UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Boštjan Bajc

Strežniški program na osnovi protokola IEC61850 za obveščanje o elektroenergetskem stanju sistema

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Zoran Bosnić

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00079/2011
Datum: 07.03.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **BOŠTJAN BAJC**

Naslov: **STREŽNIŠKI PROGRAM NA OSNOVI PROTOKOLA IEC61850 ZA OBVEŠČANJE O ELEKTROENERGETSKEM STANJU SISTEMA**

SERVER PROGRAM BASED ON THE IEC61850 PROTOCOL FOR NOTIFYING OF THE ELECTRICITY SYSTEM STATE

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V elektroenergetiki sta komunikacija z elektroenergetskimi napravami in njihovo nadzorovanje nepogrešljiva elementa delovanja sistema. Za izvedbo omenjenih funkcionalnosti je na razpolago več protokolov, od katerih se je IEC61850 uveljavil kot standard. Kandidat naj v diplomski nalogi implementira strežniški program, ki se uporablja za obveščanje o stanju elektroenergetskih naprav v sistemu in temelji na protokolu IEC61850. V nalogi naj kandidat opiše delovanje protokola in posameznih elementov komunikacijskega sistema ter predstavi razvoj in uporabnost strežniškega programa.

Mentor:

doc. dr. Zoran Bosnić

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Boštjan Bajc,

z vpisno številko 63040190,

sem avtor diplomskega dela z naslovom:

Strežniški program na osnovi protokola IEC61850 za obveščanje o elektroenergetskem stanju sistema

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Zorana Bosnića
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 10.03.2011 Podpis avtorja:

Zahvala

Iskreno se zahvaljujem vsem tistim, ki so pomagali pri nastanku diplomskega dela. Posebno zahvalo namenjam mentorju doc. dr. Zoranu Bosniću za vse koristne nasvete in korekten strokovni odnos, Iztoku Kobalu za strokovno pomoč in za zaupanje pri dodelitvi tako zahtevne naloge, Urošu Kobalu in Jaki Žvan za pomoč pri razumevanju problemske domene, celotni razvoji ekipi PRR-ZVE, ki je kakorkoli sodelovala pri razvoju IEC61850 ter sestri in Mateji, ki so me spodbujale in pomagale na celotni poti do uspeha.

Kazalo

Povzetek **1**

Abstract **2**

1 Uvod **3**
1.1 Opredelitev problema . . . 3
1.2 Namen diplomske naloge . . . 3
1.3 Struktura diplomske naloge . . . 4
1.4 Cilji . . . 5

2 Standard IEC61850 **6**
2.1 Kratek opis . . . 6
2.1.1 GOOSE . . . 6
2.1.2 MMS . . . 7
2.2 Sestavni deli konfiguracijske datoteke IEC61850 . . . 8
2.2.1 Hierarhija objektov . . . 9
2.3 Smeri pošiljanja podatkov . . . 10

3 Opis integracije IEC61850 strežnika **12**
3.1 Izvor podatkov za prenos po protokolu . . . 12
3.1.1 Podatkovna baza RTDB . . . 12
3.1.2 Datoteke XML . . . 13
3.1.3 Datoteke ASCII . . . 13
3.2 Opis preslikavanja na izvorne podatke . . . 13
3.2.1 IEC61850 DAI@sAddr . . . 14
3.2.2 Sintaksa sAddr . . . 14
3.2.3 Opis vrednosti atributa sAddr . . . 16
3.3 Arhitektura . . . 16
3.4 Način pridobivanja podatkov . . . 19
3.4.1 Nit *Cycle* . . . 19

3.4.2 Nit *PollHandler* 20
3.4.3 Nit *EventHandler* 21
3.4.4 Branje iz datotek 23
3.5 Obdelava zahtev s strani IEC61850 odjemalca 24
3.5.1 Zahteve za nadzor s strani odjemalca IEC61850 24
3.5.1.1 ReadHandler 24
3.5.2 Zahteve za upravljanje s strani odjemalca IEC61850 . . . 26
3.5.2.1 WriteHandler 26
3.5.3 Izvajanje operacij nad ukaznim objektom 26
3.5.3.1 Preverjanje pravilnosti izvajanja ukaza 27
3.5.3.2 Preslikava ukaznega objekta 29
3.5.3.3 Preverjanje zaklepanja in sinhronizacije 29

4 Uporaba v praksi **30**

5 Sklepne ugotovitve **33**

Seznam slik **35**
Seznam tabel **36**

Literatura **37**

Seznam uporabljenih kratic in simbolov

IEC61850	standard za komunikacije v sistemu energetske podpostaje
NEO3000	platforma za izvedbo sistema zaščite, nadzora in vodenja v elektroenergetiki
TMW	Triangle MicroWorks
SCL	Source Code Library
MMS	Manufacturing Message Specification - (specifikacija industrijskega sporočanja)
GOOSE	Generic Object Oriented Substation Events - (generični objektno orientirani dogodki podpostaj)
SMV	Sampled Measured Values - (vzorčne vrednosti meritev)
IED	Intelligent Electronic Device - (inteligentna elektronska naprava)
ICD	IED Capability Description - (datoteka XML s komunikacijskimi zmožnostmi IEC61850)
SCL	Substation Configuration Language - (konfiguracijski jezik razdelilne postaje)
DAI	Data Attribute Instance - (primerek podatkovnega atributa)
RTDB	Real-Time DataBase - (podatkovna baza v realnem času)
PSM	Power System Manager - (upravljalec energetskega sistema)
RTE	*NEO3000* RealTime Environment - (okolje v realnem času sistema NEO3000)
SCADA	Supervisory Control And Data Acquisition - (postajni nadzorni sistem)

Povzetek

Nepogrešljiv člen v elektroenergetiki je komunikacija z elektroenergetskimi napravami ter izvajanje upravljanja in nadzora nad njimi. Funkcionalnost komunikacije tvorita strežnik, ki se nahaja na napravi in je namenjen posredovanju informacij o stanju naprave v vsakem trenutku, in odjemalec, ki komunicira s strežnikom preko komunikacijskega protokola in izvaja upravljanje in nadzor nad napravo. Cilj naloge je ustvariti strežniški program, ki bo del IEC61850 strežnika in bo sposoben komunicirati z IEC61850 odjemalcem s standardiziranim protokolom MMS, mu posredovati podatke o procesnih spremenljivkah ter izvajati zahteve nad ukaznimi objekti. V diplomi analiziramo strukturno zasnovo celotnega komunikacijskega protokola ter način povezovanja procesnih spremenljivk na komunikacijske podatke, kjer smo za definicijo povezovanja uporabili EBNF notacijo. Sledi definicija arhitekture strežniškega programa, kjer smo za implementacijo uporabili kombinacijo programskih jezikov C in C++. Ključni del programa zahteva učinkovito serviranje podatkov in obdelovanje zahtev s strani odjemalca. Rezultat dela je delujoč strežniški program, ki obdeluje zahteve za upravljanje in nadzor nad procesnimi spremenljivkami, ki so preslikane na komunikacijski protokol IEC61850.

Ključne besede:

strežniški program, protokol MMS, strežnik, odjemalec

Abstract

Indispensable part in power industry is communication with intelligent electronic devices and to control and monitor them. The communication consists of a server, which is located on the device and is intended to pass the information of the device status and a client, which is communicating with a server through the communication protocol, to control and manage the device. The thesis describes the server program, which is a part of the IEC61850 server and is capable of communicating with IEC61850 client through the MMS protocol, passing the information of process variables and to executing control operations. In thesis we analyse the structure of the communication protocol and define the way of binding process variables to communication data, where we use the EBNF notation. Next we define the arhitecture of the server program, where we decided to use a combination of C and C++ programming languages. The key part of the server program requested effective serving of data and processing of client requests. The result is a working server program, which can process the control and monitor demands over the process variables, which are mapped to IEC61850 protocol.

Key words:

server program, protocol MMS, server, client

Poglavje 1

Uvod

1.1 Opredelitev problema

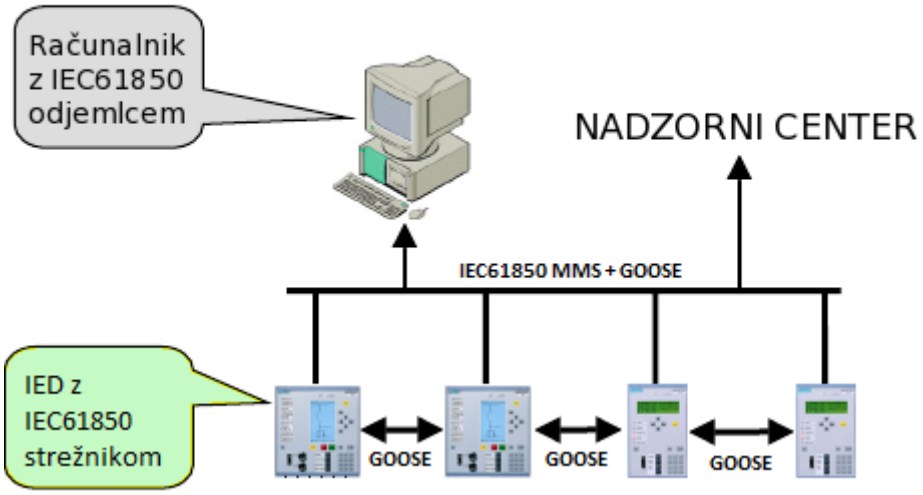
Z razvojem komunikacijskih sistemov in z globalizacijo energetskih trgov se je pojavila potreba po standardizaciji komunikacij za inteligentno opremo v energetskih sistemih. Standardizacija omogoča povezovanje naprav različnih proizvajalcev v celovite sisteme.

V Evropi je bila pred uvedbo IEC61850 standarda za komunikacijo v energetskih sistemih razširjena uporaba protokola IEC60870-5-103, ki ne omogoča horizontalne komunikacije med napravami in zaradi te omejitve otežuje interoperabilnost [1]. V Ameriki so proizvajalci uporabljali arhitekturo UCA2.0, ki je predhodnica standarda IEC61850, vendar med uporabniki nikoli ni bila sprejeta kot ustrezna rešitev. Leta 2004 je prišlo do združitve tehnologij v skupni standard IEC61850.

