УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

НОВИ САД

Департман за рачунарство и аутоматику

Одсек за рачунарску технику и рачунарске комуникације

ДИПЛОМСКИ – МАСТЕР РАД

Кандидат: **Војкан Стефановић**

Број индекса: **10391**

Тема рада: **Једно решење клијентског протокола SCADA система**

Ментор рада: **проф. др Бранислав Атлагић**

Нови Сад, April 2012.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:			
Идентификациони број, **ИБР**:			
Тип документације, **ТД**:		Монографска документација	
Тип записа, **ТЗ**:		Текстуални штампани материјал	
Врста рада, **ВР**:		Дипломски – мастер рад	
Аутор, **АУ**:		**Војкан Стефановић**	
Ментор, **МН**:		**проф. др Бранислав Атлагић**	
Наслов рада, **НР**:		**Једно решење клијентског протокола SCADA система**	
Језик публикације, **ЈП**:		Српски / латиница	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		**2012.**	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		**7/40/0/0/19/0/1**	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кqучне речи, **ПО**:		**SCADA, репликација података, протокол, WPF, XML**	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		Комуникација са оператером је једна од приоритетних функција SCADA система. У раду је реализован клијентски протокол и прототип графичке корисничке спреге аквизиционо управљачког система oSCADA. Развој ових компоненти извршен је коришћењем актуелних технологија, тако да се клијентска страна прилагођава произвољном формату примљених SCADA података. Нови протокол назван PAYC заснован је на XML порукама, а клијентска спрега је реализована као WPF апликација која се генерише на основу примљених метаподатака.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	**проф. др Мирослав Поповић**	
	Члан:	**проф. др Горан Швенда**	Потпис ментора
	Члан, ментор:	**проф. др Бранислав Атлагић**	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		**Vojkan Stefanović**	
Mentor, **MN**:		**PhD Branislav Atlagić**	
Title, **TI**:		**Implementation of a client protocol for a distributed SCADA system**	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		**2012.**	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		**7/40/0/0/19/0/1**	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S/KW**:		**SCADA, RAUS, protocol, WPF, XML**	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		Contemporary SCADA systems emphasize the importance of flexible and effective HMI subsystem. It relays basically on clear definition of formats of data used within the system. The aim of the thesis presented here was to develop client protocol and GUI for the transfer and the presentation of SCADA data. New communication protocol, called RAUS, is based on XML messages to simplify GUI development. Use of XML enables easy data parsing, which made possible generation of WPF graphical user interface according to metadata transferred to the client station.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	**PhD Miroslav Popović**	
	Member:	**PhD Goran Švenda**	Menthor's signature
	Member, Mentor:	**PhD Branislav Atlagić**	

Zahvalnost

Rad posvećujem celokupnoj porodici, kao zahvalnost za podršku koju su mi nesebično pružali tokom svih ovih godina.

SADRŽAJ

SPISAK SLIKA

SKRAĆENICE

SCADA - *Supervisory Control and Data Acquisition*, Akvizacioni upravljački sistem

RTU - *Remote Terminal Unit,* Udaljena terminalna jedinica

PLC - *Programmable Logic Controle,* Programibilni logički kontroler

LAN - *Local Area Network,* Lokalna računarska mreža

WAN - *Wide Area Network,* Globalna ili geografsko široko rasprostranjena računarska mreža

CLR - *Common Language Runtime*, Zajednički izvršni jezik

GPU - *Graphics Proccesing Unit,* Grafička procesorska jedinica

CPU - *Central Proccesing Unit,* Centralna procesorska jedinica

XAML - *Extensible Application Markup Language,* Proširivi aplikativni metajezik za označavanje

XML - *Extensible Markup Language,* Proširivi metajezik za označavanje

GUI - *Graphical User Interface*, Grafička korisnička sprega

TCP - *Transmission Control Protocol,* Protokol sa kontrolom prenosa

1. Uvod

Zbog sve veće kompleksnosti proizvodnih procesa, upravljanje savremenim industrijskim postrojenjima je nezamislivo bez pomoći integrisanog akviziciono upravljačkog sistema, tj. bez primene SCADA sistema (eng. SCADA – *Supervisory Control and Data Acquisition*) radi potpunijeg nadzora i bolje kontrole proizvodnog procesa. Ovi sistemi su opšte prisutni u raznim granama privrede, a pre svega u procesnoj industriji, energetici, telekomunikacijama, poljoprivredi, industriji nafte i gasa, transportu i sl. Nadzor ili nadgledanje postrojenja, kao osnovna funkcija SCADA sistema, koristi komunikacione protokole i grafičku korisničku spregu kako bi krajnjem korisniku - operateru dostavili podatke o stanju sistema i omogućili njihov pregled na što pregledniji način. Na toj osnovi, operater sistema može na kvalitetan način proceniti situaciju i doneti poslednju odluku o evenutalnim promenama unutar sistema. Grafička korisnička sprega operatorske radne stanice je veoma bitan segment SCADA sistema, jer je to kontakt operatora (čoveka) sa procesnim sistemom. Zato prilikom njenog razvoja treba koristiti najnovije tehnologije, koje pružaju najbolje okruženje za realizaciju složenih korisničkih zahteva.

Zadatak ovog rada je implementiranje komunikacionog protokola i osnove korisničke konzole u okviru novog akviziciono upravljačkog sistema oSCADA, koji je trenutno u fazi razvoja. Ovako definisan zadatak zahteva razvoj poslužioca (servera) SCADA podataka, i klijenta koji realizuje grafičku korisničku spregu. Pošto sama komunikacija sa operaterom ne mora imati karakteristike tvrdog realnog vremena, komunikacioni protokol ne mora biti optimizovan do najvećeg stepena. Sam protokol i format podataka koji se prenose treba odrediti tako da se olakša njihova obrada na klijentskog strani.

Takođe, prilikom razvijanja komunikacionog protokola cilj je da na osnovu promenljivog formata podataka koji se prenosi, klijentska strana, u okviru oSCADA sistema, može lako da

dinamički izgeneriše izgled grafičke korisničke sprege. Prenošeni podaci su zbog toga sačinjeni od XML (eng. *Extensible Markup Language*) poruka koje se lako obrađuju tj. parsiraju.

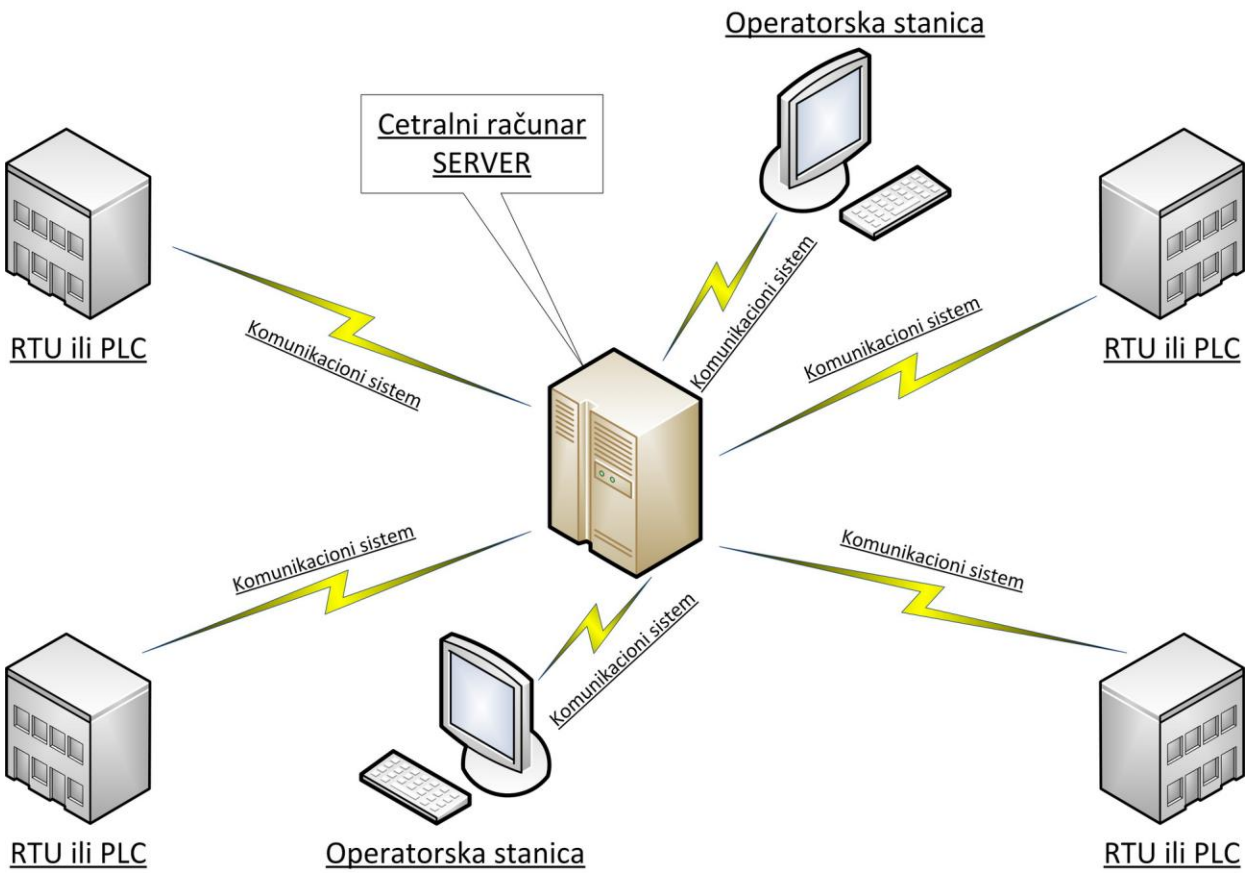
Za izradu korisničke konzole korišćena je *Microsoft* WPF (eng. *Windows Presentation Foundation*) tehnologija koja pored lake obrade XML poruka, pruža i savremen alat za kreiranje grafičke korisničke sprege koja se može izvršavati kao samostalna aplikacija, ali i u okviru internet pretraživača nezavisno od rezolucije ekrana.

Izlaganje je organizovano u sedam poglavlja. Drugo poglavlje opisuje opšte osobine i delove SCADA sistema. U trećem poglavlju dat je pregled korišćenih tehnologija u realizovanju ovog rada. Četvrto poglavlje posvećeno je opisu postojećim SCADA sisitemima kao i realizovanim klijentskim protokolom. U petom je opisana realizovana klijentska aplikacija. Šesto poglavlje sadrži zaključak rada, a sedmo spisak literature koja je korišćena prilikom izrade. Dodatak A sadrži primere XML poruka RAUS protokola, realizovanog u ovom radu.

2. SCADA sistemi

2.1 Opis i osobine SCADA sistema

Od vremena početka razvijanja digitalnih tehnologija i računarstva postoje namere da se sa njima nadziru i upravljaju proizvodni procesi. Razvojem industrije i usložnjavanjem proizvodnih procesa, digitalni uređaji i računari postali su njihovi neizostavni sastavni deo. Tako su se formirali i razvili akviziciono upravljački sistemi koji se najčešće nazivaju SCADA sistemi.



Slika 2.1 Šema opšteg savremenog SCADA sistema

SCADA sistem je skup namenskih, prostorno distribuiranih, međusobno povezanih računarskih modula, čiji je zajednički cilj ostvarenje funkcija nadzora i/ili upravljanja fizičkim procesom u realnom vremenu. Slika 2.1 prikazuje jedan opšti savremeni SCADA sistem.

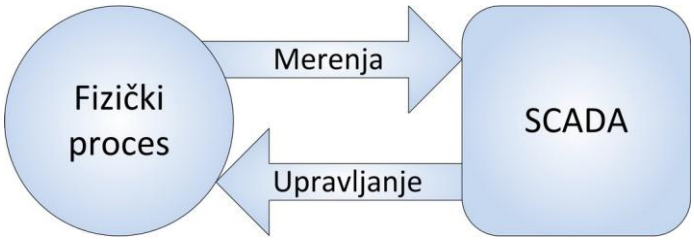
Ovaj sistem se sastoji od:

- procesnih uređaja – senzora i aktuatora,
- jednog ili više procesnih kontrolera. Obično su to udaljene telemetrijske stanice (eng. *RTU – Remote Telemetry Unit*) ili programibilni logički kontroleri (eng. *PLC – Programmable Logic Controllers*),
- komunikacionog sistema za prenos podataka,
- centralnog računara – servera,
- jedne ili više operatorskih stanica.

Kako bi SCADA sistem bio efikasan, mora ispuniti sledeće zahteve:

- *Rad u realnom vremenu* – radi pravovremenog reagovanja na poremećaje u okviru fizičkih procesa kojim se upravlja,
- *Distribucija računarskih resursa u okviru sistema* – objekti nadzora i upravljanja SCADA sistema prostorno su udaljeni u oblasti industrijskog postrojenja, ali i u mnogo širem geografskom području. Zbog toga strukturu SCADA sistema sačinjavaju prostorno distribuirane autonomne računarske komponente, koje su komunikacionom mrežom spregnute u jedistven sistem,
- *Pouzdanost* – direktno je vezana za kvalitet ugrađene fizičke arhikteture i programske podrške. Izražava se srednjim vremenom između ispada sistema,
- *Raspoloživost* – izražava se kao odnos vremena u kome je sistem funkcionaisao, i pored otkaza pojedinih komponenti.

Osnovna funkcija SCADA sistema je ciklična akvizicija digitalizovanih podataka različitih fizičkih veličina koje određuju stanje proizvoljnog tehnološkog (fizičkog) procesa. Formiranjem baze merenih podataka u računaru, stvara se mogućnost za proveru i prikaz trenutnog stanja fizičkog procesa, odnosno za obavljanje efikasnog nadzora nad njim. Takođe SCADA omogućava, izvršenjem aplikativne upravljačke podrške, zahtevanje kontrolnih aktivnosti, tj. upravljanje nad fizičkim sistemom (Slika 2.2).



