Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.4
DOI: https://doi.org/10.24867/08BE19Jelic

РАЗВОЈ SDK БИБЛИОТЕКЕ ЗА ИМПЛЕМЕНТАЦИЈУ ИНДУСТРИЈСКИХ ПРОТОКОЛА

DEVELOPMENT OF THE SDK LIBRARY FOR THE IMPLEMENTATION OF INDUSTRIAL PROTOCOLS

Бранко Јелић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом раду представљено је једно решење Software Development Kit (SDK) библиотеке за потребе развоја индустријских протокола. Поред описаног концепта решења, у раду су описане основе SCADA система, комуникационих и индустријских протокола.*

Кључне речи: SCADA, RTU

Abstract – *This paper introduces one solution of the Software Development Kit (SDK) library for industrial protocol development purposes. In addition to the described solution concept, the paper describes the basics of SCADA systems, communication and industrial protocols.*

Key words: SCADA, RTU

1. УВОД

Непрекидан развој информационих технологија условио је паралелан напредак у индустрији. Многи индустријски процеси потпуно су аутоматизовани и укључени у системе за надзор и управљање подацима. Системи за надзор и управљање, познатији у свету као *SCADA (Supervisory Control And Data Acquisition)* системи, врше прикупљање и обраду велике количине података. Размена података између самог *SCADA* система и опреме за аутоматизацију у оквиру постројења врши се путем различитих индустријских комуникационих протокола. Само тестирање имплементације различитих комуникационих протокола је често отежано и ризично радити на самом индустријском систему. Због те чињенице се имплементирају симулатори индустријских система за различите комуникационе протоколе.

Идеја овог рада је развој *Software Development Kit (SDK)* библиотеке за потребе развоја индустријских протокола. Сам задатак има за циљ имплементацију и опис библиотеке која ће омогућити развој различитих индустријских протокола у сврху симулације индустријског система. Решење треба да омогући да развој библиотеке конкретног индустријског протокола обухвати само имплементацију специфичности самог протокола.

НАПОМЕНА:

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

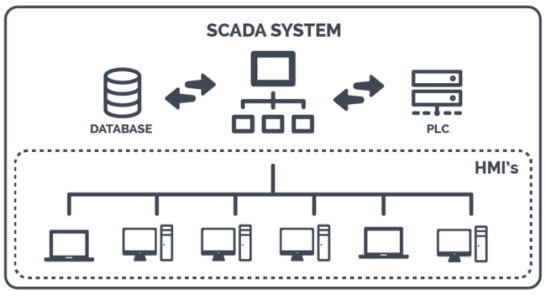
2. ТЕОРИЈСКЕ ОСНОВЕ

2.1 *SCADA* систем

Аквизиционо управљачки системи (*AUS*) представљају скуп просторно дистрибуираних и међусобно повезаних рачунарских модула који за заједнички циљ имају остварење функције надзора и управљања физичким процесом у реалном времену. Често се овакви системи (специфичне намене и структуре) у литератури означавају термином *SCADA* (*Supervisory Control And Data Acquisition*). Кључни захтеви које аквизиционо управљачки систем опште намене мора испунити су:

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

Са становишта примењених алгоритама управљања и географске топологије постоје два основна типа аквизиционо управљачких система. Први тип представља управљање у географски размештеним системима (системима за пренос енергената и других сировина). Овакви системи имају инсталиране једноставне процесне контролере који се називају удаљене телеметријске станице (*RTU – Remote Telemetry Unit*). Код оваквих система комуникациони подсистем је често веома сложен и релативно спор у погледу брзине преноса података (укључује разнородне преносне медије). Други тип аквизиционо управљачких система се односи на управљање у просторно ограниченим индустријским постројењима (индустријска постројења код којих се углавном све налази у оквиру истог постројења). Код оваквих система комуникација се одвија поузданије и постоји висок ниво аутоматизације самих управљачких активности [1].



Слика 2.1.1 Архитектура SCADA мреже

Водећа процесна јединица или *MTU* (*Master Terminal Unit*) у *SCADA* систему представља уређај који се користи за издавање команди над *RTU* уређајима, прикупља захтеване податке, складишти податке, обрађује информације и исте приказује у различитим формама (слике, графови, табеле) на корисничком приказном уређају (*HMI*) и на тај начин олакшава оператерима доношење одлука (слика 2.1.1). *MTU* се налази у оквиру контролног центра [1].

2.1.1 Прикупљање података у *SCADA* системима

Прикупљање података у *SCADA* системима се врши слањем поруке на *RTU* уређај и чекањем да стигне одговор. *RTU* уређај има везу са физичким светом у коме се врше мерења напона, струје, фазе... Одговор који стиже назад од *RTU* уређаја садржи конкретне вредности затражених тачака.