1.2 Namen diplomske naloge

Prednost uporabe IEC61850 strežnika je v tem, da uporablja omrežni protokol za komunikacijo z odjemalcem in je zato zelo razširjen, ponavadi celo zahtevan, saj se uporablja v elektroenergetiki po celem svetu. Brez podpore komunikacijskega protokola IEC61850 danes skorajda ni več mogoče prodati elektroenergetske naprave. Ravno zato so cilji naloge implementirati strežniško komponento, ki bo na osnovi IEC61850 komunikacijskega protokola servirala informacije o stanju elektroenergetskega sistema, ter pridobiti certifikat o skladnosti delovanja komponente s standardom in s tem povečati možnosti konkuriranja na svetovnem trgu.

Z uvedbo standarda IEC61850 smo tako pridobili horizontalno komunikacijo med napravami, ki poteka s telegrami IEC61850 GOOSE in vertikalno komunikacijo z nadzornimi centri ali računalniki, ki uporabljajo IEC61850 odjemalce (Slika 1.1). Za razliko od protokola IEC60870-5-103 lahko z IEC61850 nadzorujemo skorajda celotno stanje naprave ter jo z ukazi nad ukaznimi objekti tudi upravljamo.



Slika 1.1: Komunikacijska infrastruktura

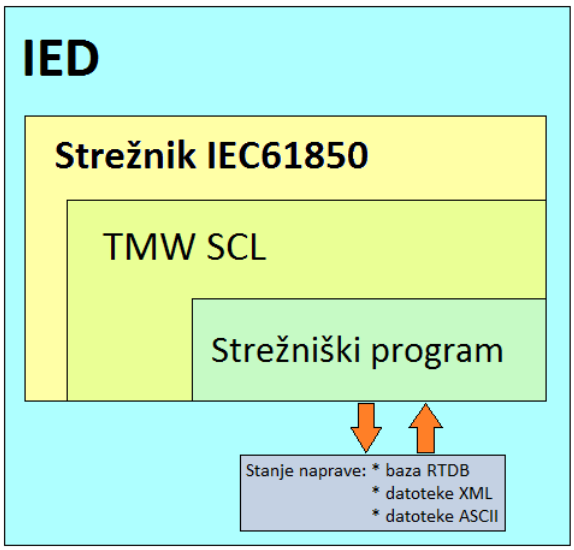
1.3 Struktura diplomske naloge

Teoretični del diplomske naloge bo zajemal splošno predstavitev komunikacijskega protokola IEC61850, njegovo uporabo in glavne lastnosti. V praktičnem delu bomo opisali celoten potek razvoja, od analize do končnega produkta. Predstavili bomo posamezne postopke razvoja in podrobno opisali nekatere funkcionalnosti s strani uporabnika. V Poglavju 2 je na kratko opisana struktura standarda, v Poglavju 3 pa je opisano, kako smo definirali obseg podatkov, ki se prenašajo po protokolu, in načini dostopa do njih. V zadnjih dveh poglavjih, 4. in 5., pa so opisani dosežki, težave pri nastajanju diplomskega dela in možnosti za nadaljnji razvoj.

1.4 Cilji

Ker je standard IEC61850 zelo obsežen in bi bilo s tem razvojno obdobje celotnega komunikacijega protokola zelo dogotrajno, smo se odločili, da implementacijo protokola kupimo od ameriškega proizvajalca rešitev za komunikacijske protokole Triangle MicroWorks (TMW). S tem smo si prihranili nekaj let trdega dela. Kljub temu sama vgradnja komunikacijskega protokola v sistem NEO3000 ni bila tako preprosta, kot se je kazalo v začetku. TMW SCL podpira celotno komunikacijo s strani IEC61850 strežnika na vseh nivojih komunikacijskega protokola do odjemalca. Da smo integrirali celoten komunikacijski protokol v sistem NEO3000, je bilo potrebno realizirati sistem, ki bo TMW SCL serviral procesne vrednosti podatkov iz naprave, ta pa potem naprej do IEC61850 odjemalcev. Cilji naloge so tako:

- implementacija strežniškega programa, ki bo del procesa TMW SCL strežnika, naloga programa pa pridobitev informacij o stanju naprave, kar je razvidno na sliki 1.2,
- definiranje virov podatkov, ki opisujejo stanje naprave in
- način pridobitve podatkov ter njihov nadzor in upravljanje.



Slika 1.2: TMW SCL IEC61850 strežnik z strežniškim programom v IED napravi

Poglavje 2

Standard IEC61850

2.1 Kratek opis

IEC61850 je standard v elektroenergetskih sistemih, namenjen za potrebe snovanja avtomatizacije v razdelilnih transformatorskih postajah. Namen uprabe standarda je potreba po komuniciranju naprav neodvisno od proizvajalca opreme. Sam standard opisuje komunikacijske zahteve, funkcionalne karakteristike, strukturo podatkov v napravi, dogovore o poimenovanju podatkov ter kako aplikacije komunicirajo z napravo in jo nadzirajo.

V standardu obstajajo različne implementacije protokolov. Trenutno implementacija obsega protokole MMS, GOOSE, SMV in spletne storitve. Mi smo izvedli zgolj podporo za GOOSE in MMS protokole, od katerih bom GOOSE le kratko označil, glavna tema diplomske naloge pa bo navezava IEC61850 MMS protokola na procesne informacije. Ti protokoli se lahko implementirajo v TCP/IP omrežjih in v lokalnih omrežjih razdelilnih transformatorskih postaj.

2.1.1 GOOSE

GOOSE sporočila so namenjena prenašanju podatkov med IED-ji in so le en del standarda IEC61850. GOOSE je mehanizem za hiter prenos dogodkov iz razdelilne postaje, kot so ukazi, alarmi, indikacije in sporočila. Posamezno sporočilo GOOSE, poslano s strani IED-ja, lahko sprejme in uporabi več prejemnikov. Standard zahteva, da je za generiranje posameznega sporočila GOOSE in posredovanje le-tega od trenutka, ko se dogodek zgodi, dovoljeno največ tri milisekunde, zato se po navadi generiranje telegramov GOOSE implementira v okolju, ki zahteva odziv v realnem času, saj ne sme prihajati do sistemskih prekinitev.

2.1.2 MMS

MMS je mednarodni standard (ISO/IEC 9506), ki opisuje mehanizem sporočanja za prenos podatkov v realnem času in nadzor kontrolnih informacij med napravami v omrežju in računalniškimi aplikacijami [2].

MMS določa naslednja pravila:

- Skupino standardnih objektov, ki morajo obstajati v vsaki napravi, nad katerimi je mogoče izvesti operacije, kot so branje, pisanje, sporočanje dogodkov in podobno. Glavni objekt protokola je virtualna industrijska naprava (VMD), vsi ostali objekti, kot so spremenljivke, domene, dnevniki in datoteke, pa se nahajajo pod VMD.

- Skupino standardnih sporočil za sporočanja med IEC61850 odjemalci in strežniki za potrebe nadziranja (*Monitoring*) in upravljanja (*Controlling*) objektov.

- Skupino kodirnih pravil za preslikavo teh sporočil na bite in bajte ob prenašanju.

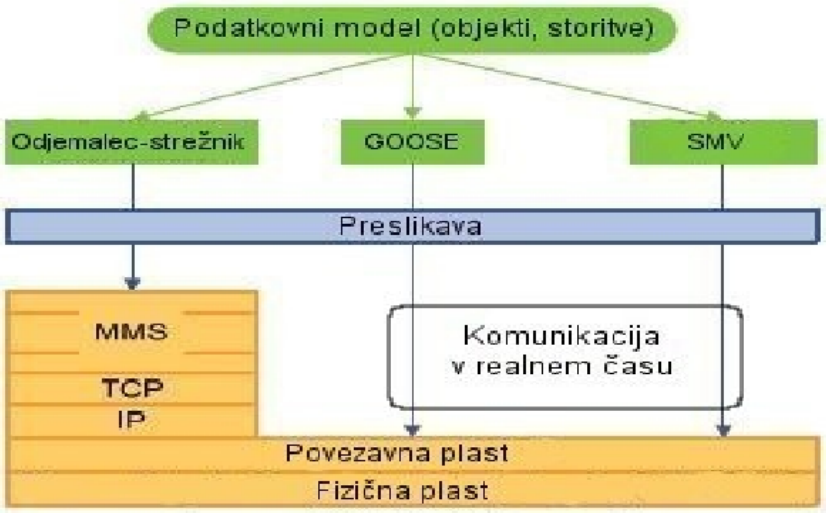
MMS objektni modeli, servisi in sporočila so enaki ne glede na tip naprave, če so to roboti ali programabilni logični kontrolerji.

Na sliki 2.1 vidimo, da je MMS protokol implementiran na zgornjih plasteh komunikacijskega modela, in sice na:

- *aplikacijski plasti*, ki je vmesnik med uporabnikom in komunikacijskim omrežjem,

- *predstavitveni plasti*, ki skrbi za uskladitev različnih načinov predstavitve podatkov (kompresija, dekompresija, nabor znakov, šifriranje, podatkovne formate) in

- *sejni plasti*, ki določa vzpostavitev, vzdrževanje in prekinitev seje ter vrsto komunikacije, ki je lahko enosmerna, izmenično dvosmerna ali dvosmerna.



Slika 2.1: IEC61850 komunikacijske plasti

Prokola GOOSE in SMV pa zahtevata komunikacijo v realnem času in se povezujeta neposredno na povezavno plast OSI, ki skrbi za določanje enote sporočil, način ugotavljanja napak med dvema sosednjima vozliščema, odpravo napak, omrežno topologijo, mehanizme dostopa do prenosnega medija in kontrolo pretoka [6].

2.2 Sestavni deli konfiguracijske datoteke IEC61850

Del standarda IEC61850 je konfiguracijski jezik razdelilne postaje (*SCL*). *SCL* je skupni jezik, baziran na XML jeziku in opisan z XML shemo in ga lahko uporabljamo za izmenjavo informacij neodvisno od proizvajalca naprave. V SCL jeziku je vsebovana predstavitev podatkovnega modela in komunikacijskih servisov, ki so določeni s standardnimi dokumenti IEC61850. V teh dokumentih so predstavljeni podatki za entitete podpostaj.Eden od teh dokumentov je tudi XML datoteka z IEC61850 komunikacijskimi zmožnostmi za posamezno inteligentno elektronsko napravo *IED*. Vsebuje vse informacije o njej in jo uprabljamo za konfiguriranje strežnika IEC61850. V grobem se deli na tri dele.

V prvem delu, ki se imenuje *Header*, identificiramo SCL konfiguracijsko datoteko in verzijo le-te, lahko opišemo ime orodja, s katerim je bila datoteka generirana, in revizijo datoteke.

Drugi del, ki se imenuje *Communication*, opisuje možnosti direktnih komunikacijskih povezav med logičnimi vozlišči glede na pomen podomrežja in IED dostopnih točk.

