c:cd:04:xx:xx*. Zde se nachází první rozdíl oproti GOOSE paketu, který má definovanou cílovou multicastovou MAC adresu jako *01:0c:cd:01:xx:xx*. Zdrojová MAC adresa samozřejmě závisí na MAC adrese odesílatele.

Dalším polem je pole pro prioritizaci rámců. Prioritizování je uskutečňováno pomocí prioritizačních značek, neboli tagů. V tomto poli je definováno TPID, jehož hodnota je 0x8100. Tato hodnota je definována samotným protokolem 802.1Q ze standardizační sbírky IEEE. Na základě této hodnoty je přepínač schopen určit, že se jedná skutečné o rámec, který byl tagován podle standardu 802.1Q. Tato hodnota je stále součástí standardní ethernetové hlavičky a není definována protokolem SMV. [10]

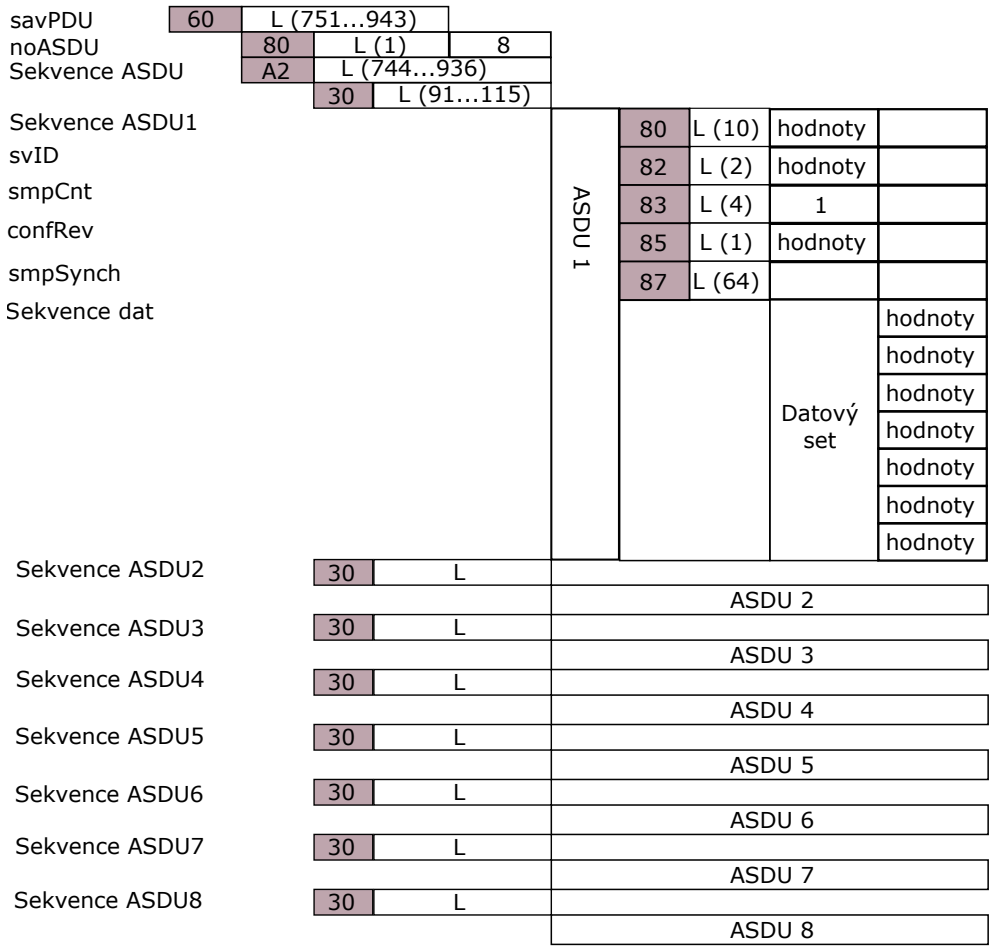
Stále je definováno pole pro prioritizaci rámců, které se skládá z menších jednotek. Konkrétně se jedná o pole *User priority* a *VID*. V poli *User priority* jde nastavit rámci priorita, která je definovaná uživatelem. U protokolu SMV se zprávy s prioritou v rozmezí 1 - 3, považují za zprávy z nízkou prioritou. naopak zprávy s prioritou 4 - 7, jsou v protokolu SMV považovány za zprávy s vysokou prioritou.[10]

Pole *VID* je vyhrazeno pro ID virtuální sítě LAN. Zkratka pochází z anglického Virtual Local Area Network ID. Jde o číslo, které identifikuje zařazení rámce do určité virtuální lokální sítě, která byla nakonfigurována na daném zařízení. Virtuální lokální sítě slouží k logickému rozdělení provozu na nezávislé linky, i přesto, že je použito jedno zařízení. Hodnota VID může nabývat hodnot 0 - 4095. [10]

Veškerá pole, která byla zmíněna, jsou součástí standardní hlavičky ethernetového rámce. Jinak tomu není ani u dalšího pole *EtherType Ethernetový typ*, které v sobě nese informaci o konkrétním ethernetovém typu, který se v rámci nachází. Standard IEC 61850 jasně definuje hodnoty, kterých může toto pole nabývat, aby komunikující zařízení věděla, o jaký komunikační protokol se jedná. Tyto hodnoty jsou normovány podle standardu IEEE, tudíž je možné je využít i na běžných přepínačích, které odpovídají tomuto standardu. Hodnoty pro jednotlivé protokoly dle standardu IEC 61850 jsou vyobrazeny v tabulce 2.1. Z tabulky je patrné, že pro protokol SMV bude pole *EtherType* nabývat hodnoty *0x8ba*. [3]

Při rozboru klasického ethernetového rámce by dalším polem byla datová část, neboli *Payload*. Právě v tomto poli se ukrývá protokol SMV, definovaný standardem IEC 61850. Stejně jako protokol GOOSE má nejprve svou hlavičku a následně datovou část. Hlavička protokolu je totožná s protokolem GOOSE a její podrobný rozbor je uveden v kapitole 2.1.2. Nicméně hodnoty jednotlivých polí mohou být rozdlné. Rezervované části v rámci nebo jeho délka jsou samozřejmě indivuduální na dané komunikaci. Ovšem pole APPID, jehož význam a podrobný popis je opět definován v kapitole 2.1.2, nabývá oproti GOOSE protokolu rozdílných hodnot. Zatímco u GOOSE protokolu nabývá toto pole hodnot 0x0000 – 0x3FFF. [3] U protokolu SMV je tato hodnota v rozmezí 0x4000 – 0x7FFF. [10]

Posledním polem SMV protokolu je pole APDU neboli Application Protocol Data Unit. Umístění v těle ethernetového rámce je opět totožné s protokolem GOOSE, nicméně zde jejich podobnost končí. V poli APDU se totiž nachází data samotné SMV zprávy. V každém APDU je možné umístit až 8 ASDU.Zkratka ASDU vychází z anglického Application Specific Data Unit. Volným překladem do češtiny by se mohlo jednat o specifickou aplikační datovou jednotku. Každé ASDU má unikátní identifikační hodnotu protokolu SMV. Zároveň také ASDU obsahuje jedno třífázové měření proudu a napětí. [10] Obsah pole Application Protocol Data Unit je zobrazen na obrázku 2.2.



Obr. 2.2: Obsah pole APDU, převzato z [10].

Je definováno přímo standardem IEC 61850, že každé ASDU pole se musí skládat z následujících dalších polí:

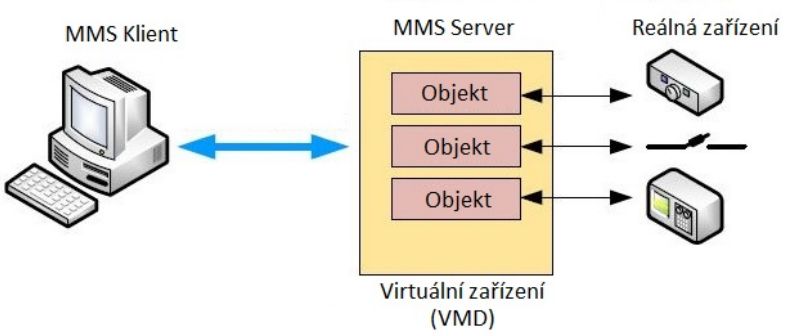
- svID - Sampled Values IDentifier. Jedná se o unikátní identifikátor zvolený uživatelem, který je používán pro multicastový odběr.
- smpCnt - index Sampled Measured Values zprávy
- confRev - revize konfigurace
- smpSynch - toto pole definuje synchronizační mechanismus hodin, které jsou používány pro odesílání SMV zpráv. Může nabývat pouze tří hodnot:
  - 0 - není využit žádný synchronizační mechanismus
  - 1 - je využit lokální synchronizační mechanismus
  - 2 - je využit vzdálený synchronizační mechanismus
- Sequence of Data - sekvence naměřených hodnot proudu a napětí

2.3 MMS protokol

Zkratka MMS vychází z anglického Manufacturing Message Specification. Tento typ protokolu je zpravidla využíván pro horizontální komunikaci typu klient – server. Z praktického hlediska to znamená, že komunikace pomocí protokolu MMS je použita pro komunikaci s nadřazenými systémy jako na například SCADA nebo HMI. Už ze samotného umístění protokolu v topologii je patrné jeho primární využití. Protokol MMS slouží ke správě a monitoringu zařízení (IED), které jsou v daném systému. Přenáší zpravodajské služby jako například vzniklé alarmy nebo události. Dohled a vzdálena správa pomocí MMS protokolu je nezbytně důležitá pro funkčnost celé rozvodny.

Jak již bylo zmíněno, protokol MMS využívá komunikační model typu klient – server. Tento model je obecný a jedná se spíše o popsání principu komunikace, než o jedinečný způsob využívaný protokolem MMS nebo standardem IEC 61850. V principu se jedná o komunikaci dvou zařízení. Tyto zařízení spolu nenavazují spojení, nýbrž jedno zařízení odpovídá na dotazy toho druhého, čili princip by se dal taktéž popsat jako dotaz - odpověď. [11]

U komunikace pomocí protokolu MMS jako klient vystupuje síťová aplikace či zařízení. Většinou se jedná například o dohledový systém, nebo centrum správy rozvodny. Server, je taktéž zařízení nebo aplikace. Dané zařízení obsahuje takzvané VMD, včetně objektů, kterými mohou být například proměnné. Zkratka vyplývá z anglického Virtual Manufacturing Device. VMD bude blíže popsáno v kapitole 2.3.1. Při komunikaci pomocí MMS protokolu, klient vyšle dotaz servisní služby a server na jeho dotazy odpovídá. Protokol MMS je objektově orientovaný, čili využívá objektové třídy, instance a metody. Například třídy: *Named Variable, Domain, Program invocation* nebo instance a metody: *read, write, store, start, stop*. VMD objekt je definován jako nosič, ve kterém jsem umístěny všechny další objekty. K těmto objektům může protokol MMS přistupovat. Náčrt principu komunikace protokolu MMS je zobrazen na obr. 2.3 [3]



Obr. 2.3: Komunikace MMS. Převzato z [3].

2.3.1 VMD model a MMS objekty

Co se týče samotného VMD modelu, ten definuje *objekty (objects)*, *služby (services)* a *chování (behavior)*. VMD model je zaměřen pouze na síťovou komunikaci. Definuje pouze viditelné aspekty komunikace. To znamená, že princip, jakým způsobem je VMD model implementován do reálného zařízení„ nebo jakým způsobem dané zařízení model implementuje, není definováno protokolem MMS. Jak již bylo zmíněno, VMD model definuje objekty, služby a chování. Jejich význam je následovný:

- Objekty (objects): mohou to být například proměnné nebo také atributy jako *name, value, type*. Jedná se o ty proměnné nebo atributy, které jsou obsaženy na serveru.
- Služby (services): zde mohou být uvedeny například *read* nebo *write*. Služby slouží k přístupu a správě jednotlivých objektů.
- Chování (behavior): jak již z názvu vyplývá, zde VMD model definuje, jaké chování by mělo zařízení vykazovat při zpracování daných služeb. [3]

Protokol MMS definuje řadu objektů, které mohou být nalezeny v mnoha zařízeních. Ze standardu ISO 9506-2[18] je pro každý objekt definována korespondující služba. Přehledné rozdělení MMS objektů a jejich mapování na objekty IEC 61850 je vyobrazeno na obr. 2.4. To, co je na obrázku označováno jako *MMS services* neboli "MMS služby", jsou vlastně metody, které pracují s MMS objekty. [3]

MMS Object	IEC 61850 Object	MMS Services
Application Process VMD	Server	Initiate
		Conclude
		Abort
		Reject
		Cancel
		Identify
Named Variable Objects	Logical Nodes and Data	Read
		Write
		InformationReport
		GetVariableAccessAttribute
		GetNameList
Named Variable List Objects	Data Sets	GetNamedVariableListAttributes
		GetNameList
		DefineNamedVariableList
		DeleteNamedVariableList
		Read
		Write
		InformationReport
Journal Objects	Logs	ReadJournal
		InitializeJournal
		GetNameList
Domain Objects	Logical Devices	GetNameList
		GetDomainAttributes
		StoreDomainContents
Files	Files	FileOpen
		FileRead
		ObtainFile
		FileClose
		FileDirectory
		FileDelete

Obr. 2.4: MMS služby. Převzato z [3].

Přístup k jednotlivým objektům muže být řízen a omezován. Povolení k přístupu je udělováno na základě přístupového seznamu. Tento přístupový seznam slouží definování klientů, kteří mají právo do objektů nejen přistupovat, ale také je modifikovat, případně i mazat. Tento seznam se nachází v objektu *Access Control List*. Jedná se v podstatě o speciální objekt, díky kterému je možné kontrolovat ostatní objekty. [3]

Jak již bylo zmíněno, MMS služby jsou de facto MMS metody. Tyto metody mohou modifikovat objekty na přístupového seznamu. Konkrétně mohou objekty například vytvořit či smazat (*creation, deletion*), vyčítat hodnoty objektů (*get, report*), upravovat hodnoty objektů (*write, alter*), stahovat či nahrávat (*domains, files*) nebo provádět příkazy (*start, stop,...*). [3]

2.3.2 Zapouzdření

Zapouzdření komunikace protokolu MMS je poměrně rozsáhlé téma. Jeho kompletní rozbor je proto vynechán a tato kapitola rozebírá pouze obecné základní zapouzdření protokolu pro jeho funkčnost v počítačových sítích.

Samotný protokol MMS nespecifikuje, jakým způsobem adresovat servery a jejich klienty. Vzhledem k tomu, spoléhá na pravidla schémata ostatních protokolů, které pro svou komunikaci a přenos dat využívají. V praxi jsou například klienti servery adresování pomocí Internet Protokolu, čili IP adresou. Samotný protokol MMS je následně zapouzdřen až na čtvrté vrstvě modelu ISO/OSI, konkrétně do TCP protokolu, na portu číslo 102. Port 102 patří mezi (dobře) známé porty, tak zvané *well known ports*. Nicméně je nutné podotknout, že právě port 102 je vyhrazen pro *ISO TSAP Class 0*. [3]

Vyšší vrstvy pro komunikaci využívají ISO identifikátorů. Mezi tyto identifikátory patří například TSAP (Transport Service Access point, zdrojové a cílové reference protokolu COTP (Connection-Oriented Transport Protocol) a tak dále. Zapouzdření obsahuje několik ISO protokolů, které jsou součástí tak zvaného ISO protokolového zásobníku. Výraz protokolový zásobník může být též označován jako sada protokolů. I přesto, že se termíny často používají jako synonyma, jejich význam je lehce odlišný. Přesně řečeno, sada protokolů je definice protokolů, zatímco protokolový zásobník je jejich softwarová implementace.

Layer	PDU	Protocols
Application (L7)	APDU	Manufacturing Message Specification (MMS): ISO 9506
		Association Control Service Element (ACSE): ISO/IEC 8650/X.227
Presentation (L6)	PPDU	OSI Connection Oriented Presentation ISO 8823/X.226
Session (L5)	SPDU	OSI Connection Oriented Session: ISO 8327/X.225
Transport (L4)	TPDU	Connection-Oriented Transport Protocol: ISO/IEC 8073/X.224
		ISO Transport over TCP (TPKT): RFC 1006
		Transmission Control Protocol (TCP): RFC 793
Network (L3)	NPDU	Internet Protocol (IP): RFC 791
Data Link (L2)	Data Frame	Ethernet: ISO/IEC 8802-3
Physical (L1)	Bits	

Obr. 2.5: Zapouzdření MMS. Převzato z [3].

Protokol MMS nevyužívá všechny protokoly z protokolového zásobníku ISO pro každou svoji zprávu. Nicméně přehled protokolů, z protokolového zásobníku ISO, které mohou být použity při komunikaci MMS protokolu jsou vyobrazeny na obr. 2.5. [3]

3 Praktická část

Z hlediska komunikačních síťových protokolů, které jsou využívány v elektroenergetice, je praktická část této práce zaměřena na rozbor protokolů SMV, GOOSE a MMS. Pro tyto účely byl vytvořen simulátor, který koresponduje s reálným zapojením rozvodny. V tomto simulovaném prostředí jsou generována data, která odpovídají hodnotám reálných zařízení.

Pro správné fungování uvedených protokolů je nezbytná také časové synchronizace, která je zajištěna pomocí síťového protokolu SNTP. Konkrétní využití jednotlivých protokolů a jejich aplikace v rámci simulovaného prostředí, je rozebrána později v této práci.

V první fázi vývoje byla zařízení vytvořena a zprovozněna pouze ve virtuálním prostředí, pomocí softwaru VMware workstation 15. Tento software je využíván k virtualizaci zařízení. Díky tomu je možné na jediném hardwaru spustit více virtuálních strojů. Právě tato možnost umožnila vývoj několika rozdílných zařízení bez investic do dalšího hardwaru. V druhé fázi byly již hotové programy přesunuty na zařízení Raspberry Pi.

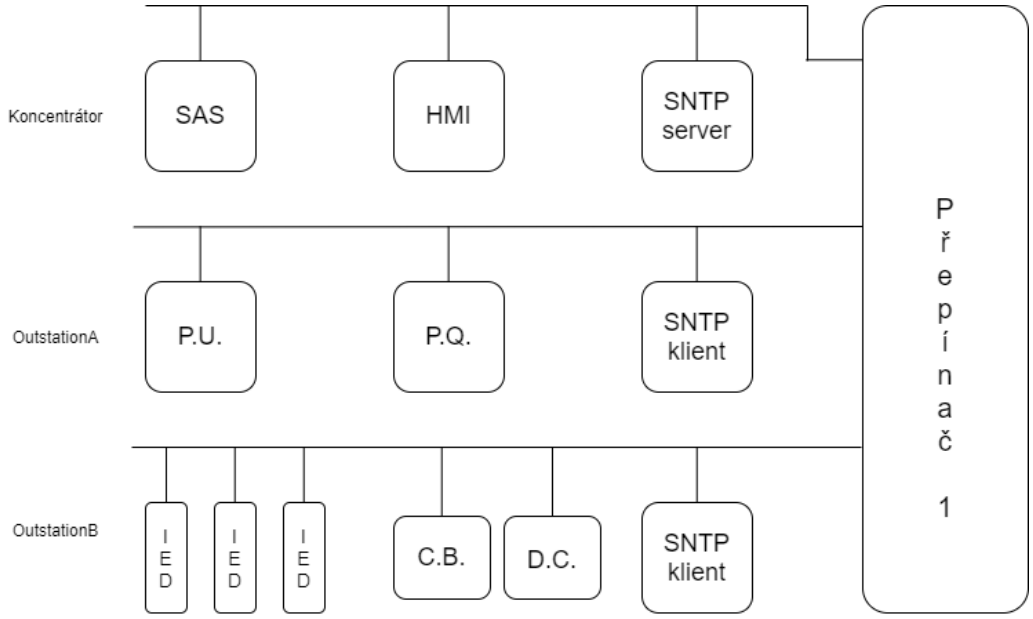
3.1 Simulované prostředí

Zařízení jsou zapojena do topologie, která je vyobrazena na obr. 3.1. Všechny zařízení Raspberry Pi pracují s operačním systémem Raspbian, což je operační systém odvozený z jedné z distribucí Linuxu, konkrétně z Debianu.

