Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.4
DOI: https://doi.org/10.24867/07BE26Jeftic

РАЗВОЈ БИБЛИОТЕКЕ ЗА ИНДУСТРИЈСКИ КОМУНИКАЦИОНИ ПРОТОКОЛ *SERIES V*

DEVELOPMENT OF THE LIBRARY FOR THE INDUSTRIAL COMMUNICATION PROTOCOL SERIES V

Филип Јефтић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом раду представљено је једно рјешење програмске подршке за управљање Slave страном у SCADA системима употребом Series V протокола. Поред описаног концепта рјешења, у раду су описане основе SCADA система, комуникационих и индустријских протокола као и самог Series V протокола.*

Кључне ријечи: SCADA, RTU, SERIES V

Abstract – *This paper presents a software solution for Slave side managing in SCADA systems using Series V protocol. In additional to concept of the solution, there are described theoretical concepts of SCADA systems, communication and industrial protocols as same as the Series V protocol.*

Keyords: SCADA, RTU, SERIES V

1. УВОД

Данас су људи, индустрија и велике корпорације постали зависни од електричне енергије, то јесте од њеног континуалног напајања. И најмањи испади у електро системима могу довести до штете која се мјери милионима евра. С тога је основни и главни циљ електро инжењера да учине тај систем што робуснијим, поузданијим и сигурнијим. Међутим, ту се крије велика одговорност и обавеза, јер је у питању огромна количина података које је потребно обрадити у реалном времену. Ријеч је о десетинама милиона тачака. Данас се слободно може рећи да податак представља главни и најважнији ресурс и с тога је неопходно правилно руковати са њим. Тако долазимо до термина *SCADA* (*Supervisory Control And Data Acquisition*) система и индустријских комуникационих протокола.

Идеја овог рада је приједлог рјешења за имплементацију програмске библиотеке на примјеру *Series V* индустријског комуникационог протокола. Циљ је да се одради и детаљно опише сам поступак имплементације библиотеке. Поред описане имплементације и концепта рјешења, у раду су описане теоријске основе *SCADA* система, комуникационих и индустријских протокола.

НАПОМЕНА:

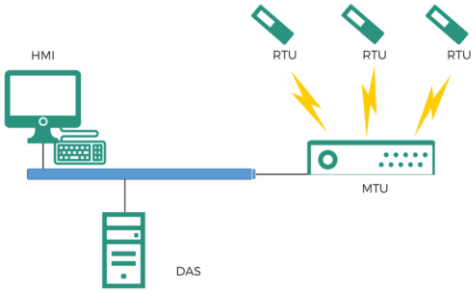
Овај рад проистекао је из мастер рада чији ментор је био др Бранислав Атлагић, доцент.

2. ТЕОРИЈСКЕ ОСНОВЕ

2.1 *SCADA* систем

Аквизиционо управљачки системи (*АУС*) су базирани на употреби рачунара и дигиталних рачунарских компоненти. Њихов основни циљ је обезбјеђење ефикасног надзора и управљања над произвољним физичким процесом. Овакви системи, специфичне намјене и структуре, у литератури се најчешће означавају термином *SCADA* (*Supervisory Control And Data Acquisition*). Фундаментални захтјеви које аквизиционо управљачки систем опште намјене мора испунити су:

- Рад у реалном времену,
- Дистрибуција рачунарских ресурса у оквиру аквизиционо управљачког система,
- Постизање максималне поузданости и расположивости.



Слика 2.1.1 Архитектура SCADA мреже

MTU (Master Terminal Units) или водећа процесна јединица у *SCADA* систему представља уређај који издаје команде над *RTU* уређајима, прикупља захтјеване податке, складишти информације, процесира информације и приказује их у форми слика, графова и табела на корисничком приказном уређају (*HMI*) и на тај начин помаже да се лакше доносе одлуке (слика 2.1.1). Ово је задатак *MTU-а* лоцираног у контролном центру. *DAS (Direct-attached storage)* представља дигитално складиште података прикључено на рачунар који му приступа [2].