Tretji del se imenuje *IED* in opisuje pred-konfiguracijo inteligentne elektronske naprave, kot so dostopne točke, logične naprave in logična vozlišča, instancirana iz logičnih naprav. Definira še zmožnosti inteligentne elektronske naprave s komunikacijskimi servisi skupaj s tipi logičnih vozlišč, podatkovnimi objekti in njihovimi privzetimi konfiguracijskimi vrednostmi. Vsa logična vozlišča in njegovi podelementi so instancirani iz dela *DataTypeTemplates*.

V delu *DataTypeTemplates* definiramo tipe logičnih vozlišč, za katere je nato moč narediti primerek v IED delu ICD datoteke. Tip logičnega vozlišča je referenciran vsakokrat, ko je potrebno narediti primerek znotraj IED dela datoteke.

2.2.1 Hierarhija objektov

IEC61850 deli podatke v 13 različnih logičnih skupin, ki jih prikazuje tabela 2.1. Namen je, da se vse podatke, ki izhajajo iz podpostaj, dodeli eni od teh skupin:

SKUPINA LOGIČNIH VOZLIŠČ	OZNAKA
System logical nodes (sistemska logična vozlišča)	L
Protection functions (zaščitne funkcije)	P
Protection related functions (funkcije, povezane z zaščitami)	R
Supervisory control (nadzor vodenja)	C
Generic references (generične reference)	G
Interfacing and archiving (povezovanje in arhiviranje)	I
Automatic control (avtomatski nadzor)	A
Metering and measurement (meritve in števci)	M
Switchgear (preklapljanje)	X
Instrument transformer (spreminjanje orodij)	T
Power transformer (spreminjanje moči)	Y
Further power system equipment (oddaljena oprema)	Z
Sensors (senzorji)	S

Tabela 2.1: Skupine logičnih vozlišč

Vsaka od skupin je naprej razčlenjena na 86 različnih tipov logičnih vozlišč in vsak od njih je sestavljen iz podatkov, ki opisujejo pomen glede na specifično

aplikacijo. Funkcije avtomatskega nadzora na primer vsebujejo pet različnih logičnih vozlišč, kot prikazuje tabela 2.2.

LOGIČNO VOZLIŠČE	OZNAKA
Neutral current regulator	ANCR
Reactive power control	ARCO
Resistor control	ARIS
Automatic tap changer controller	ATCC
Voltage control	AVCO

Tabela 2.2: Logična vozlišča iz skupine avtomatskega nadzora

Logična vozlišča se nato delijo na 355 podatkovnih razredov, ki so razdeljeni v 7 kategorij, ki jih prikazuje tabela 2.3, in katerih končni elementi so podatki.

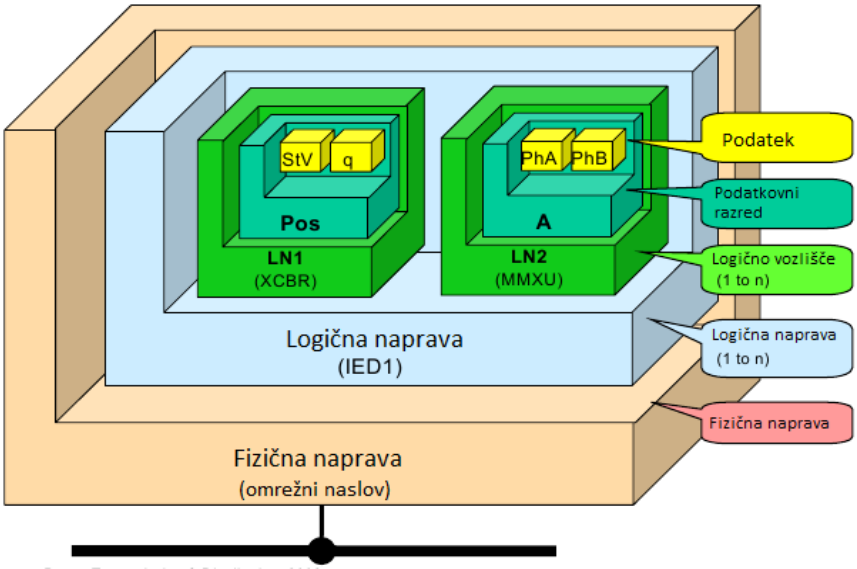
KATEGORIJA PODATKOVNIH RAZREDOV	ŠT. RAZREDOV
System information (sistemske informacije)	13
Physical device information (informacije naprave)	11
Measurands (meritve)	66
Metered values (števci)	14
Controllable data (podatki vodenja)	36
Status information (statusne informacije)	85
Settings (nastavitve)	130

Tabela 2.3: Kategorije podatkovnih razredov

Celotna hierarhija objektov je prikazana s sliko 2.2.

2.3 Smeri pošiljanja podatkov

IEC61850 je v grobem deljen na dva načina oziroma smeri komunikacije. To sta nadzor (monitoring) in upravljanje (controlling). Vsak podatkovni atribut ima po standardu določeno funkcijsko omejitev, ki klasificira specifično uporabo in določa servise, ki se lahko izvajajo nad podatkovnim atributom. Te funkcijske omejitve se uporabljajo v definiciji podatkov (vsebovanih v logičnih



Slika 2.2: Logične skupine (naprava, vozlišča, razredi in podatki)

vozliščih) in raznovrstnih kontrolnih blokih. Nad vsemi podatkovnimi atributi se ne glede na njihovo funkcijsko omejitev lahko izvaja nadzor podatkov, medtem ko je upravljanje dovoljeno le nad podatkovnimi atributi, ki imajo funkcijsko omejitev, ki jim to dovoljuje. Razlog za tako omejitev so različne vrste podatkov, ki se prenašajo po komunikaciji, od katerih nekatera predstavljajo izhodne vrednosti posameznih funkcionalnosti. Te lahko spreminjajo zgolj procesi katerim pripadajo in ne zunanji dejavniki. Zato je upravljanje dovoljeno le nad tistimi podatkovnimi atributi, ki predstavljajo vhodne podatke posameznim funkcionalnostim.

Poglavje 3

Opis integracije IEC61850 strežnika

3.1 Izvor podatkov za prenos po protokolu

Prva naloga pri implementiranju strežniškega programa je bila pridobiti vse možne izvore podatkov in definiranje načinov, kako do njih dostopati med samim delovanjem. Standard IEC61850 zahteva, da naprava pošilja več različnih vrst podatkov. Najpogostejši in najpomembnejši, ki se prenašajo, so podatki o procesih, ki se izvajajo na napravi, in podatki, ki opisujejo vrsto, tip in proizvajalca naprave. Da smo lahko pokrili vse te zahteve, smo določili tri vire podatkov, s katerimi smo lahko opisali celotno stanje in informacije o napravi:

- podatkovna baza RTDB,
- datoteke XML in
- datoteke ASCII

3.1.1 Podatkovna baza RTDB

Podatkovna baza RTDB je tako imenovana registrska baza. V sistemu NEO3000 je to glavni vir podatkov za delovanje celotne naprave. Sem se vpisujejo podatki v realnem času, ki odražajo stanje naprave v določenem trenutku. Hrani lahko le numerične vrednosti do velikosti 32-bitnega števila, ustvari pa se na podlagi datoteke XML, s katero je opisana struktura podatkovne baze, ki se deli na dva nivoja. Prvi nivo so podatkovni bloki, ki opisujejo in hranijo podatke o posameznem procesu v napravi in so oštevilčeni z indeksom bloka, ki

je unikaten na nivoju celotne naprave. Podatkovni bloki so nato sestavljeni iz registrov, ki predstavljajo izhodne podatke za posamezen proces in so prav tako oštevilčeni z indeksom registra. Vsak podatek v podatkovni bazi je tako unikatno na nivoju naprave označen z indeksom podatkovnega bloka in registra.

3.1.2 Datoteke XML

Nekateri od virov podatkov za delovanje naprave so tudi v XML. Ti se uporabljajo za potrebe kreiranja podatkovne baze in za hranjenje ostalih konfiguracijskih podatkov za same procese v napravi, ki so lahko numeričnih ali tekstovnih vrednosti. Vse datoteke XML so opisane in omejene s shemo XSD, ki narekuje, kako je datoteka XML zgrajena. V shemi je določena struktura datoteke XML in restrikcije glede vrednosti podatkov v njej.

3.1.3 Datoteke ASCII

Eden od virov podatkov v sistemu NEO3000 so tudi tekstovne datoteke, ki se ponavadi uporabljajo za konfiguracijo aplikacij v napravi. Največkrat hranijo vrednosti spremenljivk, ki služijo kot zagonski parametri različnim aplikacijam.

3.2 Opis preslikavanja na izvorne podatke

Datoteka ICD je datoteka, namenjena opisu konfiguracije komunikacijskega protokola IEC61850. Kot je opisano v prejšnjih poglavjih, je sestavljena hierarhično. Najnižje v hierarhiji je podatkovni atribut (DAI), ki vsebuje atribut *sAddr* in je namenjen povezavi komunikacijskega podatkovnega elementa z dejansko procesno spremenljivko, v večini primerov podatkovnim registrom.

Sintakso polja *sAddr* mora upoštevati tako implementacija integracije komunikacijskega protokola kot tudi grafični uporabniški vmesnik PSM (Power System Manager), ki generira konfiguracijski datoteki za napravo in strežnik IEC61850.

Notacija, ki smo jo uporabili za predstavitev sintakse preslikavanja, je računalniško splošno uporabljena Extended Backus-Naur (EBNF) notacija, s katero se običajno predstavljajo sintaktična pravila programskih ali drugih formalnih jezikov [3].

3.2.1 IEC61850 DAI@sAddr

sAddr je atribut DAI v konfiguracijskem dokumentu IEC61850, namenjen uporabniškemu naslavljanju informacijskih elementov specifičnega sistema. Sintaksa vrednosti atributa *sAddr* po standardu IEC61850 ni predpisana. V sistemu NEO3000 se vsebina *sAddr* nanaša na naslavljanje:

- registrov RTDB,
- XML elementov ali atributov,
- posameznih delov sekvenčnih ASCII datotek.

Vsak od načinov naslavljanja ima za sistem NEO3000 specifično sintakso, ki je opisana v naslednjem poglavju.