Постоје разни примери захтева, помоћу аналогних улаза се углавном добијају мерене вредности (струја, напон...), дигитални улази представљају најчешће стања прекидача, док се помоћу бројачких или имплусних улаза мери проток разних супстанци.

У зависности од типа опреме, улоге опреме у систему, као и ризика који носи са собом процес прикупљања података се може вршити сваких пар секунди, минута, сати, дана, месеци или година.

Увек треба рачунати на непредвиђене ситуације које су саставни део овако критичних система. *SCADA* систем треба да буде спреман да адекватно одреагује у складу са тренутно насталом непредвиђеном ситуацијом и да спречи упад система у неконзистентно стање. На пример постоји процес који прикупља податке сваких 20 минута, а у 10. минуту се догоди испад (прекорачење). Део опреме где је измерено прекорачење треба да обавести *RTU* уређај који ће након тога иницирати раније повлачење података [1].

2.2 Комуникациони протоколи

Комуникационим протоколом се означава скуп правила и процедура који контролишу ток комуникације и обезбеђују успешну интеракцију између удаљених процеса. Протокол дефинише формат и редослед порука које се размењују, као и сваку акцију која се иницира пријемом одређене поруке. За опис протокола се користи дијаграм промене стања или неки од виших програмских језика. Основне и најбитније функције сваког комуникационог протокола су контрола грешака и управљање током података у мрежи. Контрола грешака обезбеђује поуздан пренос и интегритет порука, док је примарни циљ управљања током избегавање загушења у саобраћају и дељење комуникационих ресурса између више корисника.

Структура комуникационих система је због своје комплексности строго хијерархијска и реализује се кроз више нивоа (слојева).

Неки од разлога који су условили слојевитост структуре протокола су следећи [1]:

- *Независност између нивоа.* Под овим се подразумева да виши ниво протокола није условљен начином на који је имплементиран претходни.
- *Флексибилност.* Све док је постојећа спрега идентична, промена на једном нивоу не утиче на остале нивое.
- *Физичко одвајање.* Неки од слојева протокола могу бити реализовани на различитој физичкој основи, уз коришћење предности најсавременије технологије.
- *Једноставнија имплементација и одржавање.* Као последица модуларности слојева и декомпозиције укупних мрежних услуга, олакшан је развој и испитивање опреме и програмске подршке.
- Могућност стандардизације. Услуге и спреге појединих нивоа се могу стандардизовати, уколико се прецизно спецификују функције.

2.3 Индустријски протоколи

Индустријски протоколи и индустријске мреже је уобичајено име за класу протокола специфичних за *SCADA* системе. Према типу учесника у комуникацији и њиховим улогама у АУС, индустријски протоколи се могу класификовати у неколико типичних категорија. Неки протоколи имају примене у више суседних категорија.

- *Сензорски протоколи* омогућују повезивање са „*smart*“ сензорима и извршним уређајима, у циљу примарне аквизиције података из постројења.
- *Fieldbus протоколи* примарно су оријентисани комуникацији са *UI* модулима и контролерима повезаних процесном магистралом у индустријску fieldbus мрежу.
- *Телеметријски протоколи* контролишу комуникацију између процесних контролера и централне *SCADA* станице. Због тога се код њих много пажње посвећује минимизацији обима података који се преносе.
- *Међусистемски протоколи* реализују спрегу између два аутономна али кооперативна система, без обзира да ли се ради о два *SCADA* система, вези *SCADA* система са симулатором, или нечему другом.

Поред наведених класа протокола, реални *SCADA* системи морају имплементирати клијентски протокол који преноси *SCADA* податке до оператерских радних станица или пословних апликација. *Modbus* је најпопуларнији индустријски протокол.

У електроенергетским *SCADA* апликацијама, у Европи су доминантни протоколи *IEC 60870-5101/104*, а у Америци протокол *DNP3*. Оба протокола имају заједничко порекло и развијени су баш за примену у системима за пренос и дистрибуцију електричне енергије.

Спрега између аутономних *SCADA* система, стандардизована је углавном кроз два стандарда: *ICCP* (*Inter-Control Center Communications Protocol*), и *OPC* (*OLE for Process Control*). *ICCP* је ту једини релевантан стандард, јер је независан од програмске платформе и једини нуди адекватне перформансе.

