UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Primož Rebec

Vmesniki OPC v industrijskih nadzornih sistemih

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: izr. prof. dr. Uroš Lotrič

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

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

Št. naloge: 00045/2010
Datum: 04.11.2010

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

Kandidat: **PRIMOŽ REBEC**

Naslov: **VMESNIKI OPC V INDUSTRIJSKIH NADZORNIH SISTEMIH**
OPC INTERFACES IN INDUSTRIAL CONTROL SYSTEMS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V zadnjem času je v industriji vedno večja potreba po integraciji nadzornih sistemov ter ostalih višjih sistemov vodenja z računalniškimi sistemi za avtomatizacijo proizvodnih procesov. Za povezovanje teh sistemov se danes v veliki meri uporabljajo vmesniki OPC. Vmesnike OPC v nalogi preglejte in jih ovrednotite na realnem problemu povezovanja lastnega nadzornega sistema s klasičnim računalniškim sistemom za vodenje procesov.

Mentor:
prof. dr. Uroš Lotrič

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Primož Rebec,

z vpisno številko 63040141,

sem avtor/-ica diplomskega dela z naslovom:

naslovtodo

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  izr. prof. dr. Uroša Lotriča

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 18. 4. 2011 Podpis avtorja/-ice: _____

Zahvala

Zahvaljujem se mentorju na fakulteti izr. prof. dr. Urošu Lotriču za pomoč in usmerjanje pri izdelavi diplomske naloge. Zahvaljujem se tudi staršem za vso podporo pri študiju.

Kazalo

Povzetek

V diplomski nalogi smo opisali implementacijo programskih arhitektur odjemalcev za vmesnike OPC. OPC je standard, ki se v avtomatizaciji industrijskih procesov uporablja za povezovanje višjenivojskih programskih sistemov s tistimi na nižjih nivojih. Višjenivojski sistemi predstavljajo nadzorne sisteme SCADA, sisteme za upravljanje proizvodnje MES in sisteme za planiranje proizvodnje ERP. Z arhitekturo odjemalec-strežnik OPC jih povežemo z nizkonivojskimi sistemi, kot so krmilni računalniki in porazdeljeni sistemi DCS.
V industrijski avtomatizaciji je že dobro uveljavljen standard klasičnega OPC: OPC DA in njegovih izpeljank. V letu 2009 je fundacija OPC, ki skrbi za ta standard izdala specifikacijo nove, poenotene arhitekture OPC Unified Architecture (OPC UA). V primerjavi s klasično arhitekturo OPC prinaša nove funkcije s poudarkom na zmogljivosti, platformni neodvistnosti, varnosti ter omrežnih in internetnih komunikacijah. Sledje so bile v klasični arhitekturi slabo podprte, kar je bila za nas ena od glavnih motivacij za implementacijo in testiranje.
Različne arhitekture OPC smo vgradili in testirali v sistemu SCADA, ki smo ga v celoti razvili sami. Aplikacija kot celota pa je bila namenjena nadzoru ogrevanja in ostalih instalacij v industrijski zgradbi. Testiranje različnih arhitektur odjemalcev in strežnikov OPC je vključevalo postavitev in konfiguracijo strežnika OPC, konfiguracijo krmilnih računalnikov, razvoj odjemalcev in njihovo vključitev v sistem SCADA ter testiranje funkcij, ki jih določen tip arhitekture ponuja. V zaključnem delu smo opisali bistvene ugotovitve in predloge za nadaljevanje razvoja na sicer zelo širokem področju arhitekture OPC.

Ključne besede: avtomatizacija, arhitektura odjemalec-strežnik OPC, poenotena arhitektura OPC UA, nadzorni sistemi SCADA.

Abstract

The thesis describes the implementation of client software architecture of different OPC interfaces. OPC is a standard used in industrial process automation to compile high-level software systems with low-level software systems. High-level systems are SCADA supervisory control and data acquisition systems, MES manufacturing execution systems and ERP enterprise resource planning systems. OPC client–server architecture connects them with low-level systems, for instance with PLC programmable logic controller and DCS decentralized systems.
The standard of classic OPC: OPC DA and other interfaces, that are mostly implemented in addition to DA. OPC foundation, which provides for this standard, has in 2009 introduced the specification of the OPC Unified Architecture (OPC UA). Compared to the classical architecture OPC provides new functions with the emphasis on the efficiency, safety, platform independance as well as on the network and internet communications. Since internet communication support was not very useful in the classical architecture, we decided to implement and test new ones.
We have built in and tested different OPC architectures in the SCADA system, which we developed entirely by ourselves. The application as a whole was intended to control heating and other installations in the industrial building. Testing of different OPC client–server architectures included the setting up and configuration of OPC server, the configuration of programmable logic controller, the development of clients and their inclusion in the SCADA system as well as the testing of functions, being offered by certain type of architecture.

Key words: automation, OPC client-server architecture, OPC Unifier Architecture (OPC UA) , Supervisory Control and Data Acquisition (SCADA)

1. Uvod

V zadnjem času je v industriji vedno večja potreba potreba po integraciji nadzornih sistemov ter ostalih višjih sistemov vodenja z računalniškimi sistemi za avtomatizacijo proizvodnih procesov. Ideja sicer ni nova, saj uporaba sistemov za avtomatizacijo, ki temelji na osebnih računalnikih in programski opremi narašča že od devetdesetih let. Nabor funkcij, ki jih nadzorni sistemi ali sistemi SCADA ponujajo, prav tako narašča, vzporedno pa tudi njihova kompleksnost.
Povod za nastanek te diplomske naloge je bil razvoja lastnega sistema SCADA, ki bi obvladoval nadzor ogrevanja in nekaj dodatnih funkcij v industrijski zgradbi. Težava, ki se nam je pojavila je bila v tem, kako uspešno povezati svet krmilnih računalnikov s svojim namenskim operacijskim sistemom in lastno predstavitvijo podatkov ter svet osebnih računalnikov. Na osebnem računalniku bi poganjali programsko opremo za vizualizacijo ter nadzor procesa ter vršili arhiviranje procesnih podatkov. Tu imamo nameščene večuporabniške in večopravilne operacijske sisteme, aplikacije so napisane v višjenivojskih objektno usmerjenih programskih jezikih. Potrebovali smo programsko opremo – nekakšen gonilnik za krmilni računalnik. Omogočati bi moral dostop do krmilnega računalnika in imeti lastnosti, kot so razširljivost in skladnost s standardi, veljavnimi na področju avtomatizacije. Podpirati bi moral omrežne in internetne komunikacije ter možnost implementacije v višjenivojskih objekto usmerjenih programskih jezikih in sodobnh platformah. Sami smo za razvoj lastnega sistema SCADA izbrali jezik C#, ogrodje .NET in razvojno okolje Microstoft Visual Studio. V zakup smo vzeli tudi stališča aplikativnega inženirja, ki razvija avtomatiziran sistem in bi rad hitro in neboleče povezal dva sistema med seboj, brez da bi moral sam razvijati gonilnik do naprave.
Odgovor na to je bil standard OPC (angl. Object Linking and Embedding for Process Control), ki je namenjen natanko temu kar potrebujemo: povezovanje programov višjega nivoja vodenja s programi in napravami na nižjih-procesnih nivojih. OPC je odprt standard, ki v osnovi temelji na tehnologijah operacijskega sistema Windows. Razvit je bil leta 1995 s strani podjetij, ki delujejo na področju industrijske avtomatizacije. Neprofitna organizacija, ki skrbi za ta standard, njegov razvoj in razpečevanje je fundacija OPC. Izdaje tega stanandarda, vse do današnjih dni, smo pregledali in opisali v poglavju 2. V tretjem poglavju smo se posvetili krmilnim računalnikom in sistemom SCADA na splošno. Krmilni računalniki, programska oprema za vizualizacijo in nadzor ter sistem za arhiviranje podatkov, ki smo jih že omenili, so tipični podsistemi v sistemu SCADA. Opisali smo uporabo teh in ostalih podsistemov, ki smo jih vgradili v naš testni nadzorni sistem. Na tem mestu smo opisali še programska orodja, ki smo jih uporabili pri izdelavi. V četrtem poglavju smo se posvetili testiranju različnih arhitektur vmesnikov OPC in sicer dveh klasičnih primerkov vmesnikov za dostop do procesnih podatkov OPC Data Access (OPC DA) in OPC DA z vgrajenim prehodom XML: OPC XML DA. Na koncu smo testirali še novo, leta 2009 predstavljeno poenoteno arhitekturo OPC Unifier Automation OPC UA. Tu nas je zanimala predvsem podpora internetnim komunikacijam, ki so v tej arhitekturi po zagotovilih izdajalca-fundacije OPC dobro podprte. Klasične arhitekture so imele internetne, pa tudi omrežne komunikacije slabo podprte. To sta bile poleg kronične platformne odvisnosti od operacijskega sistema Windows glavne slabosti klasičnih arhitektur OPC. Kako dobro se je odrezala nova arhitekura OPC UA in kako bomo delo na tem področju nadaljevali, si bralec lahko prebere v četrtem in petem poglavju.

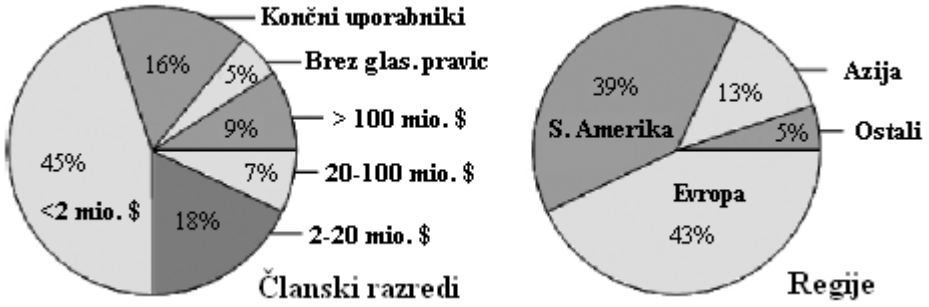
2. Vmesniki OPC

2.1. Uvod v OPC

Uporaba sistemov za avtomatizacijo, ki temeljijo na osebnih računalnikih (PC) in programski opremi narašča že od zgodnjih devetdesetih let. Osebni računalniki, še posebej tisti z operacijskim sistemom Microsoft Windows so v takšnih sistemih uporabljeni za namene vizualizacije in krmiljenja. V zadnjih letih je bilo vloženega veliko napora za standardizacijo programske opreme, ki se uporablja v avtomatizaciji. V napravah, vgrajenih v takih sistemih, se namreč uporablja skorajda nešteto mnogo različnih vodil, protokolov in vmesnikov.
Podobna težava je v računalništvu obstajala v času operacijskega sistema DOS in sicer s tiskalniki. Vsaka aplikacija je namreč morala imeti napisane svoje lastne gonilnike za dostop do vseh tiskalnikov, ki jih je podpirala. Kasneje so to težavo rešili z vgradnjo tiskalniških gonilnikov v operacijski sistem Windows, ki so zagotavljali dostop do tiskalnikov vsem nameščenim aplikacijam. Gonilniki so zagotovljeni s strani proizvajalcev tiskalnikov in ne več s strani razvijalcev aplikacij.
Dobavitelji oziroma ponudniki vmesnikov človek-stroj ali vmesnikov HMI (angl. Human-machine Interface) in proizvodnih nadzornih sistemov ali sistemov SCADA (angl. Supervisory Control and Data Acquisition) so imeli podobne težave. Tako so si leta 1995 takrat vodilna podjetja na tem področju zadala cilj definirati standard, ki bi na sistemih Windows omogočal enostaven in standardiziran dostop do podatkov.
Rezultat tega dela je bila specifikacija OPC Data Acces, izdana avgusta 1996. Neprofitna organizacija, ki skrbi za ta standard, je Fundacija OPC (angl. OPC Foundation). Skoraj vsi večji ponudniki sistemov za avtomatizacijo v industriji so njeni člani. Foundacija OPC je bila sposobna specificirati in sprejeti v praksi uporabljene standarde mnogo hitreje kot druge podobne organizacije. Eden od razlogov za ta uspeh je bila dobra zasnova aplikacijskega programskega vmesnika ali vmesnika API (ang. Application Program Interface) z relativno majhnim številom funkcij. Poudarek na pomembnih funkcijah in uporaba obstoječih tehnologij COM (ang. Component Object Model) in porazdeljenih tehnologij DCOM (angl. Distributed Component Object Model), razvitih za operacijski sistem Windows, je dovoljevala hitro osvojitev standardov za namensko uporabo.
Rezultat izkušenj različnih razvijalcev produktov je bila druga verzija specifikacije OPC Data Access, predstavljena leta 1998. Ta verzija specifikacije je danes še vedno najpomembnejši vmesnik za produkte OPC. [1]
Sistemi SCADA, HMI, porazdeljeni sistemi DCS (ang. Distributed Control Systems), krmilni sistemi na osnovi osebnih računalnikov PC in sistemi za upravljanje proizvodnje MES (angl. Manufacturing Execution System) dandanes v večini vsi podpirajo vmesnike OPC. OPC je univerzalno sprejet standard, ki omogoča izmenjavo podatkov med različnimi sistemi za avtomatizacijo v proizvodnji in procesni industriji.

2.2. Fundacija OPC

Fundacija OPC ima po 15 letih delovanja več kot 400 [2] članov in vključuje vse pomembne dobavitelje sistemov za avtomatizacijo po vsem svetu. Slika 2.1 prikazuje demografske podatke članov fundacije OPC skupaj s članskimi razredi in regijo. Uvrstitev v članske razrede za podjetja je osnovana na podatkih o prodaji posameznega podjetniškega člana. Ostali razredi so namenjeni končnim uporabnikom in članom brez glasovalnih pravic, kot so univerze in ostale organizacije. Fundacijo OPC vodi uprava, ki jo volijo člani, ima pa tudi ostale direkcije, oddelke in različne delovne skupine.



Slika 2.1. – demografija članov fundacije OPC

Fundacija OPC ima na seznamu preko 1500 produktov, osnovanih na OPC. Te produkte so prispevali samo člani fundacije, celoten trg produktov OPC ima preko 2500 ponudnikov in preko 15000 produktov.

Ta velik uspeh zahteva določene mehanizme za preverjanje ali so vsi produkti OPC kompatibilni med seboj ter za zagotavljanje določenega nivoja kvalitete. Zaradi teh razlogov je nastal program imenovan OPC Compliance Program namenjen preverjanju skladnosti produktov OPC, ki je bil poleg razvoja novih standardov glavna dejavnost delovnih skupin v fundaciji OPC.

V programu OPC Compliance Program sta definirana dva nivoja certificiranja. Prvi nivo je kombinacija samo-certificiranja in delavnic o iteroperabilnosti, ki jih organizira fundacija. Fundacija OPC ponuja orodja za testiranje skladnosti (angl. Compliance Test Tools) za vse pomembne standarde OPC. Orodja so uporabljena za testiranje produktov, rezultate se nato ustrezno kriptira in pošlje fundaciji OPC v pregled. Ti testi so uporabljeni za testiranje funkcionalnosti na nivoju vmesnika. Delavnice o interoperabilnosti so letni dogodki v Evropi, Severni Ameriki in na Japonskem, kjer lahko različni ponudniki testirajo sposobnosti delovanja svojih produktov med seboj. Produkti, ki prestanejo samo-certificirni test, lahko uporabljajo poseben logotip Self-Tested, ki nakazuje osnovni nivo skladnosti s standardom OPC (slika 2.2.).