Slika 2.2 Opšta šema povezanosti fizičkog procesa i SCADA sistema

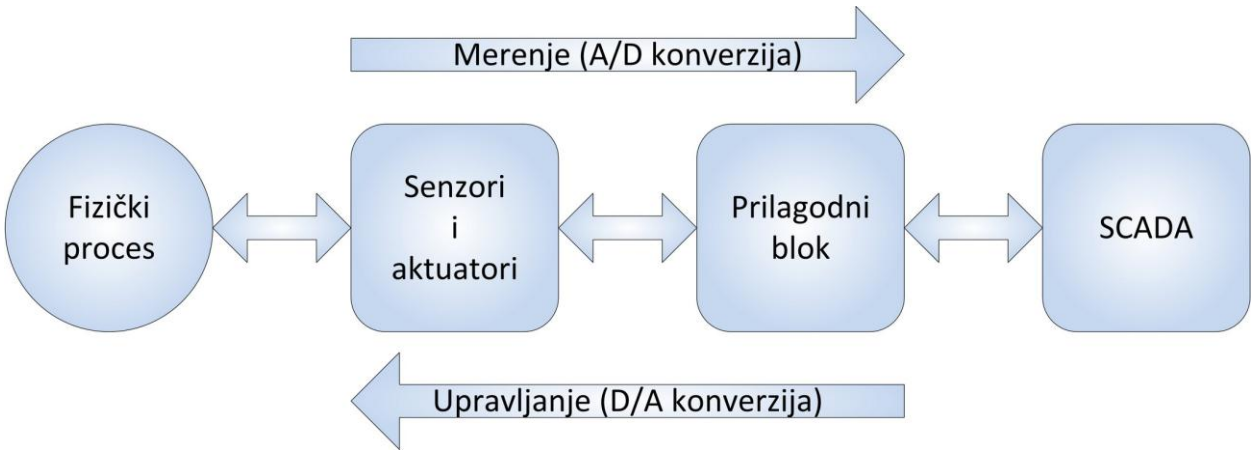
2.2 Klase SCADA sistema

Prema geografskoj razuđenosti sistema kao i algoritmima upravljanja, postoje dve osnovne klase SCADA sistema:

- *Upravljenje složenim, geografsko distribuiranim sistemima* – Zbog kompleksne prirode procesa i uticaja komunikacionih problema, zadaci automatskog upravljanja procesom su uglavnom ograničeni. Odluke kao što su promena radnog režima i iniciranje komandnih procedura, se ostavljaju čoveku - operatoru (dispičeru) sistema. Naglasak je prvenstveno na kvalitetnom i pouzdanom nadzoru procesa, uz formiranje što potpunijeg istorijata stanja i događaja u sistemu. Rad ovih sistema odvija se pod kontrolom složenih programskih paketa. Primeri ovih sistema su: sitemi za prenos energenata (gasovodi, naftovodi, elektroenergetska mreža), meteorološki akvizicioni sistemi, sistemi za kontrolu vazdušnog saobraćaja itd.
- *Upravljanje prostorno ograničenim industrijskim postrojenjijma* – Ovu klasu karakteriše manja dislokacija pojedinih elemenata sistema, pouzdano odvijanje komunikacije i visok stepen automatizacije upravljačkih aktivnosti. Često se naziva industrijski SCADA sistem. Upravljački agloritni su često vrlo složeni i obuhvataju automatsko vođenje proizvodnog procesa.

2.3 Sprega sa fizičkim procesom

Slika 2.3 prikazuje šemu povezivanja SCADA sistema sa fizičkim procesom.



Slika 2.3 Šema sprege SCADA sistema sa fizičkim procesom

Senzori i aktuatori imaju direktnu vezu sa fizičkim procesom i zbog toga predstavljaju bazu merno – regulacionog dela SCADA sistema. Uređaji poput merača nivoa rezervoara,

pokazivača trenutne pozicije ventila, termometara, protokomera, barometra itd. su primeri senzora, dok su uređaji poput električnih ventila, električnih hemijskih dozera, kontrolisanih električnih sklopki itd. primeri aktuatora.

Prilagodni blok vrši galvansko razdvajanje SCADA sistema i procesnih uređaja, kao i usklađivanje signala između signala procesnih uređaja na jednoj i signala ulaza/izlaza SCADA sistema na drugoj strani. Kvalitet analognih komponenata u prilagodnom bloku direktno utiče na kvalitet merno/upravljačkih signala, kao i na pouzdanost ukupnog sistema.

2.3.1 Procesni ulazi - senzori

Senzori su pretvarači neke fizičke veličine u ekvivalentni električni signal i najniži su elementi hijerarhije nadzorno – upravljačkog sistema. Nakon digitalizovanja električnog signala vrši se SCADA obuhvat podataka tj. fizička akvizicija posredstvom procesnih ulaza SCADA sistema. Standardni tipovi procesnih ulaza u akviziciono upravljačkom sistemu su:

- *Analogni ulazi* – Izvršavaju merenja nivo kontinualnog električnog signala (strujnog ili naponskog) na analognim ulazima SCADA sistema koji je u svakom trenutku proporcionalan trenutnoj vrednosti merene fizičke veličine. Tipični opsezi ulaznih električnih signala su: 4-20 mA, ±5V, 0-10 V, 0-100 mV i sl. Izvorne fizičke veličine koje se na ovaj način mere su pritisak, temperatura, masa i sl. Digitalizacija analognih ulaznih signala vrši se korišćenjem A/D konvertora, a rezolucija i tačnost konverzije određena je brojem bita digitalizovanog odmerka.
- *Brojački (impulsni) ulazi* – Učestanost električnih impulsa koji se prihvataju predstavlja meru trenutne vrednosti fizičke veličine. Impulsni pretvarači se najčešće koriste za merenja protoka tečnosti ili gasova, merenje brzine rotacije (taho - generator), ugla zakretanja i sl. Frekvencija generisanih impulsa u praksi je najčešće manja od 1 KHz, a njihov naponski nivo ne veći od 24 Vdc. Pošto se u okviru SCADA sistema impulsni signali prihvataju posredstvom digitalnih brojačkih kola, ovi procesni ulazi se nazivaju brojačkim ulazima.
- *Digitalni ulazi* – Izvršavaju merenja fizičkih veličina koje su diskretne po svojoj prirodi. Najčešće se preko njih prati stanje izvršnih elemenata u postrojenju, kao što su ventil (otvoren/zatvoren) ili sklopka (uključena/isključena). Izvor ovakvih signala su i razni graničnici (pun/prazan), sigurnosni prekidači poput presostata (natpritisak/normalno), i sl. Ulazni električni signal je naponski, nivoa 24 V DC ili 220 V AC. Digitalni ulazni signali se očitavaju posredstvom ulaznih registara digitalnog sistema.

2.3.2 Procesni izlazi – aktuatori

Aktuacija na neki fizički proces ostvaruje se izvršavanjem upravljačkih funkcija SCADA sistema na izvršnim elementima (uređaja). U opštem slučaju, to su elektro-mehanički uređaji direktno uključeni u samo procesno postrojenje. SCADA sistem posredstvom procesnih izlaza kontroliše izvršne elemente generisanjem pobudnih električnih signala. Standardni su sledeći tipovi procesnih izlaza:

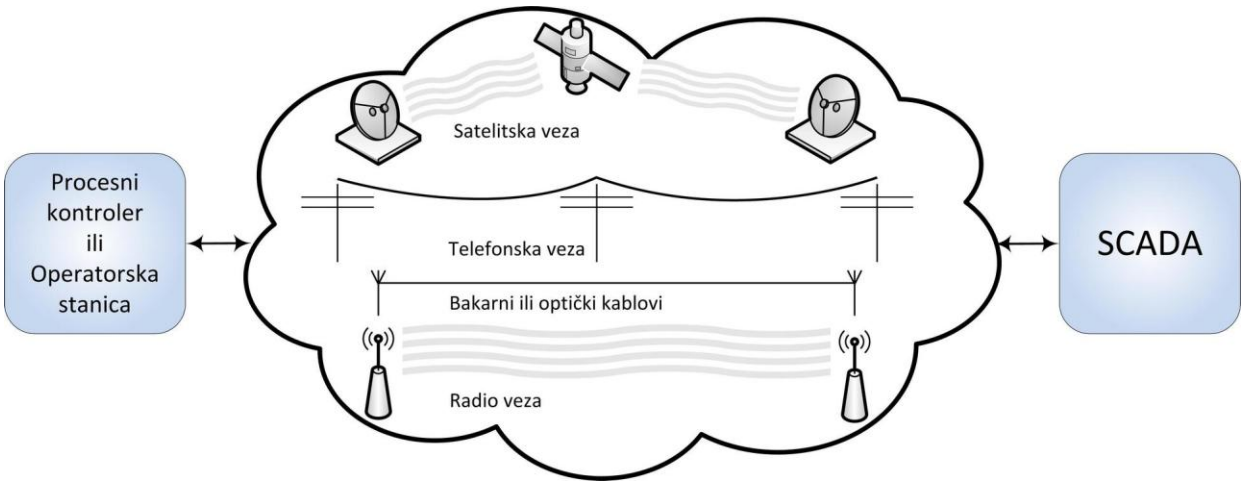
- *Analogni izlazi* – Pobudnim signalom analogne tj. kontinualne prirode, zadaje se radna tačka izvršnog elementa. Na ovaj način se obično utiče na poziciju regulacionog ventila, referentnu vrednost eksternog PID regulatora i sl. Električni opseg analognog procesnog izlaza SCADA sistema je obično reda jačine struje 0-20mA i napona 0-5V. Izlazni signal ovog aktuatora generiše se korišćenjem D/A konvertora preciznosti 8-12 bita.
- *Digitalni izlazi* – Njihovim posredstvom ostvaruje se upravljanje tipa uključi/isključi ili otvori/zatvori, za uređaje kao što su sklopke, kontaktori, aktuatori ventila i sl. Digitalni izlazi se najčešće opisuju radnim naponom i maksimalnom strujom koju mogu da propuste. Tipične vrednosti su 24Vdc/1A, 220Vac/3A, i sl. U okviru SCADA sistema digitalni izlazi se pobuđuju posredstvom izlaznih registara.

2.4 Komunikacioni sistem za prenos podataka

Namena ovog sistema je prenos podataka između:

- SCADA centralnog računara – servera i procesnog kontrolera
- SCADA centralnog računara – servera i operatorske stanice.

Kao medijum prenosa može se koristiti radio, kablovska ili satelitska veza, kao i bilo koja kombinacija pomenutih (prikazano na Slika 2.4). Koja će se vrsta prenosa koristiti zavisi od količine podataka koju treba preneti, raspoloživosti telekomunikacione infrastrukture, geografskog položaja delova SCADA sistema kao i od ekonomske isplatljivosti.



Slika 2.4 Uopšteni šematski prikaz komunikacije unutar SCADA sistema

Na samom početku razvoja SCADA sistema, komunikacioni sistemi su imali svoje autonomne mreže za prenos podataka. Povećanjem broja lokalnih računarskih mreža (eng. *LAN – Local Area Network*) unutar industrijskih prostorija, kao i širenje globalne računarske mreže (eng. *WAN – Wide Area Network*), došlo je do mogućnosti njihove integracije sa komunikacionim SCADA podsistemima. Najveća prednost ovako uređenog sistema je korišćenje već postojeće telekomunikacione infrastrukture, tj. izbegavanje investiranja u zasebnu računarsku mrežu.

2.5 Arhitektura SCADA sistema i tipovi upravljanja

Tip upravljačkog SCADA sistema zavisi od njegove arhitekture i može biti lokalni, centralizovani i distribuirani.

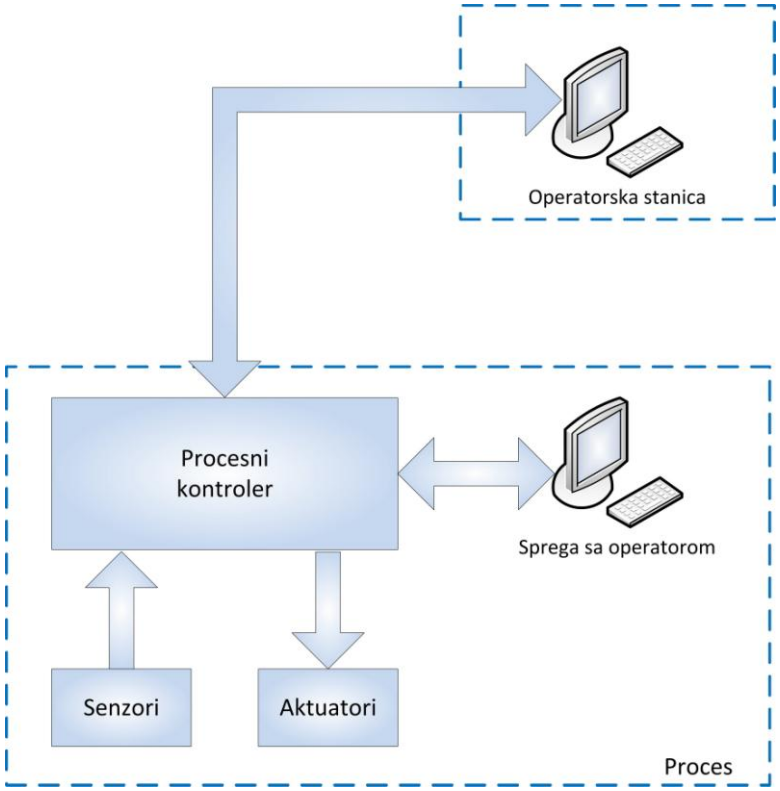
2.5.1 Lokalno upravljanje

Slika 2.5 prikazuje arhitekturu SCADA sistema gde su senzori, aktuatori i procesni kontroler međusobno u neposrednoj blizini. Tipičan slučaj je upravljanje pojedinačnom mašinom, ili nekim autonomnim delom postrojenja.