3.2.2 Sintaksa sAddr

Sintaksa *sAddr* atributa datoteke ICD je zapisana z Extended Backus-Naur Form notacijo, ki se uporablja za opis sintakse jezika v računalništvu. S to notacijo smo določili vsebino atributa, s katero je določeno, na kateri vir podatka v napravi se DAI nanaša, s kakšnim kodiranjem je zakodiran in na kakšen način se bo podatek osveževal ali spreminjal. V nadaljevanju je celotna EBNF sintaksa atributa tudi prikazana:

sAddr =[BS_conversion, “;”],
[DAI_reference], “:”, info_source, info_address, [“:”,specific_attributes],
[{ “;” [DAI_reference], “:” , info_source , info_address ,
[“:” , specific_attributes] }];

BS_conversion = “BS:” , “1N” | “GRAY” | “BCD”;

DAI_reference = [number], [„.“ , number];

number = digits | digits, number;

digits = “0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9” ;

info_source = “R” | “X” | “F” |;

info_address = R_address | X_address | F_address;

R_address =? RTDB datablock id ? , “.” , ? RTDB register ID ? ,
[“.”, R_type];

R_type = “B” | “I” | “C” | “F” ;

X_address =? pot do XML datoteke ? , “:” ,
? pot do elementa v XML datoteki ?

F_address =? filename URL ?, “:”,
? ime elementa v datoteki ? “:”
? ločilo ki loči ime element od vrednosti elementa ?

specific_attributes =[refresh_attributes] ,
[[„,“], submit_attributes] ,
[[„,“], conversion_attributes]

refresh_attributes =poll_class |
„refresh_only_once“ |
„refresh_everytime“ |
„refresh_on_file_modified“

submit_attributes = „submit_everytime“ | „submit_on_changed“

conversion_attributes =„conv-“, „S“ , [„-“] ,
scaling_factor , „+“ | „-“,
scaling_offset

scaling_factor = number, [“.”, number]

scaling_offset = number, [“.”, number]

poll_class = ? polll cycle time amount in milliseconds ?

Z zgornjo sintakso atributa *sAddr* smo določili vse možne kombinacije preslikave podatka na komunikacijski protokol, upoštevati pa jo mora strežnik ob serializaciji konfiguracijske datoteke strežnika in grafični uporabniški vmesnik *PSM*, ki jo generira.

3.2.3 Opis vrednosti atributa sAddr

V tem poglavju bomo predstavili nekaj primerov opisa atributov *sAddr*, s katerimi smo preslikali podatkovne atribute IEC61850 komunikacijskega protokola na podatke v napravi tipa NEO3000:

sAddr = ":R1.2" DAI je mapiran na 2. register iz 1. podatkovnega bloka. Ker ni definiranih specifičnih atributov, s katerim bi definirali način osveževanja, gre za informacijo, ki jo osvežuje proces EventHandler.

sAddr = "0:R2.1:1000;1:R2.2:1000,conv-S2.5-0.5;2:R2.3:1000" Preslikani so prvi trije elementi podatkovnega atributa, ki je polje, na prve tri registre 2. podatkovnega bloka, ki naj se osvežujejo v 1000 milisekundnem ciklu; DAI vrednost drugega registra je skalirana po formuli: $DAI = (R * 2.5)$–0.5

sAddr = ".7:XSettings.xml:/NEO3000/RT/Rtdb/Name:on_file_modified" Podatkovni atribut je preslikan na logični (boolean) parameter iz datoteke XML Settings.xml kot osmi bit v vrednosti podatkovnega atributa, pri čemer osvežimo vrednost v IEC61850 bazi TMW SCL ob vsaki zaznani spremembi datoteke Settings.xml, kar se preveri ob vsaki Monitoring zahtevi s strani IEC61850 odjemalca.

3.3 Arhitektura

Ključni del programa je zahteval učinkovito serviranje podatkov in enostavno dodajanje novih funkcionalnosti, pri čemer smo izkoristili zmožnosti objektno usmerjenega programskega jezika C++ [4]. Ker pa je TMW SCL del kode, ki se uporablja kot navezava na strežniški program napisana v programskem jeziku C, je bilo potrebno del kode napisati tudi v tem jeziku. Zato je strežniški program implementiran v kombinaciji jezikov C in C++.

Ob zagonu IEC61850 strežnika se najprej izvede serializacija konfiguracijske datoteke ICD, v kateri smo predhodno parametrizirali vse podatkovne atribute, ki naj bi jih naprava podpirala tako, da smo vsakemu določili atribut *sAddr*.

Serializacija se deli na serializacijo začetka in konca logičnih vozlišč, začetka in konca podatkovnih objektov, podatkovnih atributov in inicialnih vrednosti.

Za vsak DAI, ki je z atributom *sAddr* pravilno mapiran na procesno informacijo, ustvarimo nov objekt tipa *CDataAttributeUserData*. Če kateri od DAI nima definiranega atributa *sAddr* ali pa le ta ni v pravilnem formatu,

vse nadzorne in upravljalne funkcije nad podatkovnim atributom samim prevzame TMW SCL s svojimi funkcionalnostmi. V nasprotnem primeru smo zanj implementirali svoje funkcije za obvladovanje zahtev s strani odjemalca.

Ob klicu funkcij za obvladovanje zahtev TMW SCL kot argument poda kazalec na memorijsko lokacijo, ki mu jo določimo ob serializaciji DAI, zato ga nastavimo na referenco *CDataAttributeUserData* objekta, saj v njem hranimo vse potrebne informacije, ki jih posredujemo odjemalcu.

V nadaljevanju bomo na kratko opisali zgolj glavne objekte in njihove komponente, ki se uporabljajo za hranjenje procesnih informacij DAI in komunikacijo med njihovo podatkovno bazo in podatkovno bazo strežnika. Na sliki 3.1 je z UML diagramom razredov prikazana relacija med spodaj opisanimi razredi [5].

CDataAttributeUserData je osnovni razred, ki predstavlja podatkovno bazo komunikacije IEC61850. Vsebuje informacijo o preslikavanju na živo procesno informacijo in vse potrebne spremenljivke za podporo zahtevam nadziranja in upravljanja ter tudi inicializaciji. Ker so nekateri podatkovni objekti sestavljeni iz podatkovnih atributov, ki opisujejo vrednost podatka, čas zadnje spremembe in kvaliteto podatka, objekt hrani tudi kazalce na objekte istega tipa za povezovanje le-teh glede na zahteve standarda.

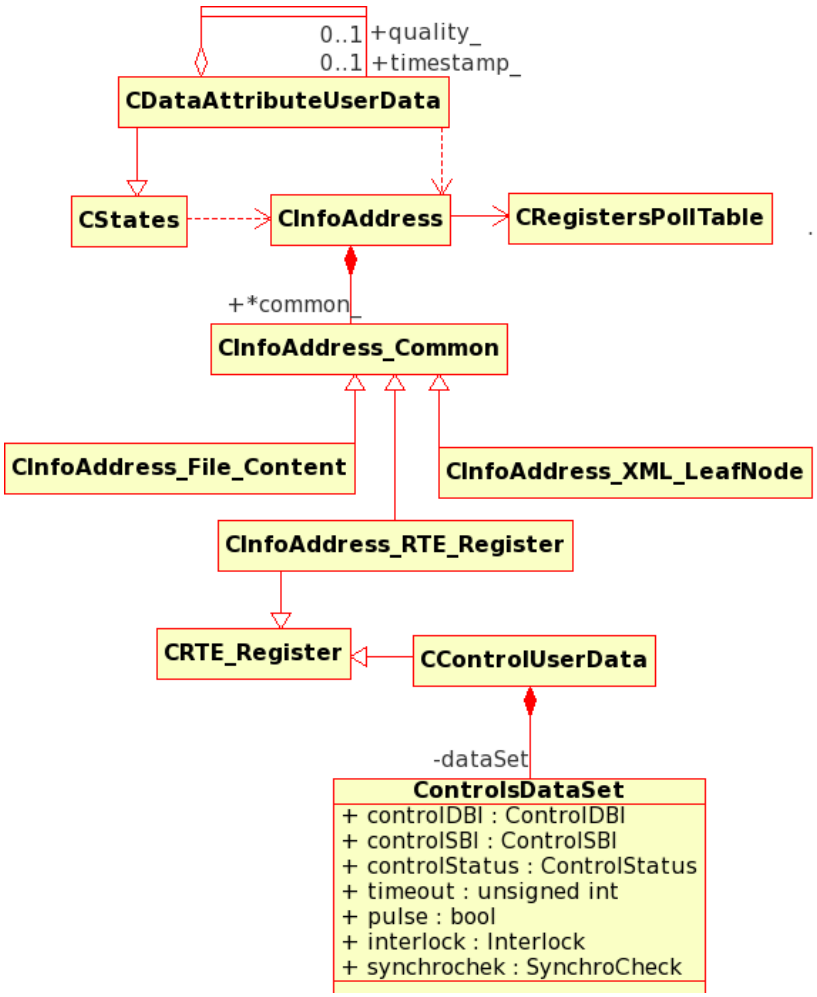
CInfoAddress je razred, ki vsebuje naslavljanje glede na vse potrebne vire informacij. Vsebuje objekt tipa *CInfoAddress_Common*, ki glede na vir podatka implementira objekt tipa *CInfoAddress_RTE_Register* za podatke, preslikane na RTDB bazo, *CInfoAddress_XML_LeafNode* za podatke, preslikane na XML dokument, ali *CInfoAddress_File_Content* za podakte, preslikane na ASCII dokument.

CStates je razred, ki hrani vse potrebne podatke o vrednosti DAI, kot so trenutna vrednost, zadnja poslana vrednost, tip podatka, velikost podatka in podatek o tem, ali se je vrednost spremenila, od kar je bila nazadnje poslana odjemalcu. Vsebuje tudi metode za kodiranje in dekodiranje podatkov, če so ti kodirani z metodami BCD, GRAY, 1N ali BitString.

CRegisterPollTable je tabela osveževanja periodičnih informacij, ki jo uporablja nit *PollHandler* in nastane v začetni fazi, ko se serializira konfiguracijska datoteka IEC61850.

CControlUserData je razred, ki izvaja operacije nad ukaznim objektom. Glavna struktura objekta je *ControlDataSet*, ki hrani informacijo o tipu

ukaza ter reference na registrsko bazo za preverjanje zaklepanja in sinhronizacije. *CControlUserData* predstavlja avtomat stanj ob izdaji zahteve za select ali operate. Objekti se ustvarijo v času serializacije v začetni fazi.