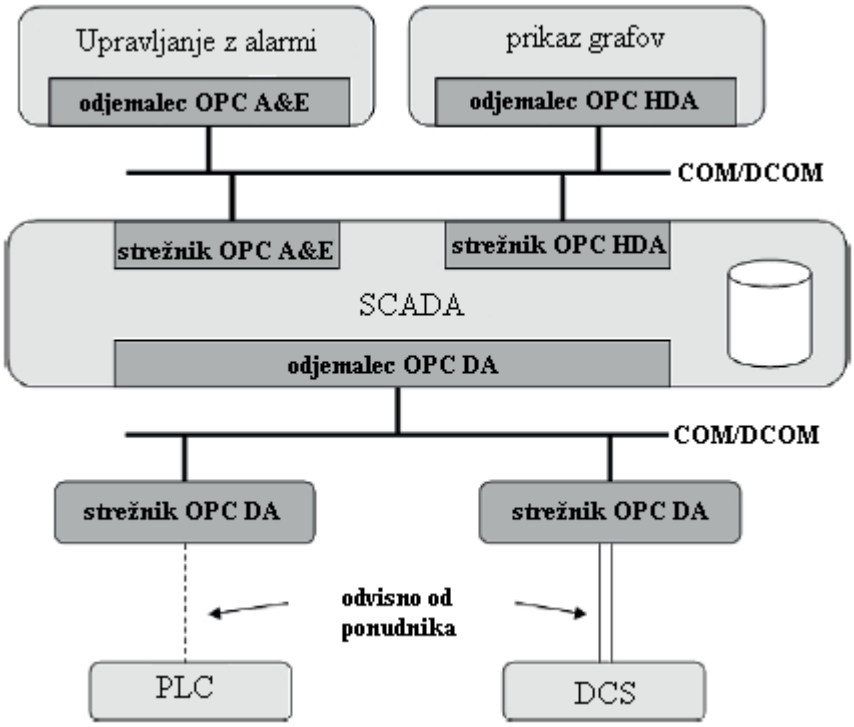
Drugi nivo certificiranja produktov poteka v neodvisnih laboratorijih za certificiranje (angl. Certification Test Labs). Pooblaščeni neodvisni laboratoriji testirajo produkte OPC s širokim spektrom testov. Poleg testiranja osnovnih funkcionalnosti s prej omenjenimi orodji za testiranje skladnosti, poganjajo tudi teste obnašanja produktov pod obremenitvijo, izvajajo stresne teste, testirajo njihovo interoperabilnost in uporabnost. Produkte, ki prenesejo neodvisno certificiranje, opremijo z logotipom OPC Certified (slika 2.2.), ki izkazuje visok nivo kvalitete in skladnosti s standardom OPC.

Slika 2.2. – logotipa samo-certificiranega in neodvisno certificiranega produkta OPC

Priporočljivo je, da končni uporabniki kupujejo samo certificirane produkte. S tem se izognejo težavam z interoperabilnostjo in si zagotovijo zanesljivost in uspešnost svojih produktov na osnovi OPC.
Kot zanimivost naj še omenimo, da sta trenutno v Sloveniji samo dve podjetji člana fundacije OPC. [2]

2.3. Klasični OPC

Fundacija OPC je do sedaj definirala številne vmesnike v namen standardizacije podatkovnega toka iz procesnega nivoja v višje nivoje vodenja. Ti vmesniki so v veliki večini uporabljeni v aplikacijah v avtomatizaciji v industriji, kot so sistemi HMI in SCADA. Služijo za zajem podatkov iz krmilnih računalnikov in ostalih naprav na nižjih nivojih vodenja ter jih prikazujejo v realnem času in shranjujejo.
Glede na različne zahteve industrijskih aplikacij na trgu, so bile razvite tri glavne specifikacije OPC: dostop do procesnih podatkov DA (angl. Data Access), alarmi in dogodki A&E (angl. Alarm & Event) ter dostop do zgodovinskih procesnih podatkov HDA (ang. Historical Data Access). Dostop do trenutnih procesnih podatkov je opisan v specifikaciji DA, specifikacija A&E opisuje vmesnik za informacije na osnovi dogodkov, vključno s potrjevanjem procesnih alarmov. Specifikacija HDA opisuje funkcije za dostop do arhivskih oziroma zgodovinskih podatkov. Vsi vmesniki ponujajo orientacijo po naslovnem prostoru naprave in zagotavljajo informacije o podatkih, ki so na voljo.
Specifikacija OPC za izmenjavo podatkov uporablja arhitekturo odjemalec-strežnik. Strežnik OPC povzame procesne informacije iz vira oziroma napravo in jih nato ponudi preko svojega vmesnika. Odjemalec OPC se poveže na strežnik OPC in lahko dostopa do ponujenih podatkov. Tako odjemalec kot strežnik sta lahko izvor in ponor podatkov. Slika 2.3. prikazuje tipičen primer uporabe strežnikov in odjemalcev OPC.



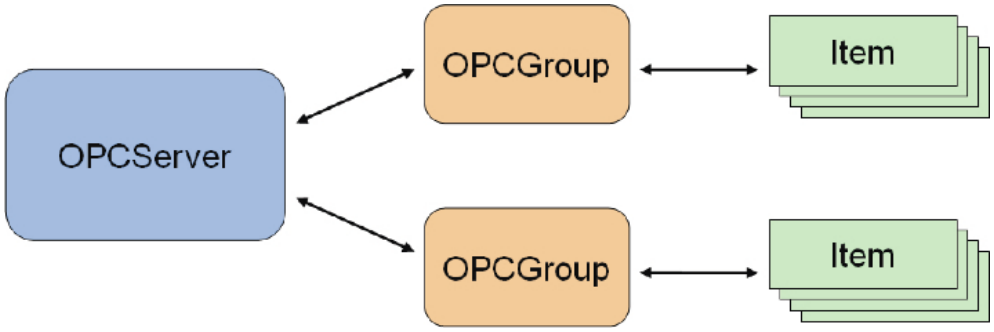
Slika 2.3. – tipičen primer uporabe odjemalcev in strežnikov OPC

Klasični vmesniki OPC so osnovani na Microsoftovem modelu COM (angl. Component Object Model) in porazdeljenem modelu DCOM (angl. Distributed Component Object Model). Prednost takega pristopa je bilo zmanjšanje specifikacije le na definicijo različnih aplikacijskih programskih vmesnikov za različne specialne namene, a brez potrebe, da bi definirali še omrežne protokole oziroma mehanizme za komunikacijo med procesi. Tehnologiji COM in DCOM zagotavljata transparenten mehanizem za odjemalčev klic metod na strežnikovem objektu COM. Le-ta je lahko isti proces, nek drugi proces ali pa proces na nekem drugem računalniku v omrežju. Tehnologijo COM/DCOM se lahko uporablja na vseh računalnikih z operacijskim sistemom Windows, zmanjša se čas za razvoj specifikacij in produkta, zmanjšan je čas od pričetka razvoja do prodaje produkta na trgu. To so glavne prednosti, ki so zaslužne za uspeh produktov OPC.
Dve veliki slabosti pa sta navezava na operacijski sistem Windows in zelo zahtevna administracija v zvezi z DCOM pri omrežni komunikaciji produktov OPC. Nastavitve v zvezi z DCOM so kompleksne in zapletene. Poleg tega so časovne zakasnitve ob javljanju napak v komunikaciji (angl. timeout) zelo dolge in se jih ne da nastavljati, zato tehnologije DCOM ni priporočljivo uporabljati za komunikacijo med sistemi preko interneta.

2.3.1. Vmesnik OPC Data Access

Vmesnik za dostop do procesnih podatkov OPC Data Access oziroma OPC DA omogoča branje, pisanje in spremljanje spremenljivk trenutnih procesnih podatkov. V večini primerov se ga uporablja za izmenjevanje podatkov v realnem času med krmilnimi računalniki (angl. Programmable Logic Controller, PLC), porazdeljenimi sistemi (angl. Distributed Computer

Systems, DCS) in ostalimi krmilnimi napravami ter vmesniki HMI in ostalimi odjemalci. Vmesnik OPC DA je še vedno najpomembnejši vmesnik OPC. Dandanes je uporabljen v 99% produktov, ki uporabljajo tehnologijo OPC. Vsi ostali klasični vmesniki OPC so večinoma izpeljanke vmesnika OPC DA.
Odjemalci OPC DA eksplicitno izberejo spremenljivko oziroma točko (angl. OPC item), ki jo želijo brati, pisati ali pa jo le spremljati na strežniku. Odjemalec OPC vzpostavi povezavo s strežnikom s tem, da ustvari objekt *OPCServer*. Strežnikov objekt ponudi metode za navigacijo po naslovnem prostoru strežnika, kjer odjemalec lahko najde zahtevane točke in njihove lastnosti (podatkovni tip, dostopne pravice, ...).
Za dostop do podatkov odjemalec združi točke OPC v skupino (angl. group) z istimi nastavitvami, kot je na primer čas osveževanja, v objekt *OPCGroup*. Slika 2.4. prikazuje različne objekte, ki jih odjemalec ustvari na strežniku OPC.



Slika 2.4. – objekti, ki jih ustvari odjemalec OPC za dostop do podatkov

Po dodajanju točk v skupine so le-ti na voljo za branje ali pisanje s strani odjemalca. Najboljši način za periodično branje podatkov je spremljanje sprememb vrednosti spremenljivk na strežniku. Odjemalec nastavi čas posodabljanja (angl. update rate) na skupini, ki vsebuje želeno spremenljivko oziroma točko. Nastavljeni čas posodabljanja je na strežniku uporabljen za periodično pregledovanje sprememb vrednosti točk. Po vsakem končanem preverjanju strežnik pošlje odjemalcu samo vrednosti, ki so se spremenile.
OPC zagotavlja podatke v realnem času, ki niso nujno trajno dostopni. Povezava med strežnikom in napravami se lahko začasno prekine. Klasična tehnologija OPC takšne situacije obvladuje tako, da vsak podatek opremi še z dvema metapodatkoma: časovno znamko (angl. timestamp) in kvaliteto (angl. quality). Kvaliteta podatka je dobra (angl. good), če je podatek točen. Podatek, ki ni na voljo, je slab (angl. bad), neznana kvaliteta pa se označi z negotov (angl. unceartain).

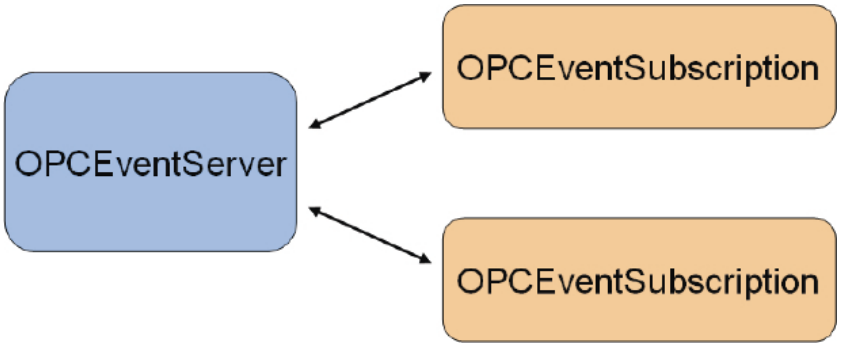
2.3.2. Vmesnik OPC Alarm & Event

Vmesnik za alarme in dogodke OPC A&E omogoča sprejemanje obvestil o alarmih in dogodkih. Dogodki so enkratna obvestila, ki posredujejo odjemalcu podatke o pojavitvi dogodka. Alarmi so obvestila o spremembah stanja v procesu. Na primer, pri spremljanju nivoja tekočine v rezervoarju se sprememba stanja lahko zgodi, ko je presežen maksimalen nivo tekočine oziroma, ko ta pade pod minimalni dovoljeni nivo. V praksi veliko procesnih alarmov zahteva potrditev, ki je ravno tako mogoča preko vmesnika OPC A&E.

Prejemanje obvestil si odjemalci zagotovijo tako, da se povežejo na strežnik in se naročijo na obvestila (angl. subscribe). Nato prejemajo vsa obvestila, ki so sprožena na strežniku, z določenimi kriteriji pa si jih lahko omejijo oz. filtrirajo.
Odjemalci se na strežnik OPC A&E povežejo v dveh korakih:

1. odjemalec ustvari objekt *OPCEventServer,*
2. odjemalec ustvari objekt *OPCEventSubscription* s katerim se naročijo na prejemanje določenih obvestil.

Omejitve oziroma filtri se lahko konfigurirajo ločeno za vsako naročnino posebej. Slika 2.5. prikazuje različne objekte, ki jih odjemalec OPC ustvari na strežniku.



Slika 2.5. – objekti, ki jih odjemalec ustvari na strežniku OPC A&E

Z razliko od vmesnika OPC DA vmesnik OPC A&E posebej ne podaja zahtev po podatkih, na primer z branjem, ampak so odjemalcu avtomatsko dostavljena naročena obvestila o dogodkih in alarmih.

2.3.3. Vmesnik OPC Historical Data Access

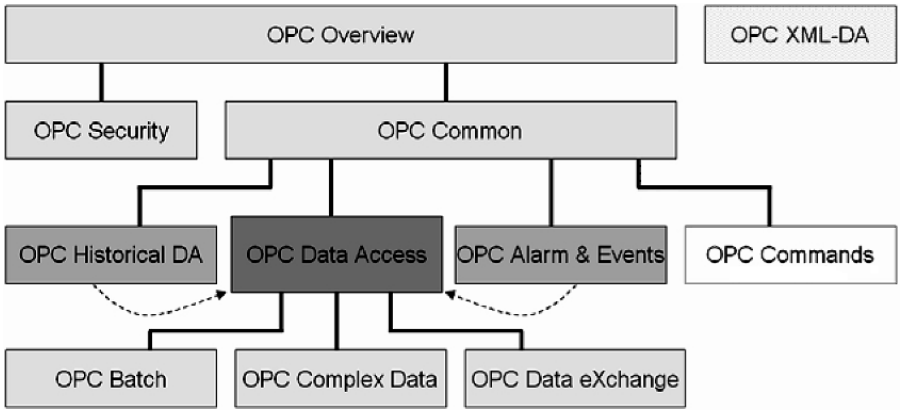
Vmesnik OPC DA, opisan v poglavju 2.3.1., omogoča dostop do neprestano spreminjajočih se podatkov v realnem času. Vmesnik za dostop do zgodovinskih procesnih podatkov OPC HDA pa ponuja dostop do že shranjenih podatkov. Pristop do zgodovinskih podatkov je enoten, pa naj gre za preprosto sekvenčno shranjevanje ali pa kompleksen sistem SCADA.
Odjemalec OPC se poveže na strežnik z ustvarjenjem objekta *OPCHDAServer* v strežniku HDA. Ta objekt ponuja vse vmesnike in metode za branje in posodabljanje arhivskih podatkov.
Glavna funkcionalnost je branje podatkov na tri različne načine.

1. Branje neobdelanih podatkov iz arhiva, kjer odjemalec določi eno ali več spremenljivk, časovno domeno in maksimalno število zadetkov. Strežnik vrne določeno število vrednosti v zahtevanem časovnem obsegu.
2. Branje ene ali več vrednosti z določeno časovno značko.
3. Metoda branja je podobna prvi, le da lahko odjemalec uporablja tudi določene združitvene funkcije. V odgovoru odjemalcu sta vedno vključena tudi metapodatka kvaliteta in časovna značka.

Vmesnik OPC HDA ponuja poleg metod za branje tudi metode za vstavljanje, zamenjevanje in brisanje zapisov iz baze zgodovinskih podatkov.

2.3.4. Ostali standardi vmesnikov OPC

Fundacija OPC je določila še nekaj standardov ki služijo kot temeljna specifikacija ali pa kot specifikacija za specializirane potrebe. Temeljni specifikaciji sta *OPC Overview* in *OPC Common*, ki določata vmesnike in obnašanje le-teh, skupno za vse specifikacije na osnovi COM. Pregled klasičnih specifikacij OPC je na sliki 2.3.4.
Vmesnik OPC za varnost *OPC Security* določa kontrolo pristopa do strežnika, na katerem se lahko nahajajo občutljivi podatki, in ščiti pred nepooblaščenim spreminjanjem procesnih parametrov.
Vmesniki *OPC Complex Data*, *OPC Batch* in *OPC Data eXchange* (OPC DX) so razširitev vmesnika OPC DA. Vmesnik Complex Data določa opis kompleksnih struktur in se ukvarja z njihovim prenosom. OPC DX določa konfiguracijo vmesnikov v odjemalcih, vgrajenih v strežnike, in njihovo obnašanje pri izmenjavi podatkov med strežniki OPC DA. Vmesnik OPC Batch razširja specifikacijo DA na področju posebnih potreb v industriji, za katero so značilni paketni procesi.



Slika 2.6. – standardi klasičnih vmesnikov OPC

Vmesnik *OPC Commands* določa mehanizem za klic metod oziroma izvrševanje programov preko sistema OPC. Ta specifikacija ni bila nikoli izdana, njen razvoj je bil zaključen po tem, ko je fundacija OPC izdala specifikacijo *OPC Unified Architecture* (OPC UA). Vsebina in funkcionalnost omenjenega vmesnika je bila popolnoma vključena v specifikacijo OPC UA.

2.3.5. Vmesnik OPC XML-DA

Vmesnik OPC XML-DA za dostop do procesnih podatkov z vgrajenim prehodom XML je bil prvi platformno neodvisni vmesnik OPC. Tehnologijo COM/DCOM je zamenjal protokol SOAP (angl. Simple Object Access Protocol) preko HTTP protokola in tehnologije spletnih storitev. Novi vmesnik je neodvisen od ponudnika in platforme in dobro sprejet med strokovno javnostjo, saj je obdržal vse pomembne funkcionalnosti vmesnika OPC DA.