Procesni kontroler ima ograničeno polje delovanja na ukupan sistem. Njegova sprega sa operatorskom stanicom podrazumeva izvršavanje komandi o pokretanju ili zaustavljanju lokalno kontrolisanih automatskih sekvenci, kao i podešavanje željenih kontrolnih granica procesa. Kontrole fizičke sprege su implementirane u njemu.

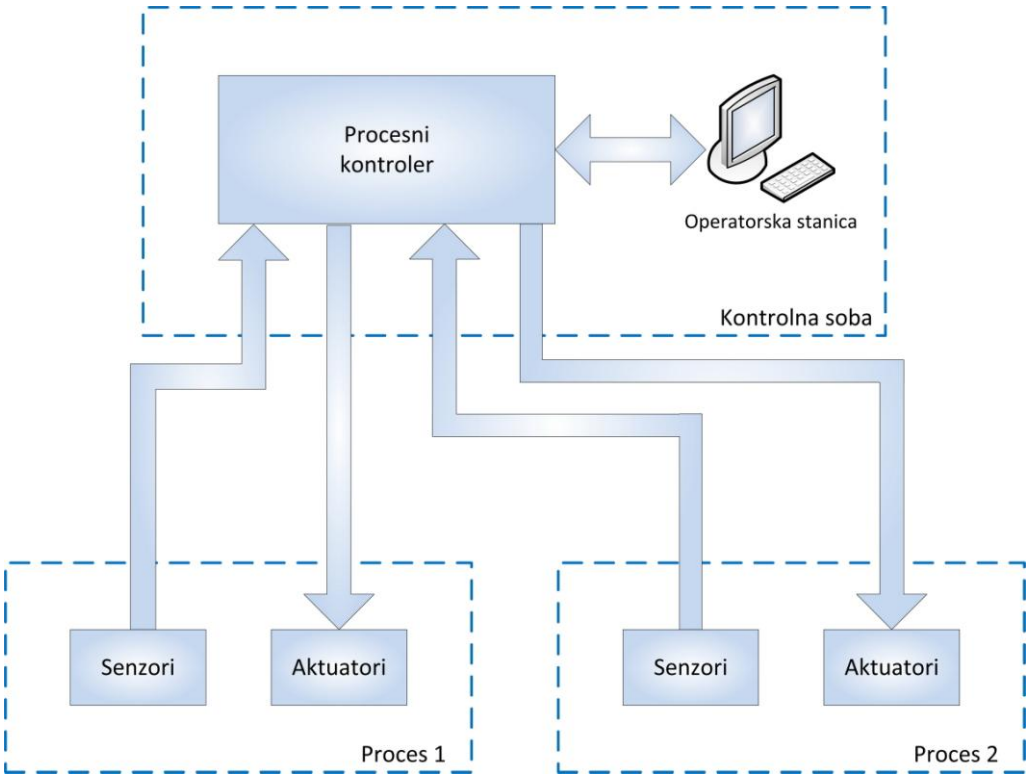
Sprega sa operatorom takođe je lokalna tj. u neposrednoj blizini same sprege sa fizičkim procesom. Ukoliko postoji udaljena operaterska stanica, ona i dalje ima funksionalnost kao i lokalna. U slučaju da postoji problem u sistemu, blizina operateru pruža olakšano rešavanje

problema. Sa druge strane, ova osobina zahteva da se operater, prilikom praćenja stanja sistema i otkrivanja eventualnih neregularnosti u okviru složenog postrojenja, kreće po objektu.



Slika 2.5 Arhitektura SCADA sistema sa lokalnim upravljanjem

2.5.2 Centralizovano upravljanje



Slika 2.6 Arhitektura SCADA sistema sa centralizovanim upravljanjem

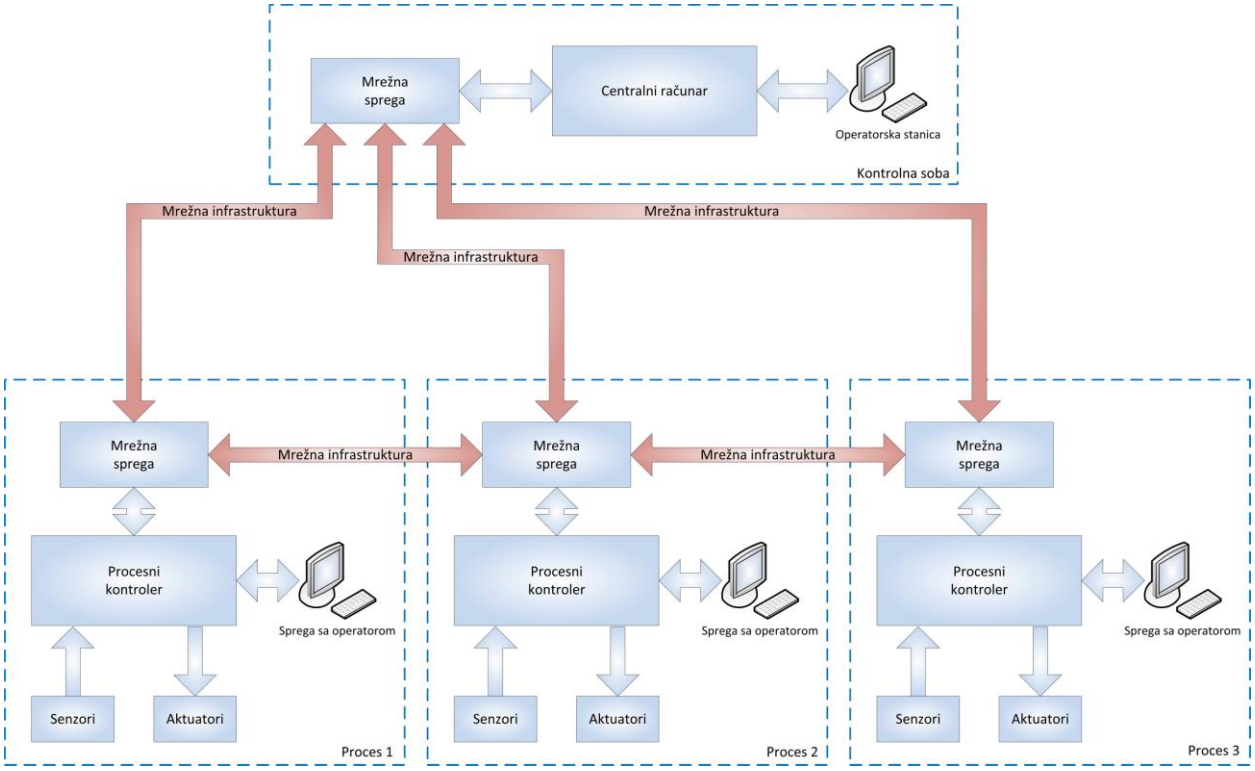
Slika 2.6 prikazuje arhitekturu sistema sa centralizovanim upravljanjem, kod koje su svi senzori, aktuatori i druga oprema u okviru postrojenja povezani na jedan ili grupu procesnih kontrolera. Upravljanje (praćenje stanja sistema i zadavanje komandi) svim procesima, kao i postojanje samo jedne operatorske stanice unutar kontrolne sobe, omogućava operateru bolji uvid u stanje sistema kao i brže reagovanje prilikom vanrednih situacija. Ova vrsta kontrole predstavlja preteču distribuiranom tipu kontrole.

2.5.3 Distribuirano upravljanje

Slika 2.7 prikazuje distribuirani tip SCADA sistema koji pruža najbolje karakteristike lokalnog i centralizovanog tipa upravljanja. U savremenim SCADA sistemima ovaj tip upravljanja i arhitekture sistema je najviše zastupljen.

Procesni kontroler deluje lokalno na spregu sa fizičkim procesima, ali je komunikacionim sistemom za prenos podataka umrežen sa centralnim računarom u kontrolnoj sobi.

Centralna operatorska stanica ima uvid u stanje celokupnog SCADA sistema, kao i ulaza i izlaza iz svakog lokalnog procesnog kontrolera u sistemu, ali poseduje mogućnost da menja upravljanje ukoliko je to potrebno.



Slika 2.7 Arhitektura SCADA sistema sa distribuiranim upravljanjem

2.6 SCADA centralni računar

Centralni računar ili SCADA server predstavlja nadzorno upravljačku stanicu. Sastoji se od jednog ili više računara povezanih u mrežu, čiji je osnovni cilj prikupljanje i obrada svih procesnih podataka kao i komunikacije sa operatorskim stanicama i operatorima. Obrada prikupljenih podataka podrazumeva proveru ispravnosti i vrednosti merenja, uz eksplicitno izveštavanje operatera o otkrivenim neregularnostima. Svi podaci se pretvaraju u oblik pogodan za prikaz, kao i za generisanje upravljačkih akcija.

2.7 Operatorska stanica i njena programska podrška

Operatorske stanice su najčešće obični računari koji su povezani komunikacionim sistemom sa centralnim računarom SCADA sistema. Sa odgovarajućom programskom podrškom (eng. *Software*) računar ima karakteristike klijenta koji prima i šalje informacije SCADA serveru, na osnovu zahteva i akcija operatera. Sprega između operatorske stanice i centralnog računara ostvaruje se posredstvom dinamički osvežavanih, posebno definisanih grafičkih i alfanumeričkih prikaza. Operaterski podsistem mora obezbediti pregledno i kvalitetno izveštavanje o promenama u sistemu, kao i brzo i tačno prihvatanje operaterskih komandi.

3. Razvojno okruženje i tehnologije

3.1 Programski jezik C++

Programski jezik C++ nastao je kao proširenje jezika C 1980. godine kada su se pojavile nove tehnike u programiranju, prvenstveno pojavom objektno orijentisanog programiranja. Nakon dodavanja klasa, provere i konverzije tipova itd. dobijeni jezik nazivao se C sa klasama. Sa dodavanjem virtuelnih funkcija i preklapanja operatora, 1983. godine jezik je dobio današnje ime C++. Tek 1997. godine usvojen je ANSI standard za C++ sa kojim je jezik dobio znatna proširenja. Njegova specifičnost je usvajanje, bez izmena, čak 95% jezika C a sa tim i vrlo blizak odnos sa fizičkom arhitekturom računara. Pri izradi programske podrške korišćen je najnoviji C++11 standard programskog jezika sa akcentom na realizovanje rešenja koje će biti nezavisno od operativnog i računarskog sistema (eng. *Cross Platform*).

3.2 Boost C++ biblioteke

Boost je skup biblioteka koje proširuju funkcionalnost C++ programskog jezika. Napravljene su namerom da rade sa standardnim C++ bibliotekama. Korisne su u rešavanju velikog broja svakodnevnih problema i mogu se implementirati u široki spektar aplikacija. Trenutno, izdanje sadrži preko osamdeset biblioteka, među kojima se nalaze biblioteke za strukture podataka, liearnu algebru, konkurentno programiranje, obradu slike, testiranje itd. Ovaj skup se često ažurira i održava, a kako su reference implementacija dostupne, većina biblioteka ulazi u nove C++ standarde.

3.2.1 boost::property_tree biblioteka

Property_tree biblioteka obezbeđuje duboko ugnježdenu strukturu podataka tipa stabla koja je na svakom nivou indeksirana po ključu. Pristup bilo kom elementu strukture moguć je spajanjem (konkatenacijom) više ključeva. Takođe, biblioteka poseduje parsere i generatore za više formata podataka koji sa kojim se struktura može prikazati. Jedan od njih je i XML (eng. *XML - Extensible Markup Language*) parser.

3.3 Microsoft .NET okruženje

.Net razvojno okruženje je sastavni deo Windows operativnog sistema koji omogućava razvoj i korišćenje nove generacije programa i mrežnih (internet) servisa. Daje mogućnost korišćenja velikog broja programskih jezika kao što su C#, C++, Visual Basic, JScript i drugi. Sastoji se iz dve osnovne komponente:

3.3.1 Zajednički izvršni jezik (eng. *CLR – Common Language Runtime*)

Predstavlja temelj .NET okruženja. Programska rešenja, pisana za ovo okruženje, izvršavaju se u virtuelnom programskom okruženju. Takođe, pruža servise kao što su bezbednost, upravljanje memorijom i rukovanje izuzecima. Kod koji CLR mehanizam izvršava naziva se kontrolisani kod (eng. *managed*), a kod koji se izvršava izvan kontrole CLR-a je nekontrolisani kod (eng. *unmanaged*).

3.3.2 Klasna biblioteka .NET okruženja (eng. *.NET framework class library*)

Sastoji se od širokog spektra gotovih rešenja za česte probleme koji nastaju prilikom pisanja programa. Najčešće su to metode za rad sa tipovima podataka, datotekama, bazama podataka, mrežnim povezivanjem, numeričkim algoritmima itd.

3.4 WPF

WPF (eng. *Windows Presentation Foundation*) je grafičko okruženje nove generacije za razvoj korisničke sprege u vidu samostalnih aplikacija i aplikacija za internet pretraživače. Uključen je u Microsoft .NET okruženje, pa se prikiom razvoja aplikacija mogu koristiti i drugi elementi iz biblioteke klasa .NET okruženja. Umesto oslanjajna na stari GDI (eng. *Graphical Device Interface*) WPF kao bazu koristi DirectX grafičku podršku. Ona je sačinjena od skupa API funkcija sa kojim se mogu iskoristiti resursi fizičkih arhitektura savremenih grafičkih uređaja (eng. *GPU – Graphics Processing Unit*) u računarima što sve zajedno omogućava iscrtavanje grafičkog okruženja bazirano na vektorima koje je nezavisno od rezolucije. Ovim

postupkom se oslobađaju (rasterećuju) resursi centralne procesorske jedinice (eng. *CPU – Central Proccesing Unit*) što dovodi do poboljšanja performansi i subjektivnog komfora rada na računaru. Osnovnu DirectX podršku WPF proširuje sveobuhvatnom kolekcijom razvojnih karakteristika kao što su 2D i 3D grafika, animacije, multimedija, tekst, tipografija itd.