Циљеви које треба остварити у току пројектовања и извођења аквизиционо управљачких система су сљедећи [1]:

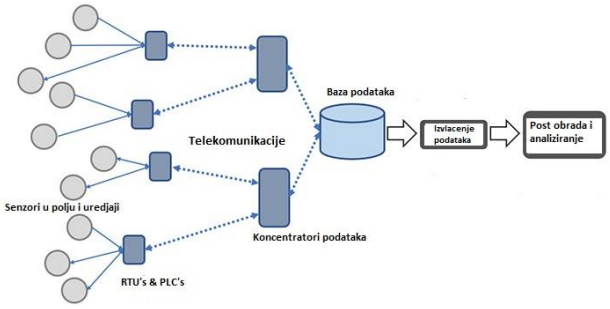
- *Смањење трошкова производње.* Оно се остварује на два начина. Први је везан за мањи обим потребне радне снаге, смањење трошкова превоза, повећање квалитета производње и сл. Други аспект се огледа у повећаном степену сигурности извршења технолошких процеса који

често могу бити екстремно опасни по околину или животе људи.

- *Расположивост и интегритет система*. АУС мора обезбиједити наставак рада и у случају отказа појединих компоненти. Интегритет (конзистентност) података и укупно функционисање не смије бити угрожено.
- *Флексибилност и проширивост система*. У току употребе аквизиционо управљачког система често су потребне измјене његове конфигурације. Оне су најчешће условљене промјенама у производном процесу који се надзире.
- *Поузданост система*. Поузданост АУС система условљена је поузданошћу његових саставних компоненти.
- *Перформанса система*. Генерално, перформанса се дефинише помоћу времена одзива и пропусности. Вријеме одзива се може смањити нарочито уколико се главнина обраде врши локално, без потребе за чекање управљачког сигнала од централне станице. *Пропусност* се односи на обим података који се могу пренијети и обрадити у оквиру аквизиционо управљачког система.

2.1.1 *Polling* у *SCADA* системима

Када причамо о *SCADA* системима, често се провлачи појам *Polling-a* (*слика 2.1.1*). Он представља процес којим *SCADA* прикупља податке. Порука се пошаље до *RTU* уређаја и чека се на одговор. Тај уређај посједује везе које га вежу за физички свијет, у којем се врше мјерења напона, струје и фазе. Одговор који се шаље назад су специфичне вриједности захтјеване преко *polling-a* [6].



Слика 2.1.1.1 Polling података

Poll захтјев може да садржи генерални захтјев као на примјер читање вриједности напона, или једноставно може бити фокусиран на специфична мјерења. Један примјер захтјева може бити рецимо прибављање температуре трансформатора. Тај захтјев се шаље када рецимо желимо да знамо да ли се одређени трансформатор прегријао [6].

Овај процес *polling-a* се може вршити свако пар секунди, минута, сати, дана, мјесеци или година. Процес је зависан од типа опреме, њене улоге у систему и ризика који та опрема носи са собом [6].

Међутим, увијек постоји могућност појаве непредвиђене ситуације, што је уједно и битна карактеристика критичних система. На примјер, постоји *polling* процес који се извршава свако 30 минута, а рецимо у 15 минуту се десио неки испад, односно неко прекорачење. Тај дио опреме који је измјерио прекорачење ће да обавијести *RTU* уређај који ће затим да изда захтјев за *poll* који се зове *unsolicited poll response* (више о овоме у поглављу *2.3.1*). Овај захтјев ће се затим послати кроз *SCADA* систем, који ће препознати о којем типу захтјева је ријеч. Ово је јако чест процес [6].