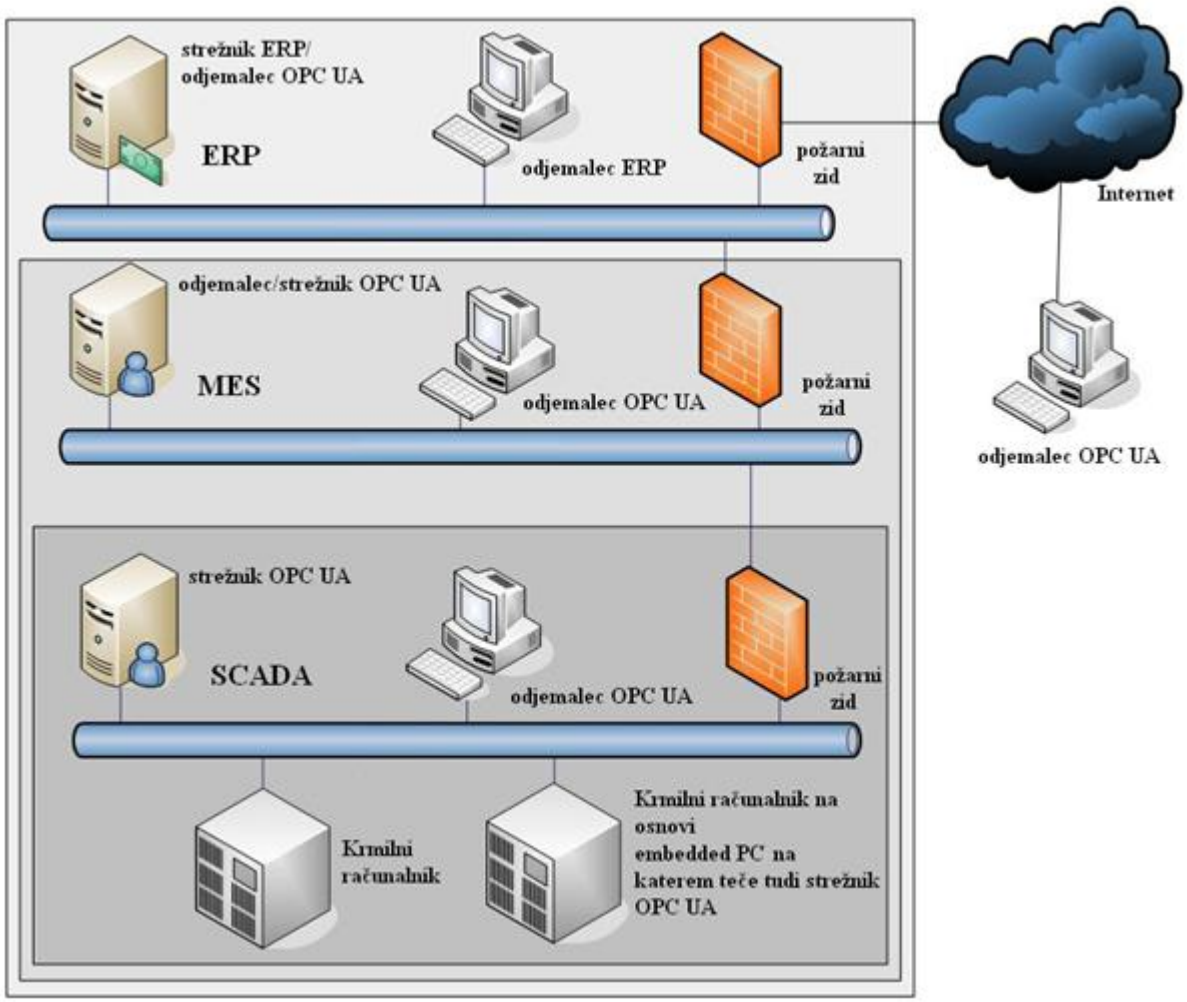
Funkcionalnost vmesnika je zmanjšana na minimalen nabor metod za izmenjano podatkov OPC DA. Ključne funkcije vmesnika za dostop do procesnih podatkov pokriva osem metod:

- *GetStatus* za preverjanje statusa strežnika,
- *Read* za branje vrednosti ene ali več točk,
- *Write* za pisanje vrednosti ene ali več točk,
- *Browse* in *GetProperties* za dostop do informacij o dosegljivih točkah,
- *Subscribe* za izvedbo naročila na seznam določenih točk (podobno kot pri vmesniku OPC A&E),
- *SubscriptionPolledRefresh* za izmenjavo spremenjenih vrednosti v seznamu naročenih točk in
- *SubscriptionCancel* za preklic naročnine.

Vmesnik OPC XML-DA je bil zasnovan za internetno uporabo. Glede na to, da je neodvisen od platform, je bil večinoma uporabljen v vgradnih sistemih in platformah, ki niso bile osnovane na Microsoftovih produktih. Slabost tega vmesnika je požrešnost računalniških virov med delovanjem in omejene zmogljivosti v primerjavi z vmesnikom OPC DA. To sta glavna vzroka, zakaj ta vmesnik ni tako uspešen, kot bi se sicer pričakovalo za tovrstne aplikacije.

2.4. Vmesnik OPC Unified Architecture

Poenotena arhitektura OPC Unified Architecture ali OPC UA je bila zastavljena kot želja po resnični zamenjavi vseh obstoječih specifikacij na osnovi tehnologij COM/DCOM vendar brez izgub njihovih funkcij in zmogljivosti. Izpolnjevati bi morala zahteve po neodvisnosti od platforme in biti dovolj razširljiva, da bi z njo opisali tudi zelo kompleksne sisteme. Širok spekter področij, ki jih pokriva OPC, sega od vgradnih sistemov preko sistemov SCADA do sistemov MES in sistemov planiranja proizvodnje ERP (angl. Enterprise Resource Planing). Najpomembnejše zahteve za vmesnik OPC UA so zbrane v tabeli 2.1., kompleksnost takega sistema pa je prikazana na sliki 2.7.
Zahteve lahko razdelimo na dva dela: na zahteve, povezane s komunikacijo porazdeljenih sistemov med seboj in na definicijo podatkovnega modela, ki opisuje vmesnik in informacije, ki so na voljo.
Klasični vmesnik OPC je bil sprva zasnovan kot gonilnik naprave. Dandanes pa je vmesnik OPC uporabljen kot sistemski vmesnik, saj je komunikacija med porazdeljenimi sistemi zelo pomembna in mora biti zato zelo zanesljiva. Zato vmesnik OPC zaradi potreb po visoki razpoložljivosti daje pri komunikacijah poseben poudarek na njihovo robustnost, odpornost proti napakam in redundanco. Za integracijo vmesnikov OPC neposredno v različne platforme sta nujni neodvisnost od platforme in razširljivost. V novem OPC vmesniku mora biti mogoča komunikacija prek interneta, kar seveda brezpogojno vključuje tudi požarne zidove. Varnost in kontrola dostopa je torej še dodatna pomembna zahteva. Najpomembnejša zahteva pa še vedno ostaja interoperabilnost med sistemi različnih proizvajalcev.



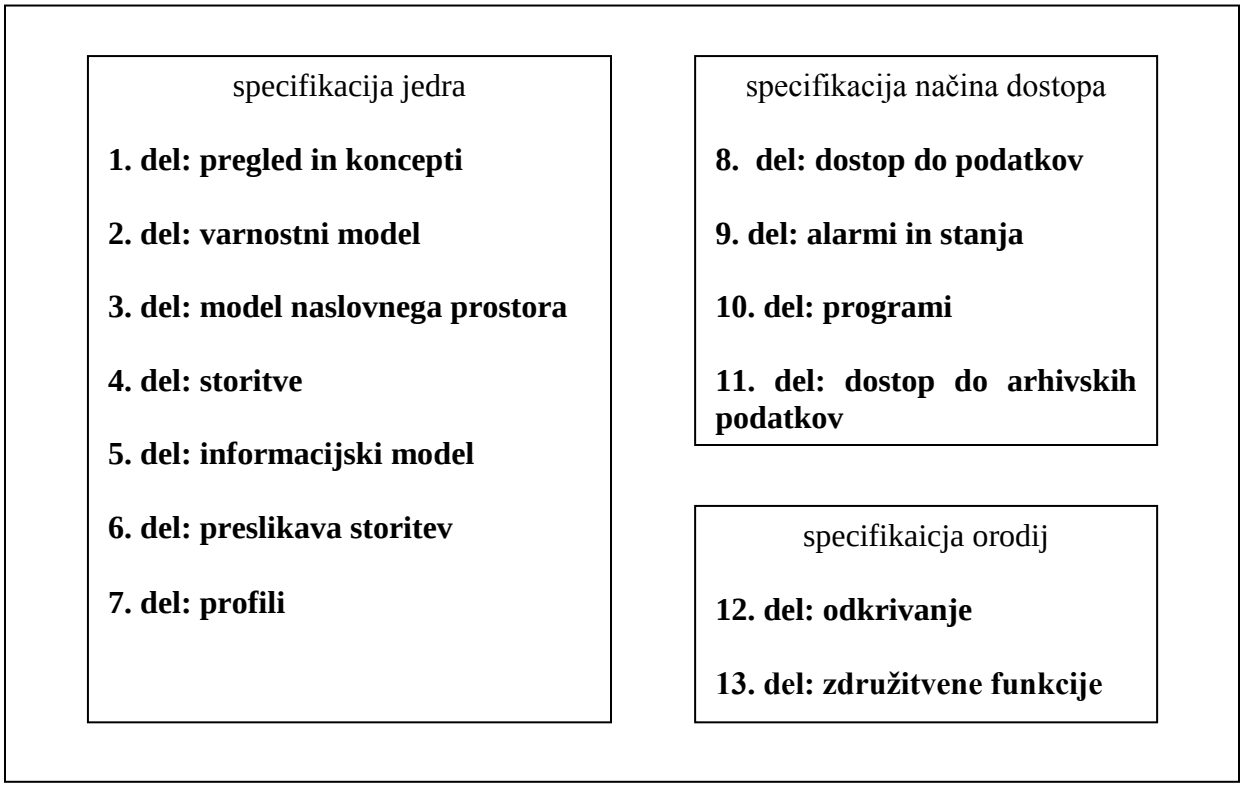
Slika 2.7. – primer kompleksnega sistema z OPC UA

Tabela 2.1.. – zahteve za vmesnik OPC UA

Komunikacija med porazdeljenimi sistemi	**Podatkovni model**
- robustnost in odpornost proti napakam - redundanca - platformna neodvisnost - razširljivost - visoka zmogljivost - komunikacija prek interneta in požarnih zidov - varnost in kontrola dostopa - interoperabilnost	- skupen podatkovni model za vse podatke OPC - objektno orientiran - metapodatki - kompleksni podatkovni tipi in metode - razširljiv od preprostega do kompleksnega modela - abstrakten osnovni model - temelj za ostale standardne podatkovne modele

2.4.1. Specifikacija

Specifikacija OPC UA je razdeljena na več delov, kot prikazuje slika 2.8.



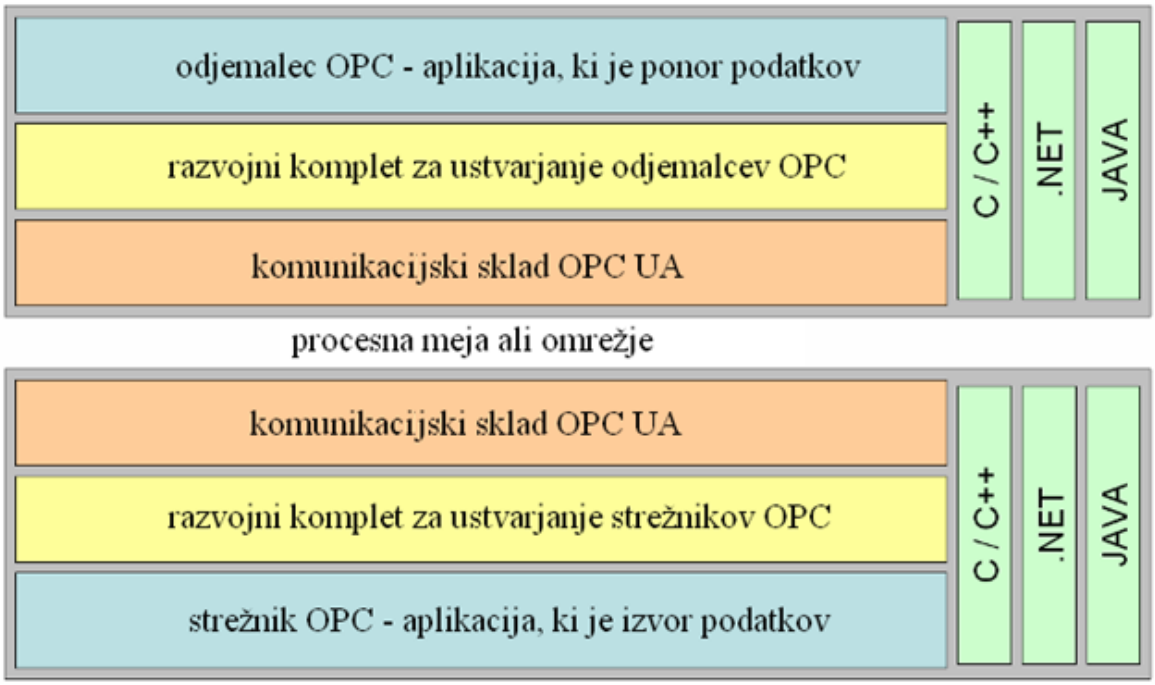
Slika 2.8. – večdelna specifikacija OPC UA

Prvi del govori o konceptih (angl. concepts) in podaja pregled nad OPC UA. Drugi del opisuje varnostne zahteve za vmesnik (angl. security model). V tretjem delu je opisana zbirka informacij, ki strežnik OPC UA naredi vidnega svojim odjemalcem, in kako je naslovni prostor zgrajen (angl. address space model). Abstraktna definicija storitev (ang. services) je opisana v četrtem delu. Predstavlja možne interakcije med odjemalci in strežniki UA. Odjemalci uporabljajo storitve za dostop do informacij, ki jih nudi strežnik. Opis storitev je abstrakten, saj ne vsebuje nobenih konkretnih predstavitev in nobenih konkretnih vmesnikov, ampak samo določa, katere informacije bi se morale prenašati med aplikacijami UA.
Preslikavo storitev na sporočila (podatkovne enote ki se prenašajo med strežniki in odjemalci), varnostni mehanizmi, ki se uporabljajo v njih, in konkreten opis sistema prenašanja sporočil so opisani v šestem delu (angl. service mappings). Temeljni informacijski model (angl. information model), določen v petem delu zagotavlja ogrodje za vse informacijske modele, ki uporabljajo OPC UA.
Profili (angl. profiles), opisani v sedmem delu, določajo uporabne podmnožice funkcij OPC UA. Vsaka aplikacija OPC UA mora vključevati kompletno implementacijo določene podmnožice funkcij. Na ta način se zagotovi interoperabilnost za različne podmnožice. Specifikacija določa podmnožice funkcij na dveh nivojih. Prvi nivo so skladnostne enote (angl. conformance units) ki predstavljajo majhne nabore funkcij, ki so vedno uporabljene skupaj, in jih je mogoče preveriti z orodji za preverjanje skladnosti (poglavje 2.2.) in jih enotno overiti. Drugi nivo so profili (angl.

profiles), ki predstavljajo seznam skladnostnih enot. Ravno tako kot skladnostne enote morajo biti tudi profili kompletno implementirali v aplikaciji OPC UA. Tudi testi certificiranja se izvajajo na kompletnem naboru enot posameznega profila. Odjemalec in strežnik si med vzpostavljanjem povezave izmenjata seznam podprtih in uporabljenih profilov. S tem aplikacija ugotovi, ali so določene funkcije pri komunikacijskem partnerju na voljo.
Model dostopa do procesnih podatkov (angl. data access) je opisan v osmem delu in določa način predstavitve podatkov in ostale lastnosti kot so recimo inženirske mere. Deveti del, ki govori o alarmih in stanjih (angl. alarms and conditions) določa procesne alarme, spremljanje stanj npr. stroja in tipe dogodkov. Deseti del specifikacije, programi (angl. programs), določa izvajanje, manipulacijo in spremljanje programov. Informacijski model dostopa do zgodovinskih podatkov (angl. historical access) v enajstem delu določa uporabo storitev za dostop do teh podatkov in način predstavitve podatkov in zgodovine dogodkov. Dvanajsti del specifikacije imenovan odkrivanje (angl. discovery) določa načine, kako odjemalci poiščejo strežnike v omrežju in kako pridobijo potrebne informacije za vzpostavitev povezave z določenim strežnikom. V trinajstem delu so opisane združitvene funkcije (angl. aggregates), ki se uporabljajo za izračun vrednosti, kot so, na primer, minimum, maksimum in povprečje iz surovih, neobdelanih podatkov. Uporabne so tako za dostop do zgodovinskih podatkov kot za spremljanje le-teh v realnem času.