Microsoft Silverlight je redukovan WPF koji omogućava izvršavanje aplikacija u okviru internet pretraživača. Podržano je iscrtavanje vektorske grafike ali ne i 3D funkcije. Ovako realizovane aplikacije mogu se izvršavati i na Linux operativnom sistemu.

3.4.1 XAML

Razvoj korisničke sprege (eng. *GUI – Graphical User Interface*) je moguć i novim kodnim jezikom XAML (eng. *Extensible Application Markup Language*) koji predstavlja alternativni način .NET okruženju (C#, Visual Basic .NET…) za definisanje grafičkih komponenti i njihovog odnosa sa ostalim komponentama. To je ujedno i jedna od najbitnih karakteristika WPF-a koja pruža konzinstentan model programiranja u kojoj je omogućeno razdvajanje razvoja grafičke korisničke sprege i razvoja logike aplikacije. Njih prave različiti razvojni timovi (dizajnera i programera) i to donosi nov pristup razvoja aplikacije. Kako je XAML jezik modifikovana verzija XML-a timovi mogu da implementiraju i menjaju XAML kod bez kompajliranja. Pravljenje ili izmena XAML datoteka moguće je uraditi upotrebom alata Microsoft Visual Studio ili alata za grafički dizajn Microsoft Expression Blend.

3.5 XML

XML (eng. *Extensible Markup Language*) je proširivi metajezik za označavanje koji predstavlja standardni skup pravila za definisanje formata podataka u elektronskoj formi. Specifikacija ovog metazika je predastavljena od strane W3C – WWW Konzorcijuma (eng. *World Wide Web Consortioum – W3C*), međunarodnoj organizaciji koja se bavi standardizacijom interneta.

Podaci su smešteni kao znakovni nizovi (eng. *String*) koji se nalaze između tekstualnih oznaka (eng. *tags*) koje ih opisuju. Oznake bi trebalo da budu lako čitljive, tj. da imaju poznato ili lako razumljivo značenje jer one opisuju sadržaj koji se nalazi unutar njih i to ovaj format čini samodokumentujući. Jedna od karakteristika XML-a mogućnost da ga lako čitaju ljudi a da može biti jednostavno obrađen računarom. Podatak i etiketa čine logičku celinu koja se naziva elemenat. Počinje sa početnom oznakom (eng. *start tag*) a završava se sa završnom oznakom (eng. *end tag*). Primer sintakse jednog elementa tj. XML-a:

```
<početnaOznaka>
    Podatak koji sadrži jedan elemenat
</završnaOznaka>
```

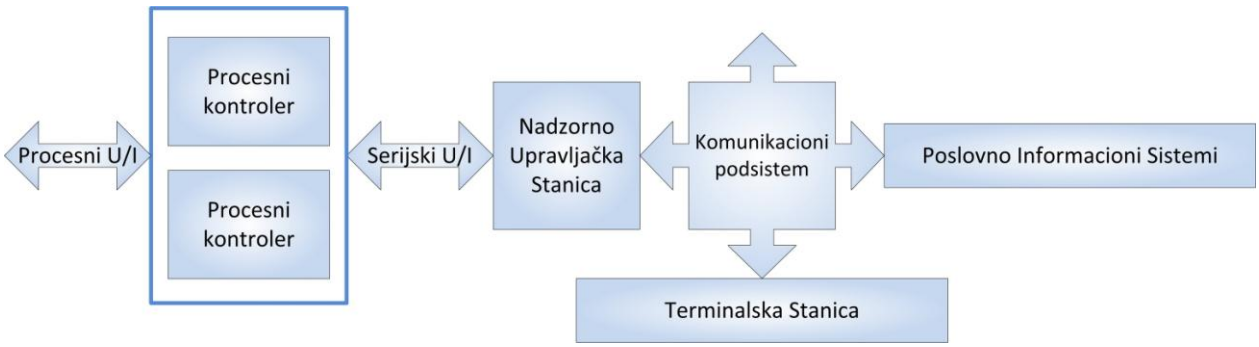
Struktura XML dokumenta je takva da postoji samo jedan korenski (eng. *root*) element, a svi ostali su ugnježdeni elementi (podelementi) korenskog elementa. Kako su podaci i oznake tipa tekstualnih podataka, organizacija elemenata, pa i celog XML-a je u potpunosti prilagodljiv svim oblastima, kao što su razmena podataka, čuvanje podataka itd. Takođe, ova osobina donosi prednost da je XML čitljiv, tj. primeljiv na svim operativnim sistemima koji imaju mogućnost rukovanje tekstualnim sadržajem.

4. oSCADA akviziciono upravljački sistem

Kao server u SCADA sistemu u kojem je implementiran novi protokol korišćeno je rešenje pod nazivom oSCADA, koje predstavlja naslednika akviziciono upravljačkog sistema opšte namene GAUS (*Generalizovani Akviziciono Upravljački Sistem*). Oba sistema su razvijena na katedri za računarsku tehniku Fakulteta Tehničkih nauka u Novom Sadu. Radi lakšeg razumevanja u ovom poglavlju biće objašnjeni elementi GAUS-a, oSCADE i novih implementiranih elemenata servera.

4.1 GAUS

Slika 4.1 prikazuje arhitekturu GAUS sistema čiji će osnovni gradivni elementi biti objašnjeni u daljem tekstu.



Slika 4.1 Arhitektura GAUS programskog rešenja

4.1.1 Procesni kotroler (PK)

Procesni kontroler je mikroprocesorska stanica kojoj je najvažniji zadatak obuhvat mernih signala sa senzora, kao i generisanje pobude nad izvršnim elementima. Bitan deo procesnog kontrolera je podsistem serijskog ulaza/izlaza (sU/I) čija je funkcija razmena poruka sa Nadzorno Upravljačkom Stanicom (NUS). Ukoliko više procesnih kontrolera komuniciraju sa nadzorno upravljačkom stanicom putem zajedničkog komunikacionog kanala, oni se razlikuju po adresi. Procesni kontroler poseduje set funkcija koje omogućuju prosleđivanje prikupljenih podataka ka nadzorno upravljačkoj stanici ili izvršenje primljene komande. Ukoliko je reč o distribuiranom SCADA sistemu procesni kontroleri se nazivaju Udaljene Telemetrijske Stanice (eng. *RTU – Remote Telemetry Station*).

4.1.2 Nadzorno upravljačka stanica (NUS)

Nadzorno upravljačka stanica (NUS) poseduje većinu osobina centralnog računara SCADA sistema. Kao što je već napisano, primarna funkcija NUS-a je objedinjavanje, obrada i prikaz podataka primljenih od procesnog kontrolera. Obrada podataka predstavlja proveru ispravnosti i vrednosti svakog od merenja jer se o eventualnim otkrivenim neregularnostima operater eksplicitno obaveštava formiranjem alarma. Svi podaci se pretvaraju u oblik pogodan za za generisanje upravljačkih akcija, kao i njihov prikaz. Operaterski podsistem unutar nadzorno upravljačke stanice mora obezbediti pregledno i kvalitetno izveštavanje o promenama u sistemu, kao i brzo i tačno prihvatanje operaterovih komandi posredstvom dinamički osvežavanih, posebno definisanih grafičkih i alfanumeričkih prikaza. Podsistem komunikacije sa operaterom je izuzetno značajan elemenat nadzorno upravljačke stanice na osnovu koga se često formira mišljenje o kvalitetu programskog rešenja SCADA sistema.

Na nadzorno upravljačkoj stanici se centralizuju naprioritetnije upravljačke funkcije SCADA sistema što znači da izvršenje dela upravljačkog algoritma može biti distribuirano na procesni kontroler, ali iniciranje svih kontrolnih funkcija i krajnja verifikacija njihovog izvršenja se vrši isključivo na nadzorno upravljačkoj stanici.

Takođe, funkcije nadzora mogu biti distribuirane na dodatne, prostorno udaljene stanice - terminalske stanice (TS). Tako se proširuje mogućnost pristupa podacima o stanju nadziranog procesa, a pristup ovim podacima može biti potpun (svim podacima) ili parcijalan (u skladu sa nekim kriterijumum izbora).

Sličnu funkciju ima i međusobno povezivanje autonomnih nadzorno upravljačkih stanica. Razmenom podataka između dva nezavisna SCADA podsistema, koji kontrolišu različite segmente istog tehnološkog procesa, stiče se celovita slika o njegovom trenutnom stanju. Sličan

cilj ima i hijerarhijsko povezivanje stanica nadzorno upravljačkih stanica gde se superpozicijom podataka primljenih od stanica nižeg nivoa formira zajednička baza podataka. Na osnovu nje, postoji mogućnost da se na višem nivou odredi adekvatna upravljačka akcija, koja se u formi naloga prenosi podređenim stanicama NUS-a. Na najnižem nivou NUS-a (direktno iznad procesnog kontrolera), primljeni nalog se prevodi u niz konkretnih izvršnih akcija. U oba slučaja, izvršne upravljačke funkcije se nikada ne dele između dva autonomna sistema SCADA.

Kako su tehnološki procesi koji SCADA sistem kontroliše često kompleksni, a u cilju ostvarenja potpunije kontrole, klasični SCADA sistem se proširuje stanicom za *podršku u odlučivanju.* Podsistem za podršku u odlučivanju ima kao osnovni cilj da sa dodatnim procesiranjem telemetrisanih podataka obezbedi efikasniji i potpuniji nadzor nad fizičkim procesom. Od njega se, takođe, zahteva da predvidi kritične događaje u sistemu pre nego što oni nastanu ili da ukaže na korektivnu akciju ukoliko se poremećaj već desio. Procedure obrade su u opštem slučaju složene i usko vezane za prirodu nadziranog fizičkog procesa. Osnova za njih su podaci koje SCADA sistem obezbeđuje, uključujući i trend kritičnih procesnih veličina.

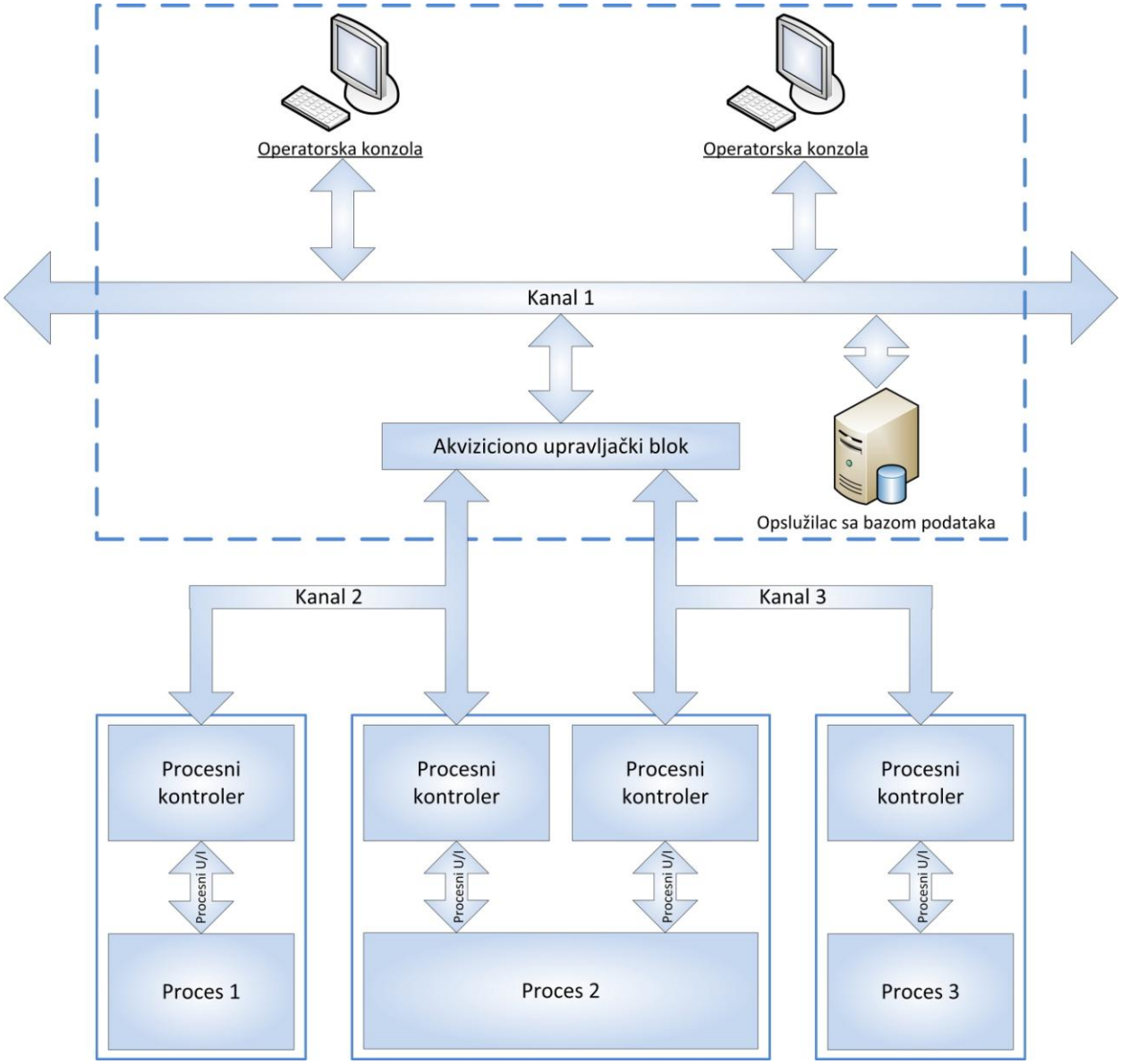
Radi povećanja pouzdanosti, kao jednog od bitnih karakteristika SCADA sistema, često se primenjuje udvajanje fizičke arhitekture na kojoj se programska podrška izvršava u tzv. „tandem" konfiguraciji. U toku rada jedna od tandem stanica je „vodeća", što znači da ona kontroliše komunikaciju sa mrežom procesnih kontrolera. Svi prikupljeni podaci se predaju „pratećoj" stanici, tako da je u svakom trenutku baza podataka u memoriji oba računara identična. Na ovaj način se obezbeđuje mogućnost da u slučaju otkaza vodeće stanice, prateća nastavi dalji rad nesmetano.