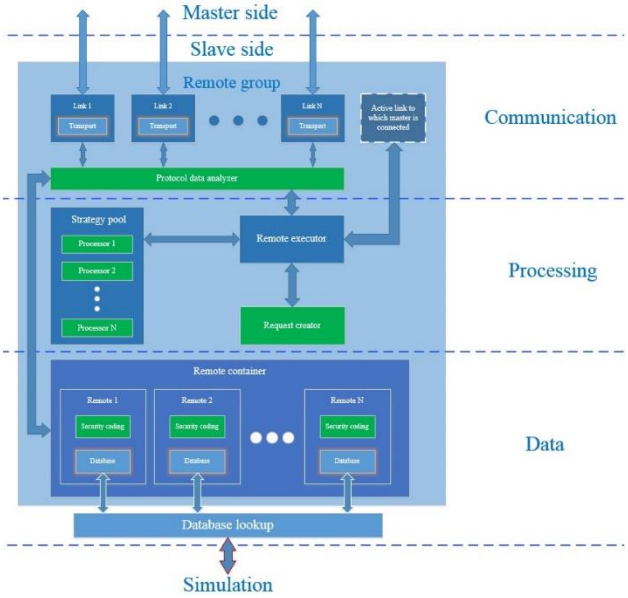
2.3.1 Комуникациони приступи

Комуникациони приступи подржавају ефикасну комуникацију преко мреже широког подручја до *RTU*, за прикупљање процесних података и пренос управљачких команди, који се могу оптимизовати и са становишта безбедности и трошкова. Комуникациони приступи подржавају различите конфигурације. Данас су најпопуларније у употреби [1]:

- *Multidrop* је радијална конфигурација где се *RTU* позивају у низу преко једног комуникационог канала. Ово даје јефтиније решење на рачун времена одзива.
- *Point-To-Point* намењује један комуникациони канал за један *RTU*. Обично се користе за главне трафостанице или концетраторе података који имају *RTU* са великим улазно/излазним захтевима.
- *Loop* (петља) ради у конфигурацији отворене петље подржане са два комуникациона приступа, где је сваки канал *multidrop* типа.
- *Star* (звезда) конфигурација је комбинација *point-to-point* до *RTU* концентратора података, који управља приступом подацима до подређених RTU конфигурисаних као *point-to-point* или *multidrop*. Ове конфигурације се користе у аутоматизованим дистрибуцијама где мешовито време одзива може бити економски планирано.

3. КОНЦЕПТ РЕШЕЊА

На слици 3.1 је приказана архитектура самог решења подељена у више целина (делова). Целине су комуникација (*communication*), обрада (*processing*) и подаци (*data*).



Слика 3.1 Приказ везе између основних компоненти у архитектуре решења

Разлог овакве архитектуре решења је у томе што је подржана *multi drop* веза у пољу између више *RTU*-ова једног комуникационог протокола. Дакле ако је реч о *Point-To-Point* вези између *master* стране и *RTU* онда ће *Remote container* садржати само један *Remote*.

Уколико се ради о *multidrop* вези више *RTU*-ова *Remote container* ће имати више *Remote*-ова.

Важна напомена је да су плавом бојом на слици 3.1 означене компоненте које су имплементиране у оквиру библиотеке, а зеленом бојом оне компоненте које ће корисник имплементирати по унапред дефинисаним интерфејсима приликом имплементације индустријског протокола.

Свака *Remote group* може имати више комуникационих канала (*Link*) путем којих размењује поруке са *master* страном.

Remote executor представља компоненту која је задужена за координацију осталих компонентама из више различитих целина.

4.1 Комуникација (*Communication*)

Као што се може видети на слици 3.1 комуникациони део је одговоран за размену (пријем и слање) података са *master* страном.

Као што је раније речено свака *Remote group* има своје *Link*-ове. Кључни разлог оваквог дизајна лежи у томе што сваки *RTU* уређај у пољу је повезан са остатком система путем више различитих комуникационих веза (*Link*-ова). У сваком тренутку *RTU* размењује податке са *master* страном путем искључиво једне од комуникационих веза (по природи *RTU* редно обрађује захтеве пристигле са *master* стране и размена података путем више комуникационих веза није могућа) и она се назива активна комуникациона веза (*Active link*). Изузетак је кад су у квару све комуникационе везе, тада *RTU* нема активну комуникациону везу.

Веома важан део сваког *Link*-а представља коришћени протокол за размену порука на транспортном нивоу (*TCP* или *UDP* су протоколи који се користе на транспортном нивоу). Он одређује на који начин се врши пријем и слање порука преко мреже.

Када се *master* страна повеже на неки од Link-ова он постаје активан (*Active link*) тек онда када *master* страна прекине везу са свим осталим *Link*-овима.

Оног тренутка кад неки од *Link*-ова постане активан, *Remote executor* почиње пријем, обраду и слање података са тог *Link*-а.

Основна замисао је да се кориснику омогући имплементација специфичности протокола кроз *Protocol data analyzer*.

Приликом пријема података константно ће бити консултован *Protocol data analyzer* који би требало да препозна да ли је пристигла порука примљена у потпуности и уколико јесте да изврши валидацију пристиглих података. У супротно треба да одбаци поруку или настави пријем док порука не буде примљена у потпуности. *Protocol data analyzer* има задатак да одреди за који *Remote* из *Remote container*-а је стигла порука.