2.4.2. Programske plasti OPC UA

Vmesnik OPC UA uporablja podoben koncept odjemalec-strežnik kot klasični vmesnik OPC. Aplikacija, ki nudi svoje podatke drugim aplikacijam je strežnik, aplikacija ki zahteva oziroma porablja podatke pa odjemalec. Težnja pa je, da bi bilo čim več aplikacij hkrati odjemalec in strežnik. Eden od razlogov je, da bodo strežniki OPC UA vgrajeni neposredno v naprave (primer je krmilni računalnik na osnovi vgradnih (angl. embedded) sistemov na sliki 2.7.). Vgradnja odjemalca OPC UA bi med napravami omogočila neposredno komunikacijo. Drugi razlog je uporaba OPC UA tudi za potrebe konfiguracije odjemalca in njenega vmesnika OPC UA. V tem primeru bi morali biti odjemalci tudi strežniki, ki bi jih konfigurirali preko tega istega vmesnika.
Tipična aplikacija OPC UA je sestavljena iz treh plasti, kot je prikazano na sliki 2.4.2. OPC UA po definiciji ni omejen na določen programski jezik ali razvojno platformo, vendar so komunikacijski skladi in razvojni kompleti za ustvarjanje aplikacij OPC UA (angl. Software Development Kit), ki jih fundacija OPC nudi svojim članom, trenutno na voljo le za okolja C/C++, .NET in Java.

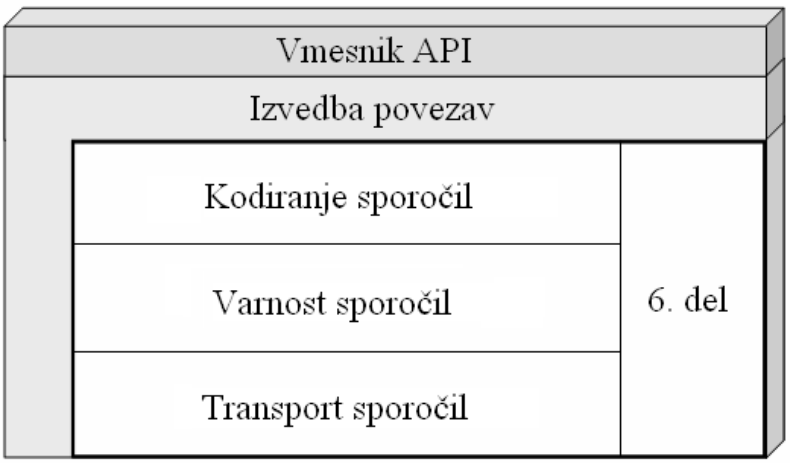


Slika 2.9. – programske plasti arhitekture OPC UA

Aplikacija OPC UA je sistem, ki hkrati služi kot izvor in ponor podatkov preko vmesnika OPC UA. Vsebuje namenske funkcionalne posebnosti in preslikavo podatkov na specifikacijo OPC UA preko komunikacijskega sklada in razvojnega kompleta za ustvarjanje programske opreme (SDK).
Strežniki in odjemalci OPC UA izrabljajo skupne funkcionalnosti OPC UA na nivoju aplikacije, komunikacijski sklad pa uporabljajo samo za komunikacijo. Komplet za ustvarjanje programske opreme OPC UA pripomore k lažjemu razvoju aplikacij OPC UA in interoperabilnosti med njimi.
Komunikacijski sklad OPC UA, opisan v šestem delu izvaja preslikave in kodiranje na transportnem nivoju. Sklad prikliče storitve OPC UA preko procesnih meja ali preko omrežja. Standard OPC UA določa tri plasti komunikacijskega sklada:

1. kodiranje sporočil določa serializacijo parametrov storitev OPC UA v binarni format in format XML (angl. Extended Markup Language),
2. varnost določa varnostne mehanizme v sporočilih z uporabo standardov varnosti spletnih storitev ali binarne verzije OPC UA tega standarda,
3. transportna plast določa omrežne protokole, ki so lahko UA TCP, HTTP ali SOAP za spletne storitve.

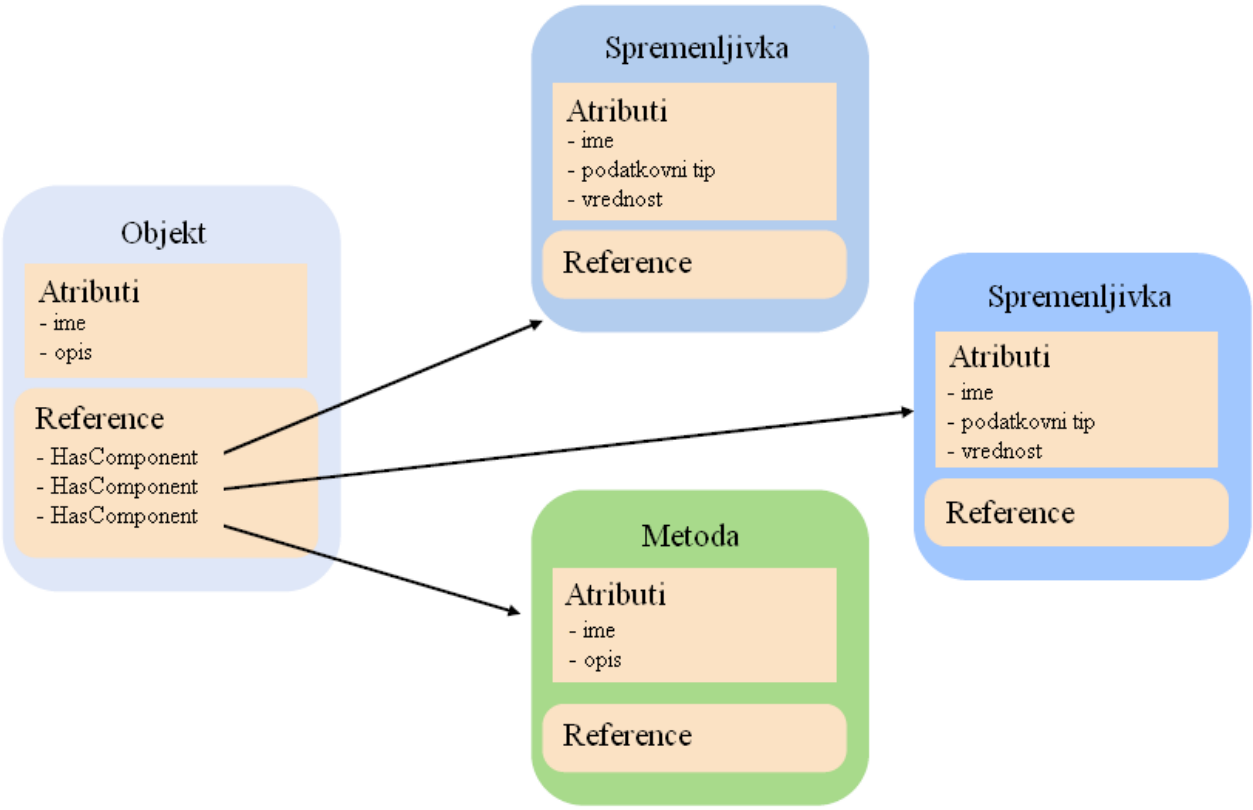
Slika 2.9. ponazarja plasti komunikacijskega sklada UA. Implementacija teh plasti v skladu in pripadajoči vmesnik API nista del specifikacije OPC UA. Komunikacijski sklad OPC UA ponuja vmesnike API za razvoj aplikacij strežnikov in odjemalcev OPC UA. Ti vmesniki so neodvisni od platforme in programskega jezika. Storitve OPC UA in pripadajoči parametri pa so si podobni, vsi so osnovani na abstraktnem opisu storitev, opisanem v 4. delu specifikacije.



Slika 2.10. – plasti komunikacijskega sklada OPC UA

2.4.3. Struktura naslovnega prostora strežnika OPC UA

V poglavju Vmesnik OPC Unified Arhitecture smo v tabeli 2.4., ki podaja zahteve za vmesnik OPC UA, navedli objektno usmerjenost poenotenega sistema. Posamezen element v naslovnem prostoru strežnika OPC UA, ki mu rečemo vozlišče (angl. node) izgleda kot primerek določenega razreda, spremenljivka (ang. variable) ali pa metoda (angl. method). Vsa vozlišča imajo skupne atribute, kot na primer ime in opis ter specifične atribute, kot je vrednost spremenljivke. Seznam atributov je fiksen in ga ni mogoče razširiti, možno pa je dodajati dodatne informacijo o vozlišču, to so lastnosti (angl. properties). Lastnosti so poseben tip spremenljivk.
Vozlišča so medsebojno povezana z referencami (angl. references), ki so tipizirane. Določena sta dva glavna tipa referenc in sicer hierarhične ter nehierarhične reference. Primer hierarhične reference je referenca *HasComponent* za komponente objekta. Nehierarhične reference imajo spremenljivke in metode, primer je referenca *HasTypeDefinition* za povezavo objekta in tipa objekta. Slika 2.4.4. prikazuje primer povezave vozlišč s svojimi referencami.



Slika 2.11. – primer povezav vozlišč z referencami v naslovnem prostoru strežnika OPC UA

Seznam razredov vozlišč prikazuje tabela 2.2. Seznam je fiksno določen v specifikaciji in ga ni mogoče razširiti.

Tabela 2.2. – razredi vozlišč v naslovnem prostoru strežnika OPC UA

Razred vozlišča	**Opis**
Objekt	Objekt se uporablja kot tipiziran vsebovalnik za spremenljivke, metode in dogodke.
Spremenljivka	Spremenljivka predstavlja podatke določenega objekta, lahko pa nastopa tudi kot lastnost vozlišča.
Metoda	Metode so komponente objekta, imajo lahko seznam vhodnih in izhodnih parametrov. Parametri so določeni kot lastnosti.
Pogledi (angl. view)	Pogledi predstavljajo del naslovnega prostora. Pri brskanju po velikih naslovnih prostorih lahko s pogledi omejimo število vidnih vozlišč in referenc.
Tipi objektov (angl. object types)	Tipi objektov podajajo informacije o strukturi ali komponentah objekta.
Tipi spremenljivk (angl. variable type)	Tipi spremenljivk opisujejo, katere lastnosti podatkovnih tipov so na voljo preko referenc spremenljivke.
Podatkovni tipi (angl. data types)	Podatkovni tipi opisujejo vsebino vrednosti spremenljivke.

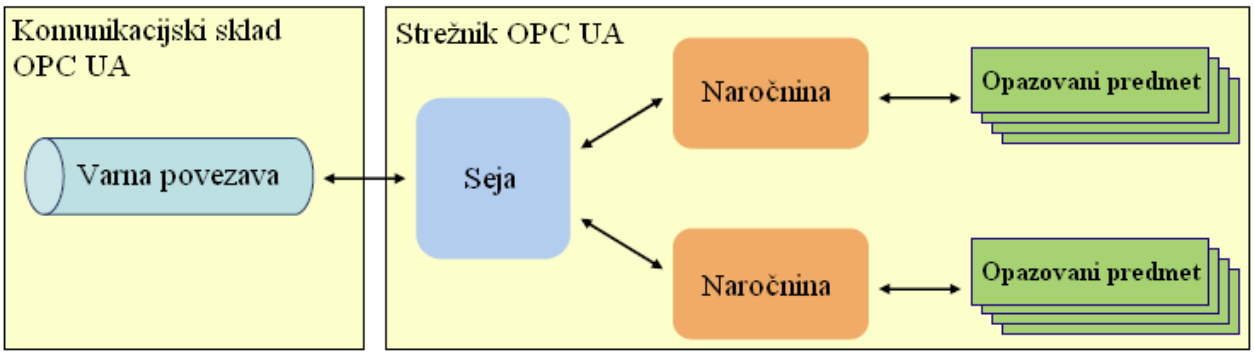
Vsako vozlišče v naslovnem prostoru strežnika OPC UA je enolično definirano z atributom NodeId. Ta atribut je izpeljan iz imenskega prostora (angl. namespace), da lahko razlikujemo identifikatorje v različnih podsistemih. Identifikator je lahko številska vrednost, niz znakov ali identifikator GUID (angl. globally unique identifier) [4]. Identifikator GUID je posebna vrsta identifikatorja, ki se uporablja za zagotavljanje globalne unikatne vrednosti, na katero se lahko sklicujemo. Predstavljen je kot 32-bitni šestnajstiški niz.
Atributi vozlišč so navedeni v tabeli 2.3. Zbrani so le najpomembnejši, glavni poudarek je na spremenljivkah.

Tabela 2.3. – najpomembnejši atributi vozlišč v naslovnem prostoru strežnika OPC UA

Atribut	**Relevantnost za razrede vozlišč**	**Opis**
NodeID	Vse	Unikaten naslov vozlišča.
DisplayName	Vse	Ime vozlišča v specifičnem jeziku. Odvisen je od jezika, ki ga zahteva odjemalec in jezikov, ki jih podpira strežnik.
BrowseName	Vse	Ime vozlišča neodvisno od jezika.
Description	Vse (opcijsko)	Opis vozlišča v specifičnem jeziku.
Value	Spremenljivke	Vrednost spremenljivke. Kot za vse ostale atribute sta pri branju na voljo metapodatka časovna značka in status.
DataType	Spremenljivke	Podatkovni tip spremenljivke ali atributa vrednosti. Podatkovni tipi so v OPC UA določeni kot Int32, Double, String ali strukture.
ValueRank	Spremenljivke	Hrani podatek o tem, ali je vrednost skalar, tabela ali večdimenzionalna tabela.
AccessLevel	Spremenljivke	Določa način dostopa do spremenljivke (branje ali pisanje)

2.4.4. Vmesnik za dostop do naslovnega prostora strežnika OPC UA

Slika 2.4.5. prikazuje komunikacijski kanal, ki se vzpostavi ob prenosu podatkov med strežnikom in odjemalcem OPC. Objekti so opisani v tabeli 2.4.



Slika 2.12. – komunikacijski kanal med odjemalcem in strežnikom OPC

Tabela 2.4. – objekti komunikacijskega kanala med odjemalcem in strežnikom OPC

Objekt	Opis
Varna povezava (angl. Secure Channel)	Varna povezava je realizirana preko komunikacijskega sklada. Objekti na aplikacijski plasti so neodvisni, prosto živeči, ustvarjeni, spremenjeni in uporabljeni so pa lahko samo v kontekstu varne povezave. Če pride do prekinitve varne povezave in ponovne vzpostavitve le-te, mora biti dodeljena seji na aplikacijskem nivoju.
Seja (angl. Session)	Seja je logična povezava med odjemalcem in strežnikom. Vsebuje uporabniške informacije in jezikovne nastavitve za povezavo. Seja propade, če ni odjemalčevih klicev po določeneni časovni omejitvi. Časovno omejitev določi odjemalec. Seja je neposredno povezana s komunikacijskim kanalom, vendar se ji ob prekinitvi lahko dodeli novega.
Naročnina (angl. Subscription)	Naročnino ustvari odjemalec na skupino točk oz. spremenljivk, ki jih spremlja (angl. monitoring items). Spremlja lahko spremembe vrednosti ali prejema sporočila o dogodkih. Strežnik prekine naročnino, če ne more poslati podatkov ali signala *KeepAlive* po določeni časovni omejitvi. Omejitev določi odjemalec.

Objektom komunikacijskega kanala pripadajo metode za:

- vzpostavljanje povezave,
- sejo,
- naročnino.

Metode, ki smo jih uporabili pri implementaciji odjemalca za strežnik OPC UA, so opisane v poglavju 4.3.