Slika 3.1: Diagram razredov

3.4 Način pridobivanja podatkov

Cilj pri implementiranju strežniške komponente IEC61850 je bil, da so informacije, s katerimi razpolaga strežnik, vseskozi ažurne in tako ob vsakokratni zahtevi po podatku s strani IEC61850 odjemalca že pripravljene za pošiljanje. Ker pa je narava naprave taka, da se nekateri podatki v napravi spreminjajo zelo pogosto, drugi pa dokaj redko, smo v sistem vpeljali različne načine osveževanja podatkov. Tako je komponenta IEC61850 sestavljena iz treh niti in nekaj vmesniških in podatkovnih struktur. Niti, ki se izvajajo v sistemu so:

- Cycle,
- PollHandler in
- EventHandler.

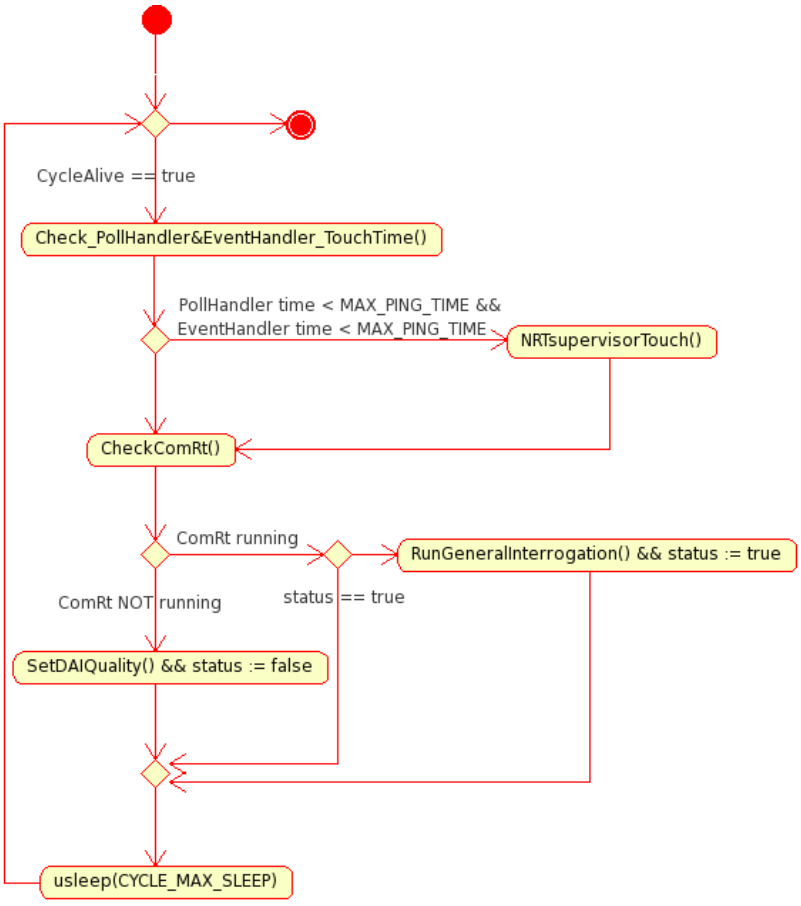
3.4.1 Nit *Cycle*

Cycle je osnovna nit komponente IEC61850 in izvaja nadzorne funkcije, diagnostiko niti *PollHandler* in *EventHandler* ter vzdržuje komunikacijo s procesom *NRTsupervisor*. Nit *Cycle* preverja, ali so instance *ComRt* procesa v realnem času, ki skrbijo za prenos informacij med podatkovno bazo *RTDB* in *PollHandler* nitjo, delujoče. Če katera od instanc *ComRt* ni v delujočem stanju, se vsem DAI, ki opisujejo kvaliteto vrednosti in so preslikane na *RTDB* podatkovno bazo, postavi bite, ki odražajo neažurnost vrednosti, vse dokler *Cycle* ne ugotovi, da so instance zopet v delujočem stanju.

Naloga niti *Cycle* je tudi preverjanje stanja niti za osveževanje vrednosti *PollHandler* in *EventHandler*. Če ugotovi nedelovanje katere koli od njih, *Cycle* preneha z odgovarjanjem procesu *NRTsupervisor*, kar bo posledično pripeljalo do ponovnega zagona strežnika, saj je delovanje le-tega brez komunikacije s procesnimi informacijami nesmiselno ali pa celo zavajajoče, česar pa si ne želimo. Delovanje niti je prikazano z UML diagramom aktivnosti na sliki 3.2.

NRTsupervisor je uporabniška aplikacija, ki teče v svojem procesu. Njena naloga je preverjanje stanja aplikacij, ki so se prijavile v njen sistem. Ob prijavi aplikacije v sistem *NRTsupervisor* določimo časovni interval preverjanja stanja posamične aplikacije in na kakšen način naj se aplikacijo restavrira, po navadi s sistemskimi klici, če ta ni več odzivna. Komunikacije med aplikacijo *NRTsupervisor* in aplikacijami, ki jih nadzira, potekajo po FIFO povezavah, in

sicer za vsako nadzorovano aplikacijo po dve FIFO povezavi in sicer za branje in pisanje.



Slika 3.2: Diagram aktivnosti niti Cycle

3.4.2 Nit *PollHandler*

Nit *PollHandler* je zadolžena za periodično prenašanje tistih informacij iz podatkovne baze *RTDB* proti sistemu, katerih vrednosti se v sistemu spreminjajo dokaj pogosto. Ko *PollHandler* prebere vrednosti iz registrske baze, se te takoj shranijo v podatkovno bazo strežniškega programa, s čimer smo zagotovili ažurnost baze v vsakem trenutku.

Nit *PollHandler* uporablja za komunikacijo s podatkovno bazo *RTDB* dva primerka *ComRt*, ki tečeta v delu naprave, ki se izvaja v realnem času, in sicer eno za branje in eno za vpisovanje vrednosti vanjo.

Vsak DAI ima v *sAddr* atributu določen način osveževanja informacije, ki določa, kako pogosto naj se vrednost osvežuje, kar je v *EBNF* zapisu notacije atributa *sAddr* zapisano kot:

- poll_class,
- refresh_only_once,
- refresh_everytime ali
- refresh_on_file_modified.

Edini način, ki zadeva *PollHandler*, je *poll_class*, ki je celo število in predstavlja število milisekund, ki naj pretečejo med vsakim ciklom osveževanja podatka iz baze *RTDB*. Ostali trije načini osveževanja so uporabljeni zgolj za osveževanje podatkov, ki se pridobivajo iz XML ali ASCII datotek.

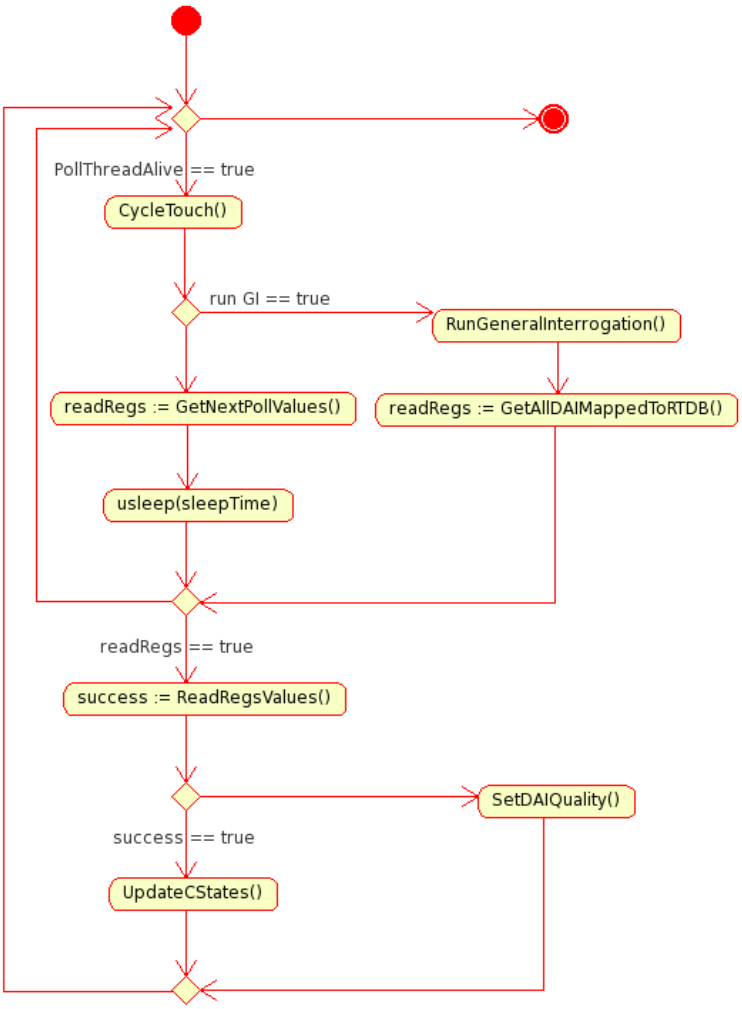
Sistem osveževanja smo realizirali tako, da smo podatke razdelili po skupinah, in sicer na način, da so v isti skupini podatki, ki imajo enak interval osveževanja. Če je zahteva za branje vrednosti iz baze *RTDB* neuspešna, se podobno kot pri niti *Cycle* postavi bite tistim DAI, ki opisujejo kvaliteto, katerih vrednosti so se v tistem ciklu osveževale. Biti so postavljeni vse dokler *PollHandler* zopet uspešno ne prebere vrednosti iz baze *RTDB*. Delovanje niti je predstavljeno z UML diagramom aktivnosti na sliki 3.3.

3.4.3 Nit *EventHandler*

Nit *EventHandler* je prav tako zadolžena za prenos podatkov iz baze *RTDB*, vendar za tipe podatkov, katerih vrednosti se spreminjajo dokaj redko.

Komunikacija niti *EventHandler* z instanco module *Event*, ki teče v delu naprave, ki se izvaja v realnem času, poteka po FIFO povezavi. Naloga modula *Event* je zaznavanje sprememb vrednosti registrov v podatkovni bazi RTDB glede na tipe dogodkov, ki so lahko

- enobitni,
- dvobitni ali
- analogni dogodki.



Slika 3.3: Diagram aktivnosti PollHandler niti

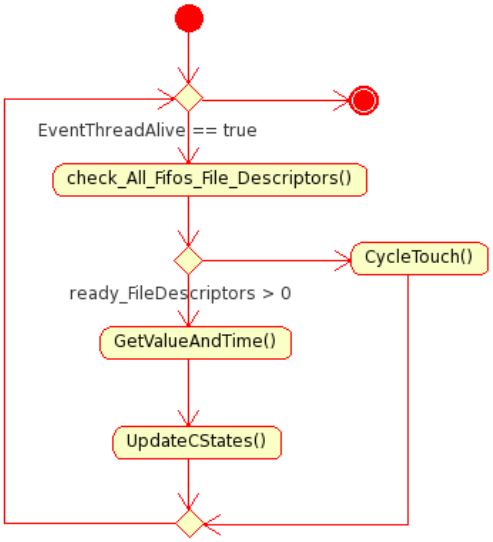
Kot že samo ime pove, je vrednost enobitnega dogodka lahko ena ali nič, spremembo vrednosti pa Event pošlje glede na stanje parametera, s katerim določimo, ali se dogodek pošlje ob spremembi iz ena na nič, iz nič na ena ali pa ne glede na spremembo.