Podaci prikupljeni SCADA sistemom su često od velikog interesa i u drugim segmentima poslovnog sistema, kao što su službe održavanja, kontrole kvaliteta, komercijale, planiranja i sl. Iz tog razloga potrebno je obezbediti spregu SCADA sistema sa klasičnim, poslovno - tehnološkim informacionim sistemom, što podrazumeva prenos podataka u vidu datoteka sa podacima iz SCADA sistema do poslovno informacionog sistema putem komunikacionog podsistema.

4.1.3 Komunikacioni podsistem

Komunikacioni podsistem u okviru SCADA sistema čine komunikacioni prenosni kanali i komunikacioni kontroler (server) nadzorno upravljačke stanice. Komunikacioni kanali povezuju nadzorno upravljačku stanicu sa mrežom procesnih kontrolera ili drugim serijski spregnutim komponentama SCADA sistema. Komunikacioni kanal obuhvata svu opremu potrebnu za kvalitetnu razmenu poruka (modemi, koncentratori, prilagodni elementi, i sl.). Kako

bi SCADA sistem bio efikasniji, brzina i propusnost komunikacionog kanala moraju biti na odgovarajućem nivou.



Slika 4.2 Topologija GAUS programskog rešenja

4.1.4 Organizacija podataka unutar GAUS-a

Podaci dobijeni sa senzora, koji su u sirovoj formi i predstavljaju potpuno stanje sistema, nisu pogodni za tumačenje od strane čoveka. Izmerene vrednosti koje predstavljaju kontinualne veličine potrebno je prebaciti u numeričku veličinu označenu jedinicom mere, tzv. inženjersku jedinicu. Diskretne veličine svoje binarno kodovanje zamenjuju opisom stanja fizičkog uređaja.

Na najnižem nivou GAUS posmatra fizički proces kao skup procesnih ulaza i izlaza. U GAUS-u **procesne promenljive** predstavljaju pojedinačne ili grupisane (u slučaju digitalnih

uređaja) opise procesnih ulaza i izlaza, tj. opisa analognih ulaza i izlaza, digitalnih ulaza i izlaza i brojača kao najmanjih celine SCADA sistema. Sistem koristi isključivo ove strukture za rad.

Sve strukture podataka su grupisane u tri manje, karakteristične podgrupe:

- ***fiksna*** – sadrži deo procesne promenljive koji se ne menja, postavlja se isključivo prilikom konfigurisanja sistema. Ova podgrupa obeležja nosi podatke o identifikaciji i opisu promenljive u obliku razumljivom za čoveka, kao i neke bitne fabričke karakteristike senzora ili aktuatora.
- ***privremena*** – sadrži deo koji se privremeno menja i to isključivo od strane operatera SCADA sistema. Vrednosti promenljivih podešavanja uređaja su ručno podešene.
- ***promenljiva*** - sadrži trenutno stanje procesne promenljive i menja se na svaku akviziciju fizičkog sistema.

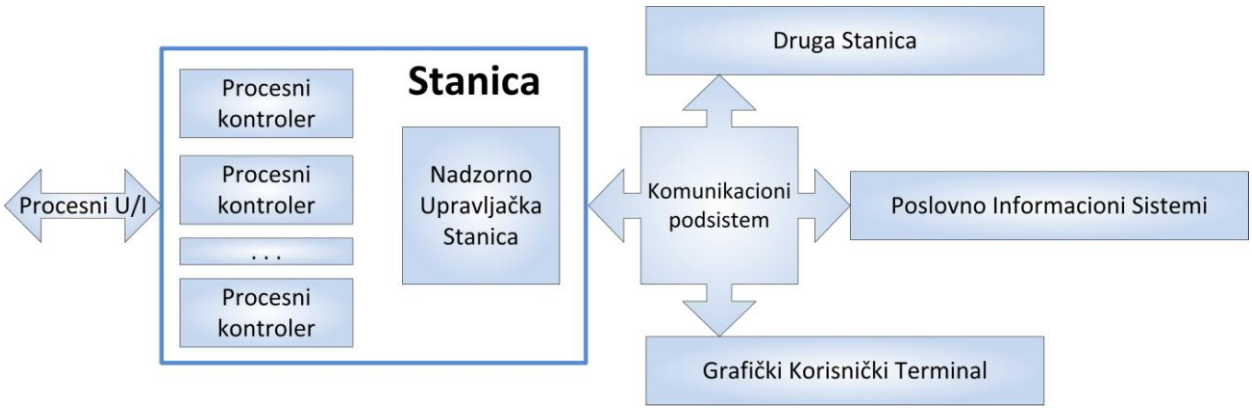
Kako bi se obezbedio efikasni pristup do proizvoljne strukture u bazi podataka GAUS-a, svakoj strukturi se dodeljuje jedinstvena binarna identifikacija (ključ). U sistemu GAUS, ova veličina se naziva **identifikacijom procesne promenljive (PVID).**

4.2 oSCADA

oSCADA je novonastalo rešenje, koje je nasledio generalne koncepte GAUS programskog rešenja koji su se u praksi pokazali kao dobri, ali sa naglaskom na prilagođavanje objektnoj realizaciji programskog rešenja i iskoristivosti iste. Moguće je izvršavanje na Windows i Linux operativnom sistemu, i to na x86 i na x86_64 fizičkim platformama.

4.2.1 Stanica unutar oSCADA sistema

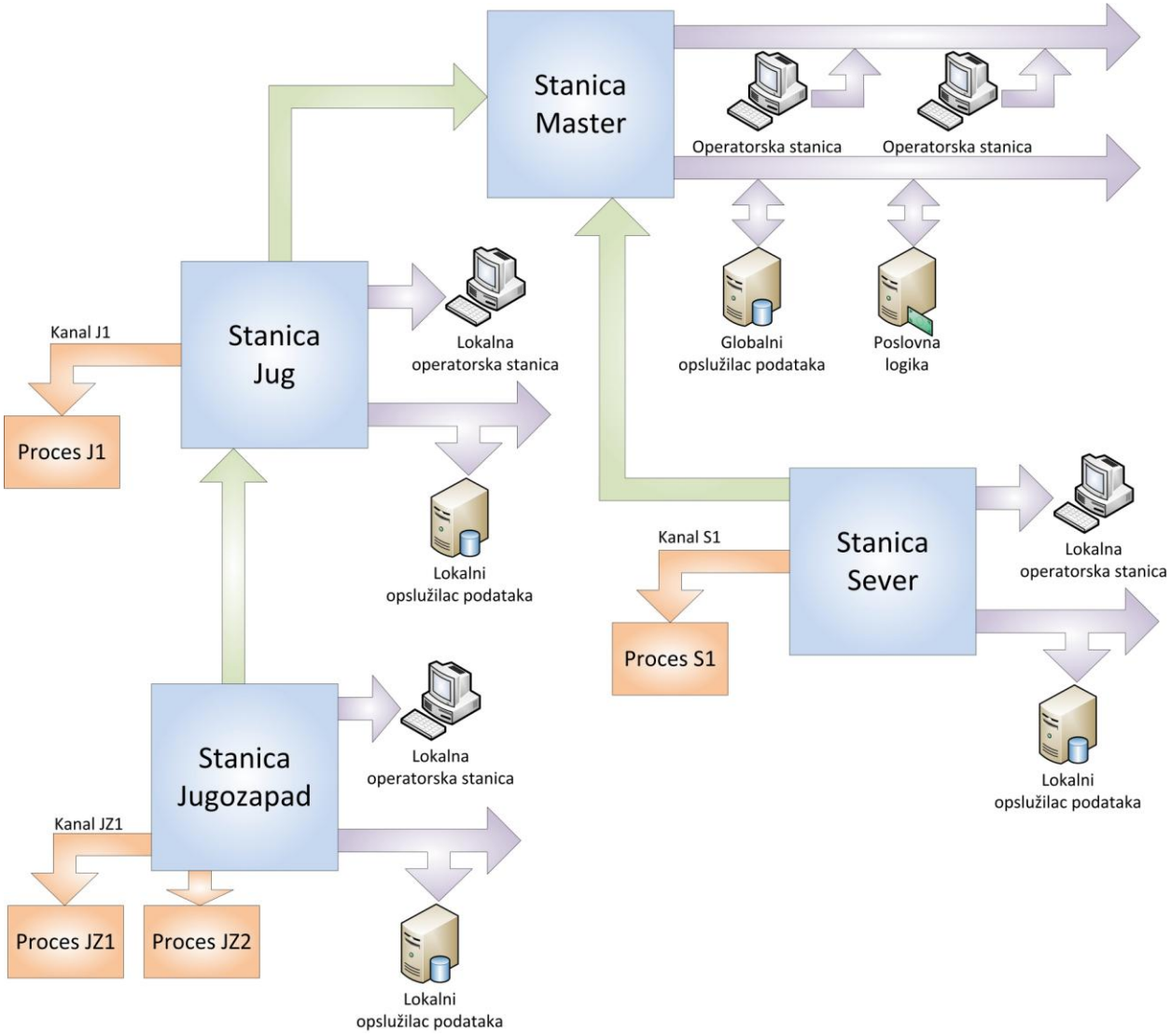
U okviru strukture GAUS programskog rešenja je uočeno da su celine procesni kontroler i nadzorna upravljačka stanica u velikoj meri slične, odnosno da rukuju istim podacima i vrše istu obradu. Zbog poboljšanja performansi celokupnog sistema uvodi se pojam stanice, kao celine koja objedinjuje udaljenu telemetrijsku stanicu i nadzorno upravljačku stanicu (Slika 4.3).



Slika 4.3 Arhitektura oSCADA sistema

Spajanjem ovih celina sistema (procesnog kontrolera i nadzorno upravljačke stanice) izbacuje se deo serijske komunikacije, a sama obrada podataka se približava mestu akvizicije. Distribucijom obrade smanjuju se zahtevi prema procesorskoj moći fizičke arhitekture (jer se obrađuju manje količine podataka), uz pozitivan bočni efekat smanjivanja vremena upravljačke reakcije nakon akvizicije podataka.

U slučaju geografske raširenosti fizičkog procesa kontrolisanog sa strane SCADA, postoji mogućnost vezivanja stanica u proizvoljnu hijerarhijsku organizaciju, gde je svakom od čvorova omogućeno kako pristup fizičkim vrednostima, tako i vrednostima na drugim, podređenim stanicama. Što znači, da stanica može i da prikuplja podatke i upravlja sa uređajima preko industrijskih protokola (Modbus, Hart, …) ili sakuplja podatke od stanica koje su joj po hijerarhiji podređeni. Ovakvom organizacijom podataka je omogućena podela sistema na podsisteme (manje celine), samim tim je omogućena i podela kako akvizicije, tako i upravljanja. Zbog mogućnosti podele sistema na podsisteme, nameće se i minimalizacija količine podataka na jednoj stanici, samim tim, operateru stanice sa daju na raspolaganje isključivo oni podaci koji pripadaju njegovoj oblasti odgovornosti. Radi lakšeg praćenja procesa i upravljanja, svakoj stanici je omogućeno da se na nju prikači operaterska konzola. Hijerarhijom stanica se uvodi i hijerarhijsko klasifikovanje operatera, gde je operater stanice na višem nivou hijerarhije u mogućnosti da nadglasa akcije operatera stanice na nižem nivou hijerarhije (Slika 4.4).

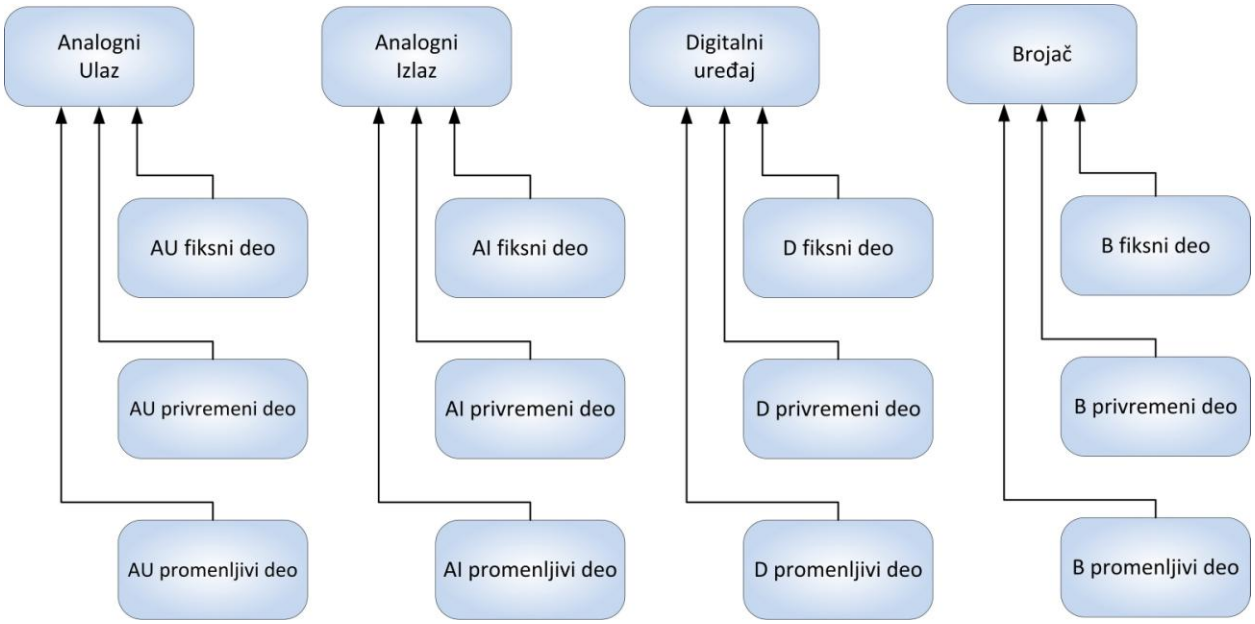


Slika 4.4 Primer hijerarhijske konfiguracije sistema

Prikupljanje podataka od podređenih stanica i propagiranjem istih prema instancama više odgovornosti omogućuje postojanje centralne baze podataka u realnom vremenu, što je pogodno za opsluživanje svih vrsta poslovnih aplikacija kao i za dobijanje uvida u celokupni sistem.