2.4.5. Prehod iz klasičnega vmesnika OPC na vmesnik OPC UA

Vmesnik OPC UA ponuja različne migracijske strategije za različne potrebe in različne nivoje posvojitev funkcij OPC UA. Osnovni nivo ne zahteva nikakršnih sprememb na obstoječih produktih. Fundacija OPC ponuja različne ovojnice (angl. wrappers) in nadomestki (angl. proxy component), ki prevedejo klasične vmesnike OPC v OPC UA in obratno. Ta nivo je primeren za integracijo obstoječih produktov OPC v komunikacijska omrežja OPC UA.
Ovojnice OPC UA so programska oprema, ki odjemalcem OPC UA omogoča dostop do klasičnih strežnikov OPC in imajo istočasno dve funkciji:

- služijo kot odjemalec OPC za eno od klasičnih arhitektur strežnikov OPC in
- služijo kot strežnik OPC UA, ki omogoča odjemalcem OPC UA, da komunicirajo s klasičnim strežnikom OPC.

Nadomestki so oprema OPC UA, ki omogoča klasičnim odjemalcem OPC dostop do strežnikov OPC UA in ima istočasno dve funkciji:

- služi kot odjemalec OPC UA za strežnik OPC UA in

- služi kot posrednik, ki ustvari vmesnik COM klasičnega strežnika OPC, ki omogoča klasičnim odjemalcem OPC dostop do strežnika OPC UA.

Ovojnice in nadomestki ne prinašajo nobenih dodatnih funkcij za OPC UA, ampak samo prevajajo funkcionalnosti klasičnih arhitektur OPC v arhitekturo OPC UA in obratno.
Glavno področje uporabe je integracija OPC UA z obstoječim naborom produktov OPC. To je tudi največji cilj OPC UA, saj je omogočen preprost prehod iz klasičnega OPC v OPC UA. S tem so pretekle investicije v OPC zaščitene, obstoječa oprema je lahko osnova za OPC UA. Produkti za prehod iz starejše v novejšo arhitekturo OPC lahko tečejo na istem sistemu vzporedno, brez izgube obstoječe funkcionalnosti.
Fundacija OPC svojim članom ponuja dokumentacijo in izvorno kodo za ovojnice in nadomestke. Trenutno je koda in dokumentacija na voljo za vmesnike OPC DA, OPC AE in OPC HDA za Microsoft .NET ogrodje. Poleg fundacije OPC pa že izdelane rešitve ponujajo številna podjetja, ki so ponudniki produktov OPC.

3. Krmilni računalniki in sistemi SCADA

3.1. Krmilni računalniki

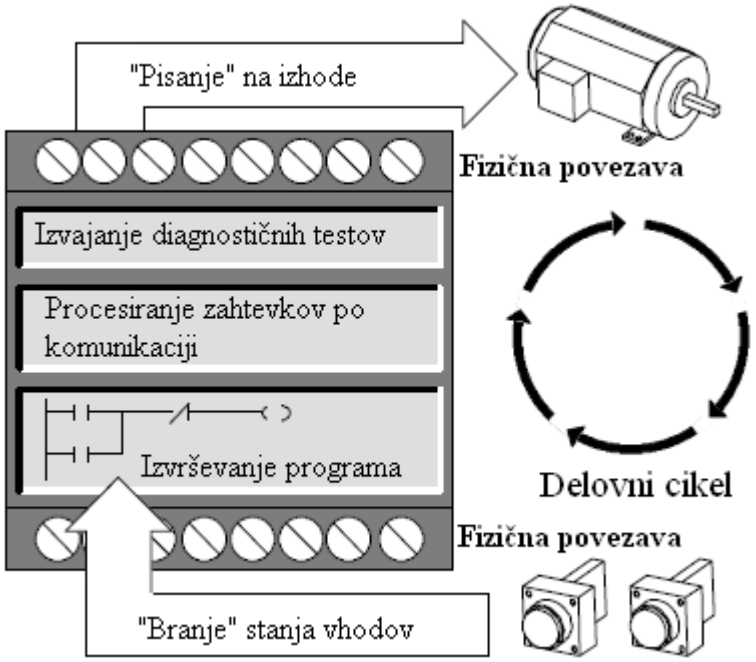
Krmilni računalniki ali programirljivi logični krmilniki (PLK) (angl. Programmable Logic Controllers - PLC) so pogosto opisani kot miniaturni industrijski računalniki, sestavljeni iz strojne in programske opreme. Uporablja se jih v vseh vejah industrije za krmiljenje najrazličnejših funkcij. PLK je sestavljen iz dveh osnovnih sestavnih delov:

- centralno-procesna enota (CPE) in
- vhodno-izhodnega sistema.

Centralna procesna enota, ki nadzira vse funkcije PLK, je razdeljena na procesni in pomnilniški del. Procesni del prevzema in izvršuje ukaze iz pomnilnika, v pomnilniku pa so shranjeni ukazi in operandi. Tudi pomnilnik je sestavljen iz več pomnilniških tehnologij, vsaka služi svojemu namenu (npr. ločen pomnilnik za program in podatke).
Vhodno-izhodni sistem je fizično povezan s področnimi napravami (stikala, senzorji, ventili, signalne lučke, ...) in tako predstavlja vmesnik med napravami, ki veličine iz realnega sveta pretvorijo v električne signale in napravami, ki električne signale pretvarjajo v dejanja. Prvim omenjenim napravam rečemo senzorji, drugim pa aktuatorji. [3]
S senzorji zajemamo veličine iz okolja, kot so npr. temperatura, tlak, sila, pretok, hitrost itd. Aktuatorji so glede na tehnologijo izdelave lahko električni, pnevmatski ali hidravlični. Preko njih lahko učinkujemo na okolico, na primer, vključimo črpalko za ogrevalno vodo.
Krmilni računalnik ciklično izvaja (angl. scanning) naslednje operacije [6]:

- branje vhodov, ko se stanje iz fizičnih vhodov prepiše v vhodni register,
- izvajanje krmilnega programa, v katerem CPE izvede ukaze in shrani vrednosti operandov v različna pomnilniška področja,
- procesiranje zahtev za komunikacijo,
- izvajanje diagnostičnih testov s katerimi se preveri pravilnost delovanja operacijskega sistema, programskega pomnilnika in vhodno-izhodnega sistema in
- pisanje vrednosti na izhode, s čimer se vrednosti v izhodnem registru prepišejo na fizične izhode.

Ponazoritev delovanja je na sliki 3.1.



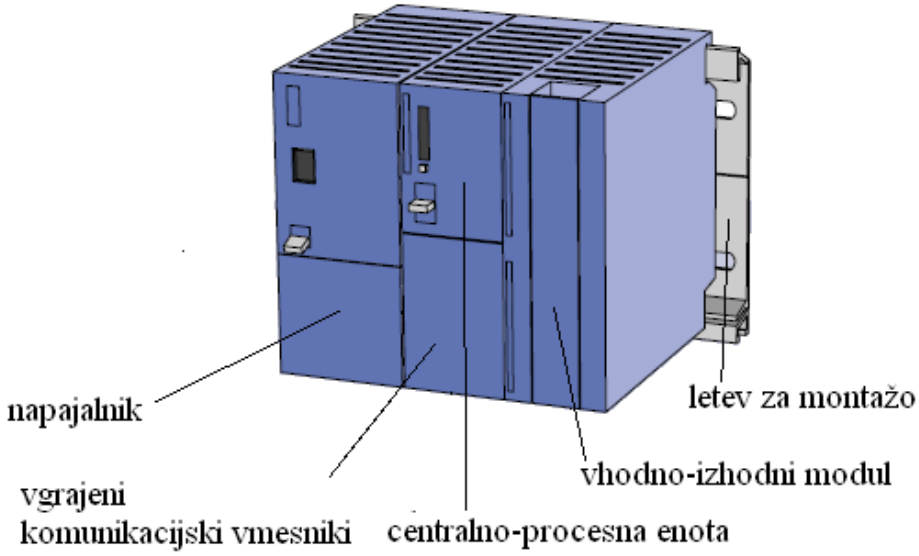
Slika 3.1. – Delovni cikel krmilnega računalnika

Krmilni računalnik se je prvič pojavil leta 1968, ko so za podjetje General Motors razvili napravo, imenovano digitalni modularni krmilnik (angl. Modular Digital Controller for General Motors). Znan je bil tudi pod imenom MODICON, z njim pa naj bi v podjetju GM zamenjali tradicionalno stikalno (relejsko) tehniko v krmilnih sistemih. Krmilni računalniki so se hitro vpeljali, predvsem zaradi manj potrebnega ožičenja v krmilnih omarah, preprostega odpravljanja napak in relativno nezapletenega programiranja.
Krmilni računalniki so do današnjih dni precej napredovali in ponujajo:

- pester izbor komunikacijskih protokolov,
- hitrejše programske cikle,
- manjše in zmogljivejše vhodno-izhodni sisteme,
- vmesnike za posebne področne naprave,
- naprednejšo diagnostiko in
- programska orodja za programiranje in parametriranje PLK z več programskimi jeziki in metodami programiranja.

3.1.1. Krmilni računalnik Siemens Simatic S7-300

Za namene testiranja smo v tej diplomski nalogi uporabili sodoben, modularen krmilni računalnik Siemens serije S7-300 [9]. Slika 3.2. prikazuje modularno zgradbo tega PLK.



Slika 3.2. – modularna zgradba krmilnega računalnika Siemens Simatic serije S7-300

Glede na zahtevnost aplikacije, ki jo želimo avtomatizirati je pri seriji S7-300 na voljo 13 različnih procesorjev v osnovni ali kompaktni izvedbi. Pri slednji so k samemu procesorju vgrajene tehnološke funkcije kot so na primer števniki, digitalne in analogne vhode in izhodi, komunikacijski vmesniki (MPI, PROFIBUS, Ethernet, PROFINET…).
Zmogljiveši procesor ima krajše cikle izvajanja programa in večji pomnilnik. Tudi število naprav in razširitvenih modulov, ki jih lahko priključimo na krmilni računalnik, se z zmogljivostjo veča. Tabela 3.1. prikazuje tehnične podatke nekaterih standardnih procesorjev iz serije S7-300:

Tabela 3.1. – tehnični podatki nekaterih krmilnih računalnikov serije S7-300

Tip krmilnika	**312**	**314**	**315-2 DP/ 315-2 PN/DP**	**317-2 DP 317-2 PN/DP**
				
Programski pomnilnik	32 kB	96 kB	128/256 kB	512/1024 kB
Maks. št. digitalnih vhodov/izhodov	256	1026	1024	1024
Maks. št. analognih vhodov/izhodov	64	256	256	256
Čas izv. operacij: bit/beseda fiksna/plavajoča vejica	0,2/2 µs 5/6 µs	0,1/1 µs 2/3 µs	0,1/1 µs 2/3 µs	0,05/0,2 µs 2/3 µs
Zastavice (angl. flags)	1024	2048	16384	32768
Števci	128	256	256	512
Maks. št. povezav	6	12	16	32
Komunik. vmesniki	MPI	MPI	PROFIBUS-DP Master/Slave PROFINET/Ethernet	PROFIBUS-DP Master/Slave PROFINET/Ethernet MPI uporaben kot DP

V našem primeru smo uporabili krmilni računalnik tipa 314 IFM ki ima

- vgrajenih 16 digitalnih vhodov in 16 digitalnih izhodov in
- vgrajen funkcijski modul,

dodali pa smo mu še:

- modul s 16 digitalnimi vhodi in 16 digitalnimi izhodi,
- modul z analognimi vhodi in
- komunikacijski modul za protokol PROFINET/Ethernet.

Podrobnejša konfiguracija testnega sistema je opisana v poglavju 3.3.

3.2. Sistemi SCADA

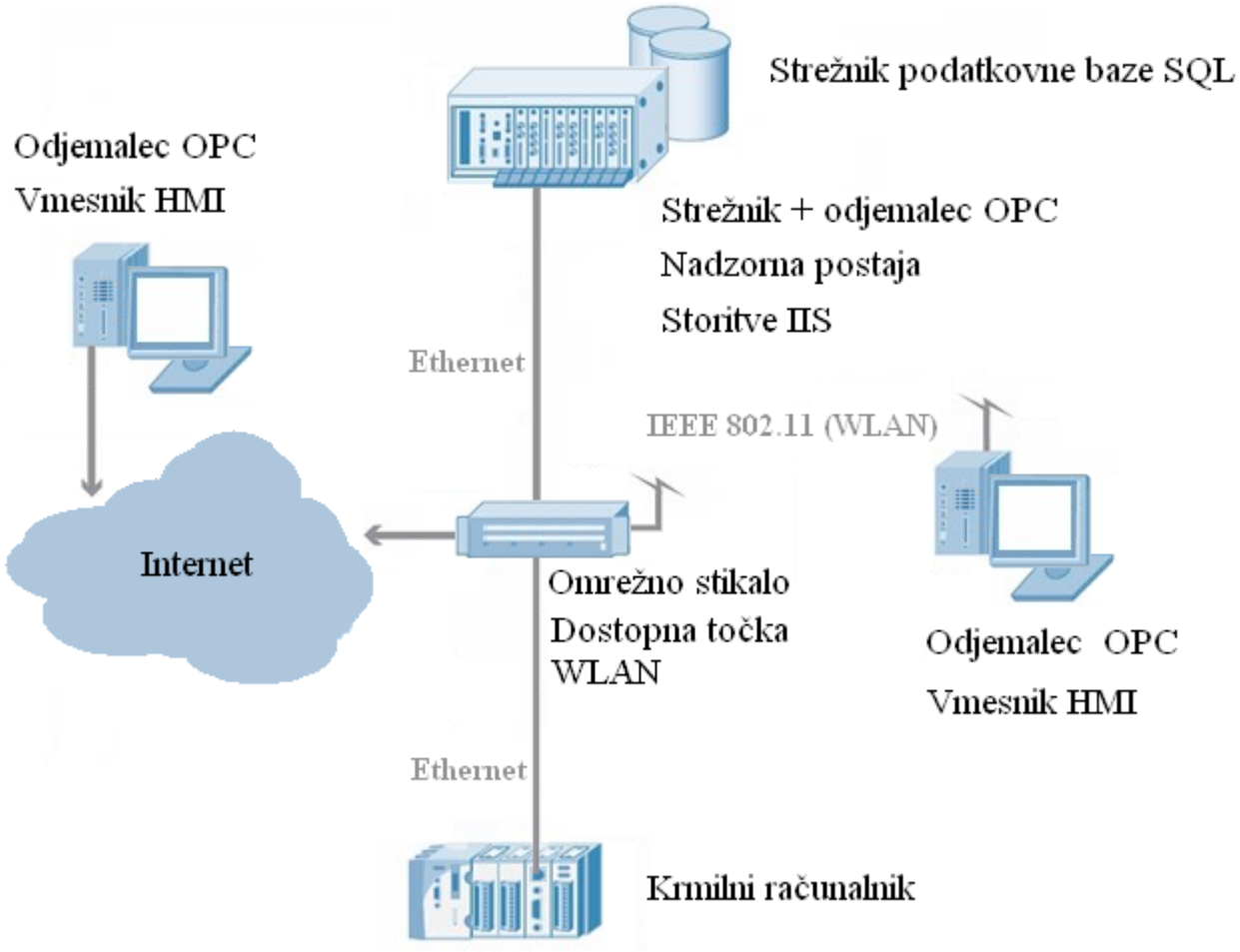
Proizvodni nadzorni sistemi SCADA (angl. System Control and Data Acquisition) predstavljajo industrijske sisteme za nadzor in krmiljenje, kjer za različne procese skrbi računalniški sistem. Poleg različnih področij v industriji se se ti sistemu uporabljajo še v procesih in infrastrukturi, na primer, v oskrbi s pitno vodo, ravnanju z odplakami, oskrbi z električno energijo, v velikih telekomunikacijskih sistemih, plinovodih, naftovodih.
Vesoljske postaje, različna plovila (tovorne, potniške ladje), industrijske, komercialne in javne zgradbe imajo ravno tako vgrajene sisteme SCADA za kontrolo pristopa, spremljanje porabe električne energije in ostalih energentov, ter nadzor nad gretjem, prezračevanjem in klimatizacijo.
Sisteme SCADA sestavljajo naslednji podsistemi: [13]

- vmesniki človek-stroj (angl. Human Machine Interface, HMI) so naprave, ki operaterju prikazujejo procesne podatke in mu omogočajo nadzor nad procesom in njegovo spremljanje,
- nadzorna postaja ali nadzorni računalnik, zajema potrebne podatke, jih po potrebi hrani in pošilja procesu ukaze,
- enote RTU (angl. Remote Terminal Units) in krmilni računalniki, ki so priključeni na senzorje v procesu in pretvarjajo signale v digitalno obliko, primerno za pošiljanje podatkov nadzornemu sistemu ter
- komunikacijska infrastruktura med krmilnimi računalniki oziroma enotami RTU in nadzornim sistemom.