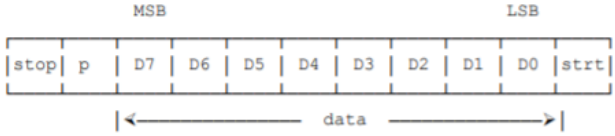
3. *SERIES V* ПРОТОКОЛ

Series V SCADA Communication protocol је асинхрони *byte* оријентисани протокол чији је примарни циљ командовање и аквизиција над *RTU* уређајима у пољу, користећи исти комуникациони канал. Дизајниран је тако да се повеже директно на рачунарске комуникационе портове. Може се користити у *point-to-point* или *multi-drop* конфигурацији. Комуникација између двије компоненте се може остварити у *half* или *full-duplex* варијанти [4][5].

Сваки вид комуникације је иницран од стране водеће јединице (*MTU*). Удаљена процесна јединица (*slave* страна) не може да иницира никакав вид размјене информација са водећом јединицом, нити може да директно адресира или комуницира са другом удаљеном процесном јединицом. Она ће да врати одговарајући испроцесирани одговор за сваку валидну поруку која је стигла и која је адресирана на дату процесну јединицу. Ово је небалансирани режим комуникације, о коме је било више ријечи у претходном поглављу (2.3.1). Једини изузетак за ово је тип поруке која се шаље свакој станици (*broadcast message*) и у том случају ниједна удаљена јединица не враћа одговор. Приликом сваког пријема поруке се врши провјера исправности послатих података. За то се користе два уобичајена и на широко коришћена сигурносна механизма: *CRC* и *LRC*. Уколико неки од *byte*-ова није валидан, удаљена процесна јединица ће игнорисати послату поруку и никакав вид акције неће бити подузет [5].

3.1 Структура карактера

Подаци се преносе у стандардном 10-обитном или 11-обитном асинхроном бајт оријентисаном формату. Сваки од бајтова се састоји од почетног бита (*start bit*), 8 бита који представљају саме податке, опциони бит за паритет и крајњег бита (*stop bit*) [4].



Слика 3.1.1 Структура једног бајта поруке

Кратак опис садржаја једног бајта: [4]

- *strt* - почетни бит, који означава почетак бајта,
- *stop* - крајњи бит, који означава крај бајта,
- *D0-D7* - подаци, гдје је *D0* најмање значајан бит (*LSB*), а *D7* најзначајнији бит (*MSB*).
- *p* - бит паритета

3.3 *Series V* поруке

У суштини, све поруке унутар *Series V* индустријског протокола се могу подијелити у два основна типа: захтјеви за подацима и контролни захтјеви. Први тип захтјева податке у виду вриједности са *RTU* уређаја. Ови подаци могу бити дискренти, аналогни, акумулатор, израчунате варијабле, параметри удаљене једи-

нице, статус *RTU* уређаја, аналогни излази и дискретни излази.
Други тип захтјева је дефинисан као било која порука упућена од стране водеће процесне јединице која захтјева промјену стања неког од уређаја у пољу или модификацију интерног стања удаљене процесне јединице [5].

3.3.1 Структура поруке

Ради лакоће приказа, почетни, крајњи и бит паритета ће бити изостављени у наредним илустрацијама. За више информација о њима, погледајте поглавље 3.1.
Такође, у наставку је приказана структура порука са *LRC* сигурносним механизмом, што значи да је величина поруке 5 бајта.
Све поруке које се размјењују између *MTU* и *RTU* имају неку од основних структура наведених у наставку [4].

3.3.1.1 *Master-to-Remote* порука

Структура поруке је илустрована на слици 3.3.1.1.1.

BYTE	7	6	5	4	3	2	1	0	FUNCTION
1	1	I6						I0	Station ID
2	COSR	ABER	O5					O0	Opcode
3	D7							D0	data
4	D7							D0	data
5	L7							L0	LRC

Слика 3.3.1.1.1 Структура поруке од MTU до RTU

3.3.1.1 *Remote-to-Master* порука

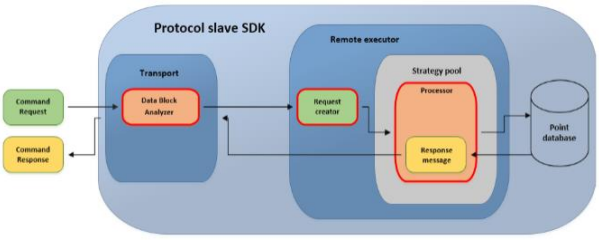
Структура овог типа поруке је приказана на слици 3.3.1.2.1.