Zařízení v této práci jsou označována jako stanice Outstation1, Outstation2 a Koncentrátor. Každé z těchto zařízení simulují jinou funkci rozvodny, vyobrazené na obr. 3.1. Jako Koncentrátor je označován SAS neboli Substation Automation System. Koncentrátor slouží k přijímání a zpracování dat přijatých z podřízených stanic, v této práci nazývaných jako Outstation.

To znamená, že na koncentrátoru se nachází GOOSE subscriber a SMV subscriber. Blíže popsáno v kapitolách 3.2.1 a 3.2.1. Mimo jiné také Koncentrátor slouží k ovládání jednotlivých stanic pomocí protokolu MMS a také jako SNTP server. Popis fungování Koncentrátoru se nachází v kapitole 3.2.3.

Pro realizaci Koncentrátoru a obou Outstation byl při vývoji na virtuálních zařízeních použit použit operační systém Debian jen se základními systémovými nástroji a implementovanou knihovnu pro realizaci standardu IEC 61850. Podobně tomu tak je i u Raspbianu, který využívají zařízení Raspberry Pi. Koncové stanice generují hodnoty, které co nejvěrněji simulují reálné hodnoty, které by mohly být naměřeny v rozvodně. Následně jsou data zpracována v Koncentrátoru a zobrazena přímo v CLI, které simuluje uživatelské rozhraní. Samotná komunikace je detailně popsána v kapitole 3.2.1.



Obr. 3.1: Zapojení simulátoru IEC 61850

3.2 Simulované prvky

Účelem této práce je simulovat komunikaci v rámci energetické rozvodny, tak jak popisuje kapitola 3.1. Každá stanice, která je spuštěna na zařízení Raspberry Pi, simuluje několik nezbytných součástí reálných rozvoden. Tato kapitola pojednává o jednotlivých stanicích a jejich významu v rámci simulace. Vzhledem k tomu, že se jedná pouze o simulaci, nikoli o vytvoření jednotlivých hardwarových prvků rozvodny, existují případy, kdy jsou v rámci simulace využity stejné metody u všech stanic.

V určitých případech mají stanice za úkol tutéž funkci, a proto by opakování duplicitních věcí bylo redundantní. Vzhledem k tomuto faktu jsou v jednotlivých kapitolách popsány prvky, které jsou charakteristické pro danou stanici. Postupy, které jsou společné pro obě stanice jsou popsány vždy v té kapitole, kde se daná problematika hodí více do kontextu. V případě, že je nutné zmínit postup, který byl popsán již v předešlých kapitolách, je vždy uveden křížový odkaz na danou kapitolu, pro lepší orientaci čtenáře.

Tato práce vychází z prostředí navrženého v práci [1]. V původní verzi spolu sice zařízení komunikovala pomocí protokolů standardu IEC 61850, nicméně tato komunikace nenabývala reálných hodnot. Protokoly přenášely pouze hodnoty datové typu *Boolean*, čili 1 nebo 0. Pro účely této práce bylo nutné přenastavit stanice, aby simulovaly reálné hodnoty, které je možné měřit na rozvodnách, které jsou aktivně nasazeny v síti. Je logické, že pokud by měla být monitorována například úroveň napětí, hodnoty 1 a 0 jsou nedostačující.

Mimo jiné byla práce byla také rozšířena o další prvky, které přibližují simulátor reálné rozvodně tak, jak je zobrazeno na obr. 3.1. Taktéž byl do simulátoru také implementován protokol MMS pro ovládání jednotlivých stanic. Přidáno je například také logování odeslaných rámců, či časová synchronizace pomocí protokolu SNTP. Z těchto důvodů byla struktura simulátoru od základu změněna. Detailní popis jednotlivých řešení, případně i porovnání oproti původnímu řešení, je popsáno v kapitolách 3.2.1, 3.2.2 a 3.2.3.

Každá z uvedených stanic je naprogramována v jazyce C++ a využívá veřejně dostupnou knihovnu *libiec61850.*[19] Vzhledem k faktu, že knihovna libiec61850 je také psána v jazyce C++, jevilo se toto řešení jako nejvhodnější. Jak již bylo zmíněno, pro tuto práci byla vyvíjena ve virtualizovaném prostředí a následně přenesena na Raspberry Pi. Všechny stanice jsou naprogramovány tak, aby simulovaly komunikaci právě podle standardu IEC 61850.

Veškerý vývoj programu probíhal ve vývojovém prostředí NetBeans IDE 8.2, které se nachází na jednom z virtuálních zařízení, které jsou připojeny jako příloha k této práci. Tento postup byl vždy stejný, ať už se jednalo o vývoj virtuálních zařízení či Raspberry Pi. Program je přenesen do zařízení vzdáleně, tak jak popisuje kapitola 3.4. Na samotných zařízeních je program vždy pouze spuštěn, nicméně byla implementována možnost konfigurace hodnot simulátoru samotným uživatelem pomocí textového souboru. Postup spuštění samotné simulace si nachází v kapitole 3.6. A to včetně popisu zmiňované konfigurace.

3.2.1 OutstationA

OutstationA simuluje ochranu v rámci rozvodny, která má za úkol kontrolovat naměřené hodnoty. Naměřené hodnoty následně zpracovává a vyhodnotí. Za použití terminologie z obr. 3.1, se jedná o prvky P.U. z anglického Proctection Unit, neboli ochranná jednotka. Dalším zařízením je P.Q. z anglického Power Quality, neboli kvalito-metr. Zařízení mají za úkol kontrolovat, jestli jsou naměřené hodnoty v rámci definovaných mezních hodnot. Pokud je překročená mezní hodnota, která je definována uživatelem, je odeslána kritická GOOSE zpráva na koncentrátor. V případě, že tato situace nastane, koncentrátor o tomto faktu upozorní uživatele. V reálné rozvodně jsou tato zařízení využita k tomu aby bylo možné monitorovat zda nedochází v rozvodně například k přepětí nebo naopak podpětí.

Vzhledem k tomu, že se jedná pouze o simulaci, stanice neměří žádné konkrétní reálné hodnoty, nýbrž v sobě ukrývá generátor, které naměřené hodnoty simuluje. Hodnoty jsou generovány na základě rozsahu, který je určen přímo v kódu samotného programu, jak blíže specifikuje tato kapitola.

Z hlediska síťové komunikace standardu IEC 61850 stanice OutstationA využívá protokolů SMV, GOOSE a MMS. Veškeré vygenerované hodnoty jsou odesílány na Koncentrátor pomocí protokolu SMV. Jedná se tedy pouze o surová data, která jsou následně zobrazena uživateli. Tak jak funkcionalitu protokolu SMV definuje standard IEC 61850.

Protokol GOOSE je využit pro pravidelné hlášení své aktivity tím, že zasílá tak zvané "I'm alive"rámce, které jsou odesílány v intervalu pěti sekund, tak jak definuje standard IEC 61850. Další využití protokolu GOOSE je již zmiňovaná kritická GOOSE zpráva, která je odeslána vždy při překročení definované mezní hodnoty. Pro co nejrealističtější simulaci a využití této možnosti, bylo nutné do stanice implementovat také funkci, která má za úkol tyto chyby generovat. Samozřejmě je možné upravit frekvenci výskytu chyb, na které je nutné upozornit kritickou GOOSE zprávou. To blíže popisuje kapitola 3.2.1.

Protokol MMS je na stanici OutstationA využit pro spínání nebo vypínání generování specifikovaných hodnot. Na základě MMS příkazu, který je odeslán z koncentrátoru je možné ovlivnit, jestli bude nebo naopak nebude určitá veličina generována. Tato funkce má za úkol simulovat možnost sepnutí či vypnutí IED v reálné rozvodně. V rámci simulátoru je tedy možnost vypnout generování napětí, proudu či frekvence.

Generování naměřených hodnot

Pro provedení simulace komunikace podle standardu IEC 61850 je nutné mít hodnoty, které se budou jednotlivými protokoly posílat. Práce v tomto ohledu přináší značné vylepšení oproti původnímu modelu. Původní model obsahoval program v jazyce Python, který vytvořil seznam hodnot datového typu Boolean a uložil je do textového souboru. Stanice následně tento soubor otevřela a odeslala zmiňované hodnoty. Toto řešení bylo nedostačující, a proto bylo nutné systém generování hodnot změnit. Důvodem byla další práce s generovanými hodnotami, ale především další přiblížení simulátoru realitě. Na jednotlivých stanicích je generování lehce rozdílné.

Princip generování naměřených hodnot je stejný u obou stanic. Byly ovšem implementovány dva rozdílné způsoby určování rozsahu, ve kterém se má generátor pohybovat. Rozsah generování pro OutstationA je definován přímo ve zdrojovém kódu. OutstationA generuje hodnoty pro napětí, proud a frekvenci. V tomto pořadí jsou také zapsány mezní rozsahy přímo v kódu. Definované hodnoty jsou ukládány do proměnných třídy Sensor, která je dále používá k dalšímu zpracování. Jak je možné vidět na následujícím výpisu, všechny proměnné využívají datového typu Float.

Výpis 3.1: Definování rozsahu generování pro OutstationA

```
1 Sensor(float _min_volts, float _max_volts, float _min_amp,
2 float _max_amp, float _min_freq, float _max_freq)
3 ....
4 Sensor my_sensor = Sensor(230, 235, 15, 18, 47, 55);
```

Výhodou tohoto typu konfigurace je, že mezní hodnoty jsou nastaveny uvnitř samotného programu, ještě před nahráním souboru do zařízení. To znamená, že pro koncového uživatele stačí simulátor pouze spustit, bez nutnosti dalšího ovládání či konfigurace.

Další výhodou je bezesporu fakt, díky uložení hodnot proměnných přímo v kódu, je eliminována možnost poškození, či smazání předdefinovaných uživatelem, jako tomu může být při nastavení mezních hodnot například z textového souboru. Program není závislý na externím médiu, což snižuje pravděpodobnost chyby. Je možné tvrdit, že tento způsob eliminuje chyby způsobené lidským faktorem.

To, co může být považováno za výhodu, může být také považováno za nevýhodu. Absence možnosti konfigurace ze strany koncového uživatele sice eliminuje výskyt chyb způsobených lidským faktorem, nicméně případná změna rozsahu generovaných hodnot musí být provedena odborným zásahem do kódu celého programu, což může být pro koncového uživatele značně zdlouhavé a nepraktické. Mimo jiné je pro tuto úpravu nutné mít přístup do konfigurační stanice, ze které je následně celý program znovu sestaven na koncovou stanici, kde je následně spuštěn.

V této práci je generátor implementován přímo do stanic OutstationA a OutstationB. Program obsahuje smyčku, která cyklicky generuje potřebná data. Takto vygenerovaná data slouží jako náhrada naměřených hodnot z reálného IED. Proto je v jazyce C++ vytvořen generátor náhodných hodnot, který by měl simulovat hodnoty, které mohou být naměřeny například v rozvodnách. Generátor je ovšem možné ovlivnit uživatelským vstupem, tudíž generované hodnoty mohou být takřka jakékoliv.

V případě, že se jedná o datový typ Float, funkčnost simulátoru není ovlivněna konkrétní hodnotou, která je do proměnné uložena. Je tedy zcela na uživateli, jaké hodnoty chce generovat. Vždy je ovšem nutné definovat v jakém číselném rozsahu bude simulátor čísla generovat. Pro účely této práce byly zvoleny dva postupy zadávání tohoto rozsahu. Zadefinováním maximální a minimální hodnoty je taktéž určena i tak zvaná mezní hodnota. Tato hodnota určuje mez, kterou daná veličina nesmí přesáhnout. V případě, že k takové situaci dojde, je odeslána kritická GOOSE zpráva.

Jak již bylo zmíněno, pro simulaci reálných hodnot, které mohou vznikat při měření v rozvodně byl definován datový typ Float. Pro generátor byla vytvořena nová třída *Sensor*, ve které byly deklarovány potřebné proměnné typu Float. Ukázka kódu je zobrazena ve výpisu níže.

Výpis 3.2: Vytvoření třídy Sensor

```
1 class Sensor{
2 public:
3     float volt, min_volts, max_volts;
4     float amp, min_amp, max_amp;
5     float freq, min_freq, max_freq;
```

Pro co nejpřesnější simulaci měřených hodnot je vhodné, aby generovaná čísla nebyla stejná. To zajistí využití tak zvaného "random seedu". Český ekvivalent tohoto pojmu je "náhodné semínko". Jedná se o náhodné číslo, které je použito při inicializaci generátoru. Díky jinému inicializačnímu číslu, je při každém spuštění programu generovaná jiná sekvence pseudonáhodných dat. Tato práce pro inicializaci svého generátoru využívá random seed, který je generován z aktuálního času stanice. Díky tomu je zajištěno, že vygenerovaná pseudonáhodná posloupnost bude vždy jiná. Níže je ukázka implementace v jazyce C++.

```
std::srand(std::time(NULL));
```

Proměnné jsou deklarovány, mezní hodnoty generátoru zadefinovány, nicméně stále nenabývají konkrétních hodnot. Program využívá těchto proměnných právě pro ukládání náhodně vygenerovaných čísel, se kterými může dál pracovat. V této práci se jedná se o hodnoty, které by mohly být naměřeny na reálné rozvodně. Nicméně jak již bylo zmíněno, rozsah generovaných veličin je plně v rukou uživatele. Pro vygenerování hodnot a následné uložení do proměnných je vytvořena v programu nová funkce, která je zobrazena níže.

Výpis 3.3: Funkce pro generování hodnot

```
1 void init_values(){
2     volts = std::rand()/(RAND_MAX/(max_volts-min_volts))
3     + min_volts;
4     amp= std::rand()/(RAND_MAX/(max_amp-min_amp))
5     + min_amp;
6     freq = std::rand()/(RAND_MAX/(max_freq-min_freq))
7     + min_freq;
8     return;
9     }
```

Pseudonáhodný generátor v jazyce C++ nijakým způsobem neosciluje kolem vygenerované hodnoty. Čísla jsou generována z definovaného rozsahu, ovšem při každém průchodu smyčky je hodnota unikátní, bez jakékoli vazby na předchozí čísla. V reálné rozvodně může docházet ke skokovým změnám měřených hodnot, nicméně vyčítané hodnoty vždy oscilují kolem aktuálního stavu dané veličiny. V případě využití pouze neupraveného generátoru hodnot, by tedy nedocházelo k ideální simulaci. Pro vytvoření reálnějšího prostředí měření byla ve funkci `read_sensor` vytvořena podmínka, zajišťující oscilaci kolem vygenerované, tedy naměřené hodnoty. Podmínka je zobrazena na další straně.

Výpis 3.4: Podmínka pro Oscilaci kolem naměřených hodnot

```
1 float delta_volts = (std::rand()%100)/100.0;
2     if((std::rand()%100) > 50 && volts+delta_volts <
3     max_volts){
4         volts += delta_volts;
5     }
6     else if(volts-delta_volts > min_volts){
7         volts -= delta_volts;
8     }
```

Generování chyb v naměřených hodnotách

GOOSE zprávy vznikají, mimo jiné, také při náhlých událostech v rozvodně. Jedná se o tak zvanou prioritní GOOSE zprávu, která je odeslána jako upozornění na určitou nestandardní hodnotu. Vždy se jedná o překročení definované hraniční hodnoty. V reálné rozvodně může tato situace vznikat například při zatížení systému nebo při mechanické chybě. Aby byla simulace co nejblíže reálnému prostředí v rozvodně, bylo nutné tento jev implementovat i do simulátoru.

Pro tento účel, je deklarována proměnná `volts_err`, která je datového typu Integer a její výchozí hodnota je 0. Je využito opět generování náhodných čísel a pomocí matematické operace modulo, což je zbytek po celočíselném dělení, je definována pravděpodobnost, s jakou se chyba objeví. Pro procentuální vyjádření pravděpodobnosti, je použito modulo 100, které je porovnáváno s fixně deklarovanou hodnotou, která definuje právě onu pravděpodobnost. V příkladu, který je uveden níže, je znázorněna pravděpodobnost 20%. Pokud je podmínka pravděpodobnosti splněna, do proměnné `volts_err` je uložen náhodný integer v rozmezí 1 - 10, což je zajištěno opět matematickou funkcí modulo.

Výpis 3.5: Generování chyb v naměřených hodnotách

```
1 if( volts_err == 0 && (std::rand()%100) > 80){
2         volts_err = (std::rand()%10);
3     }
```

V této chvíli se v proměnné `volts_err` nachází integer v rozmezí 1 - 10. Pomocí funkce `get_volts` je vyčítána hodnota naměřeného napětí. Před samotným vyčtením proměnné, která ukrývá hodnotu změřeného napětí, funkce ještě zjistí, jestli se v proměnné `volts_err` nachází nějaký integer nebo je její hodnota nulová, tak jak byla deklarována ve výchozím stavu. V případě, že je hodnota skutečně nulová, program vypíše změřené napětí. V případě, že hodnota nulová není, program k naměřenému napětí přičítá číslo 10, čímž zajistí, že bude překročena hraniční hodnota

a systém odešle prioritní GOOSE zprávu. To, jakou hodnotu program přičítá k naměřenému napětí vychází z rozsahů, které byly použity pro generování měřených hodnot. Ve chvíli, kdy by rozdíl mezi `max_volts` a `min_volts` byl větší než 10, muselo by být samozřejmě použito větší číslo. Tato funkce je obdobně implementována i pro ostatní veličiny. Příklad funkce je uveden níže.

Před touto podmínkou se nachází ještě jedna podmínka, která určuje, jestli vůbec ke generování dojde. Jedná se o splnění podmínky, která vychází z řídícího protokolu MMS. Bližší definice této podmínky i samotného řídícího protokolu MMS se nachází v kapitole 3.2.1.

Výpis 3.6: Podmínka pro řídící protokol MMS

```
1  float get_volts() {
2      if(v_enabled) {
3          if (volts_err > 0) {
4              volts_err--;
5              return volts + 10;
6          }
7          return volts;
8      } else {
9          return -1;
10     }
11 }
```

Nastavení komunikace

Po vygenerování hodnot, které mají simulovat naměřené hodnoty, je nutné tyto hodnoty předat stanici, pro kterou mají vypovídající hodnotu. Tato práce popisuje komunikaci podle standardu IEC 61850. Jednotlivé komunikační protokoly jsou popsány v kapitole 2. Jaké hodnoty simulátor generuje je popsáno v kapitole 3.2.1. Z principiálního hlediska nezáleží, o jaké konkrétní hodnoty se jedná. Data jsou zapouzdřena do těla ethernetového rámce, kde splňují své další protokolové standardy tak, jak definuje IEC 61850.

V praxi je možné se setkat s případy, kdy pro rozvodné sítě s frekvencí 50 Hz, je využit protokol SMV, který odesílá 4000 rámců za sekundu. V rozvodných sítích s frekvencí 60 Hz SMV odesílá 4800 rámců za sekundu. [13] Pro dosažení co nejreálnější simulace je nutné tyto podmínky napodobit. V původním programu, ze kterého tato práce vychází, tento fakt zohledněn nebyl. Původní kód neřešil odesílání prioritních GOOSE zpráv při překročení nastavené mezní hodnoty.

Oproti tomu tato práce již danou problematiku zohledňuje a simulátor je tedy schopen odesílat prioritní GOOSE zprávu v případě překročení nadefinované mezní hodnoty. Opětovné odesílání je realizováno pomocí cyklu. Tento cyklus je nastaven podle definice vyplývající ze standardu IEC 61850. To znamená, že v případě překročení hraniční mezní hodnoty je spuštěn cyklus pro opětovné odeslání rámce, který při každém svém průběhu zdvojnásobí rozmezí odesílaní mezi jednotlivými rámci. Tento cyklus probíhá do té doby, dokud není synchronní s pravidelným, periodicky odesílaným rámcem, který má za úkol hlásit, jestli je dané zařízení v režimu zapnuto či vypnuto. Bližší specifikace cyklu, včetně ukázky konkrétního kódu je v kapitole 3.2.1.