Splošno povedano, sistemi SCADA ne krmilijo procesov (za to skrbijo krmilni računalniki) ampak jih koordinirajo, spremljajo in pošiljajo ukaze podrejenim sistemom v realnem času.
Za primer si zamislimo proces, kjer krmilni računalnik skrbi za proces pretoka hladilne vode skozi sistem, ki ga je potrebno hladiti. Sistem SCADA omogoča operaterju, da omogoči alarme v primeru sprememb v pretoku ali nastavi želen pretok. Prikazuje in beleži različne dogodke kot so previsoka temperatura ali anomalije v pretoku. Sistem SCADA skrbi za zanesljivo delovanje procesa kot celote, medtem ko je krmilna zanka omejena le na krmilni računalnik in pripadajoče senzorje ter aktuatorje.
Arhitekturo sistema in funkcije podsistemov SCADA bomo konkretno predstavili v naslednjem poglavju na našem testnem sistemu.

3.3. Arhitektura testnega sistema SCADA

Slika 3.3. prikazuje arhitekturo našega testnega sistema SCADA. Testni SCADA sistem je bil postavljen za nadzor ogrevanja, temperature in nekaj ostalih preprostih funkcij v industrijski zgradbi. Na tem sistemu smo testirali različne arhitekture strežnikov in odjemalcev OPC, ki smo jih opisali v prejšnjih poglavjih. Opis podsistemov in uporabljenih orodij v našem testnem sistemu sledi v naslednjih podpoglavjih.



Slika 3.3. – testni sistem SCADA

3.3.1. Funkcije krmilnega računalnika

Krmilni računalnik skrbi za:

- krmiljenje gorilnika kotla za ogrevanje, glede na temperaturo v kotlu,
- krmiljenje črpalke za ogrevalno vodo za pisarno, glede na nastavljen režim na termostatu,

- krmiljenje črpalke za ogrevalno vodo za proizvodno halo, glede na nastavljen režim na termostatu,
- krmiljenje požarne luči glede stanje svetlobnega senzorja (dan/noč),
- krmiljenje kompresorske postaje,
- zajemanje podatkov iz notranjega senzorja temperature (temperatura v pisarni) in
- zajemanje podatkov iz zunanjega senzorja temperature (temperature ozračja).

Krmilni računalnik je preko področnega vodila in omrežnega stikala povezan z nadzornim računalnikom. V našem primeru smo za področno vodilo uporabili industrijski Ethernet, ki ga podpira komunikacijski modul.

3.3.2. Industrijski Ethernet

Industrijski Ethernet se veliko ne razlikuje od običajnega, pisarniškega Etherneta. Ena od razlik med njima so redundantne povezave v industrijskem Ethernetu. V industriji je zanesljivost prenosa podatkov visoka prioriteta, zato moramo rezervno povezavo primeru izpada linije. To pri industrijskem Ethernetu zagotavljamo s topologijo v obliki obroča. [7]
V redundantnih povezavah so v industrijskem Ethernetu vpeljani standardi, ki pripomorejo k zanesljivosti prenosa podatkov. Pri klasičnem Ethernetu se namreč ne zgodi nič, če podatek ne pride na cilj. Dobimo le obvestilo, da je povezava prekinjena. Standard IEEE 802.1d, ki je uporabljen v klasičnem Ethernetu, predvideva ob prekinitvi poskus ponovne vzpostavitve povezave v 30 sekundah. To je za industrijske zahteve absolutno prepočasi.
Standard Hiper Ring, ki temelji na algoritmu vpetega drevesa v redundantnih omrežjih industrijskega Etherneta omogoča ponovno vzpostavitev povezave v 0,5 sekunde. Za ta standard skrbi podjetje Hirschmann, specializirano v avtomatizaciji in omrežnih sistemih in ga v klasičnem, pisarniškem Ethernetu ne srečamo.[11].
Industrijska mrežna oprema kot so usmerjevalniki, stikala, dostopne točke, kabli, konektorji, je prilagojena agresivnim okoljem. V industrijskih okoljih je navadno izpostavljena velikim temperaturnim razlikam, vlagi, vibracijam, prahu, elektromagnetnim motnjam in podobnim neprijetnostim. Za ta namen je oprema izdelana v skladu z višjimi standardi in posledično dražja. Tako oprema, kot sam standard industrijski Ethernet sta popolnoma združljiva z običajnim, pisarniškim Ethernetom.
Industrijski Ethernet danes vse pogosteje zamenjuje klasična področna vodila, kot so Modbus, Profibus, CANOpen, DeviceNet in ostali, (zaprti) standardi različnih proizvajalcev. Komunikacijske naprave za domačo rabo, kot so IP-telefoni, pametni mobilni telefoni z vgrajenim vmesnikom WLAN, televizorji z Ethernet vmesnikom za brskanje po spletu in ostali, dandanes vsi uporabljajo protokol TCP/IP. Razvoj v industrijski avtomatiki gre, sicer bolj zadržano, naprej v podobni smeri.

3.3.3. Programsko okolje Siemens Simatic Step 7

Programsko okolje, ki smo ga uporabili za parametriranje in programiranje krmilnega računalnika Siemens Simatic S7-300 je Siemens Simatic Step 7. Okolje je sestavljeno iz množice programskih urejevalnikov. Osnovna verzija okolja vsebuje: [8]:

- urejevalnik SIMATIC Manager, ki služi za administracijo vseh orodij in podatkov v našem projektu,
- orodje za konfiguracijo strojne opreme,
- programske urejevalnike za razvijanje in testiranje uporabniških programov po standardu IEC 61131-3,
- orodje NetPro za konfiguracijo in vzpostavitev omrežnih povezav med napravami v sistemu,
- orodje za diagnostiko strojne opreme in pregled nad stanjem naprav v sistemu,
- funkcije za izdelavo krmilnih zank PID (angl. proportional–integral–derivative controller),
- odprt ukazni vmesnik za uvoz ali izvoz podatkov iz ostalih orodij v operacijskem sistemu Windows in
- orodje za izdelavo projektne dokumentacije.

Poleg naštetih funkcionalnosti ima razširjena verzija Professional na voljo še simulator za testiranje sistema v odsotnosti strojne opreme in podporo za dva dodatna programska jezika po standardu IEC:

- S7-SCL (angl. Structured Control Language), ki je visokonivojski programski jezik, podoben jeziku Pascal in je uporaben predvsem za programiranje kompleksnih algoritmov, matematičnih funkcij in procesiranju večjih količin podatkov. [12] in
- S7-Graph (Sequental Function Chart), ki je programski jezik, podoben diagramu poteka in se uporablja predvsem za programiranje zaporednih operacij, kot je na primer koračna veriga stroja.

Programski jeziki, ki smo jih že omenili in so podprti v osnovni verziji okolja Simatic Step 7, so:

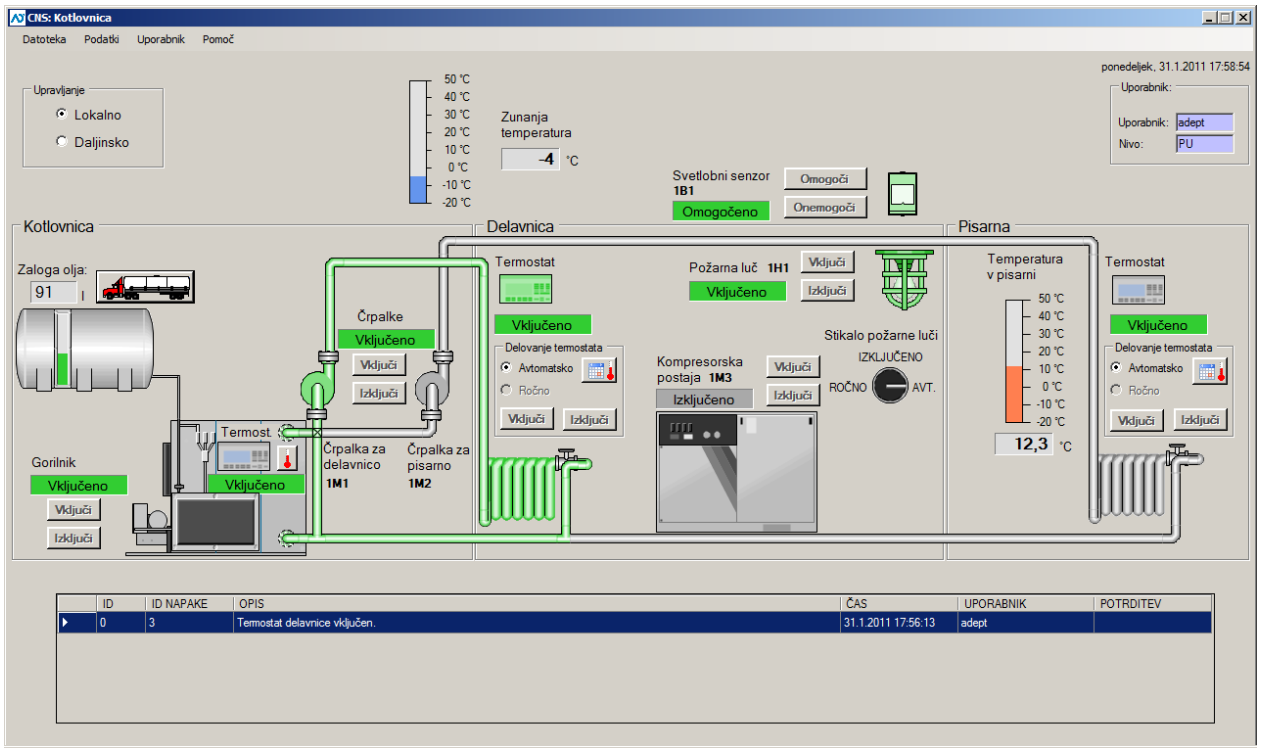
- lestvični diagram (angl. kratica LAD – Ladder Logic), grafični programski jezik s sintakso, podobno načrtu stikalne (relejske) tehnike,
- nabor ukazov (angl. kratica STL – Statement List), tekstovni programski jezik, podoben zbirniku in
- funkcijski blokovni diagram (angl. kratica FBD – Function Block Diagram), grafični programski jezik, ki v sintaksi uporablja bloke iz Boolove algebre.

V tem v tem poglavju nas najbolj zanima konfiguracija omrežja z orodjem NetPro in ne toliko sam krmilni program, ki smo ga napisali za izvajanje krmilnih funkcij. Omrežne konfiguracije testnega sistema SCADA izdelane v NetPro so predstavljene v poglavju 4.

3.3.4. Vmesnik HMI

V arhitekturi testnega sistema SCADA (slika 3.3.) sta določena dva vmesnika HMI. Prvi je na nadzorno postajo povezan prek lokalnega brezžičnega omrežja (WLAN), drugi pa prek interneta.

Funkcionalnih razlik med njima ni, so pa določene sistemske razlike, ki pa so za operaterja nevidne in nepomembne. Sistemske razlike so opisane v poglavju 4. Grafični vmesnik je prikazan na sliki 3.4. [30]



Slika 3.4. – grafični vmesnik HMI

Operater v realnem času preko vmesnika spremlja naslednje procese oz. parametre:

- delovanje gorilnika kotla za ogrevanje,
- delovanje ogrevanja za pisarno in proizvodno halo,
- delovanje svetlobnega senzorja in požarne luči,
- delovanje kompresorske postaje,
- zunanjo in notranjo temperaturo,
- stanje stikala za požarno luč,
- ogrevalne režime, nastavljene na termostatih,
- procesne alarme in dogodke.

Vsi procesi so na računalniškem zaslonu predstavljen shematsko. Stanja procesa so upodobljena z grafičnimi animacijami, s senčenjem z zeleno barvo je ponazorjena aktivnost (delovanje), s sivo pa neaktivnost oziroma mirovanje procesa. Na sliki 3.4. so shematski elementi sistema ogrevanja v proizvodni hali oziroma delavnici (črpalka za ogrevalno vodo, cevovod, radiator) obarvani zeleno, kar nakazuje na trenutno aktivnost tega procesa.
Ostali parametri so predstavljeni numerično, predstavitve temperatur so dodatno obogatene s stolpičastima grafikonoma.
Vsi podatki se arhivirajo in so na voljo za kasnejši prikaz v tabelarični in grafični obliki. Dostop do arhivskih podatkov in procesni alarmi niso neposredno realizirani preko vmesnika OPC, tako da nas na tem mestu ne zanimajo.

Operater vnese spremembe parametrov in izvede ukaze prek vnosnih polj, izbirnih gumbov in običajnih gumbov v tem istem grafičnem vmesniku. Fizični vmesnik je v našem primeru računalniška miška in tipkovnica.
Industrijski vmesniki HMI imajo lahko tudi na zaslonu vgrajene tipke, zaslon občutljiv na dotik in podobno. Na voljo so v različnih oblikah in velikostih za različne namene, od enovrstičnih tekstovnih do 19 in več palčnih grafičnih, v stacionarni ali prenosni obliki. Obstajajo različne posebne izvedenke, na primer za uporabo v prehrambni industriji ali eksplozivnih conah.

3.3.5. Programsko okolje Microsoft Visual Studio .NET

Aplikacijo, ki služi kot vmesnik HMI smo izdelali s pomočjo programskega orodja Microsoft Visual Studio. To je močno programsko okolje za razvoj aplikacij (angl. Integrated Development Environment, IDE) [14].
V grobem sestoji iz [16]:

- urejevalnika izvorne kode in grafičnega urejevalnika uporabniškega vmesnika aplikacije,
- prevajalnika,
- orodij za izdelavo izvršljive verzije programa (angl. build automation tools) in
- razhroščevalnika.

Microsoft Visual Studio omogoča razvoj različnih aplikacij v različnih programskih jezikih. Naša aplikacija je okenska in teče na operacijskem sistemu Windows. Napisana je v programskem jeziku Visual C#, ki je sodoben, višje-nivojski objektno orientiran programski jezik.
Aplikacije, napisane v programskem jeziku Visual C#, dostopajo do funkcij operacijskega sistema preko ogrodja Microsoft.NET. To ogrodje ponuja skupni jezikovni izvajalnik (angl. kratica CLR – Common Language Runtime) in predstavlja vmesno abstraktno plast med operacijskim sistemom in aplikacijo. Ponuja osnovne knjižnice za programsko nezahtevne aplikacije in različna razvojna ogrodja za večje in zahtevnejše aplikacije. [15]. Podpira več programskih jezikov, kar omogoča interoperabilnost med njimi. Ogrodje .NET odlikujejo še nekatere varnostne funkcije, preprosto uvajanje in prenosljivost. Programi, napisani z uporabo ogrodja .NET, se lahko izvajajo na kateri koli platformi, kjer je ogrodje nameščeno.
Pri Microsoftu niso nikoli implementirali ogrodja .NET na kateri drugi platformi razen na operacijskem Windows. Pojavili pa so se izvajalniki, kompatibilnih z Microsoft .NET tudi za druge platforme. Eden takih je npr. Mono [17].

3.3.6. Vpeljava vmesnika OPC in možne alternativne rešitve

Do sedaj smo opisali vse podsisteme, ki sestavljajo naš sistem SCADA (slika 3.3.). Potrebno je le še povezati dva različna svetova med seboj: krmilni računalnik s svojim namenskim operacijskim sistemom in lastno predstavitvijo podatkov ter svet osebnih računalnikov. Na osebnih računalnikih imamo nameščene večuporabniške in večopravilne operacijske sisteme, aplikacije so napisane v višjenivojskih objektno usmerjenih programskih jezikih. Potrebovali smo programsko opremo – nekakšen gonilnik za krmilni računalnik. Omogočati bi moral dostop do krmilnega računalnika in vsebovati knjižnice s čim več uporabnimi funkcijami za takojšnjo uporabo v .NET ogrodju.

Odgovor na to je bila arhitektura odjemalec-strežnik OPC. Strežnik OPC je nameščen na nadzorni postaji in služi za:

- povezavo s krmilnim računalnikom; strežnik v našem primeru vsebuje gonilnik za povezavo s krmilnimi računalniki serije Simatic S7-300 in
- povezavo z odjemalcem; omogoča mu priključitev, razkriva svoj naslovni prostor in nudi specifične storitve glede na arhitekturo in tip vmesnika.