4.2.2 Koncepti nasleđeni iz GAUS-a primenjeni u oSCADA

oSCADA kao naslednik GAUS programskog rešenja koristi koncept **procesnih promenljivih** kojima opisuje analogne ulaze i izlaze, digitalne ulaze i izlaze, i brojače. Svaka procesna promenljiva treba da sadrži tri dela (grupe) koje u potpunosti opisuju sve njene osobine. Slika 4.5 prikazuje strukturu procesnih promeljivih.



Slika 4.5 Struktura procesnih promeljivih

Kartica u oSCADA sistemu kao i u GAUS sistemu predstavlja sliku fizičkog uređaja, koja objedinjuje nekoliko procesnih promenljivih nekog postrojenja, pri tome definiše način njihove akvizicije, obrade i distribucije. Dobijene informacije kartica prosleđuje u kanal kojem je pridružena.

Kanal u GAUS-u predstavlja fizičku spregu preko koje se vrši komunikacija sa uređajem (RS232, Ethernet, ...). Apstrahovanjem fizičke sprege uvodi uniformnost između komunikacije preko različitih uređaja. Zahvaljujući ovome, u trenutku razvoja kartica, programer ne mora biti svestan fizičkog medijuma preko koje se vrši komunikacija. U oSCADA sistemu kanal dobija drugu ulogu - postaje bafer ulaznih i izlaznih poruka.

Svakom kanalu je pridružen **protokol** koji prevodi poruku iz stanice u oblik koji je razumljiv uređaju sa kojim se komunicira. Pretvorenu poruku šalje preko jednog od dodeljenih prolaza.

Prolaz predstavlja mrežni uređaj preko koga se vrši komunikacija.

Pored ovih jasno definisanih celina, oSCADA sistem kao i GAUS sadrži nekoliko programskih niti koje su zadužene za pokretanje najbitnijih zadataka sistema. Vrlo bitna je nit za akviziciju podataka, koja inicira slanje zahteva uređaju za novim očitavanjem stanja procesnih izlaza. Podatke, koji su pristigli sa fizičkih uređaja na oSCADA, obrađuje nit za obradu, koja je odgovorna i za proveravanje ispravnosti vrednosti procesnih promenljivih kao i za signaliziranje o neispravnosti u njima.

4.3 RAUS protokol

U cilju omogućavanja komunikacije između serverske stanice u oSCADA sistemu i operatorskih terminala, razvijen je protokol nazvan RAUS. Ovaj protokol omogućuje distribuciju SCADA podataka i baziran je na XML porukama.

Za potrebe protokola, formirana je RAUS kartica koja objedinjuje više procesnih promenljivih i njihove podatke iz internog oSCADA formata podataka prevodi u XML poruke RAUS protokola.

Komunikacija kroz ovaj protokol sastoji se iz dve faze. U prvoj se, samo jednom, nakon uspostavljanja veze šalju detaljni opisi podataka (metapodaci) procesnih promenljivih koje su konfigurisane za datu RAUS karticu (obično je to podskup promenljivih za čiji nadzor je zadužen operater koji se povezuje sa datom karticom – prikazano na Slika 4.6) na osnovu kojih se može generisati izgled grafičko korisničke sprege na klijentskoj strani. U drugoj fazi se šalju podaci na svaku promenu vrednosti podskupa procesnih promenljivih. Prilikom slanja se u svaku poruku integriše XML zaglavlje

```
<?xml version="1.0" encoding="utf-8"?>
```

kako bi se eksplicitno naglasilo da je reč o XML formatu 1.0 kodnog sistema UTF-8.

Slanje metapodataka (opšteg opisa podataka) se kreće od stanice prema terminalima, i prenosi opis podataka iz dostupne konfiguracije sistema datom terminalu.

```
<metadata>
     <Identifikacija procesne promenljive>
          <Tip procesne promenljive>
               <Deo Procesne promenljive>
                    <VALUE>
                         <Name>Ime promenljive</Name>
                         <Type>Tip promenljive</Type>
                    </VALUE>
                    …
               </ Deo Procesne promenljive >
               …
          </ Tip procesne promenljive >
     </ Identifikacija procesne promenljive >
     …
</metadata>
```

Slanje vrednosti prenosi aktuelne vrednosti procesnih promenljivih. U trenutnoj implementaciji RAUS protokola, slanje vrednosti se obavlja u smeru od servera prema klijentu.

```
<data>
     <Identifikacija procesne promenljive>
          <Tip procesne promenljive>
               <Deo Procesne promenljive>
                    <VALUE>
                         <Name>Ime promenljive</Name>
                         <CurrentValue>
                              Trenutna vrednost promenljive
                         </CurrentValue>
                    </VALUE>
                    …
               </ Deo Procesne promenljive >
               …
          </ Tip procesne promenljive >
     </ Identifikacija procesne promenljive >
```

Data ili Metadata predstavljaju korenski element XML poruke koja se šalje i pritom predstavljaju oznaku da li je reč o metapodacima ili trenutnim vrednostima.

Identifikacija procesne promenljive je podelement *data/metadata* elementa čije ime sadrži kombinaciju karaktera koja predstavlja jedinstveni ključ procesne promenljive unutar oSCADA sistema.

Tip procesne promenljive je podelement elementa *Identifikacija procesne promenljive*. Govori o tome koji je tip procesnog ulaza ili izlaza u pitanju (analogni ulaz/izlaz, digitalni ulaz/izlaz, brojač).

Deo procesne promenljive je podelement elementa *Tip procesne promenljive*. Daje oznaku unutar poruke koji se deo poruke opisuje (fiksni, privremeni ili promenljivi).

VALUE je podelement elementa *Deo procesne promenljive* i predstavlja oznaku elementa u čijem se sadržaju (podelementima) prenose podaci koji opisuju procesnu promenljivu.

Name je podelement elementa *VALUE* čiji sadržaj predstavlja ime podatka koji se prenosi. Polje *Name* je jedinstvenog sadržaja unutar jedne procesne promenljive, tj. u jednoj procesnoj promenljivoj se ne mogu naći dva podatka sa istim imenom.

Type je podelement elementa *VALUE* u slučaju prenosa metapodataka. Podaci koji sadrži ovaj element predstavljaju tip podataka sadržaja elementa *Name* sa kojim je u istom hijerarhiskom nivou.

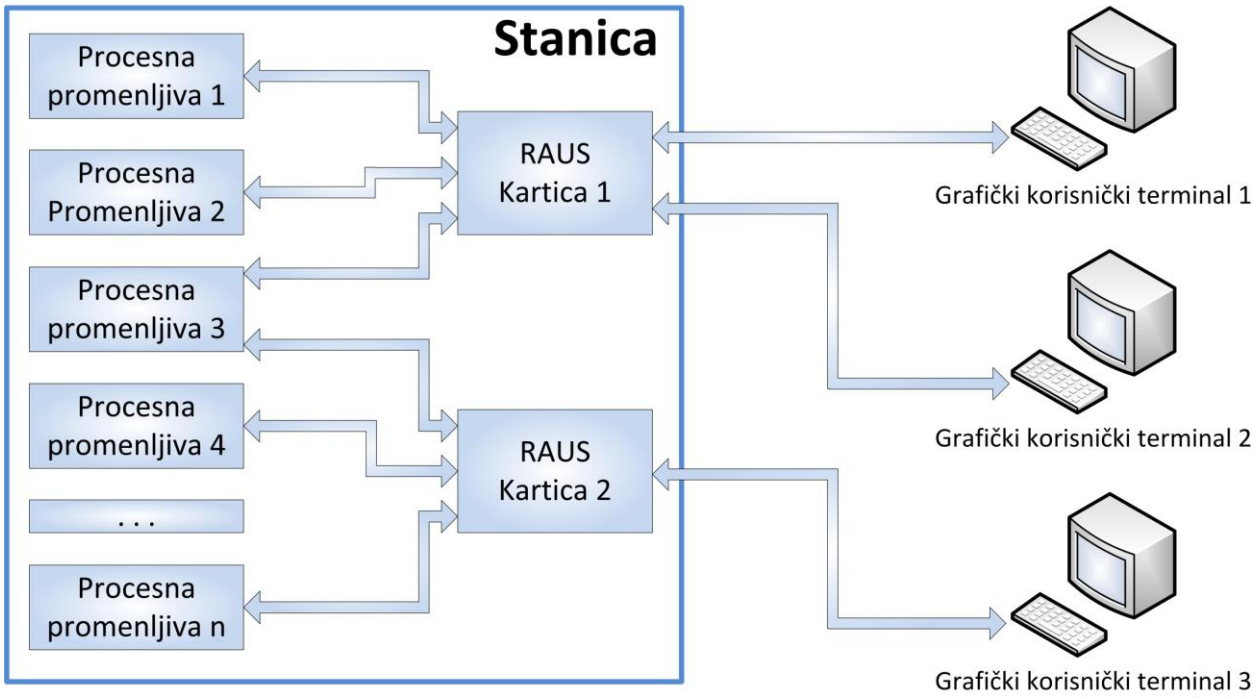
CurrentValue je podelement elementa *VALUE* u slučaju prenosa trenutnih vrednosti. Podaci koji sadrži ovaj element predstavljaju trenutnu vrednost podataka sadržaja elementa *Name* sa kojim je u istom hijerahiskom nivou.

Ukoliko je tip podataka koji se šalje klasa, njeni atributi mogu biti i nizovi, pa je to sve potrebno smestiti unutar jednog podatka elementa *CurrentValue*. Odlučeno je da znakovi za razdvajanje logičkih celina unutar jednog niza znakova budu "," "|" i "#". Spakovana struktura jedne klase bi u ovom slučaju trebalo da izgleda na sledeći način.

```
<CurrentValue>
     promenljiva1[n],njenaVrednost|
     promenljiva2[n],njenaVrednost|
     promenljiva3[n],njenaVrednost#
     promenljiva1[n+1],njenaVrednost|
     promenljiva2[n+1],njenaVrednost|
     promenljiva3[n+1],njenaVrednost
</CurrentValue>
```

Primeri slanja metapodataka za potrebe dinamičkog generisanja korisničke grafičke sprege i primeri slanja trenutnih vrednosti podataka za potrebe popunjavanja generisanih polja unutar grafičke korisničke sprege preuzeti su iz slanja oSCADA sistema i nalaze se u Dodatak A – Primeri RAUS komunikacije.

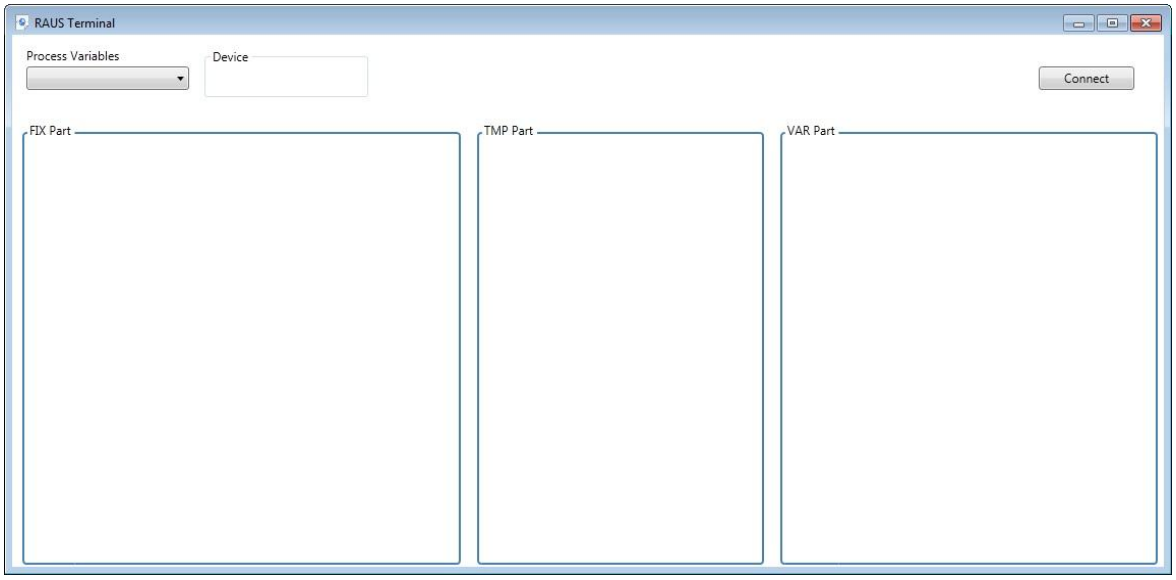
U okviru jedne stanice, moguće je imati više RAUS kartica i na svakoj od njih drugi podskup dodeljenih procesnih promenljivih (prikazano na Slika 4.6). Operater se vezuje na RAUS karticu koja sadrži podskup procesnih promenljivih za koje je on ovlašćen i nadležan.