Pouhé vložení tohoto cyklu do původního řešení nebylo dostačující. Samotné odesílání, ať už SMV nebo GOOSE rámců, je tvořeno taktéž cyklem. Jednalo se tedy o vnoření jednoho cyklu do dalšího cyklu. Problém nastal ve chvíli, kdy byla překročena nadefinovaná hraniční mez a byl spuštěn cyklus pro znovu-odeslání GOOSE zpráv. V daný moment přestal běžet primární cyklus zajišťující standardní komunikaci a čekal do doby, než bude ukončen cyklus znovu-odesílání. Po ukončení tohoto cyklu je obnoven chod primárního cyklu a obnoveno standardní odesílání. Jak již bylo zmíněno, podle standardu IEC 61850 je nutné ovšem odeslat 4000 SMV rámců, což není možné v případě, že odesílací cyklus je zastaven a blokován jiným cyklem. Dalším důvodem je, že pro standardní komunikaci při měření v rámci hraničních mezí, kdy protokol GOOSE pouze informuje HMI o své funkčnosti, by bylo odesílání 4000 GOOSE rámců velmi nadbytečné. Mimo jiné by byla taktéž zbytečně zatěžována síťová infrastruktura.

Pro eliminaci tohoto problému bylo zvoleno řešení, které rozděluje odesílání GOOSE rámců a SMV rámců do dvou nezávislých procesů. Při tomto řešení jsou tedy rámce odesílány zvlášť, podle svých individuálních nastavení a jejich odesílání není vzájemně ovlivňováno.

Pro docílení tohoto požadavku je zvolen postup, který využívá časových značek a unixového času. Unixový čas je systém pro označení časových okamžiků, zejména v systémech, které jsou na bázi Unixu. Systém identifikuje časové okamžiky pomocí počtu sekund uplynulých od okamžiku koordinovaného světového času (UTC) 00:00:00 1. ledna 1970, ale bez započítání přestupných sekund.

Jak již bylo zmíněno, odesílání komunikace probíhá pomocí nekonečného cyklu. Hned na začátku cyklu je definována podmínka, která porovná rozdíl unixového času a časové značky požadovaného protokolu, s hodnotou, která je nastavena ve

zdrojovém kódu. Například v případě protokolu GOOSE je tato hodnota nastavena na 5000 ms, tak jak stanovuje standard IEC 61850. Při splnění podmínky, tím pádem i úspěšném odeslání rámce, je proměnná definující časovou značku přepsána stávající hodnotou. V případě nesplnění podmínky program pokračuje dál, bez odeslání daného rámce. Na základě této podmínky je tedy možné definovat, jaký časový úsek musí uběhnout, aby bylo možné odeslat další rámec požadovaného protokolu. Tento časový úsek se samozřejmě u jednotlivých protokolů liší a to na základě definici vyplývajících ze standardu IEC 61850. Formální zápis zmiňované podmínky v jazyce C++ je následovný:

Výpis 3.7: Podmínka pro učení rychlosti odesílání

```
1 if(Hal_getTimeInMs() - time_goose > 5000) {
2         time_goose = Hal_getTimeInMs();
```

Samozřejmostí je primární inicializování dané proměnné ještě před spuštěním samotného cyklu.

Nastavení komunikace GOOSE

Jak již bylo zmíněno, tato práce vychází již z před připraveného datového modelu protokolu GOOSE a veřejné dostupné knihovny *libiec61850*. Podrobný popis tvorby datového modelu a práce s knihovnou je popsán v [1]. Tato práce se zabývá spíše problematikou síťové komunikace.

Hodnoty protokolu GOOSE jsou přenášeny uvnitř Ethernetového rámce, a proto je nutné vytvořit data-set pro ukládání hodnot, které nabývá GOOSE, do jednotlivých rámců.

```
LinkedList dataset = LinkedList_create();
```

Následně jsou definována jednotlivá pole GOOSE protokolu, tak jak definuje standard IEC 61850. Rozbor jednotlivých polí je uveden v kapitole 2.1.2 této práce. Implementace do kódu v jazyce C++, se nachází ve výpisu na další straně.

Výpis 3.8: Definování polí protokolu GOOSE

```
1  CommParameters cp;
2  cp.vlanPriority = 4;
3  cp.vlanId = 0;
4  cp.appId = 0x1000;
5  cp.dstAddress[0] = 0x01;
6  cp.dstAddress[1] = 0x0c;
7  cp.dstAddress[2] = 0xcd;
8  cp.dstAddress[3] = 0x10;
9  cp.dstAddress[4] = 0x00;
10 cp.dstAddress[5] = 0x00;
11
12 GoosePublisher goosePublisher = GoosePublisher_create(&cp,
13 interface);
14
15 GoosePublisher_setGoID(goosePublisher, "TEST");
16 GoosePublisher_setGoCbRef(goosePublisher, "TEST/LLN0$TEST");
17 GoosePublisher_setDataSetRef
18     (goosePublisher, "TEST/LLN0$TEST");
19 GoosePublisher_setTimeAllowedToLive
20     (goosePublisher,(1.5 * 1000));
21 GoosePublisher_setConfRev(goosePublisher, 1);
22 GoosePublisher_setSimulation(goosePublisher, 0);
23 GoosePublisher_setNeedsCommission(goosePublisher, 0);
```

Je možné si povšimnout, že již z kódu je patrná korelace mezi informacemi uvedenými v kapitole 2.1.2. Jako příklad může být uvedena definovaná multicastová cílová adresa. Jednotlivá pole odpovídají standardu IEC 61850.

Celý proces odesílání se odehrává v jedné smyčce. V této smyčce se nachází cyklus while, jehož spuštění ani ukončení není omezeno žádnou podmínkou. Ve chvíli kdy je cyklus spuštěn, je opakován do té doby, dokud není manuálně vypnut. Postupně jsou volány jednotlivé objekty a jejich funkce či proměnné, které ve své kombinaci vytvoří samotný GOOSE rámec. Kompletní procesy a deklarace jsou zapsány ve zdrojových kódech, které se nachází v příloze C.

Primárním cílem této práce je vytvoření co nejrealističtějšího simulátoru komunikace podle standardu IEC 61850. V reálných zařízeních je při překročení definované mezní hodnoty odeslána prioritní GOOSE zpráva, která je následně odesílána znova s cyklicky se zvětšující periodou do doby, než je dosažena doba pravidelného odesílání. Pro dosažení toho požadavku byla vytvořena funkce `retransmitGoose`. Tato

funkce je na stanici OutstationA využívána k monitorování přepětí a podpětí. Ve chvíli kdy dojde k přepětí či podpětí, je odeslána kritická GOOSE zpráva na koncentrátor, na jejímž základě koncentrátor zobrazí aktuální chybový stav.

Uvnitř funkce se nachází cyklus, který nastavuje spínač pro odeslání prioritní GOOSE zprávu, a zároveň definuje celkovou periodu, do které se má pravidelné odesílání GOOSE rámců vrátit. Samotný cyklus je ještě omezen podmínkou, ve které je definováno, že bude cyklus spuštěn pouze v případě, že proměnná, definující počet iterací, bude mít hodnotu nula. Implementace je následovná.

Výpis 3.9: Podmínka definující odesílání protokolu GOOSE

```
1 if (iterations == 0) {
2     while (3 * std::pow(2, iterations) < 5000) {
3         iterations += 1;
4     }
5 }
```

Standard IEC 61850 definuje, že prioritní GOOSE rámec musí být odeslán do 3ms od vzniku události, což koresponduje s nastavením uvedeným v kódu. Běžná perioda standardních GOOSE rámců je nastavena na 5000ms.

Z hlediska implementace do reálného prostředí je nutné, aby bylo z chybového GOOSE rámce jasně patrné, která hodnota překročila svoji hraniční mez a vyžaduje pozornost. To je možné samozřejmě zajistit více způsoby, nicméně pro účely této práce byl zvolen textový zápis, čili datového typu String, do samotného rámce. Z tohoto důvody jsou pomocí příkazu *switch* vytvořeny jednotlivé varianty, ke kterým může při měření hodnot dojít. Hodnota datového typu String je přidána do nadefinovaného datasetu GOOSE rámce. Zápis se nachází na další straně.

Výpis 3.10: Varianty možných chyb

```
switch (type) {
    case 0: LinkedList_add(ds, MmsValue_newMmsString("Zvysene
napeti"));
         break;
    case 1: LinkedList_add(ds, MmsValue_newMmsString("Zvyseny
proud"));
        break;
    case 2: LinkedList_add(ds, MmsValue_newMmsString("Zvysena
frekvence"));
        break;
    default:
        LinkedList_add(ds, MmsValue_newMmsString
        ("Default"));
        break;
```

Jednou z nových funkcí, které tento simulátor nabízí oproti původní verzi je již zmiňované odesílání prioritních GOOSE zpráv, které jsou popisovány i v této kapitole. Konkrétně se jedná o inicializaci funkce, která zajišťuje správný provoz. Tuto funkci je samozřejmě nutné implementovat do hlavní smyčky programu. Konkrétní kód je poté následující.

Výpis 3.11: Funkce pro odesílání prioritních GOOSE zpráv

```
float napeti = my_sensor.get_volts();
        float proud = my_sensor.get_amp();
        float frekvence = my_sensor.get_freq();

        if (napeti > my_sensor.max_volts) {
          retransmitGoose(0, napeti, dataset, goosePublisher);
        }
        if (proud > my_sensor.max_amp) {
          retransmitGoose(1, proud, dataset, goosePublisher);
        }
        if (frekvence > my_sensor.max_freq) {
          retransmitGoose(2,frekvence,dataset,goosePublisher);
        }
```

V první řadě jsou načteny hodnoty, které byly naměřeny/vygenerovány v objektu Sensor. Popis generování hodnot je podrobně popsán v kapitole 3.2.1. Následně již pomocí jednoduché podmínky IF je porovnána aktuální naměřená hodnota s maximální mezní hodnotou, která byla definována v programu. V případě, že je tato hodnota překročena, čili podmínka je splněna, je volána funkce retransmitGoose. Pro každou veličinu je připravena individuální verze funkce retransmitGoose tak, jak je popsáno výše.

Pro uvolnění paměti je vždy smazán starý dataset z předchozího cyklu pomocí:

Výpis 3.12: Smazání starého datasetu

```
1 LinkedList_destroy(dataset);
2         dataset = LinkedList_create();
```

A do nového datasetu je taktéž přidán Boolean, který má za úkol hlásit, že monitorované zařízení je v provozu. Jedná se o tak zvaný "I am alive"boolean. Níže je zobrazeno přidání do datasetu.

Výpis 3.13: Přidání nového datasetu

```
1 LinkedList_add(dataset, MmsValue_newBoolean(true));
2         GoosePublisher_publish(goosePublisher, dataset);
```

Pravidelná hlášení o funkčnosti zařízení, neboli zasílání "I am alive"paketu, je vyžadováno každých pět sekundu. Proto, byla do simulátoru implementována podmínka, která zajišťuje odesílání GOOSE rámců v této periodě. V rámci flexibility simulátoru a jeho potencionální využití například pro testování síťové komunikace podle standardu IEC 61850 je samozřejmostí, že tuto periodu lze ve zdrojovém kódu programu změnit. Hodnota se pohybuje v jednotkách milisekund tak, jak je definováno v těle samotného programu.

Nastavení SMV

Jistá míra podobnosti mezi protokoly GOOSE a SMV byla v této práci zmíněna již mnohokrát. Proto také u protokolu SMV nutné definovat jednotlivá pole protokolu a tímto způsobem vlastně sestavit samotnou zprávu, která bude následně zapouzdřena do ethernetového rámce. V první fázi je opět nastavena hlavička. Ukázka nastavení je zobrazena na následující straně.

Výpis 3.14: Definice jednotlivých polí protokolu SMV

```
CommParameters svCommParameters;

svCommParameters.appId = 0x4000;
svCommParameters.dstAddress[0] = 0x01;
svCommParameters.dstAddress[1] = 0x0c;
svCommParameters.dstAddress[2] = 0xcd;
svCommParameters.dstAddress[3] = 0x04;
svCommParameters.dstAddress[4] = 0x00;
svCommParameters.dstAddress[5] = 0x01;
svCommParameters.vlanId = 0x000;
svCommParameters.vlanPriority = 4;
```

Tak jako u protokolu GOOSE je možné si povšimnout, že hlavička koresponduje s definicí standardu IEC 61850 tak, jak je uvedeno v teoretické části této práce, konkrétně v kapitole 2.2.2. Nutno podotknout, že protokol SMV je poněkud jednodušší, než protokol GOOSE. Protokol SMV slouží v podstatě pouze k odesílání naměřených hodnot, bez větší logiky. Zjednodušeně by se dalo říct, že protokol pouze vyčte naměřenou hodnotu, kterou odesílá dál do sítě. Tato hodnota je ukládána do pole APDU (Application Protocol Data Unit). Vyjádření v jazyce C++ je následující.

Výpis 3.15: Uložení naměřené hodnoty do APDU

```
SVPublisher svPublisher = SVPublisher_create
(&svCommParameters,interface);

SVPublisher_ASDU asdugpio26 = SVPublisher_addASDU
(svPublisher, "GPIO26_RPi3", NULL, 1);
int float1 = SVPublisher_ASDU_addFLOAT(asdugpio26);
int ts1 = SVPublisher_ASDU_addTimestamp(asdugpio26);
```

Simulátor popisovaný v této práci simuluje měření tří veličin. konkrétně generuje hodnoty pro napětí, proud a frekvenci. Na příkladu uvedeném výše je zobrazena pouze ukázka práce s APDU pro vygenerované/naměřené napětí. Práce s ostatními veličinami je analogická.

Principiálně je odesílání SMV rámců totožné s odesílám rámců pomocí protokolu GOOSE. Odesílání opět běží v cyklu, který se opakuje tak dlouho, dokud není manuálně zastaven. Inicializace daného cyklu včetně proměnných, které jsou při jeho chodu použity jsou detailně rozebrány v kapitole 3.2.1.

Podstatné vylepšení oproti původnímu řešení nabízí tato práce právě při odesílání SMV rámců. Dle IEC 61850 jsou rozdíly v periodě odesílání mezi GOOSE a SMV, což v původní verzi nebylo zohledněno. Počet odeslaných SMV rámců by měl dosahovat 4000 za sekundu, [13] což je pro pravidelné neprioritní GOOSE rámce naprosto nadbytečné. Jak již popisuje kapitola 3.2.1 došlo k oddělení manipulace s jednotlivými protokoly.

Řešení dané problematiky ovšem bylo podobné. Stejně, jako je tomu u GOOSE protokolu, ideologické řešení bylo využít unixového času a v jednoduché podmínce porovnávat, jestli již uběhla zadefinovaná doba od posledního přepsání. Formální zápis je následovný.

Výpis 3.16: Podmínka pro počet odeslaných SMV rámců

```
1 if(Hal_getTimeInMs() - time_smv > 25) {
2         time_smv = Hal_getTimeInMs();
```

V podmínce je definována doba, která musí uplynout od posledního odeslání SMV rámce. Samozřejmostí je primární inicializování dané proměnné ještě před spuštěním samotného cyklu. Pomocí této podmínky je možné ovlivnit počet odeslaných rámců v čase. Bez této podmínky zařízení Raspberry Pi odesílá přibližně 9000 rámců za vteřinu, což je nadbytečné. Jednotky, které jsou použity jsou definovány v těle programu. V této práci se jedná o milisekundy.

Nastavení MMS

Protokol MMS je řídícím protokolem. Pomocí protokolu MMS je možné jednotlivé IED vzdáleně ovládat nebo vyčítat naměřené hodnoty. Komunikace funguje na principu klient – server, tudíž jedno zařízení se dotazuje a druhé zařízení mu následně odpovídá. V aktuální topologii vystupuje OutstationA jako server a Koncentrátor jako klient.

Na stanici OutstationA je protokol MMS využíván pro ovládání stanice Koncentrátorem. Konkrétně je pomocí protokolu MMS vypínáno nebo zapínáno generovaní hodnot, které simulují naměřené hodnoty z rozvodny jsou následně ukládány do ASDU rámce protokolu SMV. Pokud je na Koncentrátoru nastavena hodnota datového typu Boolean na 1, znamená to pokyn pro OutstationA, že má generovat požadované hodnoty a následně je odesílat. V případě, že je hodnota nastavena na 0, generátor na OutstationA přestává generovat náhodné hodnoty z definovaného rozsahu. Namísto toho je do ASDU vkládána hodnota -1, která je následně zobrazena i na Koncentrátoru. Při opětovném nastavení Boolean hodnoty v konfiguračním

souboru Koncentrátoru na 1, je opět spuštěno generování a vygenerované hodnoty jsou odesílány v ASDU rámce SMV. Ovládat lze jednotlivé veličiny zvlášť, je tedy možné, aby byla odesílána libovolná kombinace veličin.

Jak již bylo zmíněno, protokol MMS je řídící protokol. Pro řízení samotného MMS protokolu a datového modelu, je vytvořena nová instance objektu iedServer. OutstationA vystupuje v topologii jako server. Je tedy nutné nastavit port, na kterém bude přijímat klientské požadavky, konkrétně požadavky, které přicházejí z Koncentrátoru. Uvnitř této instance se nachází smyčka pro odesílání GOOSE a SMV zpráv, která byla popsána v této kapitole. Pro případ neúspěšného spuštění serveru je vytvořena podmínka, která uživatele na tento fakt upozorní a následně uvolní dostupné zdroje. Tím pádem zbytečně nealokuje nevyužitý prostor v paměti. Formální zápis vytvoření této instance, včetně kontrolní podmínky spuštění, je následovný.

Výpis 3.17: Instance objektu iedServer

```
1 iedServer = IedServer_create(&iedModel);
2 IedServer_start(iedServer, 102);
3
4 if (!IedServer_isRunning(iedServer)) {
5     printf("Starting␣server␣failed!␣Exit.\n");
6     IedServer_destroy(iedServer);
7     exit(-1);
8 }
```

Dále tedy probíhá smyčka programu, kde jsou sestaveny a následně odeslány GOOSE a SMV zprávy. Dále se také ve smyčce nachází funkce `IedServer_setControlHandler`. Tato funkce je zodpovědná za vykonání řídících povelů z Koncentrátoru. Po přijetí řídícího povelu, zavolá tak zvanou callback funkci `controlHandlerForBinaryOutput`. Tato funkce se v kódu nachází již před samotnou smyčkou. Má za úkol obsluhu příjmu ovládacích povelů. Dále také definuje podmínky, za kterých bude povel proveden. Výpis obou funkcí se nachází na následujících stranách.

Výpis 3.18: Řídící funkce

```
1
2  \\Výpis funkce IedServer_setControlHandler
3
4  IedServer_setControlHandler(iedServer,
5  IEDMODEL_RPi1_GGIO1_SPCSO1,
6  (ControlHandler) controlHandlerForBinaryOutput,
7  IEDMODEL_RPi1_GGIO1_SPCSO1);
8
9  IedServer_setControlHandler(iedServer,
10 IEDMODEL_RPi1_GGIO1_SPCSO2,
11 (ControlHandler) controlHandlerForBinaryOutput,
12 IEDMODEL_RPi1_GGIO1_SPCSO2);
13
14 IedServer_setControlHandler(iedServer,
15 IEDMODEL_RPi1_GGIO1_SPCSO3,
16 (ControlHandler) controlHandlerForBinaryOutput,
17 IEDMODEL_RPi1_GGIO1_SPCSO3);
```

Výpis 3.19: Callback funkce

```
1
2  \\Výpis funkce controlHandlerForBinaryOutput
3
4  static ControlHandlerResult controlHandlerForBinaryOutput
5  (void* parameter, MmsValue* value) {
6  uint64_t timestamp = Hal_getTimeInMs();
7
8  if(parameter == IEDMODEL_RPi1_GGIO1_SPCSO1) {
9      IedServer_updateUTCTimeAttributeValue
10     (iedServer, IEDMODEL_RPi1_GGIO1_SPCSO1_t, timestamp);
11     IedServer_updateAttributeValue
12     (iedServer, IEDMODEL_RPi1_GGIO1_SPCSO1_stVal, value);
13 }
14 else if(parameter == IEDMODEL_RPi1_GGIO1_SPCSO2) {
15     IedServer_updateUTCTimeAttributeValue
16     (iedServer, IEDMODEL_RPi1_GGIO1_SPCSO2_t, timestamp);
17     IedServer_updateAttributeValue
18     (iedServer, IEDMODEL_RPi1_GGIO1_SPCSO2_stVal, value);
19 }
20 else if(parameter == IEDMODEL_RPi1_GGIO1_SPCSO3) {
21     IedServer_updateUTCTimeAttributeValue
22     (iedServer, IEDMODEL_RPi1_GGIO1_SPCSO3_t, timestamp);
23     IedServer_updateAttributeValue
24     (iedServer, IEDMODEL_RPi1_GGIO1_SPCSO3_stVal, value);
25 }
26 else {
27     return CONTROL_RESULT_FAILED;
28 }
29
30 return CONTROL_RESULT_OK;
31 }
```

Pro správné fungování je samozřejmě nutné také zadefinovat proměnné, do kterých bude Boolean hodnota, neboli hodnota, která určuje jestli je generování zapnuto či vypnuto, uložena. Tato proměnná je posléze načítána ve funkci `read_sensor` a ve funkci `get_velicina` je zadefinována podmínka, díky které jsou veličiny generovány pouze pokud je Boolean hodnota. Tato podmínka se nachází v kapitole 3.2.1. Přímo funkce `get_velicina` se v kódu nenachází, je to pouze obecné označení. Každá veličina má totiž svoji funkci, tudíž i své vlastní označení.