Dvobitni dogodek mora biti povezan z dvema registroma. Vrednost posameznega registra je prav tako lahko le ena ali nič, spremebe pa se pošiljajo ne glede na smer spremembe katerega koli registra.

Analogni dogodek odraža vrednosti registrov analognih vrednosti. Ker se te spreminjajo pogosto in so lahko spremembe zelo minimalne glede na velikost

vrednosti, imajo analogni dogodki parameter, s katerim določimo, kolikšna mora biti sprememba vrednost registra, da se ga obravnava kot dogodek. Če parametra ne bi upoštevali, bi lahko velika frekvenca majhnih sprememb preobremenila odjemalca dogodkov.

Vsak od dogodkov nosi tudi informacijo o času spremembe vrednosti. Zapisan je v obliki sekund in milisekund, ki so pretekle od datuma 01. 01. 1970 naprej, vrednost pa se zapiše v DAI, ki predstavlja čas zadnje spremembe DAI in je povezan s procesno informacijo. Z UML diagramom aktivnosti je na sliki 3.4 prikazano delovanje niti *EventHandler*.



Slika 3.4: Diagram aktivnosti EventHandler niti

3.4.4 Branje iz datotek

Ker vsi podatki, ki se prenašajo po komunikaciji IEC61850, nimajo numerične vrednosti, jih ni moč hraniti v registrski bazi. Tako so nekateri podatki shranjeni v prej omenjenih tekstovnih datotekah različnih tipov, ki so lahko:

- datoteke XML ali
- datoteke ASCII.

Če hočemo vrednost DAI pridobiti iz tekstovnih datotek, je potrebno v atributu *sAddr* določiti, v kateri datoteki in na katerem mestu v datoteki se

ta podatek nahaja in na kakšen način hočemo vrednost osveževati. Ker se vsebina datotek med delovanjem naprave po navadi ne spreminja, se vrednosti najpogosteje preberejo le ob zagonu strežnika IEC61850.

3.5 Obdelava zahtev s strani IEC61850 odjemalca

IEC61850 odjemalec lahko izvaja različne zahteve proti strežniku IEC61850. Zahteve, na katere se bomo omejili tu, so zahteve za nadziranje vrednosti podatka (Monitoring) in zahteve za upravljanje vrednosti podatka (Controlling). TMW SCL ima že izdelan sistem, s katerim obvladuje celoten komunikacijski protokol, zato se nam v trežniškem programu ni bilo treba ukvarjati s samo komunikacijo do odjemalca. Potrebno je bilo le zagotoviti povezavo do vrednosti preslikanih podatkovnih atributov. V ta namen smo implementirali dve funkcionalnosti, s katerima smo zadostili zahtevam TMW SCL po upravljanju in nadziranju vrednosti DAI.

3.5.1 Zahteve za nadzor s strani odjemalca IEC61850

Zahteve za nadzor v komunikacijskem protokolu IEC61850 so zahteve, ki prihajajo s strani IEC61850 odjemalca. Ta pošilja zahteve za branje vrednosti posameznega podatka ali sklopa podatkov. Ko je zahteva poslana, mora strežnik v določenem času odgovoriti z vrednostjo podatka ali zahtevo zavrniti z negativnim odgovorom. V ta namen smo implementirali funkcijo *ReadHandler*.

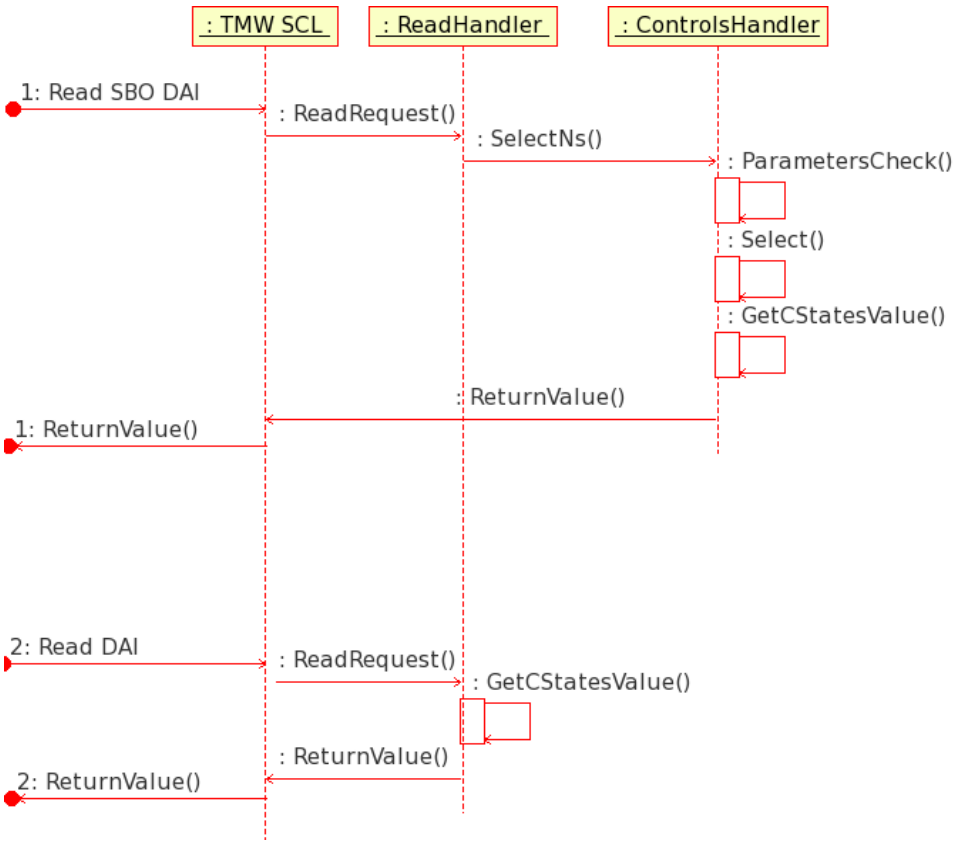
3.5.1.1 ReadHandler

Ko odjemalec pošlje zahtevo za nadzor nad posameznim podatkom ali sklopa podatkovnih atributov, se za vsak DAI podatek pokliče funkcija *ReadHandler*, ki smo jo v TMW SCL-u registrirali kot funkcijo za branje vrednosti za vse DAI, ki so pravilno preslikani na procesne spremenljivke. Ker nam TMW SCL kot glavni argument funkcije poda kazalec na objekt *CDataAttributeUserData*, ki je bil ustvarjen v času serializacije, in ker je objekt del podatkovne baze strežnika, mu za DAI, ki so mapirani na *RTDB* podatkovno bazo, le vrnemo vrednost, ki je v tistem trenutku shranjena v podatkovni bazi strežnika, saj je zaradi *PollHandler* in *EventHandler* funkcionalnosti vrednost v objektu vseskozi ažurna. Kar pa ne velja za DAI, ki so mapirani na tekstovne datoteke.

V tem primeru je potrebno glede na način osveževanja, določenega v *sAddr*, vrednost prebrati iz datoteke.

Funkcijo *ReadHandler* pa se izrablja tudi za izvajanje operacij nad ukaznimi objekti, saj se nekatere faze ukazov izvajajo prav z zahtevami za branje vrednosti podatkovnega atributa, zato v teh primerih, preden vrednost DAI vrnemo TMW SCL, izvedemo operacije nad ukaznim objektom. Te so opisane v naslednjih poglavjih.

S sliko 3.5 so predstavljeni dve zahtevi s strani odjemalca. Kot prva je predstavljena zahteva s strani odjemalca nad ukaznim objektom. Druga zahteva predstavlja branje vrednosti DAI, ki ni ukazni objekt. Vidimo, da se ob prvi zahtevi, preden se vrne vrednost podatka, preveri parametre ukaznega objekta in izvede faza izbora objekta. V drugi zahtevi pa vrnemo le vrednost podatka, nad katerim je bila izdana zahteva za nadzor.



Slika 3.5: Zahteva za nadzor s strani odjemalca

3.5.2 Zahteve za upravljanje s strani odjemalca IEC61850

Zahteve za upravljanje se izvajajo s strani odjemalca IEC61850 in se uporabljajo za upravljanje IED-ja. V komunikacijskem protokolu IEC61850 imajo le določeni podatkovni atributi možnost upravljanja, za razliko od nadziranja, ki ga je mogoče izvajati nad vsemi podatkovnimi atributi.

3.5.2.1 WriteHandler

Podobno kot *ReadHandler* je tudi *WriteHandler* funkcija, namenjena obvladovanju zahtev s strani odjemalca IEC61850, ki nam jih posreduje TMW SCL. Razlika je v tem, da se *WriteHandler* kliče v primerih, ko želi odjemalec spremeniti vrednost DAI. Tudi v tem primeru nam TMW SCL kot glavni argument funkcije poda kazalec na objekt tipa *CDataAttributeUserData*, poleg tega pa nam posreduje še novo vrednost DAI, ki jo je potrebno spremeniti tako v podatkovni bazi strežnika preko podanega objekta, kot tudi v podatkovni bazi RTDB, če je DAI preslikan nanjo, ali pa v tekstovnih datotekah.