Slika 4.6 Konfiguracija oSCADA stanice sa više RAUS kartica

5. Grafička korisnička sprega - klijentska strana

Slika 5.1 prikazuje izgled grafičke korisničke sprege - klijentske strane, nazvane "RAUS Terminal".



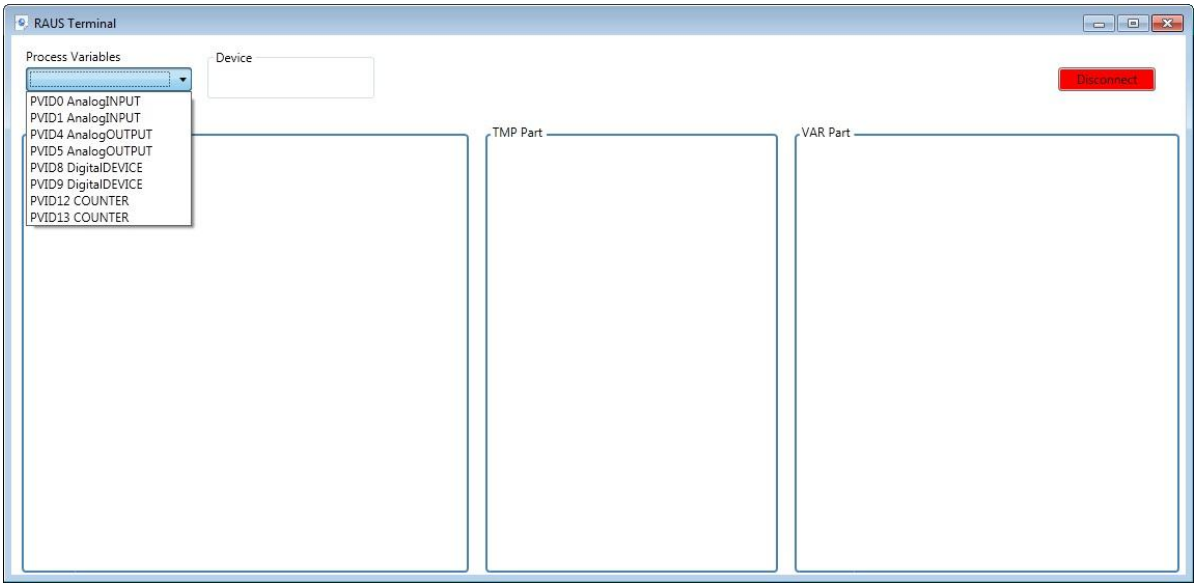
Slika 5.1 Izgled grafičke korisničke sprege – klijentske strane

Izrađena je WPF tehnologijom u jednom prozoru u kome se nalaze elementi koji će biti ukratko opisani:

- Dugme *Connect – Disconnect*:
  - u slučaju da veza nije uspostavljena pritiskom na *Connect* uspostavlja se veza i preuzimaju podaci

- u slučaju da je veza uspostavljena pritiskom na *Disconnect* prekida se veza i preuzimanje podataka.

- Padajuće polje (eng. *Combo Box*) *Process Variables* sadrži listu identifikacije procesnih promenljivih (PVID) sa njihovim tipom (Slika 5.2).

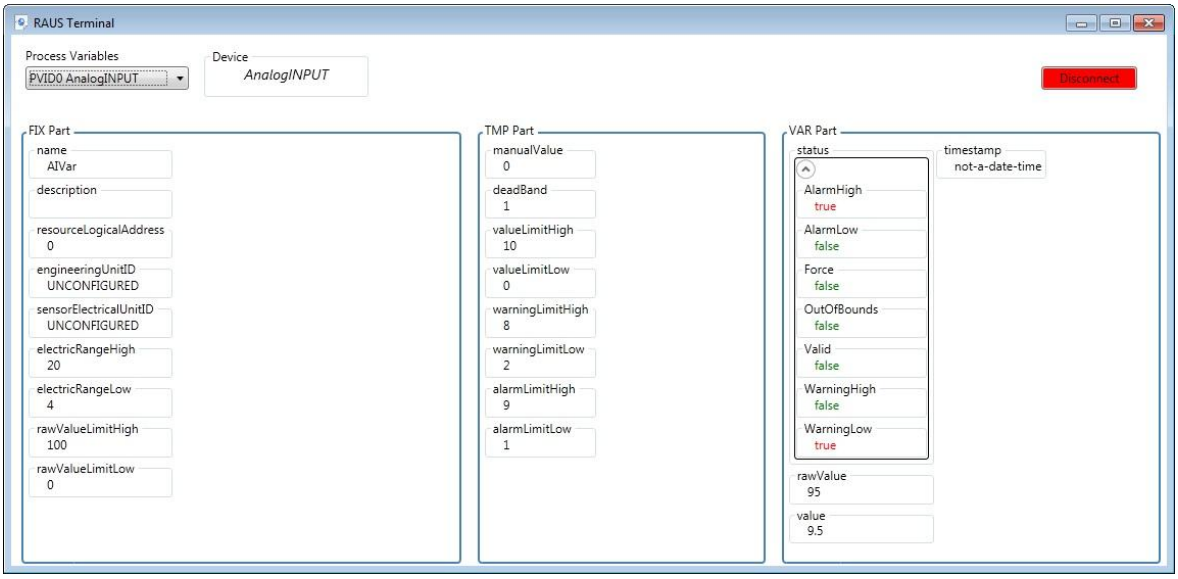


Slika 5.2 Prikaz padajućeg polja nakon uspostavljanja veze

- Tekstualno polje *Device* prikazuje tip procesne promenljive odabranog u padajućem polju *Process Variables*
- Paneli *FIX Part*, *TMP Part* i *VAR Part* za prikaz fiksnih, trenutnih i promenljivih podataka respektivno, koji opisuju izabranu procesnu promenljivu.

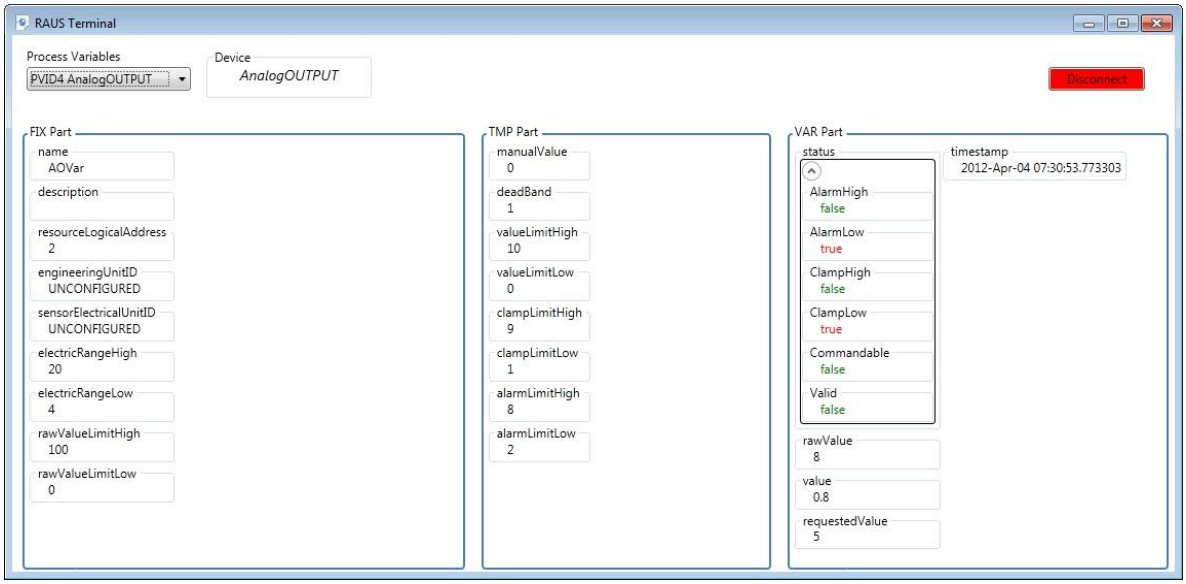
Trenutno raspoloživi tipovi procesnih promenljivih u oSCADA centralnom računaru i njihov prikaz u grafičko korisničkoj sprezi su:

- **analogni ulaz *(eng. analog in)***



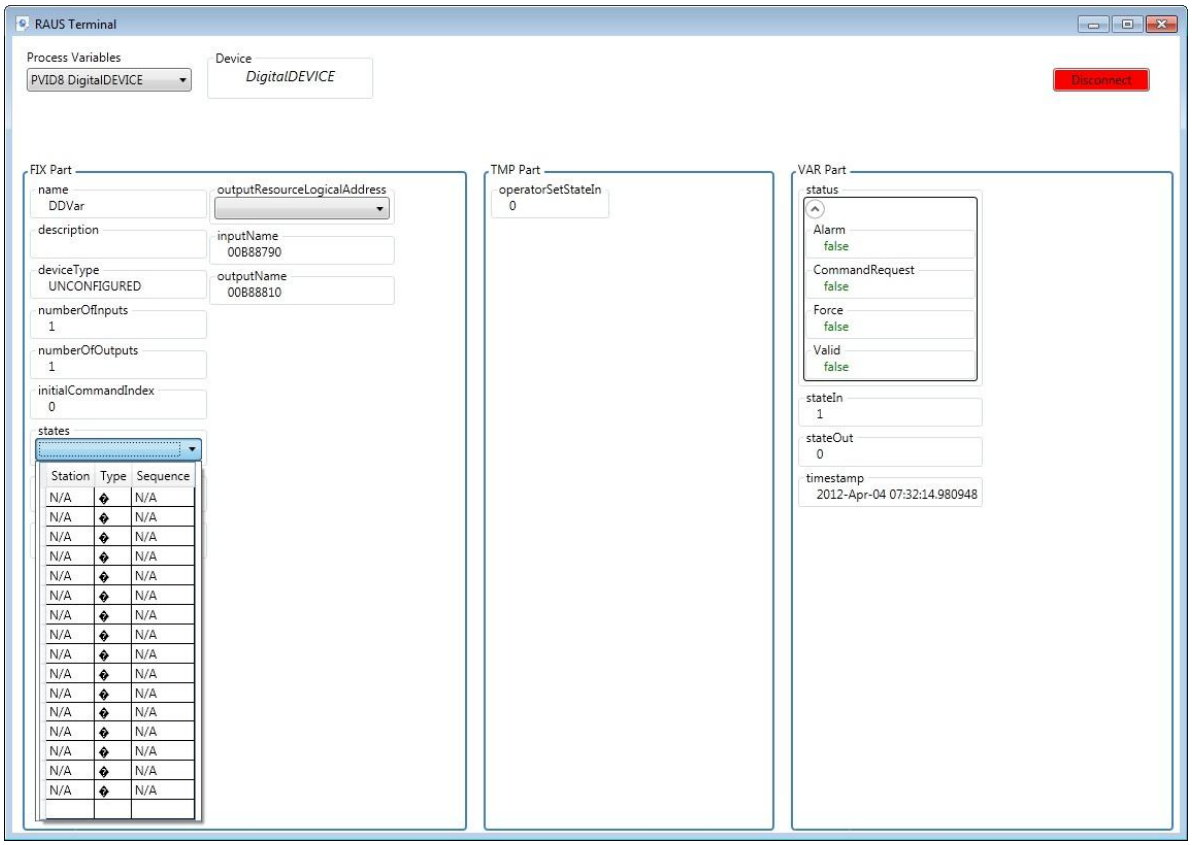
Slika 5.3 Grafičko korisnička sprega – prikaz procesne promenljive tipa analogni ulaz

- **analogni izlaz *(eng. analog out)***



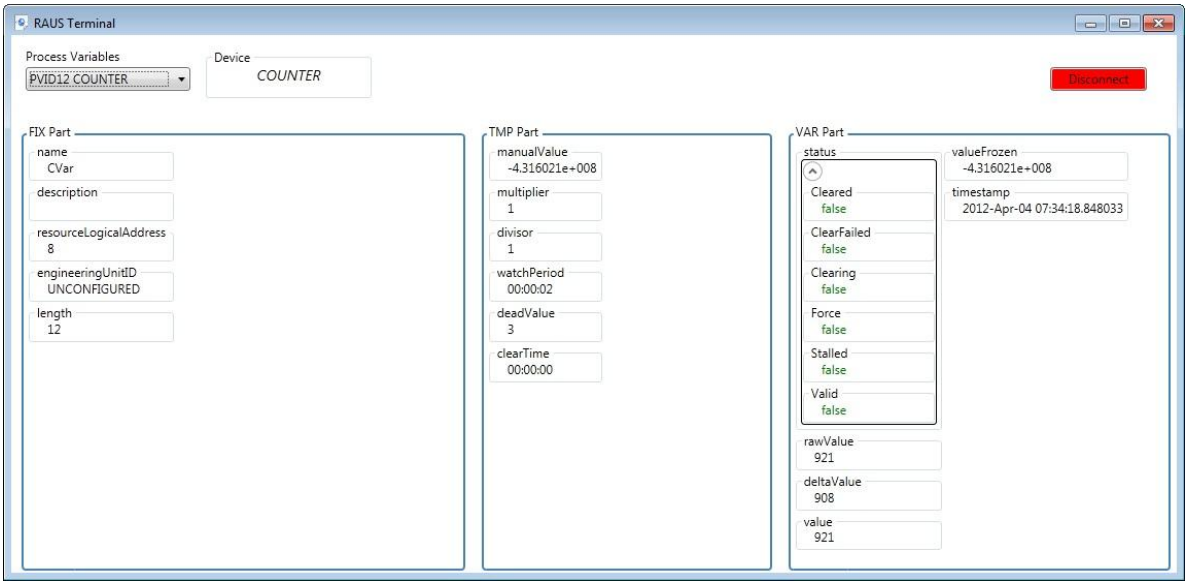
Slika 5.4 Grafičko korisnička sprega – prikaz procesne promenljive tipa analogni izlaz

- **digitalni uređaj *(eng. digital device)***



Slika 5.5 Grafičko korisnička sprega – prikaz procesne promenljive tipa digitalni uređaj

- brojač (eng. counter)



Slika 5.6 Grafičko korisnička sprega – prikaz procesne promenljive tipa brojač

5.1 Kratak opis funkcionisanja klijentske strane

U delu koji sledi biće kratko opisan algoritam *RAUS Terminal* aplikacije.