Na konci smyčky, jejíž počátek byl definován začátku této kapitoly se nachází dvě funkce, které ukončují celý proces. Konkrétně se jedná o funkci `IedServer_stop(iedServer)`, která ukončí veškeré připojení jednotlivých klientů. Jako další, je funkce `IedServer_destroy(iedServer)`, která uvolní veškeré dostupné zdroje. Zápis v jazyce C++ je následovný:

Výpis 3.20: Funkce pro uvolnění zdrojů

```
1 IedServer_stop(iedServer);
2
3 IedServer_destroy(iedServer);
```

Kompletní zdrojový kód se nachází v příloze C.

Nastavení logování

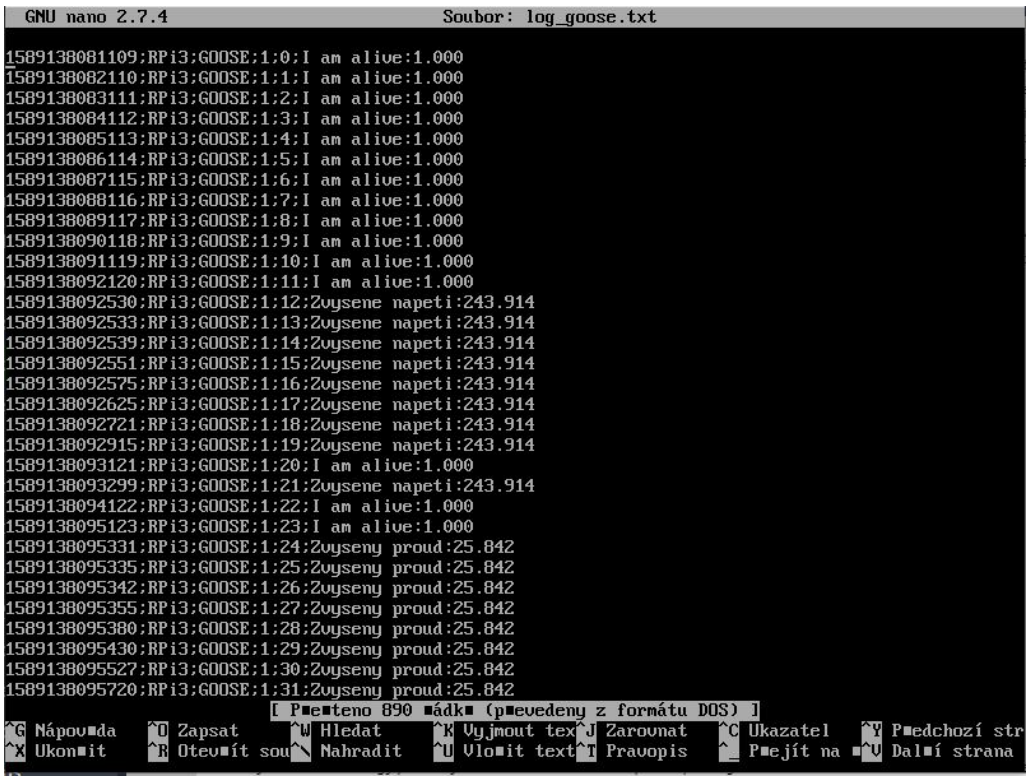
Pro samotnou tvorbu simulátoru, ale také pro případné testování a porovnávání potřebných výsledků, je důležitá funkce logování. Tato funkce je dalším z mnoha rozšíření, které tato práce nabízí oproti původnímu řešení. Jedná se vlastně o záznam přijatých, či odeslaných rámců z dané stanice a následné uložení těchto hodnot do textového souboru, pro pozdější zpracování.

Stanice OutstationA loguje vygenerované, čili neměřené hodnoty a ukládá je do různých textových souborů. Tyto textové soubory jsou rozlišeny podle protokolů, které jsou monitorovány. V případě problému, je na základě logování možné například určit, v jakou chvíli daný problém nastal nebo také frekvence jeho výskytu v rámci měření. V případě, že je tato funkce implementována na všechny stanice, je možné také lépe diagnostikovat problém v případě, že na koncovou stanici nepřichází data. Díky logování má uživatel možnost zjistit, jestli byla data ze zdrojové stanice vůbec odeslána, a nebo jestli nebyla pouze přijata koncovou stanicí.

Pro logování je vytvořena funkce, která je prakticky totožná jak pro protokol GOOSE, tak pro protokol SMV. Níže zobrazená ukázka implementace logování v jazyce C++ je pro protokol GOOSE, spolu s výsledným výpisem do příkazové řádky.

Výpis 3.21: Funkce pro logování

```
1  id logGoose(char* string, float value) {
2  FILE *fptr;
3
4  fptr = fopen("log_goose.txt", "a");
5
6  if(fptr == NULL) {
7      printf("Error opening log file!");
8  } else {
9      fprintf(fptr,"%" PRIu64 ";RPi3;GOOSE;%" PRIu32 ";%"
10     PRIu32 ";%s:%.3f\r\n", Hal_getTimeInMs(), local_stNum,
11     local_sqNum, string, value);
12 }
13 fclose(fptr);
14 return;
```



Obr. 3.2: Log protokolu GOOSE na stanici OutstationA

Textový soubor s logy, který se nachází v této práci, má jasně definovanou strukturu a obsah. Každý záznam, který je do textového souboru uložen, obsahuje vždy časové razítko, zařízení, ze kterého byl rámec odeslán, protokol standardu IEC 61850, který byl pro komunikaci použit, pořadové číslo, označení o jaký rámec se jedná a samotnou hodnotu, která byla vygenerována a uložena do těla ethernetového rámce. Ukázka ze záznamu logu pro protokol GOOSE, který je na stanici uložen v textovém souboru, je zobrazen na obr. 3.2. Záznam pro SMV je zobrazen na obr.

3.2.2 OutstationB

Principiálně jsou si stanice OutstationA a OutstationB velmi blízké, nicméně jsou mezi nimi také velké rozdíly. Každá ze stanic má za úkol simulovat jinou část rozvodny, pro dosažení co nejpřesnějším a co nejrealističtějších simulací. Z hlediska topologie vyobrazené na obr. 3.1, OutstationB simuluje 3 různá **zkied!** spolu s jističem a odpojovačem. Zkratky C.B. a D.C. na obr. 3.1 vycházejí z anglického označení pro tyto součástky. Jistič se v angličtině nazývá Circuit Breaker a odpojovač DisConnector. Jsou ovšem prvky, které mají obě stanice také společné. Jednou z těchto vlastností je například generování odesílaných hodnot nebo logování. Tato kapitola popisuje prvky a funkce, které jsou oproti stanici OutstationA rozdílné. Zbylé funkce, které jsou společné, jsou detailně popsáný v kapitole 3.2.1.

Ani jeden ze simulátorů neměří opravdové hodnoty, jako je tomu v reálných rozvodnách. Z tohoto důvody je tedy nutné měřené hodnoty simulovat, a to v obou stanicích. K tomu dochází pomocí vytvořeného generátoru, který je detailně popsán v kapitole 3.2.1. Spolu se samotným generováním měřených hodnot jsou také generovány chyby, které simulují nezvyklé situace v rozvodně, na které systém musí reagovat. Generování těchto chyb a jejich implementace do simulátoru je popsáno v téže kapitole. Jak vychází z obr. 3.1, OutstationB simuluje tři různá IED. Opět pro co nejrealističtější simulaci je nutné generovat jednotlivé hodnoty pro každé IED zvlášť.

Stanice OutstationB nabízí také oproti stanici OutstationA monitorování teploty. Teplota má opět svůj vlastní generátor, který principiálně odpovídá všem ostatním generátorům, v rámci simulátoru. Je generována teplota ambientní, neboli okolní a teplota hardwarová. Hardwarová teplota simuluje teplotu IED. Stejně jako je tomu u ostatních veličin, i teplota je plně konfigurovatelná uživatelem z textovém souboru config.cfg, který se nachází v kořenovém adresáři stanice, jak je možné vidět na obr. 3.3. Pro práci s naměřenou teplotou je využit protokol GOOSE.

Další funkcí, o kterou je doplněna tato stanice oproti nejen stanici OutstationA, ale také původní práci, je funkce jističe a odpojovače. Funkce je spouštěna pomocí stisku klávesy. Její je detailnější popis se nachází dále v této kapitole.

Tato kapitole ovšem nepojednává například o logování. Právě logování je jednou z vlastností, která je totožná pro obě simulované stanice, a proto se její popis nachází taktéž v kapitole 3.2.1. Absence popisu logování v této kapitole tedy není způsobena absencí této funkce v samotném simulátoru.

Generování naměřených hodnot

Jak již byl zmíněno, principiální postup generování samotných hodnot je na obou stanicích totožný. Nicméně oproti OutstationA je zvolen opačný přístup při definování mezních hodnot generátoru. Zde je rozmezí generovaných hodnot určeno konfiguračním textovým souborem, který je uložen v kořenovém adresáři stanice. V tomto konfiguračním souboru uživatel definuje mezní hodnoty pro jednotlivé veličiny. Z mezních veličin je následně vytvořen aritmetický průměr, který slouží jako výchozí hodnota pro generování. Screenshot konfiguračního souboru je zobrazen na obr. 3.3.

Obr. 3.3: Konfigurační soubor OutstationB

Veličiny jsou načítány z konfiguračního souboru, který je pojmenován "config.cfg". Název souboru se může lišit, nicméně je nutné aby korespondoval s názvem souboru, který je uložený na spouštěcím zařízení. V samotném kódu je název souboru, ze kterého jsou vyčítány hodnoty definován takto:

Výpis 3.22: Načtení konfiguračního souboru

```
1 load_values("config.cfg");
2 init_values();
```

Pro přenesení nadefinovaných proměnných z textového souboru do programu je vytvořena následující funkce.

Výpis 3.23: Funkce pro definování proměnných z textového souboru

```
1  void load_values(std::string filename) {
2      std::ifstream file(filename);
3      std::string line;
4      std::vector<std::string> arr;
5      int position;
6
7      while(std::getline(file, line)) {
8          position = line.find('=');
9
10         if(position != -1) {
11             arr.clear();
12             arr.push_back(line.substr(0, position));
13             arr.push_back(line.substr(position + 1,
14             line.length()));
15
16             if(arr[0] == "min_volts") min_volts =
17             stof(arr[1]);
18             else if(arr[0] == "max_volts") max_volts =
19             stof(arr[1]);
20             else if(arr[0] == "min_amp") min_amp =
21             stof(arr[1]);
22             else if(arr[0] == "max_amp") max_amp =
23             stof(arr[1]);
24             else if(arr[0] == "min_freq") min_freq =
25             stof(arr[1]);
26             else if(arr[0] == "max_freq") max_freq =
27             stof(arr[1]);
28         }
29     }
30
31     return;
32 }
```

Tato metoda je mnohem dynamičtější než varianta, která je použita na OutstationA. Pro úpravu generovaného rozsahu není potřeba zásah do zdrojového kódu, což je rozhodně rychlejší a jednodušší varianta. Koncový uživatel si tedy sám zvolí rozsah, ze kterého budou hodnoty generovány, aniž by bylo nutné program znovu sestavit.

I přesto je ovšem nutné, aby měl koncový uživatel alespoň základní znalosti práce s operačním systémem Linux. Simulátor totiž nedisponuje uživatelským prostředím a veškerou konfiguraci je nutné provádět z CLI dané stanice. V případě simulátoru vytvořeného pro účely této práce, tedy Raspbianu.

Zde vzniká prostor pro chybu způsobenou lidským faktorem. Jak již bylo zmíněno, pokud uživatel zadá hodnoty, které odpovídají datovému typu Float, simulátor bude simulovat v podstatě cokoli. V případě, že bude ovšem vložena hodnota jiného datového typu, simulátor nebude schopen s danou veličinou pracovat, tudíž nastane chyba.

Konfigurace probíhá přímo v CLI Raspbianu, což může, v případě méně zkušeného uživatele, vést například k nechtěné úpravě konfiguračního souboru do podoby, se kterou simulátor není schopen pracovat nebo dokonce k jeho smazání. V takovém případě by byl opět nutný zásah vývojáře a opětovné sestavení programu. Úpravou konfiguračního souboru do podoby, se kterou simulátor není schopen pracovat je myšleno například jeho přejmenování. Tato úprava nevyžaduje nové sestavení programu, nicméně může opět vést k dysfunkčnosti celého programu.

Nastavení komunikace

Vytvoření rámců jednotlivých protokolů je takřka totožné, oproti stanici OutstationA. Ze stanice OutstationB je odesíláno více hodnot, proto jsou GOOSE i SMV publishery rozšířeny o pole, do kterých jsou hodnoty ukládány. Tím pádem i samotné nastavení komunikace je takřka totožné. Konkrétně tedy vytvoření rámce a jeho periodické odesílání v nekonečné smyčce.

Primární rozdílem v nastavení komunikace stanice OutstationB je v rozdělení procesu odesílání GOOSE a SMV. Na OutstationB má odesílaní každého protokolu svůj vlastní program. V hlavním programu se tedy vůbec nenachází. Pro spuštění simulace OutstationB nestačí pouze spustit kořenový soubor. Je nutné sestavit na dané zařízení také další dva obslužné podprogramy, které sestaví a odešlou rámce protokolu GOOSE a SMV. Pro současné spuštění a následné ukončení

všech potřebných programů byl vytvořen jednoduchý bashový skript. Pro správné fungování je nutné skript upravit do odpovídající podoby. To znamená primárně zapsat správnou cestu ke spouštěnému souboru. Spuštění simulace pomocí tohoto skriptu detailněji popisuje kapitola 3.6.

Na stanici OutstationB může být komunikace ovlivněna také simulovaným jističem nebo odpojovačem, které jsou popsány dále v této kapitole. Ve smyčce hlavního programu probíhá komunikace právě jističe a odpojovače, kteří využívají protokol GOOSE. Pro periodické zasílání GOOSE rámců a klasické odesílání SMV rámců je nutné vytvořit separátní proces, pro zajištění požadované funkčnosti.

Zde existuje opět více řešení tohoto problému. Například v původní, semestrální práci, na kterou tato práce navazuje, bylo taktéž nutné rozdělit odesílání GOOSE a SMV rámců. V semestrální práci pro to byla využita funkce *fork*. Ta svůj účel splnila a pro semestrální práci byla dostačující.

Použití forku v práci diplomové již dostačující není. Problém vychází ze samotné podstaty forku. Hlavní vlákno, neboli rodič, je rozděleno na dvě či více vláken, přičemž jeho potomci zkopírují veškeré proměnné z procesu rodiče a tyto proměnné následně nelze upravovat, což je pro funkci jističe a odpojovače nutné. Vzhledem k tomuto faktu je do simulátoru implementován zmiňovaný bashový skript pro spuštění dvou nezávislých programů.

Jedná se o zcela nové, a jak již bylo zmíněno, nezávislé programy. Je tedy nutné nadefinovat veškerou strukturu rámců znovu tak, jak je uvedeno v kapitole 3.2.1. Jak definuje začátek této kapitoly, OutstationB simuluje tři různá IED, tudíž je nutné generovat a následně odesílat tři různé hodnoty, proto je nutné vytvořit dostatečný počet individuálních ASDU jednotek pro SMV publishera a následné odesílání dat. Příklad je uveden na následující straně. Jedná se pouze o vytvoření ASDU jednotek pro napětí u jednotlivých IED. Aplikace na ostatní veličiny je totožná, samozřejmě za použití patřičných proměnných.

Výpis 3.24: Vytváření ASDU jednotek

```
SVPublisher_ASDU volt_1 = SVPublisher_addASDU
(svPublisher, "Uab_OutB", NULL, 1);
int uab = SVPublisher_ASDU_addFLOAT(volt_1);
int uab_ts = SVPublisher_ASDU_addTimestamp(volt_1);

SVPublisher_ASDU volt_2 = SVPublisher_addASDU
(svPublisher, "Ubc_OutB", NULL, 1);
int ubc = SVPublisher_ASDU_addFLOAT(volt_2);
int ubc_ts = SVPublisher_ASDU_addTimestamp(volt_2);

SVPublisher_ASDU volt_3 = SVPublisher_addASDU
(svPublisher, "Uca_OutB", NULL, 1);
int uca = SVPublisher_ASDU_addFLOAT(volt_3);
```

Nastavení MMS

Stejně jako je tomu u stanice OutstationA, v rámci MMS komunikace vystupuje OutstationB jako server. Tento server taktéž odpovídá na požadavky klientů. V aktuální topologii pro tuto se jedná o požadavky, které přicházejí z Koncentrátoru. Na této stanici není protokol MMS využíván k ovládání prvků, nýbrž má za úkol vyčítat požadované hodnoty na základě žádostí z Koncentrátoru.

Pro inicializaci této funkce je taktéž nutné inicializovat server, tak aby naslouchal na portu 102 příchozí MMS žádosti. Opět je spuštění serveru vybaveno podmínkou, která uživatele upozorní na případný neúspěšný pokus o spuštění.

Výpis 3.25: Spuštění MMS serveru

```
iedServer = IedServer_create(&iedModel);
IedServer_start(iedServer, 102);

if (!IedServer_isRunning(iedServer)) {
    printf("Starting␣server␣failed!␣Exit.\n");
    IedServer_destroy(iedServer);
    exit(-1);
}
```

Primární funkcí pro OutstationB je nastavení přímého čtení vstupu bez ukládání do bufferu. Na základě toho může koncentrátor vyčítat požadované hodnoty. Proto je implementována funkce *setUnbufferedInput*, která je při spuštění programu nastavena do stavu *true*. Výpis funkce v jazyce C++ je na následující straně.