Такође је омогућена подршка симулационом окружењу за укључивање и искључивање *Link*-ова по жељи што омогућава симулацију прекида везе.

4.2 Обрада (*Processing*)

Обрада примљене поруке почиње у оном тренутку када *Protocol data analyzer* заврши са њеном валидацијом.

Remote executor наставља са даљом обрадом поруке. Он позива *Request creator*-а који има задатак да од примљене поруке направи одговарајућу протокол захтев.

Request creator је компонента која се креира од стране корисника библиотеке и мора бити имплементирана на унапред дефинисан начин путем њеног универзалног интерфејса.

Након тога *Remote executor* консултује *Strategy pool* на који ће од *Processor*-а послати захтев на обраду. *Strategy pool* је садржи за одређени тип захтева *Processor* који је одговоран за његову обраду.

Даља обрада се одвија на *Processor*-у који поред самог захтева добија и *Remote* на ком захтев треба да се изврши. Корисник библиотеке има задатак да у потпуности имплементира за сваки тип захтева *Processor* који ће вршити његову обраду и креирање одговора. Када *Processor* обради захтев и креира одговор *Remote executor* ће преко активног линка поруку послати назад на master страну.

4.3 Подаци (*Data*)

Компоненте одговорне за чување података за сваку *Remote group*-у су приказани на слици 3.1 у оквиру секције *Data*.

Компонента одговорна за чување *Remote*-ова у оквиру *Remote group*-е се назива *Remote container*. Ова компонента је потпуно независна од протокола.

Дизајн самог *Remote* је такође осмишљен да буде независан од индустријског протокола али са могућности проширења са специфичности конкретног индустријског протокола. Сваки *Remote* може имати (али и не мора) *Security coding* чија је имплементација препуштена ономе ко имплементира конкретан индустријски протокол.

Приликом дизајнирања решења водило се рачуна о томе да манипулацију вредностима тачака може обављати више страна. Прва страна је *master*, који путем протокол захтева има могућност промена и читања вредности тачака (у зависности од типа тачке).

Другу страну представља симулационо окружење, коме је такође омогућена промена и читање вредности тачака. Битна ствар коју треба напоменути је да симулационо окружење има могућност мењања и читања вредности свих типова тачака.

Из претходно наведених разлога сваки *Remote* има своју *Database* у оквиру које се налазе све његове тачке. Приступ *Database* се очекује искључиво приликом извршавања протокол захтева.

Database lookup компонента омогућава приступ тачкама свих *Remote*-ова једног индустријског протокола. Осмишљен је у сврху приступа тачкама и мењања њихових вредности из симулационог окружења.

5. ИМПЛЕМЕНТАЦИЈА РЕШЕЊА

Само програмско решење имплементирано је у програмском језику *C#*, коришћењем *.NET* 4.6.2 софтверске платформе у *Microsoft Visual Studio 2015* развојном програмском окружењу. Приликом израде самог решења коришћене су најбоље софтверске праксе попут: *TAP* програмирања [2], *S.O.L.I.D* принципа [3], дизајн патерна [3], *Test-Driven Development*-а [4].

6. ЗАКЉУЧАК

У овом раду је представљено програмско решење *SDK* библиотеке за потребе развоја индустријских протокола. Пре саме реализације рада било је потребно детаљно се упознати са *SCADA* системом и његовом архитектуром, као и специфичностима разних индустријских протокола. Идеја самог рада је била да се омогући лака и једноставна имплементација групе или појединачних удаљених телеметрисаних јединица (*RTU*) различитих индустријских протокола, који би се користили за опонашање реалног система.

У самом програмском решењу је остављено простора за размишљање о будућим унапређенима. Даљи правац развоја би подразумевао подршку за *unsolicited* поруке, као и пружање могућности кориснику да пресреће пристигле поруке и дефинише одговоре који ће се слати назад на *master* страну. Сама литература о индустријским протоколима је слабо јавно доступна па је самим тим и отежано узети у обзир све случајеве приликом развоја саме библиотеке. Једно од будућих унапређена би свакако било проналазак адекватне литературе и модификација постојећег решења у складу са стеченим знањем.

7. ЛИТЕРАТУРА

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

[2] *https://docs.microsoft.com/en-us/dotnet/standard/asynchronous-programming-patterns/task-based-asynchronous-pattern-tap*

[3] *Beginning SOLID Principles and Design Patterns for ASP.NET Developers, 2016, Bipin Joshi*

[4] *Test-Driven Development by Example, 2002, Kent Beck*

КРАТКА БИОГРАФИЈА

Бранко Јелић рођен је 07.09.1995 године у Сремској Митровици. Завршио је Електротехничку школу "Михајло Пупин" у Новом Саду 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.