Odjemalec OPC je implementiran v vsakem vmesniku HMI. Drugače povedano: program, ki teče na vmesniku HMI, vsebuje tudi implementacijo razredov odjemalca OPC. Njegova naloga je dostop do procesnih podatkov, ki so potem prek vmesnika na voljo operaterju za prikaz in urejanje.
Na sliki 3.3., kjer je predstavljen testni sistem SCADA, vidimo, da na nadzorni postaji tečeta dva ločena procesa: odjemalec OPC in strežnik OPC. Odjemalec OPC je v tem primeru zadolžen za dostop do procesnih podatkov in njihovo shranjevanje v sodelovanju s strežnikom podatkovne baze SQL. Vmesniki HMI nato dostopajo do arhivskih podatkov preko podatkovne baze SQL brez uporabe vmesnika OPC, kot smo že omenili v poglavju 3.3.4. S tem je naš sistem SCADA postal zaključena celota z vsemi realiziranimi podsistemi in pomembnejšimi funkcijami.
Nikakor pa ni to edini način za izdelavo tovrstnih sistemov. Na trgu je namreč kar nekaj programskih rešitev, ki v enem paketu ponujajo vsa orodja za izdelavo industrijskih SCADA sistemov. Eden takšnih namenskih paketov je Siemens Simatic WinCC 7.0. Ponuja popolno osnovno funkcionalnost za vizualizacijo procesov in njihovo posluževanje. Številni vgrajeni vmesniki in urejevalniki omogočajo prilagoditev funkcionalnosti za specifične aplikacije. [19]. Vsa orodja in vmesniki so uporabniku prijazna in vključujejo izdatno grafično podporo. Splošni, pogosto uporabljeni postopki pri razvoju sistemov SCADA so avtomatizirani in poenostavljeni. Razvijalec lahko tako rekoč z enim klikom kreira podatkovno bazo za arhivske podatke, določi povezavo med krmilnim računalnikom in vmesnikom HMI ali določi okno za procesna sporočila in alarme v uporabniškem vmesniku. Licence za tovrstna »ad-hoc« programska orodja za razvoj sistemov SCADA so nekajkrat dražja od programskih orodij za splošne namene, kot je Microstoft Visual Studio. Poleg licence za razvojno okolje pa potrebujemo tudi izvajalne (angl. runtime) licence za vsak produkt posebej, ki ga razvijemo s pomočjo teh orodij in prodamo končnim kupcem. Cene izvajalnih licenc naraščajo s številom uporabljenih procesnih spremenljivk. Za naš sistem, razvit s pomočjo orodja Visual Studio, Microsoftova politika licenciranja ne zahteva izvajalnih licenc.
Povezavo med krmilnim računalnikom in PC lahko realiziramo s programskimi vtiči (angl. socket coomunication) prek protokola TCP/IP. Za to moramo imeti ustrezno strojno opremo na strani krmilnega in nadzornega računalnika. Krmilni računalnik morata imeti Ethernet priklop im omogočati komunikacijo tipa SEND/RECIEVE [10]. Zavedati se moramo, da tak način komunikacije ponuja le transportni nivo, ostale storitve in funkcije, ki jih potrebujemo, moramo na strani aplikacije za PC napisati sami. V mislih imamo predvsem varnost, vmesnik za alarme in dogodke, skrb za pravilno interpretacijo surovih podatkov in podobno.

4. Testiranje arhitektur vmesnikov OPC

V 2. poglavju o vmesnikih OPC smo omenili, da so razne knjižnice, izvorna koda in dokumentacija za razvoj strežnikov in odjemalcev OPC na voljo članom Fundacije OPC. Tega pa si mi žal nismo mogli privoščiti in smo zato ubrali drugačno pot. Strežnike in odjemalce OPC, ki so že pripravljeni za uporabo v sistemih SCADA in podobnih, ponuja mnogo proizvajalcev. Za namene našega testiranja smo uporabili strežnike OPC proizvajalca Siemens, ki seveda ustrezajo naši strojni opremi. Odjemalca OPC za klasični primer OPC smo izbrali med produkti podjetja Advosol. Za razvoj in testiranje v .NET ogrodju nam je služila preskusna (demo) verzija komponente OPC DA .Net Client Development Component [20]. Več o sledi tem v poglavju 4.1.3.
Odjemalca OPC za ostali dve arhitekturi OPC XML-DA in OPC UA smo izdelali sami s pomočjo dokumentacije za ustrezne arhitekture Siemensovih strežnikov. V tej tehnični dokumentaciji se poleg vsebine o strežniku OPC nahaja tudi poglavje, namenjeno razvijalcem odjemalcev OPC. V naslednjih podpoglavjih smo opisali implementacijo treh različnih arhitektur vmesnikov OPC v testnem sistemu SCADA:

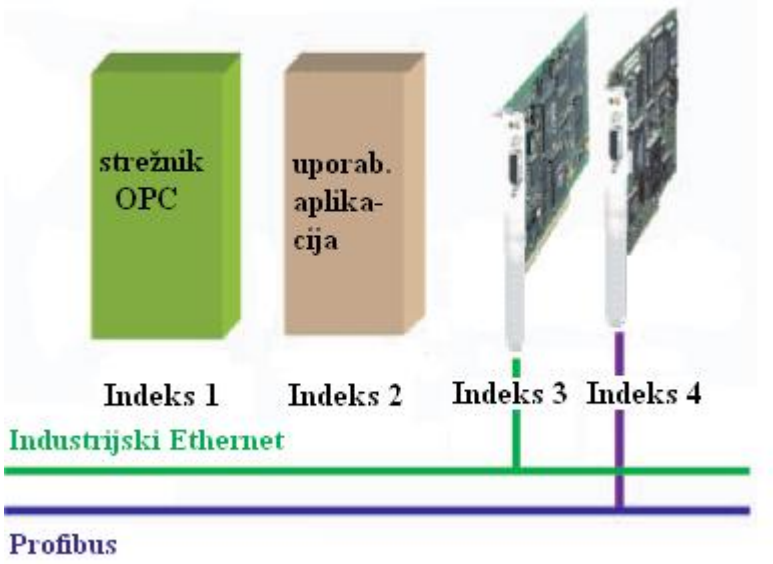
- klasični OPC za dostop do procesnih podatkov OPC DA,
- klasični OPC DA z vgrajenim prehodom XML: OPC XML-DA in
- poenoteno arhitekturo OPC UA.

4.1. Klasični OPC za dostop do procesnih podatkov OPC DA

4.1.1. Programski paket Siemens SIMATIC NET CD Edition 2008

Programski paket Siemens SIMATIC NET CD Edition 2008 [21] predstavlja programsko opremo za konfiguriranje storitev za industrijsko avtomatizacijo, osnovano na PC računalnikih. Glavna vsebina paketa:

- gonilniki in vmesniki za različne naprave, ki jih priključimo na osebni računalnik, s katerimi lahko prek množice protokolov komuniciramo s sistemi za avtomatizacijo.,
- klasični strežnik OPC DA s podporo različnim protokolom,
- program SIMATIC NCM PC, ki je v bistvu okrnjena različica SIMATIC STEP 7 programskega okolja, namenjena samo konfiguriranju osebnega računalnika za povezavo s sistemi za avtomatizacijo,
- orodje OPC Scout za odkrivanje strežnikov OPC, brskanje po naslovnem prostoru in delo s točkami, uporabno predvsem za testiranja v času razvoja sistema,
- orodje Station Configuration za sestavo avtomatiziranega sistema na PC računalniku, ki vključuje tako komponente strojne opreme kot tudi programske module (slika 4.1.) [22]



Slika 4.1. – navidezna letev s programskimi in strojnimi komponentami na PC računalniku

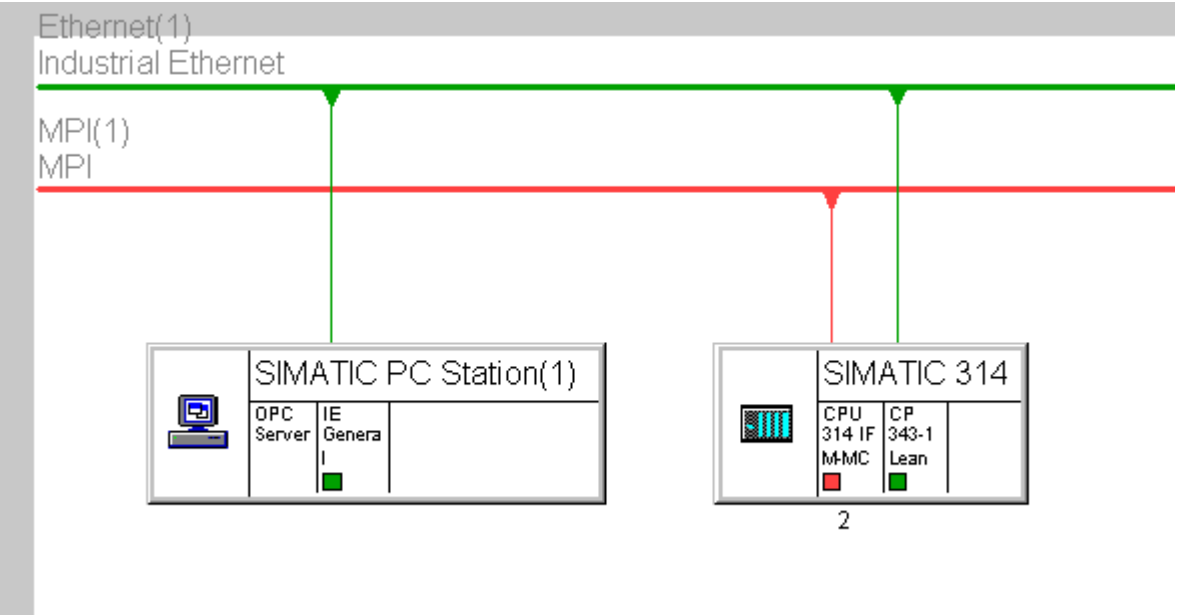
Za naša testiranja smo uporabili 14-dnevno preskusno različico programske opreme. Namestili smo jo na računalnik, ki smo ga predvideli kot strežnik. To je v našem sistemu nadzorna postaja (slika 3.3.).

4.1.2. Konfiguracija strežnika OPC

Za konfiguriranje strežnika OPC bi lahko uporabili priložen program SIMATIC NCM PC, a smo raje uporabili SIMATIC STEP 7. V njem smo obvladovali celoten projekt krmilnega sistema in konfiguracije nadzorne postaje:

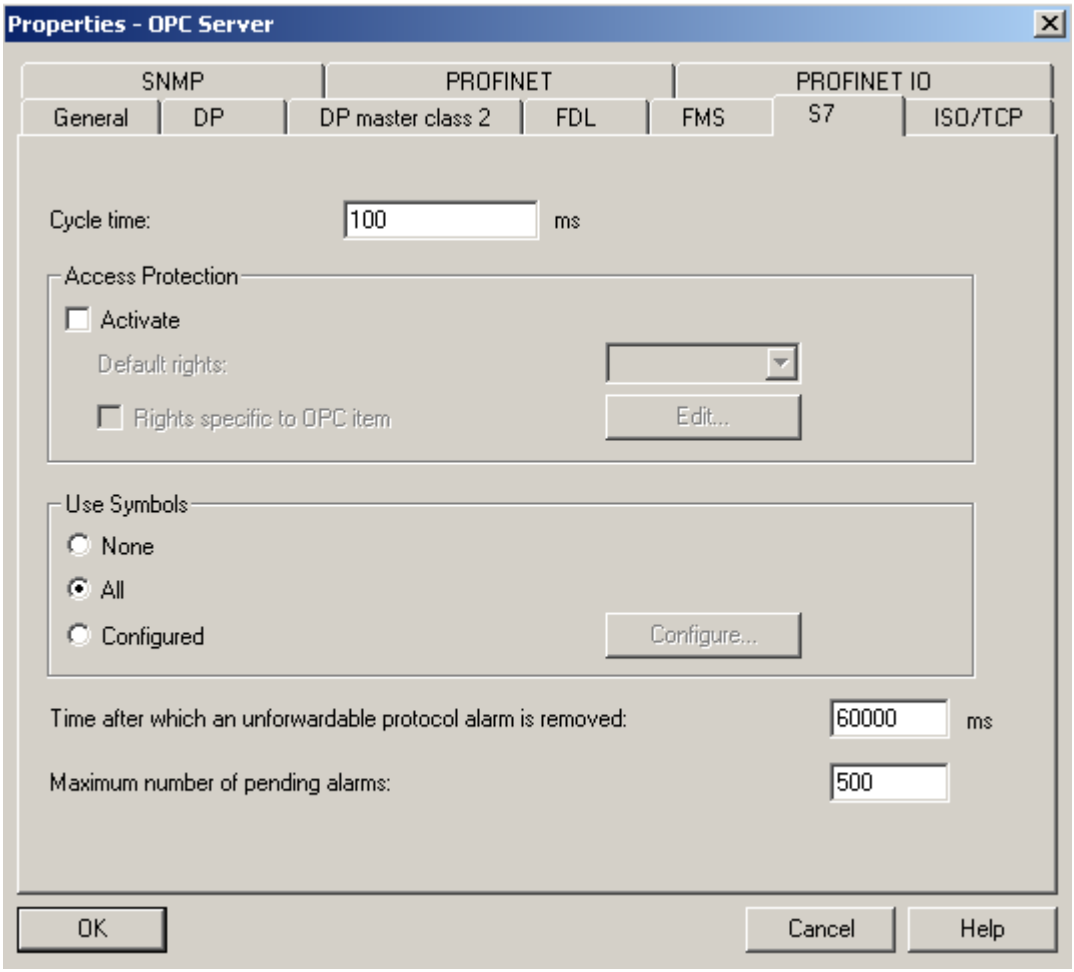
- določanje konfiguracije strojne opreme,
- razvoja krmilnega programa za PLK,
- določanje omrežnih povezav z nadzorno postajo in
- konfiguracijo strežnika OPC na nadzorni postaji.

Glavni del konfiguracije omrežnih povezav in samega strežnika OPC poteka v programu NetPro. Na sliki 4.2. so prikazane uporabljene naprave in omrežne povezave.



Slika 4.2. – konfiguracija omrežnih povezav v programu NetPro

Na strani nadzorne postaje (SIMATIC PC Station) smo v programsko-strojni konfiguraciji izbrali modul OPC Server, ki predstavlja strežnik OPC DA, in modul IE General, ki predstavlja mrežno kartico Ethernet. V našem primeru smo uporabili standardno Ethernet kartico. Pri proizvajalcu Siemens (in ostalih podobnih) so na voljo tudi industrijske izvedbe tovrstnih kartic. V primeru uporabe le-teh moramo v strojni konfiguraciji izbrati natanko tisto, ki jo imamo v resnici nameščeno. Isto velja za vse ostale namenske uporabljene komponente.
Na strani krmilnega računalnika (SIMATIC 314) smo v strojni konfiguraciji določili komunikacijski procesor CP 341-1 Lean in ga preko industrijskega Etherneta povezali z nadzorno postajo. Krmilni računalnik ima vgrajen tudi vmesnik za področno vodilo MPI, ki ga ne uporabljamo. Omrežnim napravam smo dodelili še naslov IP in masko podomrežja.
Lastnosti strežnika OPC določamo v istem programu preko okna z nastavitvami, ki ga prikazuje slika 4.3.