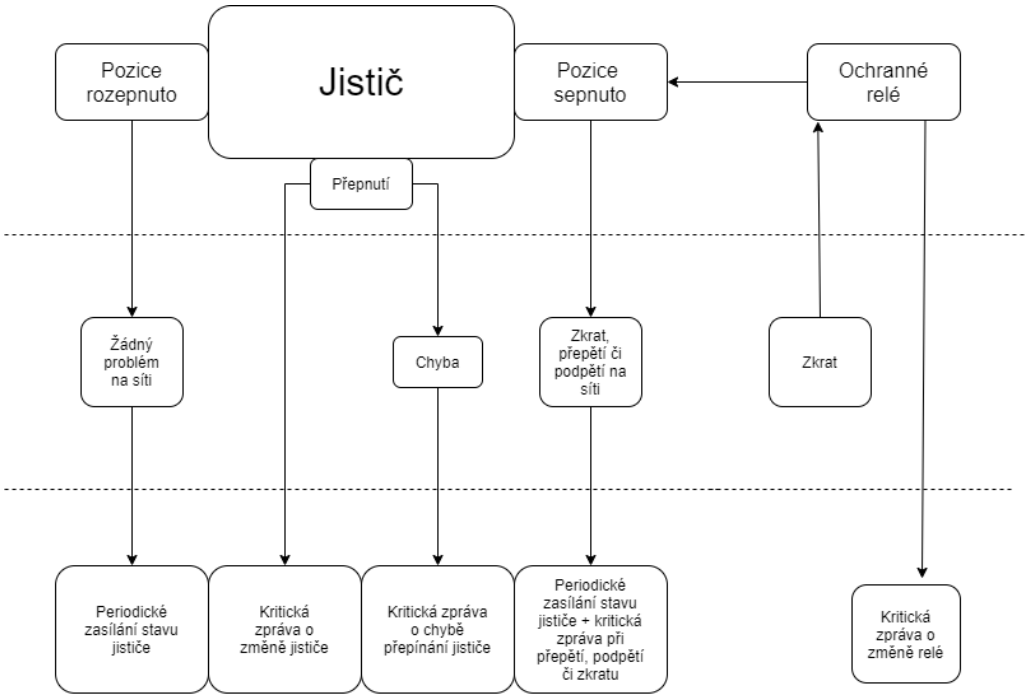
Výpis 3.26: Funkce pro přímé vyčítání hodnot

```
setUnbufferedInput(true);
...
void setUnbufferedInput(bool set) {
static const int STDIN = 0;
static struct termios term_backup;
static bool initialized = false;

if(!initialized) {
    tcgetattr(STDIN, &term_backup);
    initialized = true;
}

if(set) {
    struct termios term;
    tcgetattr(STDIN, &term);
    term.c_lflag &= ~ICANON;
    term.c_lflag &= ~ECHO;
    tcsetattr(STDIN, TCSANOW, &term);
    setbuf(stdin, NULL);
} else {
    tcsetattr(STDIN, TCSANOW, &term_backup);
}
}
```

Jistič a ochranné relé

Nedílnou součástí reálných rozvoden jsou jističe. Vytvoření jističe do simulátoru komunikace podle standardu IEC 61850 bylo jedním z primárních cílů této práce. Funkcionalita jističe je navržena podle blokového schématu vyobrazeném na obr. 3.4.

V simulátoru jistič komunikuje pomocí protokolu GOOSE. Potřeba využití právě tohoto protokolu je patrná již při bližším pohledu na blokové schéma. Znázornění jističe a jeho komunikace takřka definuje princip, využití a nasazení protokolu GOOSE do provozu. Z hlediska vývoje simulátoru byl se dal jistič definovat jako soustava na sebe navazujících podmínek, na jejichž základě jsou odesílány periodické či kritické GOOSE zprávy. Z důvodu zachování přehlednosti práce, jednotlivé podmínky implementované v jazyce C++ nejsou součástí jejího textu. Jsou součástí zdrojového kódu, který se nachází v příloze C.



Obr. 3.4: Blokové schéma jističe

Na první pohled se blokové schéma uvedené na obr. 3.4 může jevit poněkud složitě, avšak opak je pravdou. Primárně se jedná o dvě pozice, kterých může jistič nabýt. Je to buď pozice sepnuto nebo rozepnuto. Ve chvíli, kdy je jistič ve stavu rozepnuto, znamená to, že na síti není žádný problém a z jističe jsou odesílány periodické GOOSE zprávy o jeho stavu.

Druhou variantou je, že se jistič nachází v pozici sepnuto. V tuto chvíli je na síti problém, kdy mohlo dojít například ke zkratu. V takovém případě je zasílána periodická zpráva o stavu jističe, spolu s kritickou GOOSE zprávou informující o situaci která nastala.

Jak již vyplývá z textu, v případě kdy nastane na síti problém, například v podobě zkratu, dojde k přepnutí stavu jističe. K této situaci může taktéž dojít v případě uživatelského zásahu. Konkrétně tehdy, když je stisknuta řídící klávesa "j". Při této události jistič odesílá neperiodickou GOOSE zprávu, informující o stavu přepnutí. Tak, jako je tomu i v reálně rozvodně, může se stát, že k přepnutí nedojde. Pokud se tak stane, je odeslána neperiodická GOOSE zpráva informující o chybě přepnutí jističe.

Jistič je možné lokálně ovládat pomocí stisku klávesy. Při stisku klávesy "j"je jistič vypnut nebo zapnut, podle toho v jakém se aktuálně nachází stavu. Řídící

klávesu, která je zodpovědná za zapnutí vypnutí jističe je samozřejmě možné změnit ve zdrojovém kódu. Při spuštění simulace je do konzole vypsáno, pomocí kterých kláves je jistič možné ovládat. Zobrazení výpisu stanice OutstationB do lokálního CLI je popsán a vyobrazen v kapitole 3.6. Lokální klávesové ovládání jističe se nachází v hlavní smyčce programu a je zapsáno opět pomocí podmínky. Pokud uživatel stiskne klávesu "j"je přepnuta poloha jističe. Implementace v jazyce C++ je následovná:

Výpis 3.27: Funkce pro přepnutí polohy jističe

```
1 if(user_input()) {
2         input = getchar();
3         if(input == 'j') {
4             switchCircuitBreaker(true);
5          }
```

Jak již bylo popsáno v této kapitole, podobně jako u reálných rozvoden, jistič reaguje na zkrat. Pro co nejrealističtější výsledky je do simulátoru implementována funkce, která zkrat v dané topologii nasimuluje. Ve zdrojovém kódu je definována pravděpodobnost výskytu zkratu. Pravděpodobnost výskytu zkratu je možné samozřejmě měnit. nicméně při aktuálním nastavení je šance na výskyt zkratu a tím pádem i sepnutí ochranného relé 1:100. Šance na výskyt zkratu je implementována opět pomocí podmínky, která využívá matematické operace modulo.

Výpis 3.28: Podmínka definující výskyt zkratu

```
1 if(rand() % 100 + 1 == 1) {
2             switchProtectionRelay();
3         }
```

Jistič je taktéž přepínán na základě naměřeného přepětí či podpětí na monitorovaném IED. To zda v simulátoru dojde k překročení mezní hodnoty, která je již považována za podpětí či přepětí je závislé na generátoru naměřených hodnot, který byl popsán již dříve v této kapitole.

V případě, že k této situaci opravdu dojde, jsou nadále odesílány GOOSE zprávy signalizující stav jističe, nicméně v CLI Koncentrátoru již nejsou zobrazeny měřené hodnoty, jako tomu bylo doposud. Po opětovném přepnutí jističe je možné opět pozorovat naměřené hodnoty v CLI Koncentrátoru.

V CLI Koncentrátoru je možné taktéž sledovat aktuální stav jističe. Pomocí Boolean hodnoty je možné rozeznat, jestli je jistič ve stavu vypnuto či zapnuto. V případě, že dojde ke změně, dochází k odeslání neperiodické GOOSE zprávy. Tato

událost je taktéž patrná z CLI na Koncentrátoru.

Při přepínání jističe v reálné rozvodně může nastat situace, že k jeho přepnutí z nějakého důvodu nedojde. V takovém případě je nutné informovat o vzniklé události obsluhu zařízení. I tato možnost byla implementována do simulátoru, jenž popisuje tato práce. Stejně jako u simulace výskytu zkratu, je pomocí stejného principu se stejnou pravděpodobností generována šance pro neúspěšné přepnutí jističe. Vzhledem k tomu, že se jedná pouze o simulaci, jistič se podaří přepnout vždy. Jedná se pouze o přepsání hodnoty v proměnné. Při vygenerování náhodné chyby, která simuluje neúspěšné přepnutí jističe, je tedy pozice jističe přepnuta zpět do jeho původního stavu. Jistič je v původním stavu jako před přepnutím, tudíž byla chyba přepnutí úspěšně implementována. Následně je odeslána neperiodická GOOSE zpráva.

Odpojovač

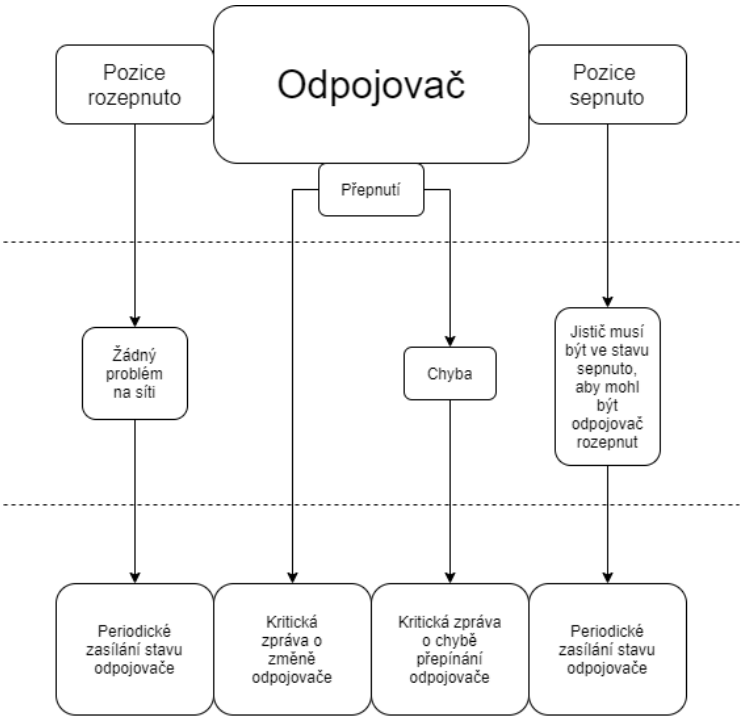
Kromě jističe byl taktéž do simulátoru implementován taktéž odpojovač, který se řídí podle blokového schématu zobrazeného na obr. 3.5. Z hlediska vývoje simulace je velmi podobný jističi, který je popsán v předchozí podkapitole. Taktéž se jedná o soustavu podmínek, které definují jaké GOOSE je odeslána. Stejně jako u jističe, je většina podmínek z důvodu vetší přehlednosti popsána pouze slovně. Jejich implementace v jazyce C++ se nachází ve zdrojových kódech, které se nachází v příloze C.

Nejedná se ovšem o dva totožné prvky. Odpojovač je dokonce na funkci jističe do určité míry závislý. Pro to, aby bylo možné odpojovač rozpojit, je nutné aby jistič byl ve stavu sepnuto. V případě, že je jistič ve stavu rozepnuto, lze odpojovač pouze sepnout. Funkčnost je zajištěna pomocí následující podmínky.

Výpis 3.29: Podmínka pro odpojovač

```
1 if(state_circuit_breaker == true) {
2     state_disconnector = !state_disconnector;
3     changed_state = true;
4 } else if(state_circuit_breaker == false &&
5      state_disconnector == false) {
6          state_disconnector = true;
7          changed_state = true;
8 }
```

Oproti jističi, nedochází k automatickému sepnutí odpojovače například při přepětí, podpětí či zkratu. Pro změnu stavu odpojovače je nutný zásah člověka. V rámci simulátoru je ovšem opět velmi podobná jističi. Ovládání probíhá pomocí stisku



Obr. 3.5: Blokové schéma odpojovače

řídící klávesy, která přepíná odpojovat. Řídící klávesou pro ovládání odpojovače je klávesa "o". Z hlediska zápisu se jedná o naprosto totožnou podmínku podmínku jako u jističe, pouze s tím rozdílem, že klávesa "j"je nahrazena klávesou "o". Pro lepší orientaci uživatele je řídící klávesa taktéž zobrazena v CLI při spuštění simulace, tak jako je tomu v případě jističe. Výpis z CLI po spuštění simulace je zobrazen v kapitole 3.6.

Tak jako u jističe, odpojovač může nabývat dvou stavů, mezi kterými je možné přepínat. V případě, že se odpojovač nachází v pozici sepnuto, není na sítí žádný problém a odchází periodická GOOSE zpráva indikující stav odpojovače. Při pozici rozepnuto je závislá na pozici jističe. Je-li jistič ve stavu sepnuto, odpojovač může být rozepnut. V opačném případě lze odpojovač pouze sepnout. Výsledkem obou variant je periodické zasílání GOOSE zpráv informující o stavu odpojovače.

Jediný případ, kdy u odpojovače dochází k odesílaní neperiodických GOOSE zpráv je při přepnutí. V případě, že přepnutí proběhlo úspěšně, je odeslána neperiodická GOOSE zpráva informující o změně stavu odpojovače. V opačném případě, je odeslána taktéž neperiodická GOOSE zpráva, která ovšem informuje o chybě přepínání odpojovače.

Aby k těmto chybám docházelo, je do odpojovače implementován generátor chyb přepnutí. Jeho princip, funkcionalita i nastavení jsou totožné s generátorem, který se nachází u jističe a je popsán v předchozí podkapitole.

3.2.3 Koncentrátor

Koncentrátor je posledním ze zařízení Raspberry Pi, které v rámci této práce simulují komunikaci podle standardu IEC 61850. V rámci topologie, vyobrazené na obr. 3.1 toto zařízení simuluje SAS neboli Substation Automation System. Primárním úkolem tohoto zařízení je sbírat odeslaná data z podřízených stanic a zobrazovat je do improvizovaného HMI. Samozřejmostí je taktéž funkce logování protokolů GOOSE a SMV pro případné porovnání či troubleshooting. Formát logů je obdobný jako u stanice OutstationA a OutstationB.

Mimo jiné, tak jak je popsáno v kapitolách 3.2.1 a 3.2.2, slouží Koncentrátor také jako ovládací prvek dalších stanic. Pomocí protokolu MMS jsou v předdefinovaném časovém intervalu vyčítány aktuální naměřené hodnoty ze stanice OutstationB. Za pomoci stejného protokolu, tedy MMS, je na stanici OutstationA ovládáno generování měřených hodnot. Z hlediska MMS komunikace koncentrátor vystupuje jako klient. Klientské žádosti jsou odesílány v pravidelných intervalech, které jsou definovatelné v konfiguračním souboru.

Z hlediska protokolů GOOSE a SMV vystupuje Koncentrátor jako subscriber, neboli odběratel. Z toho plyne, že naměřené hodnoty jsou na tomto zařízení přijímány a dále zpracovávány. Z tohoto důvodu tedy není nutné implementovat generátor měřených hodnot.

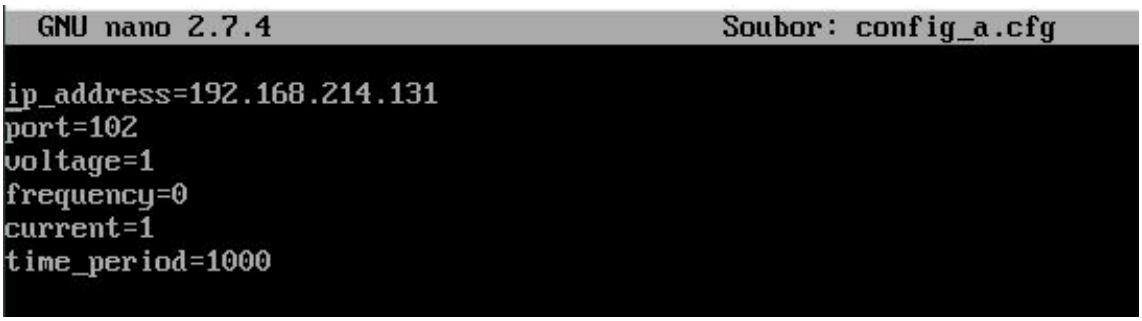
Konfigurační soubor

Stejně jako u stanice OutstationB i Koncentrátor disponuje vstupním konfiguračním souborem. Jeho význam je ovšem rozdílný. Jak již bylo zmíněno, Koncentrátor primárně sbírá data od podřízených stanic. Na stanici OutstationB jsou data vytvářena a následně odesílána, právě na základě zmiňovaného konfiguračního souboru, kde jsou nastaveny rozsahy, v jakých bude simulátor generovat hodnoty. Vzhledem k tomu, že Koncentrátor data pouze přijímá, obdobný konfigurační soubor by zde byl nepotřebný.

Vzhledem k účelu konfiguračních textových souborů na Koncentrátoru, jsou v kořenové složce uloženy hned dva soubory. Konkrétně se jedná konfigurační soubory

pro jednotlivé stanice. Díky tomu je možné ovládat centrálně více stanic z jednoho místa. V dané topologii tedy OutstationA a OutstationB.

V konfiguračním souboru má uživatel možnost definovat IP adresu zařízení se kterým chce komunikovat, port na kterém bude komunikace probíhat, nastavení protokolu MMS pro jednotlivé veličiny a časovou periodu, po které bude odeslán MMS žádost. Ukázka textového konfiguračního souboru pro stanici OutstationA na Koncentrátoru je zobrazena na obr. 3.6.



Obr. 3.6: Konfigurační soubor koncentrátoru

Naopak od ostatních stanice, Koncentrátor data přijímá, tudíž v konfiguračním souboru je možné zadefinovat IP adresu a port, na kterém bude zařízení komunikovat. V topologii, která využívá pro adresaci svých prvků dynamické přidělování adres může dojít k vypršení platnosti zápůjčky a přidělení nové IP adresy. V takovém případě tedy není pro úpravu nutné zasahovat do zdrojového kódu, nýbrž stačí drobná úprava konfiguračního souboru.

Tak jako všechny stanice v topologii i Koncentrátor využívá operační systém Raspbian, který pracuje pouze v CLI. Úprava konfiguračního souboru tedy musí probíhat taktéž z CLI. Pro uživatele je tedy nutná znalost práce se soubory v CLI operačních programů na bázi Linuxu. Po úpravě konfiguračního souboru není nutné zařízení restartovat nebo program znovu sestavit. Nová konfigurace je vždy znovu načtena při opětovném spuštění programu. Principiální implementace v jazyce C++ je naprosto totožná jako u stanice OutstationB. Rozdílné jsou pouze proměnné. Ukázka ze zdrojového kódu je tedy v kapitole 3.2.2.

Konfigurační textový soubor je na Koncentrátoru primárně využit pro ovládání protokolu MMS. Na základě Boolean hodnoty je zadefinováno, jestli má být jeho funkce využita či nikoli. Na stanici OutstationA je tímto způsobem buď sepnut, či vypnut generátor pro požadovanou veličinu tak, jak je popsáno v kapitole 3.2.1. V případě, že je Boolean hodnota 1, generování je sepnuto. Oproti tomu, pokud je Boolean hodnota 0, generování je vypnuto.

Poslední položkou konfiguračního souboru je časová perioda. Jedná se časový

interval, po kterém je cyklicky odesílána klientská MMS žádost směrem k serveru. I přesto, že je koncentrátor je topologicky nadřazený stanicím OutstationA a OutstationB, z hlediska MMS komunikace vystupuje jako klient.

Kromě IP adresy, je konfigurační soubor pro OutstationB na Koncentrátoru je naprosto totožný vůči konfiguračnímu souboru pro OutstationA. Dokonce i význam jednotlivých položek je totožný. V první řadě je nastavena IP adresa pro komunikace se stanicí OutstationB, spolu s portem.

Tak, jako je tomu u předchozí stanice, i zde další položky slouží k ovládání protokolu MMS. Nastavení protokolu MMS je ovšem odlišné. Na stanici OutstationB je protokol MMS využíván pro vyčítání měřených hodnot v určité časové periodě. K definici délky časové periody slouží poslední pole. Veškeré časové údaje uváděné v konfiguračních souborech jsou uváděny v milisekundách.

Nastavení komunikace GOOSE a SMV

Jak již bylo zmíněno, z hlediska komunikace protokolů GOOSE a SMV vystupuje Koncentrátor jako subscriber, neboli odběratel. Odebírá tedy data z okolních stanic. Pro správné fungování je nutné v první řadě zadefinovat, na jakém portu bude odběratel naslouchat. V této práci se jedná o rozhraní eth1, nicméně to se může lišit v závislosti na konkrétním zařízení, na kterém je simulace spuštěna.