BYTE	7	6	5	4	3	2	1	0	FUNCTION
1	0	I6						I0	Station ID
2	S7							S0	RTU status
3	C7							C0	COS count
4	D7							D0	1st data
.									
.									
.									
n-1	D7							D0	last data
n	L7							L0	LRC

Слика 3.3.1.2.1 Структура поруке од RTU до MTU

4. КОНЦЕПТ РЈЕШЕЊА

Тема овог рада јесте да се реализује једно програмско рјешење библиотеке на примјеру индустријског комуникационог протокола *Series V*.
Црвеним оквиром је приказана примјена већ поменуте библиотеке над *Protocol Slave SDK (слика 4.1).*



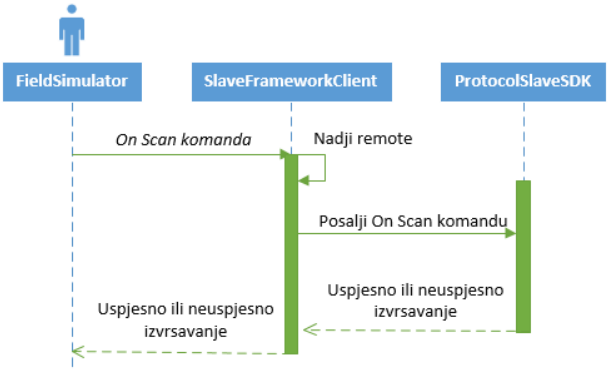
Слика 4.1 Примјена библиотеке на Protocol Slave SDK

4.1 *Protocol slave SDK*

На слици (*слика 4.1*) приказани *SDK* служи као подршка за рад са *slave* страном. У ситуацијама када програмери нису у могућности да тестирају функционалност свог софтвера над стварним уређајем у пољу, јавља се потреба за симулационим окружењем. Управо је то улога већ поменутог *Protocol slave SDK.* Међутим, оно што га раздваја од стварног уређаја је могућност манипулације и симулирања понашања свих индустријских протокола (или барем велике већине).

4.2 *Field Simulator*

За потпуни кориснички доживљај неопходан је графички приказ који омогућава директну манипулацију над *Protocol slave SDK* преко *SlaveFrameworkClient-a.*
Неке од основних функционалности које пружа *FieldSimulator* су могућност *scan on/off* свих *RTU* уређаја, канала, линкова, као и директно мијењање вриједности тачака у пољу. Пружа увид у вријеме (*TimeStamp*) посљедње очитане вриједности тачке.
Секвенцијални дијаграм (*слика 4.2.1*) приказује ток размјене порука приликом извршавања *on scan* команде преко *FieldSimulatora.*



Слика 4.2.1 Извршавање On Scan команде

4.3 *Slave Framework Client*

SlaveFrameworkClient у свој овој причи служи као као неки омотач (*wrapper*) око *ProtocolSlaveSDK* и пружа неке додатне могућности и мапирања података.
Један примјер је интерна колекција *remote* уређаја која служи за мапирање пристиглог идентификатора уређаја са конкретним уређајем.
Оно што је од велике важности је почетна иницијализација комплетног *ProtocolSlaveSDK.* Под тим се мисли на иницијализацију процесора, локаних модела *RTU* уређаја, типова захтјева, базе података и остало.

5. ИМПЛЕМЕНТАЦИЈА РЈЕШЕЊА

5.1 *Series V Data Block Analyzer*

Преко мреже стиже гомила сирових података у виду низа бајтова. Да би се ти подаци могли искористити, јавила се потреба за неким механизмом који би те податке некако прихватио, извалидирао и проследио на даљу обраду. Управо је то улога већ поменутог модула.