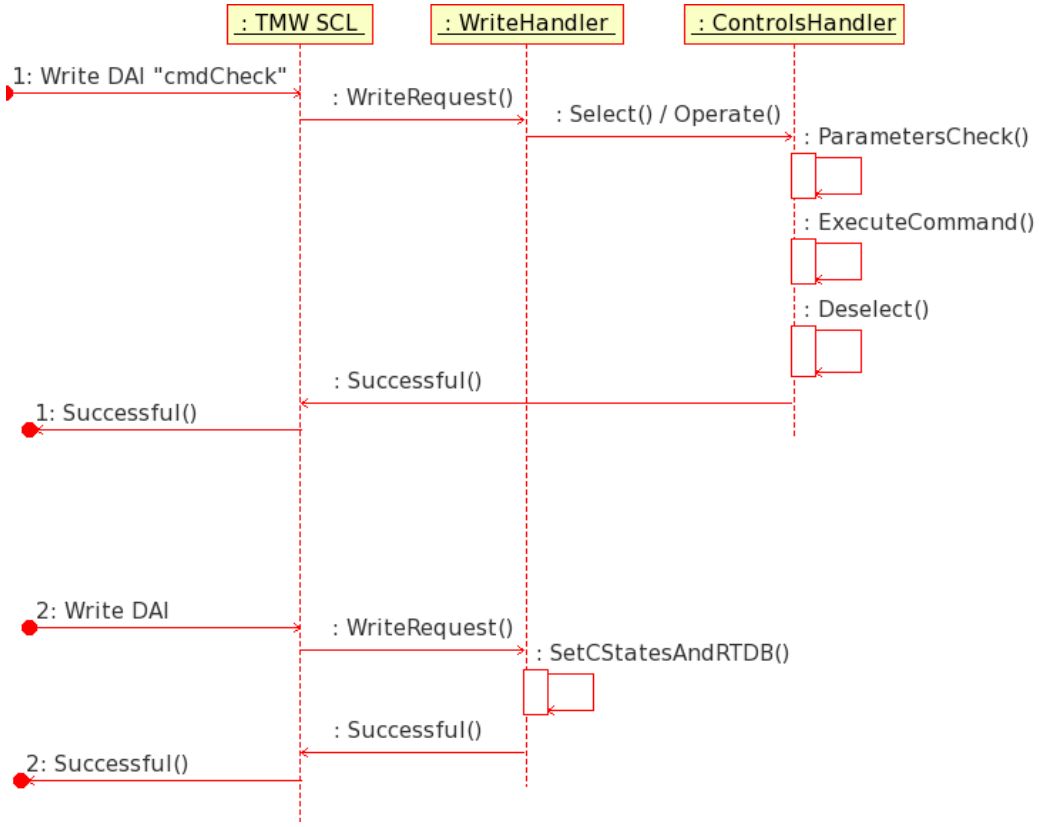
Tudi *WriteHandler* se uporablja za operacije nad ukaznimi objekti, saj se večina teh izvaja z vpisovanjem vrednosti v DAI, ki so sestavni del njih.

S sliko 3.6 je predstavljena zahteva za spreminjanje vrednosti nad DAI, ki je sestavni del ukaznega objekta kot prva zahteva in zahteva za spreminjanje vrednosti DAI, ki ni del ukaznega objekta kot druga zahteva. Vidimo, da se ob prvi zahtevi najprej preveri parametre, izvede ukaz nad objektom in nato umakne izbor ukaznega objekta. Pri drugi zahtevi pa le spremenimo vrednost v podatkovni bazi strežnika in bazi RTDB.

3.5.3 Izvajanje operacij nad ukaznim objektom

TMW SCL sam ne izvaja ukaznega procesa, posreduje le parametre ukaznega objekta, zato smo morali sami analizirati in implementirati delovanje vseh faz ob izvajanju ukaznih operacij. To je bila ena izmed zahtevnejših nalog celotnega implementiranja strežniškega programa, saj je bilo potrebno predhodno podrobno preučiti, kako naj bi se ukazi izvajali nad objekti. Standard IEC61850 natančno določa potek in vrednosti parametrov v posamezni fazi ter katere od objektov je mogoče specializirati z razširjanjem skupnih podatkovnih razredov v ukazne objekte.

Vsak od ukaznih objektov je sestavljen iz IEC61850 servisnih, statusnih in konfiguracijskih parametrov, navedenimi v Tabeli 3.1, ki so posredovani ob



Slika 3.6: Zahteva za upravljanje s strani odjemalca

izvajanju ukaza. Preslikava objekta je opravljena s kombiniranjem parametrov ter ukaznih elementov v MMS strukturo, ki je komponenta logičnega vozlišča.

3.5.3.1 Preverjanje pravilnosti izvajanja ukaza

Opracije nad ukaznim objektom se delijo na tri faze, in sicer na faze *Select*, *Operate* in *Cancel*.

Faza *Select* je namenjena temu, da inženir, če ima namen izvajati operacije nad ukaznim objektom, tega najprej izbere z fazo *Select* in s tem prepreči, da bi istočasno kdo drug nad istim objektom lahko izvedel ukaz. *Select* faze se razlikujejo glede na stopnjo varnosti, ki je določena s konfiguracijskim parametrom za posamezen ukazni objekt, v glavnem pa se razlikujejo po tem, ali mora v fazi *Select* inženir že določiti, s kakšnimi parametri želi izvesti ukaz ali pa zgolj rezervirati objekt, ne da bi predhodno določil parametre.

Ko inženir uspešno izvede fazo *Select*, lahko izvede fazo *Operate*, v kateri

PARAMETRI UKAZNEGA OBJEKTA	
Servisni parametri	
operTm	absolutni čas izvedbe ukaza
ctlNum	sekvenčna številka ukaza
T	čas zadnje spremembe statusa objekta
Test	status, ali je ukaz testni
Check	preverjanje zaklepanja in/ali sinhronizacije
origin	ozančuje klienta, ki je sprožil ukaz
Statusni parametri	
stVal	status ukaznega objekta
stSeld	stanje selektiranosti objekta
t	čas zadnje spremembe statusa objekta
Konfiguracijski parametri	
sboTimeout	čas selektiranosti objekta
sboClass	določa način izvedbe Operate faze
ctlModel	določa stopnjo varnosti

Tabela 3.1: Servisni, statusni in konfiguracisjski parametri ukaznih objektov

določi, kateri ukaz želi izvesti nad ukaznim objektom, vendar mora za uspešno izvedbo pravilno nastaviti parametre, ki morajo biti identični kot v fazi *Select*, Če je tako določeno z varnostnimi nastavitvami.

Če inženir izvede fazo *Select* in s tem izbere objekt, jo lahko prekliče z izvedbo faze *Cancel* ali pa bo preklicana samodejno po preteku časa, ki določa maksimalen čas izbranosti objekta.

Med samim izvajanjem operacij nad ukaznim objektom je poglavitnega pomena, da so vsi parametri, servisni, statusni in konfiguracijski, v stanju, ki dovoljuje izvedbo faze.

Če katera od zgoraj opisanih faz ni uspešno izvedena, mora strežnik odjemalcu odgovoriti s poročilom, kjer je s kodami opisan vzrok neuspešnosti izvedbe. Zato smo morali analizirati vse možne scenarije, ki jih lahko izvede inženir preko odjemalca nad ukaznim objektom, in za vsakega izmed njih, ki ni dovoljen, mora TMW SCL odgovoriti s kodo, ki opisuje vzrok neuspšne izvedbe, ki jo bo nato odjemalec interpretiral v inženirju razumljivo vsebino. Ker pa so določeni scenariji nemogoči zaradi narave naprave, v katero smo integrirali IEC61850 komunikacijski protokol, do nekaterih nedovoljenih stanj ne more priti, zato so v Tabeli 3.2 opisane zgolj kode implementiranih stanj.

KODA NEUSPEŠNOSTI	VZROK
ServiceError type	nedovoljen izvajalec ukaza
Blocked-by-switching-hierarchy	naprava v stanju Local
Select-failed	izbira neuspešna
Position-reached	objekt že v željenem stanju
Parameter-change-in-execution	parameter v Operate ni enak Select fazi
Blocked-by-mode	naprava v stanje Off ali Blocked
Blocked-by-interlocking	zaklepanje neuspešno
Blocked-by-synchrocheck	sinhronizacija neuspešna
Command-already-in-execution	objekt že izbran

Tabela 3.2: Možni vzroki za neuspešno izvedbo ukaza

3.5.3.2 Preslikava ukaznega objekta

Pri izvajanju ukazov nad ukaznim objektom smo morali zaradi specifike naprave pri povezovanju statusov izvedbe ukaza uporabiti konfiguracijski element *Private*. Namenjen je temu, da se ga uporablja za konfiguracijo podatkov, ki so specifični glede na proizvajalca naprave. Prav zaradi preverjanja zaklepanja in sinhronizacije ukaznega objekta, smo ga bili primorani uporabiti. Z njim smo določili vse parametre, ki smo jih potrebovali za izvedbo in pa preverjanje statusov izvajanja zaklepanja in sinhronizacije. Prav tako kot pri atributu *sAddr* je vsebina elementa *Private* določena z EBNF notacijo. Sintakso za mapiranje na podatkovno bazo RTDB smo ohranili enako kot pri atributu sAddr, saj smo imeli funkcionalnosti za razčlenjevanje vrednosti razvite tako v strežniškem programu kot v grafičnem uporabniškem vmesniku za preslikavo komunikacijskih elementov IEC61850 na procesne spremenljivke.

3.5.3.3 Preverjanje zaklepanja in sinhronizacije

Preverjanje zaklepanja in sinhronizacije se lahko izvaja pri izdajanju ukazov nad ukaznim objektom, in sicer v fazah *Select* in *Operate*. Ker mora biti preverjanje izvedeno v realnem času, so funkcionalnosti izvedene z moduli, ki delujejo v jedru operacijskega sistema. Zato je bila naloga strežniškega programa ob izdanem ukazu postaviti vhode modulom in preveriti stanje izhodov, ki jih moduli nastavljajo ob preverjanju zaklepanja in sinhronizacije. Uspešnost izdanega ukaza je tako odvisna od pravilne nastavitve servisnih, konfiguracijskih in statusnih parametrov ukaznega objekta ter stanja izhodnih podatkov modulov, ki preverjajo stanje zaklepanja in sinhronizacije.