Pro komunikaci je nutné samotného odběratele sestavit v hlavní smyčce. K tomu je zapotřebí zvolit dataset, k jehož odběru se přihlásí nově vytvořená instance objektu GooseReciever. Následně je nastaveno hexadecimální číslo, které slouží jako identifikační číslo GOOSE aplikace. Poté je inicializována proměnná, která slouží jako uživatelská vstupní proměnná pro funkci gooseListener. Tato funkce je volána více odběrateli, tudíž je žádoucí, aby pomocí této proměnné bylo definováno, ze kterého konkrétního odběratele data přichází. Posléze je k jednotlivým odběratelům přiřazen jejich vydavatel, na základě čehož může odběratel začít naslouchat GOOSE zprávám. Formální zápis je, s úpravou proměnných, pro každou dvojici vydavatel – odběratel totožný, a proto následující výpis obsahuje pouze jednu z nich. Kompletní kód se nachází v příloze C, kde je popsán a označen. Postup pro protokol SMV je totožný.

Výpis 3.30: Konfigurace GOOSE odběratele

```
GooseReceiver g_rec = GooseReceiver_create();
GooseReceiver_setInterfaceId(g_rec, interface);
GooseSubscriber g_sub_0 = GooseSubscriber_create
("TEST/LLN0$OUTB", NULL);
...
GooseSubscriber_setAppId(g_sub_0, 0x1000);
...
int g_sub_0_id = 1;
...
GooseSubscriber_setListener(g_sub_0, gooseListener,
&g_sub_0_id);
...
GooseReceiver_addSubscriber(g_rec, g_sub_0);
...
GooseReceiver_start(g_rec);
```

Jak protokol GOOSE, tak i protokol SMV sestavují svého odběratele v hlavní smyčce programu. Pro práci s přijatými daty využívají oba protokoly callback funkci, která získává naměřené hodnoty z polí přijatých rámců. V rámci těchto callback funkcí jsou přijaté hodnoty také ukládány do logů. Tak jako na ostatních stanicích i na Koncentrátoru jsou logy ukládány do textového souboru v kořenovém adresáři zařízení. Podrobný popis logování se nachází v kapitole 3.2.1.

Součástí callback funkce pro protokol SMV je taktéž výpočet výkonu. Tak, jako je tomu v reálné rozvodně, na základě naměřených veličin je vypočten činný a jalový výkon. Výpočet vychází z naměřených hodnot, čili z hodnot, které jsou uloženy v ASDU ethernetového rámce s protokolem SMV. Pomocí identifikátoru a jednoduché podmínky je determinováno, jestli jedná o naměřené hodnoty ze stanice OutstationA nebo OutstationB. Výpočet činného a jalového výkonu spolu s uložením do proměnných je zobrazen na výpisu níže. Kompletní kód se nachází v příloze C.

Výpis 3.31: Výpočet výkonu

```
if(sub_id == 2) { // OutA
    out_data->pa = out_data->uab * out_data->ia;
    out_data->qa = out_data->pa + random;
}
if(sub_id == 1) { // OutB
    out_data->pb = out_data->ubc * out_data->ib;
    out_data->qb = out_data->pb + random;
```

```
8 }
```

HMI

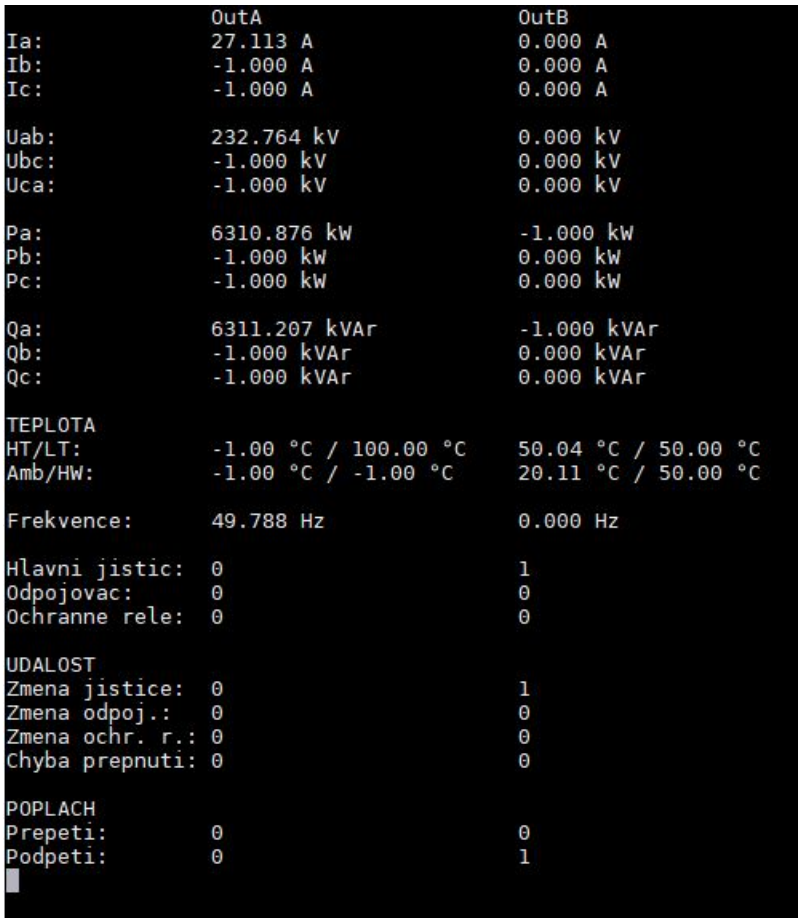
HMI neboli Human Machine Interface slouží k zobrazení naměřených hodnot v rozvodně. V rámci této práce se jedná spíše o improvizované HMI. Přijaté hodnoty jsou vypisovány do příkazové řádky. Celé rozhraní je tedy složeno pouze z textového označení jednotlivých veličin a vypsaných proměnných, které obsahují naměřené hodnoty odeslané pomocí protokolu GOOSE a SMV. Hodnoty jsou neustále aktualizovány, tudíž výpis je vždy vypsán kompletně znovu.

Při vypisování hodnot do příkazové řádky dochází k tomu, že jsou jednotlivé příkazy shlukovány pod sebe a působí velmi nepřehledně. Simulátor popisovaný v této práci zobrazuje relativně mnoho veličin a při standardním výpisu, tedy při každé změně doplnění původního výpisu o nové řádky, docházelo ke kumulaci dat, který byla velmi těžko čitelná. Pro lepší přehlednost je vždy původní výpis z příkazové řádky smazán a nahrazen novým, aktuálním. Pro uživatele je to ovšem takřka nepozorovatelné. Samotný výpis do příkazové řádky z Koncentrátoru je možné vidět na obr 3.7.

Na obr. 3.7 je možné sledovat výpis příkazové řádky Koncentrátoru, po spuštěné kompletní simulace. Na jednotlivých řádcích výpisu jsou uvedeny měřené veličiny a ve dvou sloupcích jsou vypsány naměřené hodnoty pro jednotlivé stanice. Některá pole jsou redundantní, jelikož určité funkce jsou implementovány pouze na jednu ze stanic. Ve výpisu se nachází kvůli formátování a větší přehlednosti. Zobrazený výpis příkazové řádky simuluje zmiňované HMI. Je možné sledovat, že koresponduje s funkcemi simulátoru, které byly popsány dříve v této práci.

Již na první pohled je patrné, že u stanice OutstationB nedochází k zobrazení žádných hodnot. Důvod je patrný z dalších řádků. V HMI je indikována změna jističe. U stavu jističe je možné si povšimnout, že se jistič nachází ve stavu sepnuto. To znamená, že došlo na síti k problému, obvod byl přerušen, tudíž není možné hodnoty měřit.

Boolean hodnota indikující změnu jednoho z prvků, je zobrazena vždy po dobu deseti sekund. Posléze je přepnuta zpět do stavu 0. Pokud dojde k jakékoliv změně, vždy je dočasně aktualizována Boolean hodnota v kategorii UDALOST. Spolu s tím se změní i Boolean hodnota u sledovaného prvku. Touto funkcí je vybavenou pouze stanice OutstationB.



Obr. 3.7: Výpis do příkazové řádky v koncentrátoru

Další položka, která je měřena pouze na stanici OutstationB je teplota. HMI zobrazuje v druhém řádku kategorie TEPLOTA aktuální naměřené ambientní a hardwarové teploty. Ambientní teplota simuluje teplotu v okolí, zatímco hardwarová teplota simuluje přímo teplotu monitorovaného prvku. Na prvním řádku je možné sledovat, jakým způsobem teplota zařízení oscilovala v rámci celého měření. Vlevo je zobrazena nejvyšší a vpravo naopak nejnižší naměřena hardwarová teplota. Zkratku HT a LT vycházejí z anglického označení high a low tempreature.

V kategorii POPLACH je možné sledovat, jestli se na monitorovaných prvcích objevuje přepětí či podpětí. Ve chvíli výskytu je problematická veličina opět indikována Boolean hodnotou 1. Tyto upozornění vznikají na základě kritických GOOSE zpráv, jejichž princip vzniku je popsán v kapitole 3.2.1.

Zbylé řádky slouží zobrazení naměřených hodnot, roztříděných podle jednotlivých veličin. Zobrazené hodnoty jsou vyčítány z rámců protokolu SMV. Jejich hodnota koresponduje s nastavením, které je definováno buď přímo ve zdrojovém

kódu, tak jako je tomu u stanice OutstationA. Nebo rozsah hodnot vymezen v konfiguračním textovém souboru, ze kterého následně hlavní simulační program potřebné hodnoty vyčítá, jako je tomu u stanice OutstationB.

3.3 Konfigurace SNTP

Pro komunikaci dle standardu IEC 61850 je nutné zajištění časové synchronizace. V reálných rozvodnách je pro časovou synchronizaci využíván protokol SNTP a stejný protokol využívá i simulátor popisovaný v této práci.

Komunikace pomocí SNTP protokolu je typu klient – server. Jedná se o vertikální komunikaci a je tedy nutné, aby jedno ze zařízení bylo nadřazené ostatním a na základě jejich žádostí distribuovalo požadované informace. V rámci topologie této práce vystupuje Koncentrátor jako server a stanice OutstationA a OutstationB jako klienti. Znamená to, že stanice OutstationA a OutstationB synchronizují své vnitřní hodiny podle Koncentrátoru.

Protokol SNTP nepatří do rodiny protokolů standardu IEC 61850. Vzhledem k tomu není nutná tvorba SNTP serveru a klienta přímo v programu, nýbrž je možné využít jiné postupy, například konfiguraci balíčků, které jsou k dispozici v operačních systémech na bázi Linuxu. Pro získání takových balíčků je nutné připojení k internetu.

Pro účely této práce byl zvolen rozšiřující balíček Chrony. Balíček Chrony je velmi univerzální implementací SNTP protokolu. Nabízí možnost synchronizace s nadřazeným časovým serverem, ale zároveň může sám vystupovat jako SNTP server a distribuovat čas na žádosti klientů. Synchronizace Chrony se řádově pohybuje v milisekundách. Pro komunikaci je využíván UDP port vyhrazený pro NTP, čili port číslo 123.

Balíček je Chrony je nainstalován přímo v zařízení Raspberry Pi. Není tedy přímou součástí programu simulátoru. Jeho instalace a konfigurace tedy neprobíhá z konfigurační stanice a vývojového prostředí NetBeans, tak jako tomu je u samotného simulátoru. Pro instalaci a konfiguraci je nutné se připojit přímo do Raspberry Pi pomocí zabezpečeného protokolu SSH na TCP portu číslo 22.

Pro navázání tohoto zabezpečeného připojení je nutný klient, který tento typ připojení podporuje. Pro účely této práce byl zvolen klient MobaXterm, který funguje v operačním systému Windows a je schopen navazovat SSH spojení se

stanicemi, se kterými se nachází ve stejné síti. Z hlediska hardwaru se klient MobaXterm nachází na stejném zařízení jako konfigurační stanice s NetBeans. Jak již bylo zmíněno, konfigurační stanice je virtuální zařízení. Z tohoto důvody pracuje MobaXterm a NetBeans s jinými operačními systémy, i přesto, že se fyzicky nachází na jednom hardwaru.

Po úspěšném navázání SSH spojení je možné Raspberry Pi přímo konfigurovat tak, jako by uživatel byl připojen lokálně do zařízení. Konfigurační příkazy jsou tedy pro operační systém Raspbian. Pro úspěšnou instalaci je nutné veškeré úpravy provádět z účtu, který má na zařízení nastavena administrátorská práva. Případně je možné využít také příkazu Sudo, který slouží k vykonání operace s oprávněními jiného uživatele, jímž je obvykle root.

Získání balíčku Chrony probíhá pomocí balíčkovacího systému apt-get, který usnadňuje správu softwaru na operačních systémech na bázi Unixu. Jedním z těchto operačních souborů je právě Raspbian. Systém slouží například k automatizaci vyhledávání, konfigurace a instalaci softwarových balíčků, buďto z binárních balíčků anebo při kompilaci ze zdrojových kódů. Příkaz pro spuštění instalace balíčku Chrony je následovný.

```
apt-get install Chrony -y
```

Po spuštění tohoto příkazu je balíček Chrony automaticky stažen a nainstalován do daného zařízení. V zařízení je vytvořen i konfigurační textový soubor, který definuje chování Chrony v systému. Pomocí tohoto textového souboru je Chrony nakonfigurováno.

Baliček Chrony je nainstalován, nicméně jednotlivé konfigurace jsou mírně odlišné. Tyto rozdíly vznikají na základě topologie této práce. Koncentrátor vystupuje jako časový server a pro přesný čas je tudíž nutné získat tuto informaci z internetu. Je nutné nastavit časový server, ze kterého Chrony získá aktuální čas. Pro účely této práce byl zvolen časový server, který se nachází na veřejné IP adrese 185.189.4.68.

Tato časové synchronizace funguje pouze za předpokladu, že je Raspberry Pi připojeno k internetu. V opačném případě časová synchronizace pomocí SNTP nadále funguje, ale pouze v rámci topologie. Časové značky, které se nachází například v logu jednotlivých protokolů nemusí přímo korespondovat s reálným časem.

Dalším z prvků, které je nutné nakonfigurovat pro správné fungování SNTP

protokolu v dané topologii je nastavení rozsahů IP adres, ze kterých mohou přicházet klientské žádosti a na které bude server odpovídat. Znovu se jedná pouze o konfiguraci Koncentrátoru. Tato konfigurace opět probíhá v konfiguračním textovém souboru nacházejícím se přímo na úložišti Raspberry Pi. Do konfiguračního textového souboru jsou vepsány tyto příkazy:

Výpis 3.32: Konfigurace SNTP serveru

```
1 ....
2 pool 185.189.4.68 iburst
3 ....
4 allow 192.168.253.0/24
```

Celý konfigurační soubor pro konfiguraci časového serveru na Koncentrátoru, je možné vidět v příloze B.1.

Konfigurace z hlediska klientů je obdobná, pouze s tím rozdílem, že namísto časového serveru umístěného v internetu, je jako časový server využit Koncentrátor. Pro dotazy SNTP dotazy je tedy definována IP adresa Koncentrátoru, čímž je zajištěna časová synchronizace v celé topologii.

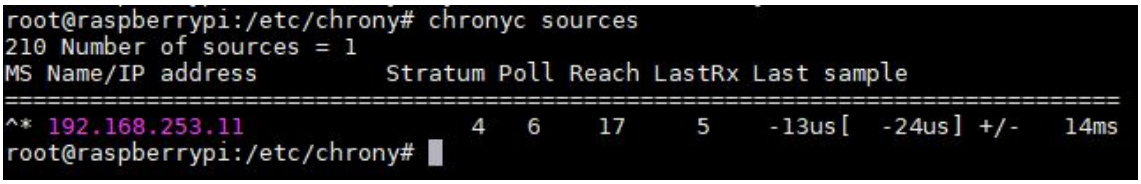
Po jakékoliv úpravě konfiguračního souboru chrony.conf je nutné balíček v rámci Raspberry Pi restartovat. V opačném případě provedené změny nebudou aktivní. Není nutné restartovat celé zařízení, stačí restartovat pouze balíček Chrony pomocí příkazu:

```
systemctl restart chrony.service
```

Po restartování je možné ověřit provedené změny a funkčnost časové synchronizace pomocí příkazu

```
chronyc sources
```

Na základě tohoto příkazu Chrony vypíše z jaké IP adresy odebírá časové informace, jestli je časový server dostupný a případnou odchylku o kterou bylo nutné vnitřní hodiny zařízení sesynchronizovat. Na obr. 3.8 je uveden příklad časové synchronizace ze stanice OutstationA.



Obr. 3.8: Zdroj pro časovou synchronizaci stanice OutstationA

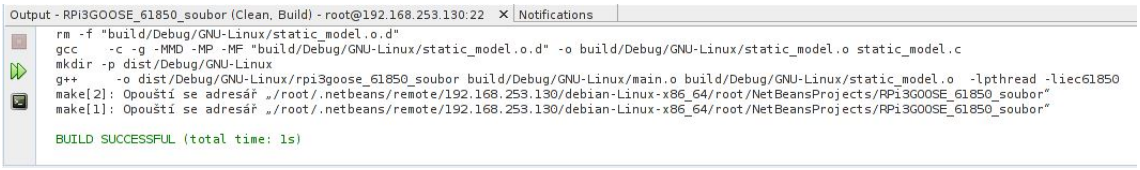
3.4 Přenos aplikace do zařízení

Jak již bylo uvedeno, zařízení jsou vytvářena na vzdálené konfigurační stanici. Program je následně přenášen po síti do jednotlivých zařízení. Každé zařízení, má svoji vlastní IP adresu. IP adresy jednotlivých zařízení vycházejí z adresního plánu, který byl pro účely této práce vytvořen a je blíže popsán v kapitole 3.5.1.

Prostředí NetBeans nabízí vcelku jednoduché uživatelské rozhraní, přes který je uživatel schopen, mimo samotné programování, také nastavit vše co je potřebné pro přenesení konfiguračního souboru do požadovaného zařízení. Spojení je navázáno pomocí SSH protokolu na TCP portu číslo 22 a soubory jsou přenášeny pomocí SFTP protokolu.

Pro nastavení IP adresy je po stisku pravého tlačítka myši na daném souboru zvolena možnost *Set Build Host -> Manage Hosts -> Add*. Nyní jsou v konfiguračním průvodci vyplněny potřebné údaje. Zde se jedná především o nastavení správné IP adresy, SSH portu a zvolení přenosu souboru přes SFTP. Pro připojení a ukládání souborů do zařízení je nutné před samotnou IP adresu ještě zvolit kořenový adresář zařízení. V dané topologii se bude jednat například o adresu Koncentrátoru *pi@192.168.253.11*. Při správném zadání jména a hesla je cesta uložena do správce.

Nyní je možné nastavit správnou cestu pro upload programu do zařízení. Cesta je opět přes pravé tlačítko myši, *Set Build Host -> Manage Hosts*, zde je vybrána relace, která byla před chvílí nastavena. Samotné sestavení programu do zařízení, ať už virtuálního nebo fyzického, probíhá pomoc *pravé tlačítko -> Clean and Build*. Tento příkaz spustí nahrávání konfiguračního souboru do zvoleného zařízení. V případě, že je vše nastaveno správně a požadované zařízení je dostupné, je v konzoli NetBeans zobrazeno *Build Sucessfull*. Příklad úspěšného vytvoření a přenosu souboru je možno vidět na obr. 3.9. Délka sestavování programu je různá, nicméně se pohybuje kolem 1 sekundy, jak je mimo jiné vidět také na obrázku.



Obr. 3.9: Úspěšné sestavení programu

3.5 Zapojení pracoviště

V praktické části této práce je realizováno pracoviště, které simuluje komunikaci podle standardu IEC 61850 v reálné rozvodně. Simulace probíhá na zařízení Raspberry Pi, které mezi sebou komunikují práce podle zmiňovaného standardu. Grafické zobrazení pracoviště je dále zobrazeno v kapitole 3.1. Pro zajištění komunikace je nutné jednotlivá zařízení mezi sebou propojit. Zařízení Raspberry Pi je vybaveno jednou síťovou kartou, což je pro potřeby práce nedostačující. Zařízení by bylo možné rozšířit například o doplňující externí síťovou kartu. V takovém případě by zařízení již mělo dostatečný počet portů pro účely této práce, nicméně by se neblížilo příliš reálnému zapojení.