5.2 *Series V Request Creator*

За сваки код операције (карактеристичан за сваки протокол) се имплементира посебан тип захтјева. Сваки појединачни захтјев се зарегиструје преко овог модула у *Series5SlaveFrameworkClient*-у. Регистрација се врши унутар локалне колекције (*Dictionary*) према коду операције.

5.2 *Processors, Requests, Responses*

Процесори су организовани тако да насљеђују базни процесор који даље насљеђује одређени *interface.* Главни задатак процесора је да одради основну логику везану за процесирање пристигле команде.

Захтјеви (*requests*) су такође организовани на начин да насљеђују базни захтјев који даље насљеђује одговарајући *interface* Пошто се приликом имплементације водило рачуна о перформансама софтвера, тако се у овом случају искористила предност секвенцијалног извршавања команде, па је извршена уштеда у меморији на начин да се не праве стално нови захтеви, већ се врше промјене над почетно креираним захтјевом.

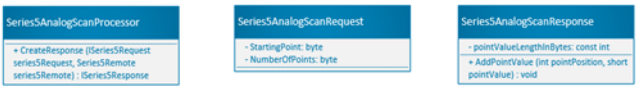
Одговори (*responses*) су реализовани преко базног одговора који даље насљеђује *interface*, али без могућности измјене вриједности над постојећим одговорима. За сваку пристиглу команду се формира посебан одговор.

Сваки захтјев, одговор и процесор из наредних поглавља насљеђују један од горе неведених класних модела. Тренутно имплементационо рјешење пружа подршку за сљедеће команде:

- *Analog scan*
- *Status point scan*
- *RTU status clear*
- *Analog change count*
- *Analog change dump*
- *Change of state queue dump*
- *Control select*
- *Control execute*
- *SOE time sync*

5.3.1 *Analog scan*

Примјер класног модела за једну од команди је приказан на слици 5.3.1.1. *Analog scan* команда враћа аналогне вриједности за захтјевани број тачака са *RTU* уређаја. Аналогни улази су нумерисани полазећи од тачке са кординатом 0. Аналогне вриједности су приказане као 12-обитне сирове вриједности у другом комплементу или као неозначене бинарне, зависно од тога како је *RTU* уређај исконфигурисан.



Слика 5.3.1.1 Класни модел за analog scan

Почетна захтјевана тачка мора бити у опсегу исконфигурисаних тачака у систему или другим ријечима, мора да постоји. Такође, крајња захтјевана тачка мора да буде мања од највеће исконфигурисане. Ако неки од претходних услова није задовољен, постављају се одговарајући *flag*-ови.

6. ЗАКЉУЧАК

У овом раду је представљено програмско рјешење библиотеке за *ProtocolSlaveSDK*, на примјеру *Series V* индустријског комуникационог протокола.

За реализацију рада је било неопходно детаљно упознавање са индустријским протоколима, као и са архитектуром и понашањем *Series V* протокола. Идеја је била да се дође до рјешења које ће бити максимално оптимизовано и перформантно, те ће на тај начин омогућити обраду огромне количине команди у кратком временском периоду.

7. ЛИТЕРАТУРА

[1] Софтвер са критичним одзивом, 2015, Бранислав Атлагић

[2] http://electricalquestionsguide.blogspot.com/2011/06/master-terminal-units-mtu-in-scada.html

[4] Series V Communication protocol, © Copyright 1992 by Valmet Automation

[5] Tejas protocol emulation, www.scribd.com

[6] https://www.taitradioacademy.com/topic/scada-and-polling-1/

Кратка биографија

Филип Јефтић рођен је 16.04.1995 године на Илици. Завршио је Гимназију “Јован Дучић” у Требињу 2014 године. Исте године је уписао основне академске студије на Факултету техничких наука у Новом Саду. Дипломски рад из области Електротехника и рачунарство – Примењени софтверски инжењеринг одбранио је 2018. године. Испунио је све обавезе и положио све испите предвиђене студијским програмом.