Poglavje 4

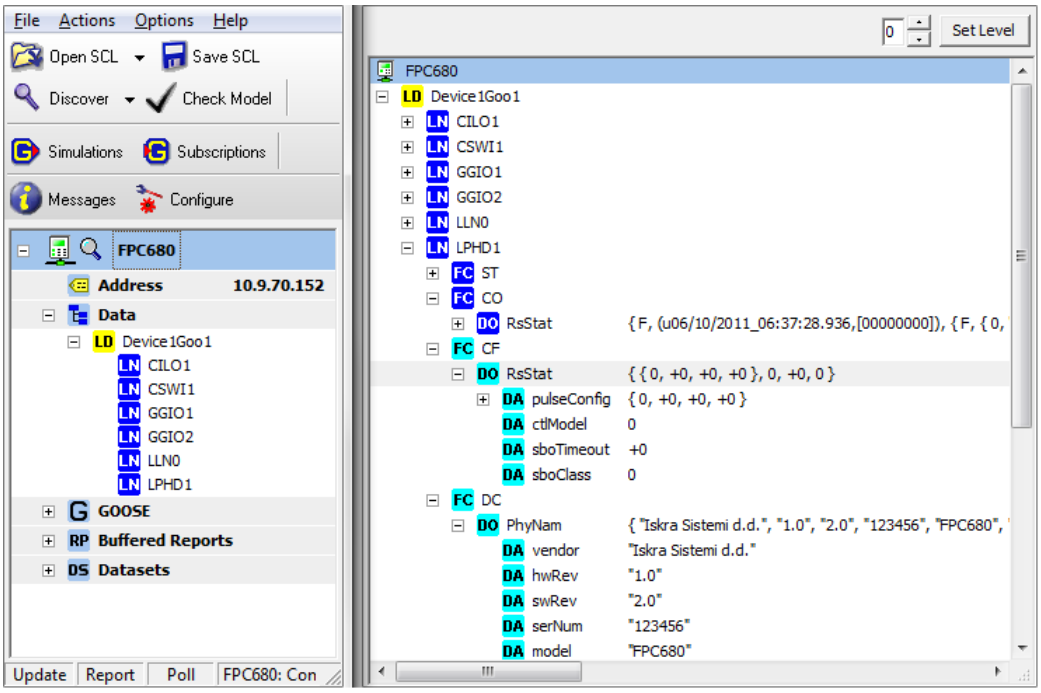
Uporaba v praksi

Za komunikacijski strežnik IEC61850, ki smo ga razvili, smo pridobili certifikat s strani mednarodne organizacije KEMA, ki je opravila teste o skladnosti delovanja strežnika s standardom IEC61850. Strežnik se trenutno uporablja na napravah *FPC* (*Feeder Protection and Control*) (Slika 4.1) podjetja Iskra Sistemi, d. d. Je celovita naprava za zaščito, nadzor, merjenje in upravljanje distribucijskih in industrijskih elektroenergetskih omrežij in je namenjena ščitenju srednje napetostnih vodov, lahko pa se uporablja tudi kot rezervna zaščita pri transformatorjih in visoko napetostnih vodih.

Nadzor voda se upravlja na daljavo preko postajnega nadzornega sistema *SCADA* (*Supervisory Control And Data Acquisition*) ali nadzornega centra vodenja, kjer komunikacija poteka po IEC61850 komunikacijskem protokolu ali preko lokalne prikazovalne enote *LDU* (Local Display Unit). Inženir lahko tako na daljavo preko odjemalca IEC61850, katerega primerek je prikazan na sliki 4.2, nadzoruje in upravlja stanje naprave tako, da določi IP naslov strežnika IEC61850 ter se z njim poveže. Strežnik nato na zahtevo odjemalca odgovori s tistimi podatki, katere smo s konfiguracijsko datoteko strežnika preslikali na procesne spremenljivke.

S tem, ko smo v sistem vpeljali nov komunikacijski protokol IEC61850, smo naredili velik korak naprej, saj trend kaže na to, da bo uporaba le tega sčasoma zmenjala vse obstoječe komunikacijske protokole kot so IEC60870-5-101, IEC60870-5-103, IEC60870-5-104, DNP3 in ostale. Z novim protokolom smo dosegli, da so podatki iz naprav vidni vsem aplikacijam v omerežju, da so se poenotila poimenovanja podatkov o stanju naprav ter konfiguracijske datoteke, ki jih opisujejo, da je možna komunikacija med napravami samimi na horizontalni ravni. Predvsem pa smo pridobili na hitrosti prenašanja podatkov, saj se sporočila po mrežni povezavi, za katero se danes uporabljajo optične

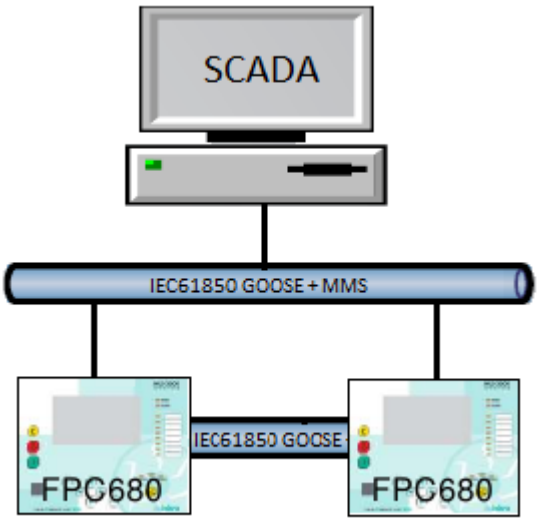
Slika 4.1: FPC680



Slika 4.2: Primer grafičnega uporabniškega vmesnika IEC61850 odjemalca

povezave, prenašajo mnogo hitreje, kot pri nekaterih ostalih protokolih, ki za prenos uporabljajo serijsko komunikacijo.

S Sliko 4.3 je ponazorjena komunikacija med nadzornim sistemom SCADA, ki uporablja odjemalec IEC61850, in napravami FPC680, ki imajo nameščen strežnik IEC61850. Na horizontalni ravni naprave med seboj komunicirajo s telegrami IEC61850 GOOSE, kar je velik doprinos h komunikaciji po protokolu IEC60870-5-103, ki se je uporabljal pred IEC61850. Na vertikalni ravni se poleg telegramov GOOSE prenašajo še telegrami MMS, in sicer za nadzor in



Slika 4.3: Komunikacija med nadzornim sistemom SCADA in napravami FPC680

upravljanje naprave.

Poglavje 5

Sklepne ugotovitve

V diplomi smo opisali razvojne iteracije strežniškega programa za komunikacijski strežnik IEC61850. Ker je standard IEC61850 obsežen, je bilo potrebno veliko truda vložiti prav v razumevanje vseh postopkov in struktur komunikacijskega protokola, kar je zelo pomembno pri načrtovanju arhitekture strežniškega programa, saj se je le tako moč izogniti kasnejšim težavam pri sami implementaciji in odpravi nepravilnosti.

Največ težav nam je povzročalo odpravljanje nepravilnosti, za katere se je izkazalo, da niso plod naše, ampak implementacije TMW SCL, saj je bilo iskanje napak v njihovi kodi zaradi skromne razvojne dokumentacije in dokaj kompleksne kode zahtevno in dolgotrajno. Tu se je izkazalo, kako pomembno vlogo pri implementiranju programskih rešitev ima razvojna dokumentacija, saj je le-ta obvezna za odpravljanje napak, ki se pojavijo takrat, ko stvari niso več tako sveže v glavah razvojnikov kot tudi takrat, ko se programska oprema še razvija. Zato smo večji poudarek dali prav dokumentiranju opravljenega dela, četudi to ni priljubljeno delo večine razvojnikov.

Nekaj težav se je med implementiranjem pojavilo tudi zaradi neizkušenosti v programskih jezikih C in C++, saj smo nevede zanemarjali problem sprostitve pomnilniških resursov, odkrivanje tovrstnih napak pa je zaradi neočitnih krivcev nemalokrat težavno. Pomembno vlogo pri razvoju ima tudi dobro načrtovano testiranje izdelka. Tu se je izkazalo, da smo tudi ta del naloge opravili zelo dobro , kar se je pokazalo pri testiranju s strani mednarodne organizacije KEMA, ki je strežnik preizkusila z skorajda vsemi možnimi vhodnimi podatki. Seveda so se pokazale nekatere pomanjkljivosti, ki pa smo jih relativno hitro odpravili, zahvaljujoč se dobri zasnovi problema in zelo strukturirani implementaciji, ki omogoča hitro odkrivanje in odpravo napak. Pomemben podatek, ki kaže na to je, da smo eni izmed redkih v svetovnem

merilu, ki smo certifikat o skaldnosti delovanja stežnika po IEC61850 standaru pridobili v prvi iteraciji testiranja.

Ker smo veliko truda vložili v razumevanje standarda IEC61850, bi bilo za nadaljnje delo mogoče dobro razmisliti o razvoju lastnega odjemalca IEC61850 in se s tem izogniti odvisnosti od tujih programskih rešitev ter posledično plačevanju licenc za uporabo le-teh. Menim, da bi zaradi znanja, ki smo ga pridobili pri implementiranju strežniškega programa, lastno rešitev za IEC61850 odjemalca relativno hitro izvedli.

Slike

1.1 Komunikacijska infrastruktura 4
1.2 TMW SCL IEC61850 strežnik z strežniškim programom v IED napravi 5

2.1 IEC61850 komunikacijske plasti 8
2.2 Logične skupine (naprava, vozlišča, razredi in podatki) 11

3.1 Diagram razredov 18
3.2 Diagram aktivnosti niti Cycle 20
3.3 Diagram aktivnosti PollHandler niti 22
3.4 Diagram aktivnosti EventHandler niti 23
3.5 Zahteva za nadzor s strani odjemalca 25
3.6 Zahteva za upravljanje s strani odjemalca 27

4.1 FPC680 31
4.2 Primer grafičnega uporabniškega vmesnika IEC61850 odjemalca 31
4.3 Komunikacija med nadzornim sistemom SCADA in napravami FPC680 32

Tabele

2.1 Skupine logičnih vozlišč 9
2.2 Logična vozlišča iz skupine avtomatskega nadzora 10
2.3 Kategorije podatkovnih razredov 10

3.1 Servisni, statusni in konfiguracisjski parametri ukaznih objektov 28
3.2 Možni vzroki za neuspešno izvedbo ukaza 29

Literatura

[1] INTERNATIONAL STANDARD IEC 61850
Communication networks and systems for power utility automation
Part 6: Configuration description language for communication in electrical substations related to IEDs
Part 7-2: Basic information and communication structure – Abstract communication service interface (ACSI)
Part 7-3: Basic communication structure – Common data classes
Part 7-4: Basic communication structure – Compatible logical node classes and data object classes
Part 8-1: Specific Communication Service Mapping (SCSM) – Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3

[2] MMS
Dostopno na: http://en.wikipedia.org/wiki/Manufacturing_Message_Specification
http://www.nettedautomation.com/standardization/ISO/TC184/SC5/WG2/mms_intro/

[3] EBNF
Dostopno na: http://en.wikipedia.org/wiki/Extended_Backus%E2%80%93Naur_Form

[4] C++
Stephen Prata, C++ Primer Plus, 5th edition, Sams, 2005

[5] UML
Bennet, Simon McRobb, Steve Farmer, Ray, Object-oriented system analysis and design : using UML, 2010

[6] ISO/OSI referenčni model
Dostopno na: http://sl.wikipedia.org/wiki/ISO/OSI_referenčni_model