V reálných rozvodnách bývají jednotlivá zařízení zapojena do přepínačů. Z tohoto důvody byl v této práci zvolen stejný postup. Samotná komunikace IEC 61850 probíhá na druhé vrstvě referenčního modelu ISO/OSI, avšak pro synchronizaci SNTP serveru je nutné připojení internetu. Na základě těchto kritérií byl zvolen přepínač, který mimo přepínání, je schopen i směrování. Kombinuje tedy funkci plnohodnotného směrovače, a zároveň pracuje jako tak zvaný Layer 3 switch. V daném zapojení slouží taktéž jako brána zmiňované sítě. Toto zařízení je v práci označován jako Přepínač 1.

3.5.1 Adresace

Pro účely této práce byl zvolen adresní rozsah 192.168.253.0/24. Adresní rozsah byl zvolen náhodně, jako libovolná síť, která využívá adresaci z tak zvaných privátních rozsahů. Tyto rozsahy byly vyčleněny pro využití v interních sítích. Standard IEC 61850 nedefinuje žádný specifický adresní rozsah. Je tedy čistě na správci dané sítě, ve které zařízení komunikují, jaký adresní rozsah pro zařízení zvolí.

Jak již bylo zmíněno, pro síť pro účely této práce má adresu 192.168.253.0/24. Stejně jako adresa sítě, ani její maska není definována standardem IEC 61850. Použitá maska /24 neboli 255.255.255.0, nabízí 254 IP adres, které mohou být v síti použity. 254 adres je pro realizaci pracoviště naprosto redundantní, ovšem

vzhledem k tomu, že se jedná o privátní síť, která netrpí nedostatkem IP adres, je možné si takovou redundanci dovolit. Síťová maska /24 byla tedy zvolena pouze z důvodu větší přehlednosti.

DHCP - Dynamic Host Configuration Protocol

Pro přidělení IP adres jednotlivým zařízením z daného rozsahu byl použit protokol DHCP neboli Dynamic Host Configuration Protocol. Jedná se o protokol, který v počítačových sítích slouží k automatické síťové konfiguraci prvků, které jsou do sítě připojeny a podporují tento protokol. Tento protokol stanici sdělí základní údaje o síti, které daná stanice potřebuje pro komunikaci. Konkrétně se jedná o IP adresu, kterou může daná stanice použít, IP adresu výchozí brány, masku sítě a adresu DNS serveru. Vzhledem k tomu, že zařízení Raspberry Pi protokol DHCP podporují, byl využit pro jejich konfiguraci.

DHCP komunikace je typu klient – server. V uvedené topologii Raspberry Pi vystupuje jako klient, tudíž je nutná konfigurace serveru. Přepínač 1 nabízí poměrně jednoduchou a uživatelsky přívětivou konfiguraci DHCP serveru. Výhodou je možnost využití GUI neboli Graphic User Interface. V českém překladu to znamená, že zařízení nabízí grafické uživatelské prostředí pro konfiguraci. Není tedy nutné, aby uživatel zařízení konfiguroval z CLI, jako je tomu u jiných výrobců.

Pro konfiguraci DHCP serveru byl na Přepínači 1 vytvořen již zmiňovaný adresní rozsah 192.168.253.0/24, přičemž Přepínač 1, vystupující jako brána, má přiřazenu IP adresu 192.168.253.1. Toto je adresa, která je distribuována pomocí DHCP protokolu do připojených Raspberry Pi. Při samotné konfiguraci DHCP serveru posléze stačí pouze potvrdit jednotlivé volby, které Přepínač 1 nabízí. Jedná se například o adresní rozsah, který má distribuovat připojeným stanicím, dobu zápůjčky přiřazených adres a také adresu DNS serveru. Na Přepínači 1 je jako DNS server použit server společnosti Google na adrese 8.8.8.8. Další z položek, kterou je nutné nakonfigurovat pro správné fungování DHCP serveru je určení rozhraní, na kterém má být DHCP server spuštěn. U Přepínače 1 bylo zvoleno rozhraní bridge1.

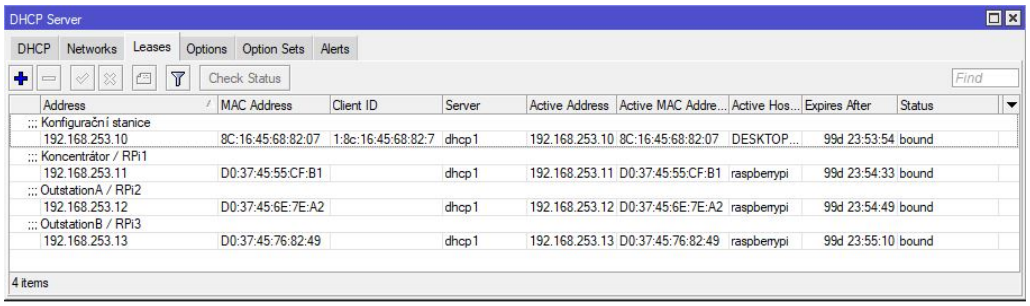
Nastavený DHCP server přiřazuje jednotlivým zařízením adresy postupně, vždy po dobu nastavené doby zápůjčky. Po uplnyutí této doby je vždy vyžádána nová adresa. Existuje několik možností, které je možné implementovat, na jejichž základě bude zajištěno správné fungování celé sítě. Jak již bylo zmíněno v kapitole 3.2.3, simulátor obsahuje konfigurační soubor, ve kterém je možné nastavit IP adresy jednotlivých zařízení. Existuje tedy vždy možnost kontroly přiřazení jed-

notlivých adres na DHCP serveru a následná úprava konfiguračního souboru. Pro účely této práce byl ovšem zvolený adresní plán, který je vyobrazen v tabulce 3.1.

Tab. 3.1: Adresní plán.

Zařízení	IP Adresa
Konfigurační stanice	192.168.253.10
Koncentrátor	192.168.253.11
OutstationA	192.168.253.12
OutstationB	192.168.253.13

DHCP je dynamický protokol, nicméně Přepínač 1 nabízí funkci, díky které je z dynamicky přidělených adres možné udělat tak zvanou statickou zápůjčku. Jedná se statické přiřazení zvolené IP adresy konkrétnímu zařízení, na základě jeho MAC adresy. Po připojení zmiňovaného zařízení do sítě, mu bude vždy přiřazena zvolená IP adresa. Ve chvíli, kdy je v DHCP servery vytvořen statický záznam, rozhraní Přepínače 1 umožňuje k tomuto záznamu přidat komentář, pro lepší přehlednost. Příklad je vyobrazen na obr. 3.10.



Obr. 3.10: DHCP server.

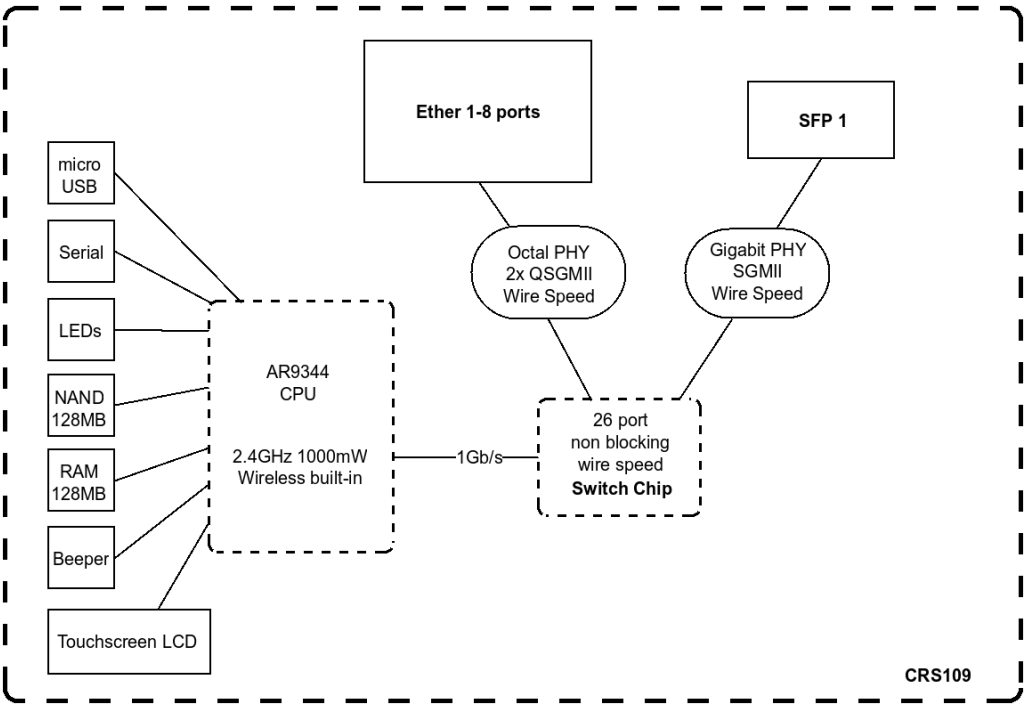
Síťový most

Pro úspěšnou komunikaci mezi jednotlivými stanicemi je nutné je vzájemně propojit. Jak již bylo uvedeno, fyzické propojení by sice možné bylo, ale neodpovídalo by příliš reálnému zapojení v rozvodně. Pro propojení je využit Přepínač 1. Komunikace standardu IEC 61850 probíhá na druhé vrstvě referenčního modelu ISO/OSI. Je tedy nutné vytvořit spojení na právě této vrstvě.

Samotná IP adresace jednotlivých stanic zaručuje pouze propojení a komunikaci na třetí vrstvě, čili je možné zjistit, jestli je daná stanice aktivní, například pomocí ICMP zprávy. Pro realizaci komunikace dle standardu IEC 61850 je nutné jednotlivé stanice propojit také na druhé vrstvě. Toho je docíleno pomocí vytvoření tak zvaného síťového mostu, který je na Přepínači 1 označen jako "bridge1".

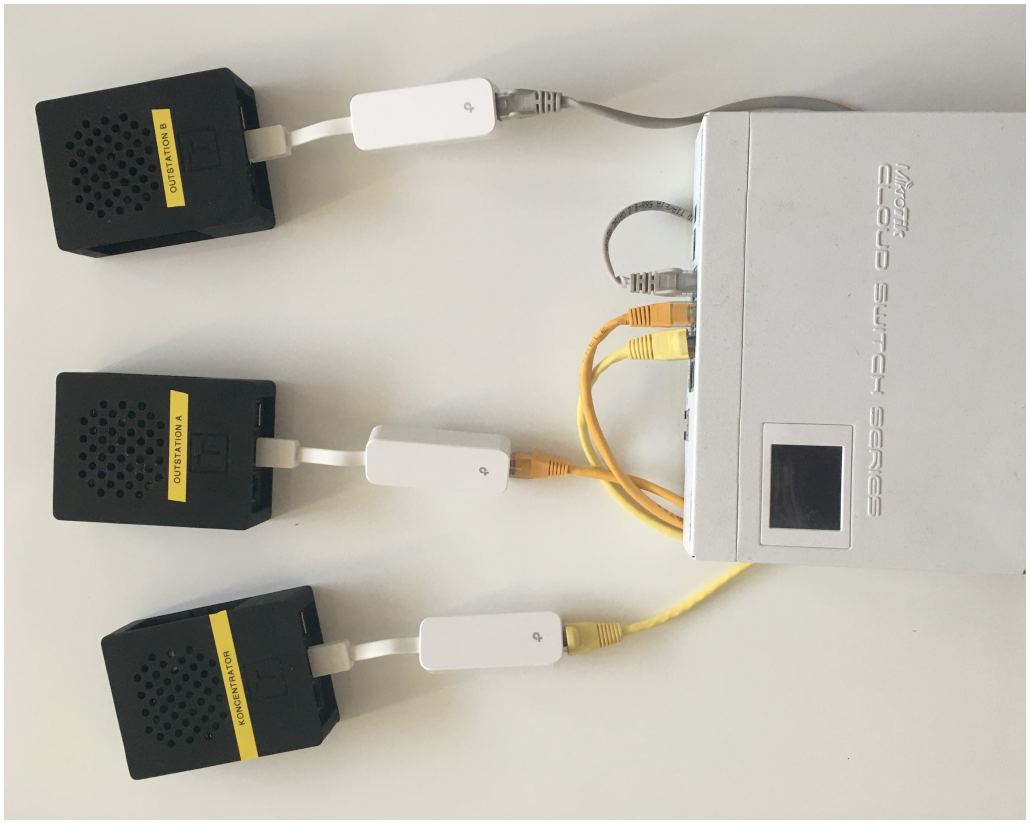
Vytvoření síťového mostu zajišťuje propojeni stanic právě na linkové vrstvě. Samotný síťový most by se dal definovat jako virtuální náhrada fyzických propojovacích kabelů. Znamená to, že není rozdíl v tom, jestli jsou stanice zapojeny pomocí fyzického média, či virtuálního síťového mostu. Mimo jiné, vytvoření virtuálního mostu znamená vytvoření rozhraní, na které je možné umístit DHCP server. Vzhledem k tomu, že spojuje všechny zainteresované stanice, stačí pro kompletní adresaci pouze jeden DHCP server.

Zapojení jednotlivých Raspberry do Přepínače 1 může být taktéž libovolné, nicméně musí korespondovat s konfigurací síťového mostu. Některé směrovače či přepínače, jsou vybaveny více jádry či přepínacími čipy. V takových případech se často administrátor uchyluje k zapojení svých zařízení do portů tak, aby byla zátěž co nejrovnoměrněji rozložena na používaný hardware. V případě Přepínače 1 jsou všechny porty připojeny pouze do jednoho přepínacího čipu, tak jak zobrazuje blokové schéma na obr. 3.11.



Obr. 3.11: Blokové schéma přepínače 1. Převzato z [20].

Výsledné reálné zapojení zařízení Raspberry Pi a Přepínače 1 pro simulaci komunikace dle standardu IEC 61850 je zobrazeno na obr. 3.12.



Obr. 3.12: Výsledné zapojení

3.6 Spuštění simulace

Kapitoly 3.2.1, 3.2.2 a 3.2.3 bylo popsáno, jakým způsobem byly jednotlivé stanice vytvořeny a nakonfigurovány. V další fázi je nutné samotnou simulaci spustit. Samotné spuštění Raspberry Pi není dostačující. Ve chvíli, kdy je Raspberry Pi spuštěno, dojde ke spuštění pouze systémových programů a balíčků. Systém běží, čeká na příkazy, ale nedělá nic. Proto je nutné do zařízení vzdáleně nahrát konfigurační soubor tak, jak je uvedeno v kapitole 3.4. Po nahrání konfiguračního souboru ještě samotná simulace neběží. Soubor je uložen v paměti zařízení, nicméně pro začátek simulace je nutné ho spustit. Jak již bylo zmíněno, koncové stanice nemají grafické uživatelské prostředí. Proto je nutné soubor spustit pomocí příkazové řádky.

Proces spouštění se na jednotlivých stanicích liší. Zatímco na stanicích OutstationA a Koncentrátor je uložen pouze jeden spouštěcí soubor, na stanici OutstationB se nachází hned tři soubory. Konkrétně se jedná o hlavní, řídící soubor,

a posléze o doplňující dva soubory, které mají za úkol odesílání zpráv pomocí protokolu GOOSE a SMV. Oddělení odesílání do dvou různých souborů bylo nutné z důvodu co nejreálnější simulace. Blíže je tato problematika popsána v kapitole 3.2.2.

Cesta k souboru je poměrně dlouhá. Operační systém Raspbian nabízí uživateli možnost doplňovaní potřebných příkazů nebo názvu složek po stisknutí klávesy *Tab*. Ve chvíli kdy uživatel napíše část příkazu do takové podoby, že již v daném adresáři není jiná možnost, po stisknutí klávesy *Tab* je příkaz či název adresáře automaticky doplněn do jeho plného znění. Pokud si uživatel není jist, jaké příkazy či adresáře jsou k dispozici, může dvakrát klepnout opět na klávesu *Tab* a operační systém Raspbian zobrazí nabídku dostupných adresářů či příkazů.

U stanic OutstationA a Koncentrátor, se samotná aplikace ke spuštění na vzdálené stanici nachází v adresáři *GNU-Linux*. Cesta do tohoto adresáře je přes `root/.netbeans/remote/192.168.253.12/debian-Linux-x86_64/root/NetBeansProjects/RPi2GOOSE_61850_soubor/dist/Debug/GNU-Linux/`.
Cesta se samozřejmě liší od konkrétního zařízení, jehož simulace je právě spouštěna. Pro srovnání například cesta ke spuštění koncentrátoru je `root/.netbeans/remote/192.168.253.11/debian-Linux-x86_64/root/NetBeansProjects/RPi1Server_61850_soubor/dist/Debug/GNU-Linux/`.
Z toho je patrné, že adresáře jsou pojmenovány podlé své vlastní IP adresy, ale také názvu souboru, který je definován již v prostředí NetBeans. Principiálně se však jedná vždy o ten samý postup. Ve složce se nachází požadovaný soubor, například `rpi3goose_61850_soubor`. Jeho spuštění je provedeno stiskem klávesy Enter. Kompletní cesta, včetně spuštění souboru pro je na obr. 3.13.

```
pi@raspberrypi:~ $ sudo .netbeans/remote/192.168.253.12/debian-Linux-x86_64/root/NetBeansPro
jects/RPi2GOOSE_61850_soubor/dist/Debug/GNU-Linux/rpi2goose_61850_soubor eth1
Using libIEC61850 version 1.3.0
Startuje proces pro SMV
Startuje proces pro Goose
```

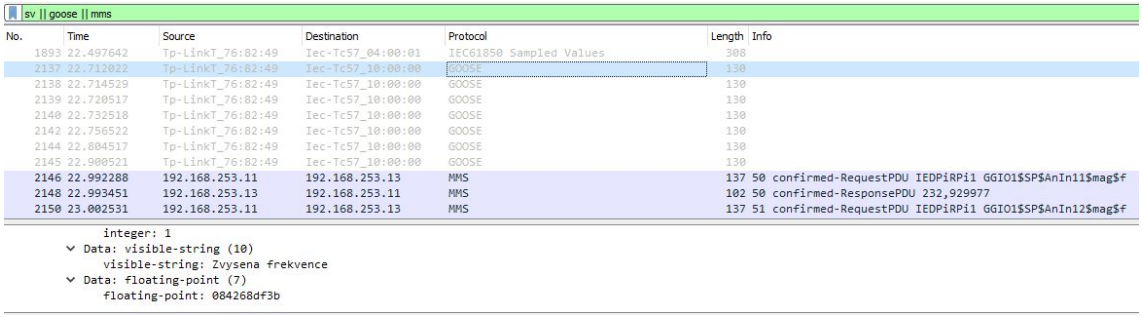
Obr. 3.13: Spuštění simulace OutstationB

Po spuštění souboru stanice začíná simulovat komunikaci. Úspěšné spuštění je signalizováno pomocí výpisu do konzole. Vždy je vypsáno jaká knihovna standardu IEC 61850 byla použita. Tato hodnota je dynamická proměnná. Za předpokladu, že by byl simulátor spuštěn na zařízení s jiným typem knihovny libiec61850, bude zde vypsána vždy aktuální knihovna.

Dále je do konzole vypsána potvrzovací hláška, že došlo ke spuštění komunikace protokolů GOOSE a SMV. Na základě toho může uživatel identifikovat zda

došlo ke správnému spuštění a zda jsou data odesílána do Koncentrátoru. To, jestli Koncentrátor přijímá data je patrné taktéž z výpisu do konzole, kde je vytvořeno improvizované HMI. Výpis je zobrazen na obr. 3.7, který se nachází v kapitole 3.2.3. Komunikace samozřejmě probíhá podle náležitostí standardu IEC 61850 tak, jak je popsáno v kapitolách 3.2.1 a 3.2.2

Protokol GOOSE v danou chvíli hlásí, že zařízení je aktivní. Při vygenerování chyby, tak jak je popsáno v kapitole 3.2.1, je odeslána také kritická GOOSE zpráva oznamující vznik dané události, následována dalšími GOOSE zprávami. Tyto zprávy při každém odeslání zdvojnásobí časový odesílací interval do doby, než je čas odeslání srovnán s periodou běžného odesílání. Na obr. 3.14 je zachycená komunikace pomocí Wiresharku. Je možné vidět, že došlo k překročení mezní hodnoty frekvence.