Prilikom pritiska na dugme *Connect*:

- uspostavlja se TCP veza prema unapred određenoj IP adresi i portu,
- pokreće se nit za primanje podataka sa te veze i samo prvi put se preuzimaju metapodaci iz kojih se izdvaja PVID lista. Nakon toga primaju se trenutne vrednosti i upisuju u rečnik *dictNameValue,* gde je ime podatka ključ a vrednost predstavlja podatak rečnika. Prijemna nit se nakon ovog procesa stavlja u stanje spavanja na dve sekunde.
- popunjava se rečnik (eng. *dictionery*) *dictPVIDObject* koji se sastoji od identifikacije procesne promenljive (koja je u svojstvu ključa) i objekata koji sadrže parsirane informacije opisa procesnih promenljivih,
- za svaki podatak iz svake procesne promenljive se prema njegovom tipu popunjava rečnik *dictNameType*, gde ime podatka predstavlja ključ a tip podatka – podatak unutar rečnika.
- prema tipu promenljive se određuje grafička komponenta u kojoj će biti prikazan i upisuje u predviđeni rečnik *dictNameGUIComponent,* gde je ime podatka ključ a grafička komponenta podatak.

Odabiranjem jedne od ponuđenih procesnih promenljvih iz polja *Process Variables:*

- pronalazi se određena procesna promenljiva čija identifikacija predstavlja ključ u rečniku *dictPVIDObject*,
- nakon toga se selekcijom uz pomoć imena podatka iz rečnika *dictNameGUIComponent* određuje grafička komponenta koja se prikazuje u odgovarajućem panelu (*FIX Part*, *TMP Part* i *VAR Part*) jer RAUS poruka sadrži jasno definisane delove poruka,
- iz rečnika se *dictNameValue* se periodično na dve sekunde, upotrebom tajmera, obrađuju podaci prema njihovom tipu i upisuju u grafičke komponente.

Prilikom pritiska na dugme *Disconnect:*

- Zatvara se TCP veza
- Završava se nit za primanje podataka

5.2 Osobine i prednosti implementirane klijentske strane

Bitne osobine i prednosti kod realizacije dinamičkog kreiranja grafičke sprege su:

- ***Proširivost*** – Ne postoje unapred napravljena polja za određene podatke koja bi se punila nakon ostvarivanja veze sa centralnim računarom. Sve grafičke komponente se generišu u odnosu na XML sadržaj RAUS poruke prilikom primanja metapodataka, a zatim se prilikom odabira procesne promenljive iscrtavaju u *wrap* panele nezavisno od broja podataka koji im se šalje niti od njihovog sadržaja. U svakom trenutku se bilo koja procesna promenljiva može proširiti sa proizvoljnim brojem akviziranih podataka. Za ispravan prikaz potrebno je samo opet primiti metapodatke. Jedino ograničenje u proizvoljnom proširenju je definisanost određenih tipova podataka u vidu struktura/klasa koje se moraju parsirati, u kojima nije implementirano parsiranje podataka pakovanog na više od tri nivoa (prikazano u poglavlju 4.3 RAUS protokol). Svi ostali nepoznati tipovi podataka prikazuju se u tekstualnom polju.
- ***Mogućnost optimizacije slanja podataka*** **–** U okviru svake procesne promenljive, postoji rečnik *dictNameValue* čijim elementima se može pristupiti uz pomoć ključa koji predstavlja ime podatka. Kao posledica ove osobine, pri akviziranju poželjno je prenositi samo vrednost podataka koji su promenjeni. To donosi značajnu optimizaciju prenosa podataka koja je i više nego potrebna, jer je količina podataka prenešena putem RAUS protokola dosta veća u odnosu na bilo koji binarni protokol.

6. Zaključak

Ovim radom je realizovano jedno rešenje klijentskog protokola SCADA sistema. Detaljno su predstavljeni formati prenosa podataka u vidu XML struktura kao i redosled slanja.

Urađena je implementacija slanja poruka RAUS protokolom u oSCADA sistemu koji je u razvoju, koristeći prednosti objektne paradigme.

Klijentska strana napravljena u WPF tehnologiji predstavlja samo jedan deo funkcionalnosti koja se zahteva od grafičke korisničke sprege, ali realizovani deo dokazuje ispravnost koncepta RAUS protokola. Dinamičko generisanja polja omogućava gotovo potpunu nezavisnost aplikacije od formata podataka poslatih sa klijentske strane a samim tim i proširivost novim tipovima procesnih promenljivih u sklopu oSCADA sistema. Pravci daljeg razvoja su:

- Optimizacija slanja podataka od stanice prema grafičkoj korisničkoj sprezi. Prilikom slanja vrednosti potrebno je poslati samo prvi put fiksne i privremene delove poruka a kasnije slati samo promenljive delove i to samo one gde postoje promene vrednosti.
- Kao sastavni deo optimizacije potrebno je razviti slanje podataka samo skupa izabranih procesnih promenljivih koje se trenutno posmatraju u okviru grafičke korisničke sprege, kao način smanjivanja opterećenja komunikacionih kanala.
- U okviru RAUS protokola planirana je binarizacija slanja trenutnih vrednosti procesnih promenljivih. Očekivani rezultat je optimizacija veličine poruka koje čine većinu podataka RAUS protokola koji zauzimaju komunikacione kanale.
- Uvođenje komunikacije u smeru od klijentske aplikacije prema centralnom računaru, radi prenosa komandi i direktiva.
- Dalji razvoj klijentske aplikacije radi upotpunjavanja njene funkcionalnosti.
- Prevođenje i izvršavanje klijentske strane sa tehnologijom Silverlight u okviru internet pretraživača.

7. Literatura

[1] *B.Atlagić*, GAUS – Generalizovani akviziciono upravljački sistem, Novi Sad, 2005.

[2] *National Communications System* - Supervisory Control and Data Acquisition (SCADA) Systems, October 2004.

[3] *Department of the Army*, *TM 5-601* - Supervisory Control and Data Acquisition (SCADA) Systems for Command, Control, Communications, Computer, Intelligence, Surveillance, and Reconnaissance (C4ISR) Facilities, 21 January 2006.

[4] *Laslo Kraus* – Programski jezik C++, Akademska misao, Beograd, 2003.

[5] *Elliote Rusty Harold, W. Scott Means*, XML za programere, Mikroknjiga, Beograd, 2006.

[6] *Adam Nathan*, WPF 4 Unleashed, 2010.

[7] *Šagi Mihalj,* Predlog jedne arhitekture savremenog SCADA sistema, Novi Sad, 2011.

[8] Microsoft Developer Network Library, www.msdn.microsoft.com

[9] Wikipedia, the free encyclopedia, www.wikipedia.com

[10] Boost C++ libraries, www.boost.org

Dodatak A – Primeri RAUS komunikacije

U nastavku, dat je primer **slanja metapodataka** za potrebe dinamičkog generisanja korisničke sprege.

```
<?xml version="1.0" encoding="utf-8"?>
<metadata>
  <PVID0>
    <AnalogINPUT>
      <FIX>
        <VALUE>
          <Name>name</Name>
          <Type>std::string</Type>
        </VALUE>
        <VALUE>
          <Name>description</Name>
          <Type>std::string</Type>
        </VALUE>
        <VALUE>
          <Name>resourceLogicalAddress</Name>
          <Type>short</Type>
        </VALUE>
        <VALUE>
          <Name>engineeringUnitID</Name>
          <Type>class CProcessVariableID</Type>
```

```
        </VALUE>
        <VALUE>
          <Name>sensorElectricalUnitID</Name>
          <Type>class CProcessVariableID</Type>
        </VALUE>
        <VALUE>
          <Name>electricRangeHigh</Name>
          <Type>float</Type>
        </VALUE>
        <VALUE>
          <Name>electricRangeLow</Name>
          <Type>float</Type>
        </VALUE>
        <VALUE>
          <Name>rawValueLimitHigh</Name>
          <Type>unsigned int</Type>
        </VALUE>
        <VALUE>
          <Name>rawValueLimitLow</Name>
          <Type>unsigned int</Type>
        </VALUE>
      </FIX>
      <TMP>
        <VALUE>
          <Name>manualValue</Name>
          <Type>float</Type>
        </VALUE>
        <VALUE>
          <Name>deadBand</Name>
          <Type>float</Type>
        </VALUE>
        <VALUE>
          <Name>valueLimitHigh</Name>
          <Type>float</Type>
        </VALUE>
        <VALUE>
          <Name>valueLimitLow</Name>
          <Type>float</Type>
        </VALUE>
```

```
        <VALUE>
          <Name>warningLimitHigh</Name>
          <Type>float</Type>
        </VALUE>
        <VALUE>
          <Name>warningLimitLow</Name>
          <Type>float</Type>
        </VALUE>
        <VALUE>
          <Name>alarmLimitHigh</Name>
          <Type>float</Type>
        </VALUE>
        <VALUE>
          <Name>alarmLimitLow</Name>
          <Type>float</Type>
        </VALUE>
      </TMP>
      <VAR>
        <VALUE>
          <Name>status</Name>
          <Type>class CAnalogInputVarStatus</Type>
        </VALUE>
        <VALUE>
          <Name>rawValue</Name>
          <Type>unsigned int</Type>
        </VALUE>
        <VALUE>
          <Name>value</Name>
          <Type>float</Type>
        </VALUE>
        <VALUE>
          <Name>timestamp</Name>
          <Type>class boost::posix_time::ptime</Type>
        </VALUE>
      </VAR>
    </AnalogINPUT>
  </PVID0>
…
</metadata>
```

Sledi primer **slanja trenutnih vrednosti podataka** za potrebe popunjavanja generisanih polja unutar grafičke korisničke sprege.

```
<?xml version="1.0" encoding="utf-8"?>
<data>
  <PVID0>
    <AnalogINPUT>
      <FIX>
        <VALUE>
          <Name>name</Name>
          <CurrentValue>AIVar</CurrentValue>
        </VALUE>
        <VALUE>
          <Name>description</Name>
          <CurrentValue/>
        </VALUE>
        <VALUE>
          <Name>resourceLogicalAddress</Name>
          <CurrentValue>0</CurrentValue>
        </VALUE>
        <VALUE>
          <Name>engineeringUnitID</Name>
          <CurrentValue>524287</CurrentValue>
        </VALUE>
        <VALUE>
          <Name>sensorElectricalUnitID</Name>
          <CurrentValue>524287</CurrentValue>
        </VALUE>
        <VALUE>
          <Name>electricRangeHigh</Name>
          <CurrentValue>20</CurrentValue>
        </VALUE>
        <VALUE>
          <Name>electricRangeLow</Name>
          <CurrentValue>4</CurrentValue>
        </VALUE>
        <VALUE>
```

```
            <Name>rawValueLimitHigh</Name>
            <CurrentValue>100</CurrentValue>
          </VALUE>
          <VALUE>
            <Name>rawValueLimitLow</Name>
            <CurrentValue>0</CurrentValue>
          </VALUE>
        </FIX>
        <TMP>
          <VALUE>
            <Name>manualValue</Name>
            <CurrentValue>0</CurrentValue>
          </VALUE>
          <VALUE>
            <Name>deadBand</Name>
            <CurrentValue>1</CurrentValue>
          </VALUE>
          <VALUE>
            <Name>valueLimitHigh</Name>
            <CurrentValue>10</CurrentValue>
          </VALUE>
          <VALUE>
            <Name>valueLimitLow</Name>
            <CurrentValue>0</CurrentValue>
          </VALUE>
          <VALUE>
            <Name>warningLimitHigh</Name>
            <CurrentValue>8</CurrentValue>
          </VALUE>
          <VALUE>
            <Name>warningLimitLow</Name>
            <CurrentValue>2</CurrentValue>
          </VALUE>
          <VALUE>
            <Name>alarmLimitHigh</Name>
            <CurrentValue>9</CurrentValue>
          </VALUE>
          <VALUE>
            <Name>alarmLimitLow</Name>
```

```
            <CurrentValue>1</CurrentValue>
          </VALUE>
        </TMP>
        <VAR>
         <VALUE>
           <Name>status</Name>
           <CurrentValue>
             AlarmHigh,false|
             AlarmLow,false|
             Force,false|
             OutOfBounds,false|
             Valid,false|
             WarningHigh,false|
             WarningLow,false|
           </CurrentValue>
         </VALUE>
         <VALUE>
           <Name>rawValue</Name>
           <CurrentValue>50</CurrentValue>
         </VALUE>
         <VALUE>
           <Name>value</Name>
           <CurrentValue>5</CurrentValue>
         </VALUE>
         <VALUE>
           <Name>timestamp</Name>
           <CurrentValue>not-a-date-time</CurrentValue>
         </VALUE>
       </VAR>
    </AnalogINPUT>
  </PVID0>
</data>
```