Obr. 3.14: Simulovaná komunikace zachycena pomocí Wiresharku

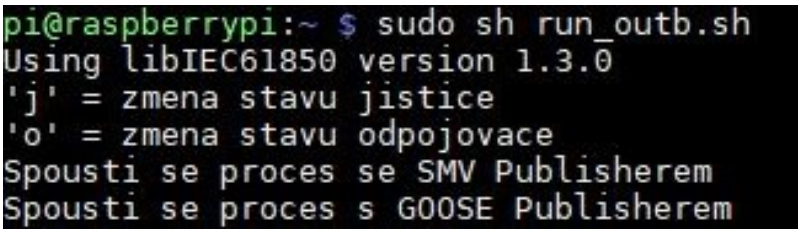
Odeslání kritické GOOSE zprávy a následná synchronizace do periodického odesíláni je patrná z časových značek jednotlivých rámců, kde je možné vidět, jak se interval zdvojnásobuje, než dosáhne požadované hodnoty.

Komunikace po spuštění programu programu probíhá principiálně stejně. OutstationB je rozšířen o několik funkcí využívajících protokol GOOSE, tak jak popisuje kapitola 3.2.2. Z hlediska spuštění programu ovšem dochází k rozdílům. U stanice OutstationB je nutné spustit hlavní a dva dílčí programy pro odesílání GOOSE a SMV. Důvody jsou popsány v již zmiňované kapitole.

Pro centrální spuštění a případné ukončení všech dílčích programů, byl vytvořen bashový skript. V případě spuštění stanice OutstationB tedy stačí spustit pouze zmiňovaný skript, který spustí zbylé podprogramy. Jeho spuštění vyžaduje administrátorské oprávnění, tudíž je nutné využít příkazu Sudo. Spuštění bashového skriptu probíhá pomocí příkazu:

```
sudo sh run_outb.sh
```

Pro správnou inicializaci všech dílčích programů, je nutné ve skriptu správně zadefinovat cestu k jejich úložišti. Konfigurace zmiňovaného skriptu se nachází v příloze C. Po úspěšném spuštění simulace je do příkazové řádky, podobně jako u OutstationA, vypsána použitá knihovna, potvrzení spuštění komunikace protokolu GOOSE a SMV. Jako další je zobrazen návod na ovládání jističe a odpojovače. Příkazová řádka po spuštění je vyobrazena na obr. 3.15.



Obr. 3.15: Spuštění simulace OutstationB

Závěr

Cílem této práce je čtenáře seznámit se základními principy síťové komunikace podle standardu IEC 61850. Konkrétně jsou rozebírány komunikační protokoly SMV (Sampled Measured Values), GOOSE (Generic Object Oriented Substation Events), MMS (Manufacturing Message Specification) a SNTP (Simple Network Time Protocol). Dalším z cílů práce je vytvoření simulátoru komunikace podle zmiňovaného standardu. Výstupem práce jsou přiložené soubory v nichž se nachází konfigurace pro jednotlivá zařízení. Při úspěšném sestavení programu, je simulace spustitelná na zařízení Raspberry Pi. Tato práce je zaměřena na snadnou rozšiřitelnost a přenositelnost knihovny standardu IEC 61850.

V teoretické části jsou blíže definovány jednotlivé protokoly, které standard IEC 61850 využívá pro svou komunikaci. Mimo jiné je také obecně popsána komunikace standardu, včetně příkladů, ze kterých je patrné, kde je možné se s tímto typem komunikace setkat.

V praktické části je popsána tvorba simulátoru a jeho úprava do co nejreálnější podoby. Byla zrealizována jednoduchá topologie tří stanic Raspberry Pi, z nichž každá simuluje určitou část reálné rozvodny. Stanice Outstation A a B simulují IED, zatímco Koncentrátor slouží jako SAS, HMI a SNTP server. Jednotlivé stanice jsou mezi sebou propojeny pomocí přepínače.

Časová synchronizace pomocí protokolu SNTP je zajištěna pomocí balíčku nainstalovaného přímo na Raspberry Pi. V případě přenosu aplikace na jiný hardware je tedy nutné časový server spolu s klienty nakonfigurovat na novém zařízení.

Stanici OutstationA je možné pomocí protokolu MMS ovládat z konfiguračního textového souboru z Koncentrátoru. Pokud je stanice sepnutá, generuje pseudonáhodné hodnoty oscilující kolem definované hodnoty a odesílá tyto hodnoty pomocí protokolu SMV na Koncentrátor. Protokol GOOSE na této stanici plní funkci ochrany, která zasílá pravidelné rámce potvrzující aktivitu stanice, a zároveň monitoruje měřené veličiny. V případě překročení mezní hodnoty odesílá kritickou zprávu. Pro realistickou simulaci je do simulátoru implementován také simulátor chyb. Z hlediska časové synchronizace stanice vystupuje jako SNTP klient. Časové značky jsou patrné například v logu.

Stanice OutstationB simuluje měřené hodnoty na základě konfiguračního tex-

tového souboru, který lze upravovat. Protokol MMS je použit pro okamžité vyčítání naměřených hodnot do Koncentrátoru, v uživatelem definované časové periodě. Stanice je také rozšířena o funkci jističe a odpojovače, které reagují automaticky na možné problémy na síti, ale zároveň je možné je ovládat i lokálně pomocí stisku předdefinované klávesy. Odeslaná data jsou taktéž zaznamenávána do logu. Z hlediska časové synchronizace stanice vystupuje také jako SNTP klient.

Koncentrátor je nadřazená řídící stanice. Slouží ke sbírání dat z podřízených stanic a vypisování přijatých hodnot do improvizovaného HMI, které je vytvořeno v příkazové řádce. Z koncentrátoru jsou ovládány podřízené stanice pomocí protokolu MMS. Přijaté hodnoty jsou zaznamenávány do logu.

Literatura

[1] STUDENÝ, Radim. *SIMULÁTOR KOMUNIKACE PROTOKOLŮ SCADA* [online]. Brno, 2018 [cit. 2019-11-03]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/110094. Diplomová práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ, FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ, ÚSTAV TELEKOMUNIKACÍ. Vedoucí práce Petr Blažek.

[2] FORGUE, Bruno a Pavel VLADYKA. IEC 61850: soubor norem pro komunikaci v energetice s velkým potenciálem výhod. In: *Automa - časopis pro automatizační techniku* [online]. Praha: Autonoma, 2010 [cit. 2019-11-02]. Dostupné z: https://automa.cz/cz/casopis-clanky/iec-61850-soubor-norem-pro-komunikaci-v-energetice-s-velkym/potencialem-vyhod-2010_03_40771_5154/

[3] MATOUŠEK, Petr. *Popis komunikace IEC 61850.* FIT-TR-2018-01, Brno, CZ: Fakulta informačních technologií VUT v Brně, 2018, 88 s. Dostupné také z: https://www.fit.vut.cz/research/publication/11832

[4] HUDEC, Jan. *GENERÁTOR ÚTOKŮ NA SCADA PROTOKOLY* [online]. Brno, 2019 [cit. 2019-11-03]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/121397?zp_id=121397. Bakalářská práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ, FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ, ÚSTAV TELEKOMUNIKACÍ. Vedoucí práce Petr Blažek.

[5] DE MESMAEKER, Ivan, et al. *Substation Automation based on IEC 61850.* In: Cigré SC B5 6th Regional CIGRÉ conference in Cairo. 2005.

[6] STODŮLKA, Ivo. *MODEL ELEKTRICKÉ STANICE S KOMUNIKAČNÍM PROTOKOLEM IEC 61850* [online]. Brno, 2012 [cit. 2019-11-05]. Dostupné z:https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=52341. Diplomová práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ, FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ, ÚSTAV ELEKTROENERGETIKY.

[7] SHARMA, Sanjay. *Substation Communication with IEC 61850 and Application Examples.* In: ABB Medium Voltage Products [online]. Chile: ABB Group, 2016, 29.11.2016, s. 46 [cit. 2019-11-05]. Dostupné z: http://www04.abb.com/global/seitp/seitp202.nsf/0/4d1c836b9e7fdb67c12580870047d7c8/file/1.Chile_+ABB+_Substatio+communication+with+IEC+61850+and+application+examples.pdf

[8] W. Huang, *"Learn IEC 61850 configuration in 30 minutes,"* 2018 71st Annual Conference for Protective Relay Engineers (CPRE), College Station, TX, 2018, pp. 1-5.doi: 10.1109/CPRE.2018.8349803

[9] Ozansoy, C.R. Zayegh, Aladin Kalam, Akhtar. (2009). *Time synchronisation in a IEC 61850 based substation automation system.* 1 - 7.

[10] IEC 61850 Sampled Values protocol. *Typhoon HIL Schematic Editor Library* [online]. , 8 [cit. 2019-11-15]. Dostupné z: https://www.typhoon-hil.com/documentation/typhoon-hil-schematic-editor-library/References/iec_61850_sampled_values_protocol.html

[11] BECKER, Farel, Stefan NOHE, Siemens a Alex ECHEVERIRA. Designing Non-Deterministic PAC Systems to Meet Deterministic Requirements. *PacWorld* [online]. 2015, 22.6.2015 [cit. 2019-11-29]. Dostupné z: https://www.pacw.org/issue/june_2015_issue/deterministic_system/designing_nondeterministic_pac_systems_to_meet_deterministic_requirements/complete_article/1.html

[12] *IED pro chránění a ovládání vývodu REF615: Aplikační manuál* [online]. In: . Trutnov: ABB s.r.o. Divize Power Systems, 2012, 14.12.2012, s. 209 [cit. 2019-11-30]. DOI: 1MRS757137. Dostupné z: https://library.e.abb.com/public/0f904438c7317b01c1257b500036486d/REF615_appl_757137_CZc.pdf

[13] SMV – IEC 61850-9-2. *SoC-e-* [online]. [cit. 2019-12-11]. Dostupné z: https://soc-e.com/smv-iec-61850-9-2/

[14] Fork() in C. *: A computer science portal for geeks* [online]. [cit. 2019-12-11]. Dostupné z: https://www.geeksforgeeks.org/fork-system-call/

[15] DITRYCH, Marek. *PROBLEMATIKA VYHODNOCOVÁNÍ BLOKOVACÍCH PODMÍNEK ROZVODNY POMOCÍ GOOSE ZPRÁV* [online]. Brno, 2019 [cit. 2019-12-15]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=191496. Diplomová práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedoucí práce Jaroslava Orságová.

[16] KRIGER, C., S. BEHARDIEN a J. RETONDA-MODIYA. A Detailed Analysis of the GOOSE Message Structure in an IEC 61850 Standard-Based Substation Automation System. *INT J COMPUT COMMUN [online]. 2013, , 14 [cit. 2019-11-15]. ISSN 1841-9836.* Dostupné z: http://univagora.ro/jour/index.php/ijccc/article/viewFile/329/pdf_66

[17] ANDREJČÁK, Michal. *Digitální rozvodny: Úvod, výhody, nabídky, reference* [online]. In: . 2017 [cit. 2019-11-19]. Dostupné z: http://search-ext.abb.com/library/Download.aspx?DocumentID=9AKK107045A6902&LanguageCode=cs&DocumentPartId=&Action=Launch

[18] *Industrial automation system - Manufacturing Message Specification: Protocol definition* [online]. In: . 2000 [cit. 2019-12-19]. Dostupné z: https://www.sis.se/api/document/preview/616829/

[19] Libiec61850: Open source library for IEC 61850, 2016, [online] Available: http://libiec61850.com/libiec61850/

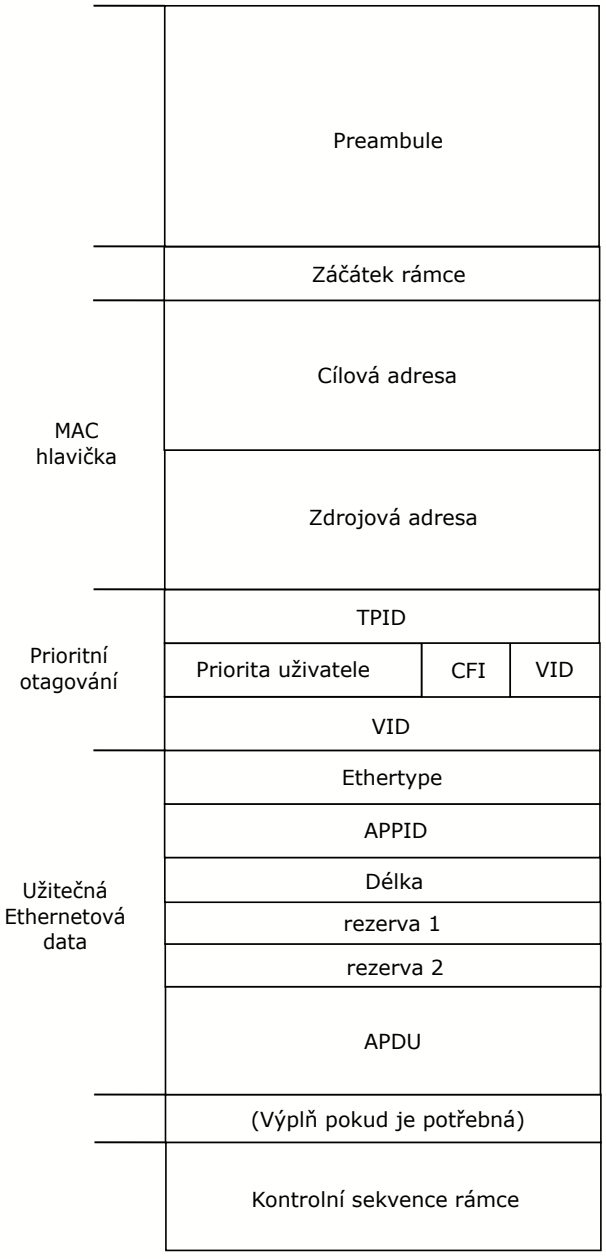
[20] *Blokové schéma Přepínače 1* [online]. In: . [cit. 2020-05-16]. Dostupné z: https://i.mt.lv/cdn/rb_files/CRS109-150409135152.png

Seznam symbolů, veličin a zkratek

SCADA	Supervisory Control And Data Acquisition
IEC 61850	Mezinárodní elektrotechnická komise – Intemational Electrotechnical Commission
GOOSE	Generic Object Oriented Substation Events
SMV	Sampled Measured Values
PD	Physical Device
IED	Intelligent Electronical Device
LD	Logical device
LN	Logical Node
DO	Data Object
MMS	Manufacturing Message Specification
SNTP	Simple Network Time Protocol
APPID	Aplication ID
APDU	Application Protocol Data Unit
HMI	Human Machine Interface
SPP	Samples Per Periode
TPID	tag protocol identifier
ASDU	Application Specific Data Unit
VMD	Virtual Manufacturing Device
TSAP	Transport Service Access point
COTP	Connection-Oriented Transport Protocol
SAS	Substation Automation System
CLI	Command Line Input
DHCP	Dynamic Host Configuration Protocol
GUI	Graphic User Interface
SSH	Secure Shell

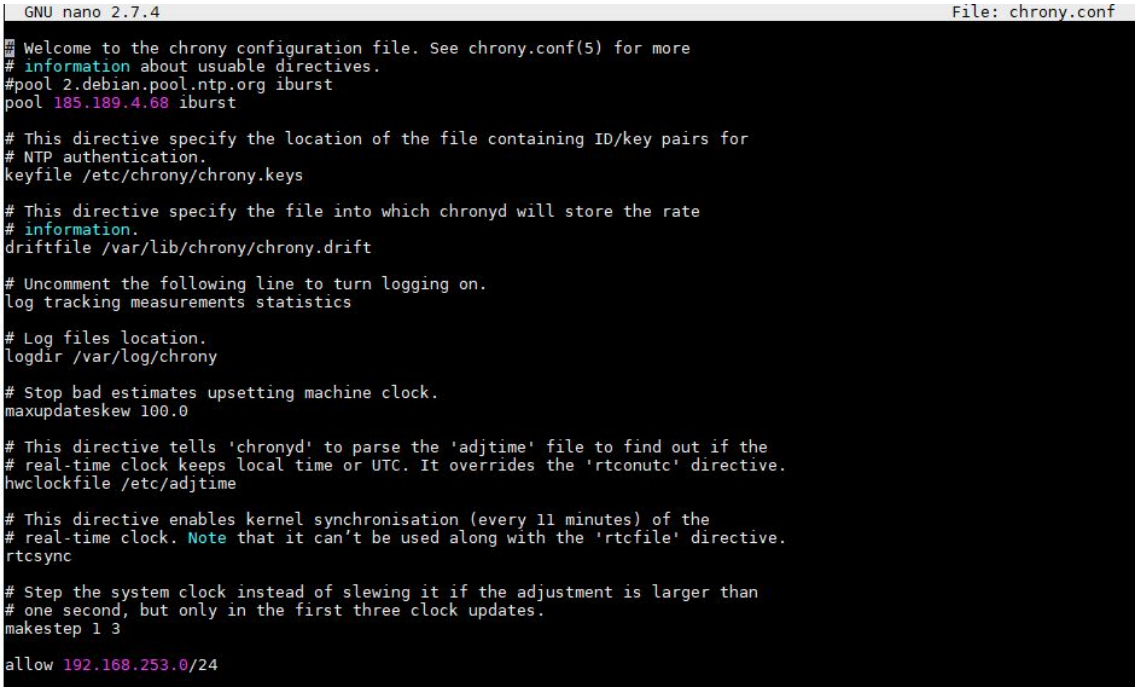
Seznam příloh

A Struktura protokolu SMV uvnitř ethernetového rámce



Obr. A.1: SMV protokol uvnitř ethernetového rámce [10].

B Konfigurace časového serveru



```
 GNU nano 2.7.4                                                File: chrony.conf

# Welcome to the chrony configuration file. See chrony.conf(5) for more
# information about usuable directives.
#pool 2.debian.pool.ntp.org iburst
pool 185.189.4.68 iburst

# This directive specify the location of the file containing ID/key pairs for
# NTP authentication.
keyfile /etc/chrony/chrony.keys

# This directive specify the file into which chronyd will store the rate
# information.
driftfile /var/lib/chrony/chrony.drift

# Uncomment the following line to turn logging on.
log tracking measurements statistics

# Log files location.
logdir /var/log/chrony

# Stop bad estimates upsetting machine clock.
maxupdateskew 100.0

# This directive tells 'chronyd' to parse the 'adjtime' file to find out if the
# real-time clock keeps local time or UTC. It overrides the 'rtconutc' directive.
hwclockfile /etc/adjtime

# This directive enables kernel synchronisation (every 11 minutes) of the
# real-time clock. Note that it can't be used along with the 'rtcfile' directive.
rtcsync

# Step the system clock instead of slewing it if the adjustment is larger than
# one second, but only in the first three clock updates.
makestep 1 3

allow 192.168.253.0/24
```

Obr. B.1: Konfigurace časového serveru pomocí balíčku Chrony.

C Obsah přiloženého Flash disku

- `/` ..... kořenový adresář přiloženého Flash disku
  - `Elektronická verze práce` ..... elektronická verze diplomové práce
    - `Srp Jakub_diplomová práce.zip`
    - `Srp Jakub_diplomová práce.pdf`
  - `OutstationA` ..... zdrojové kódy stanice OutstationA
    - `download.txt`
    - `log_goose.txt`
    - `log_smv.txt`
    - `main.cpp`
  - `OutstationB` ..... zdrojové kódy stanice OutstationB
    - `GOOSE`
      - `main.cpp`
    - `Main`
      - `main.cpp`
    - `SMV`
      - `main.cpp`
    - `download.txt`
    - `log_goose.txt`
    - `log_goose_ochrana.txt`
    - `log_smv.txt`
    - `run_outb.sh`
  - `Koncentrátor` ..... zdrojové kódy pro stanici Koncentrátor)
    - `config_a.cfg`
    - `config_b.cfg`
    - `download.txt`
    - `log_goose.txt`
    - `log_goose_ochrana.txt`
    - `log_smv.txt`
    - `main.cpp`