Slika 4.3. – določitev lastnosti strežnika OPC in nastavitve za specifične protokole

V zavihkih lahko določamo lastnosti parametrov za specifične protokole za prenos podatkov preko OPC. Na voljo imamo precej industrijskih protokolov, od različnih izvedb Profibus-a do PROFINET-a. Protokol SNMP (angl. Simple Network Management Protocol) je namenjen spremljanju aktivnih mrežnih naprav v omrežjih IP. Protokol ISO/TCP omogoča prenos podatkov preko Etherneta posredno prek vmesnika SEND/RECIEVE. To sta funkciji, ki sta del standardne Siemensove knjižnice za komunikacijo in sta na voljo v programskem urejevalniku okolja Step 7. Če hočemo vzpostaviti komunikacijo strežnika OPC s krmilnim računalnikom po tem protokolu, moramo funkcije uporabiti v krmilnem programu in jim določiti ustrezne parametre.
V našem primeru smo uporabili komunikacijo prek protokola S7. To je protokol na aplikacijski plasti, implementiran v Siemensovih sistemih [23]. Za prenos podatkov lahko izkorišča protokole iz nižjih plasti, kot sta Profibus ali industrijski Ethernet. Protokol S7 omogoča zelo tesno povezavo strežnika OPC in krmilnega računalnika Siemens. Strežniku (in posredno odjemalcu) OPC omogoča neposredno uporabo simbolov, ki smo jih določili tekom razvoja krmilnega programa. Praktično to pomeni neposreden dostop do procesnih podatkov oziroma spremenljivk v krmilnem računalniku.
V skupini Access protection lahko določamo način dostopa (branje, pisanje). Simbole, ki bodo dosegljivi preko strežnika OPC, izberemo v skupini nastavitev Use symbols (slika 4.3.). Pomembna nastavitev je še Cycle time, kjer določimo časovni interval osveževanja podatkov

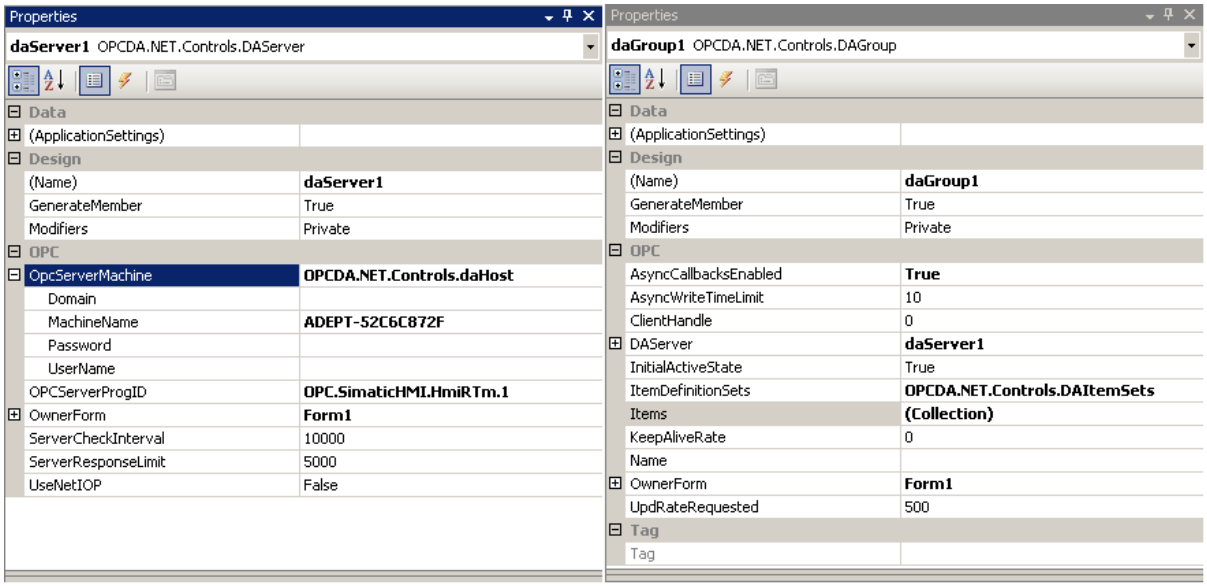
med krmilnim računalnikom in sistemom SCADA. Konfiguracijo prenesemo na strežniški računalnik z nameščenim programskim paketom Siemens SIMATIC NET CD Edition 2008 preko nastavitvene datoteke .xdb. S pomočjo orodja Station configuration uvozimo nastavitveno datoteko in tako je strežnik OPC DA v grobem pripravljen.

4.1.3. Razvojna komponenta Advosol OPC DA .Net Client Development Component

Advosol OPC DA .Net je ovojnica podjetja Advosol, namenjena razvoju odjemalcev OPC DA v programskem jeziku C# ali VB.NET. Ponuja razrede, kontrole in orodja za hiter in učinkovit razvoj aplikacij .NET. Kontrolo za dostop do strežnika OPC konfiguriramo z grafičnim orodjem Visual Studio Designer, kar zmanjša količino potrebne kode. Kontrola skrbi tudi za obravnavanje napak pri dostopu do strežnika.
Razred *QuickUse* ponuja preprosto uporabo metod za vpenjanje podatkov (angl. data binding), brskanje točk (angl. items) ter sinhrono in asinhrono branje in pisanje vrednosti točk. Uporaba te komponente ne predvideva posebnega predznanja o OPC.
V odjemalcu OPC DA smo testirali naslednje funkcije:

- spremljanje vrednosti določene točke (angl. item) v skupini OPC (angl. OPC group),
- branje vrednosti točk v skupini in
- vpisovanje vrednosti točk v skupini OPC.

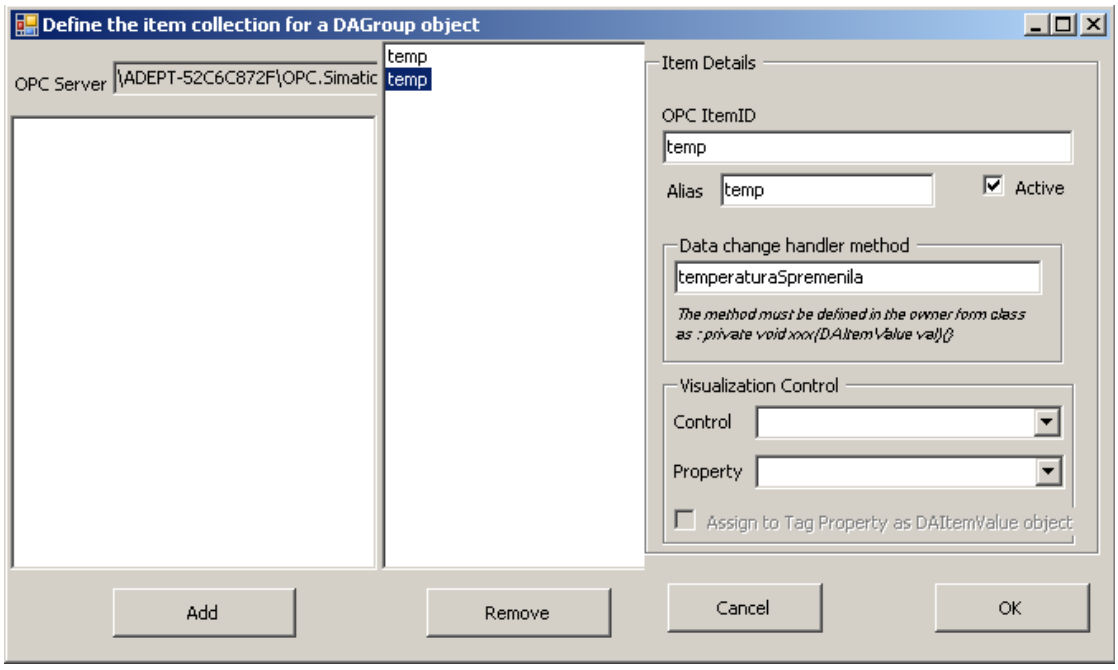
Glavni kontroli, ki omogočata povezavo s strežnikom in delo s točkami ter skupinami OPC, sta *DAServer* in *DAGroup*. Slika 4.4. prikazuje lastnosti, ki jima jih lahko določamo.



Slika 4.4. – lastnosti kontrol DAServer (levo) in DAGroup (desno)

Objekt *DAServer* omogoča, da preko okenskega vmesnika poiščemo strežniški proces *OPCServerProgID* na lokalnem računalniku ali v omrežju (lastnost *OPCServerMachine*). Na enak način s pomočjo kontrole *DAGroup* poiščemo točke v naslovnem prostoru strežnika in združimo v skupine. Pomembna lastnost je še *UpdRateRequested*, ki določa periodični čas

posodabljanja vrednosti točke na strežniku. Preko vgrajenega vmesnika nam je omogočeno, da se ob spremembi izbranega parametra izvede določena metoda, ali pa lastnosti neke druge kontrole priredimo vrednost točke. Slika 4.5. prikazuje ta vmesnik.



Slika 4.5. – grafični vmesnik za določanje zbirke točk objekta DAGroup

Točka *temp*, prikazana na sliki 4.1.4., predstavlja v našem sistemu SCADA procesni podatek notranje temperature zraka. V skupini nastavitev *Data change handler method* smo določili metodo, ki se izvede ob spremembi vrednosti te točke. Ta metoda v okenski aplikaciji služi za osveževanje stolpičastega grafikona `ccTemperatura`, ki grafično predstavlja temperaturo in pripadajoče oznake `lblTempNum` z numerično predstavitvijo tega podatka.

```
private void temperaturaSpremenila(OPCDA.NET.Controls.DAItemValue tS)
        {
            ccTemperatura.Level = double.Parse(tS.Value.ToString());
            lblTempNum.Text = tS.Value.ToString() + " °C";
        }
```

Slika 4.1.3. – metoda, ki se izvede ob spremembi vrednosti točke temp

Branje vrednosti točk in vpisovanje vrednosti v skupini OPC s pomočjo razreda *QuickUse* smo uporabili v odjemalcu OPC, ki teče na nadzorni postaji [24]. Aplikacija je zadolžena za arhiviranje procesnih podatkov; branje iz naslovnega prostora strežnika OPC in zapis v podatkovno bazo SQL.

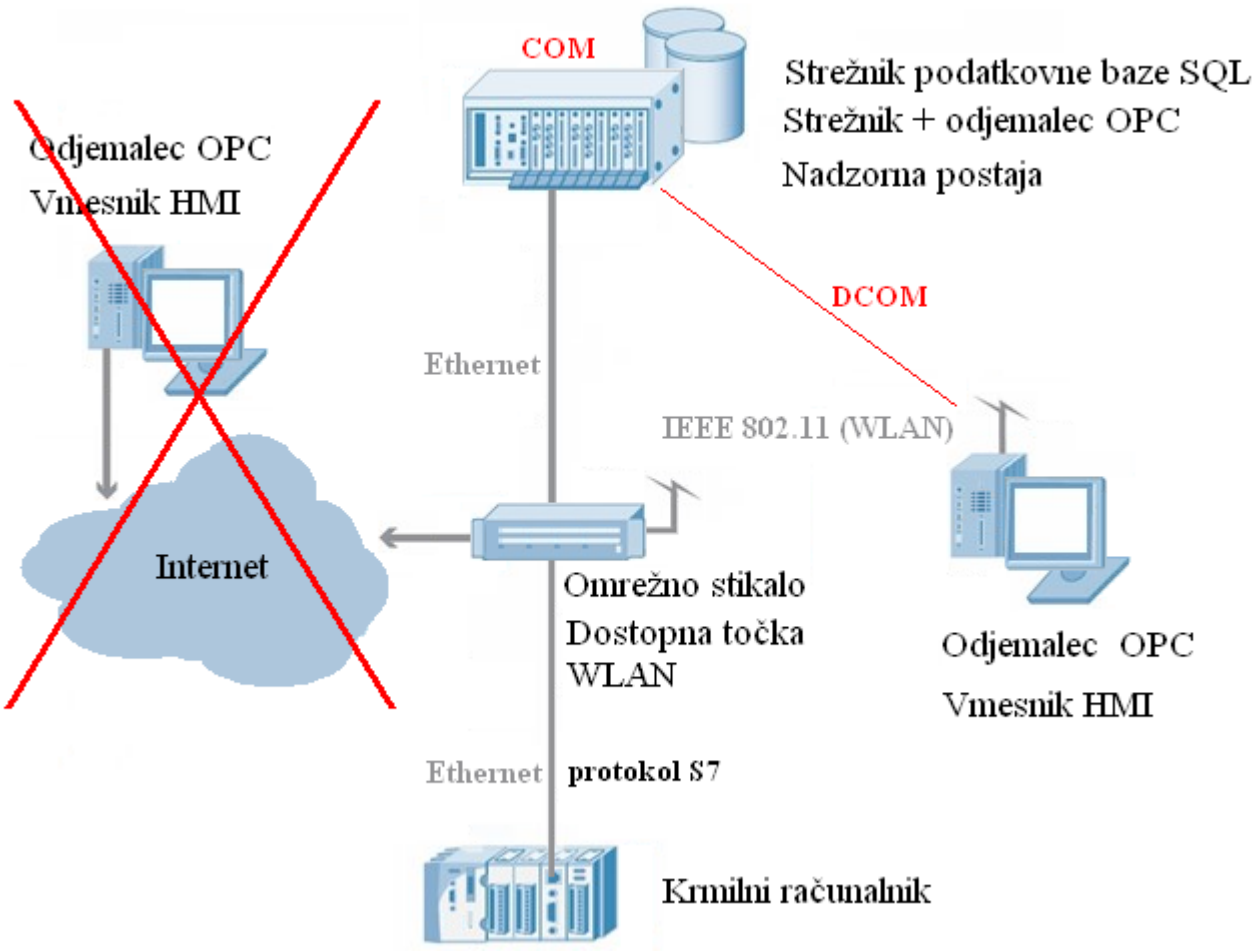
```
private void OPCRead(object sender, System.EventArgs e)
        {
            if (OpcSrv == null)
            {
                tbStatus.Text = "Ni povezave s strežnikom OPC";
                return;
            }
            // Preberemo vrednosti točk iz strežnika OPC
            // Uporablja razred OPCDA.NET QuickUse
            GetItemList();
            string txt = "";
            if (SyncRWGroup == null)      // group not yet created
                SyncRWGroup = OpcSrv.AddSyncIOGroup();
            int i = 0;
            foreach (string itemID in Items)
            {
                if (txt.Length > 0)
                    txt += "\r\n";
                OPCItemState Rslt;
                int rtc = SyncRWGroup.Read(OPCDATASOURCE.OPC_DS_CACHE, itemID,
                                           out Rslt);
                if (HRESULTS.Failed(rtc))
                {
                    tbStatus.Text = "Napaka pri branju 0x" +
                                    rtc.ToString("X"); //Heksadec. koda napake
                }
                else
                {
                    if (HRESULTS.Failed(Rslt.Error))
                        tbStatus.Text = "Napaka št. 0x" +
                        Rslt.Error.ToString("X"); //Heksadec. koda napake
                    else
                    {
                        //Pokličemo metodo za shranjevanje
                        //indeksa točke in podatka v podatkovno bazo
                        SaveToDB(i, txt);
                    }
                }
                i++;
          }
```

Slika 4.6. – branje vrednosti točk iz skupine na strežniku OPC

Metoda za zapis vrednosi točk na strežnik je podobna, le da na mesto `SyncRWGroup.Read` uporabljamo `SyncRWGroup.Write` . Tudi pri tej metodi je implementirana obravnava napak.

4.1.4. Nastavitve DCOM v operacijskem sistemu Windows

V poglavju 2.3. o klasičnem OPC DA smo omenili, da komunikacija med odjemalcem in strežnikom poteka preko objektov COM ali DCOM, ki ne podpirajo internetne komunikacije. Okrnjena arhitekturo testnega sistema SCADA je prikazana na sliki 4.1.5.



Slika 4.7. – arhitektura in protokoli testnega sistema SCADA ob uporabi vmesnika OPC DA

Na nadzorni postaji, kjer na istem računalniku tečeta tako odjemalec kot strežnik OPC DA, objekti COM ne predstavljajo večjih težav. Na edinem preostalem odjemalcu OPC DA, ki je implementiran v programu vmesnika HMI, pa obstaja logična povezava s strežnikom OPC DA na nadzorni postaji. Tu komunikacija poteka komunikacija prek objektov DCOM, kar v operacijskih sistemih Windows XP SP2 in novejših predstavlja problem. Večino težav je povezanih z varnostno politiko in administracijo uporabnikov ter uporabniških skupin. Nastavitve za uspešno omrežno komunikacijo preko DCOM okvirno obsegajo[25, 26]:

- popolno onemogočenje požarnih zidov,
- nastavljanje varnostne politike v storitvah Component Services, kjer v konfiguraciji DCOM uporabnikom in uporabniškim skupinam omogočimo dostop,
- nastavitve varnostne politike za konfiguracijo COM, podobno kot za DCOM.

Omenjene nastavitve veljajo tako za odjemalca kot za strežnik. Ni potrebno posebej poudariti, da sta takšna dva računalnika v omrežju veliko bolj ranljiva kot ostali, kjer so običajno požarni zidovi omogočeni.
Sami smo imeli pri tej konfiguraciji kar nekaj težav, tudi po prebiranju vrste priročnikov in navodil. Specifikacija za DCOM ima definiranih vrsto kod napak, ki naj bi pripomogle k

razhroščevanju. Dosegljiva dokumentacija v zvezi z njimi, še posebej tistimi, ki so tesno povezani z OPC pa je bolj skopa. Po vrsti neuspelih poskusih smo se zatekli po pomoč k spletnim forumom, kjer si razvijalci delijo mnenja. Rešitev smo našli na forumu o produktih podjetja General Electric. Izkazalo se je, da mora biti tako v odjemalčevem kot v strežnikovem operacijskem sistemu definiran uporabnik z istim uporabniškim imenom in geslom ter članstvom v isti delovni skupini (angl. workgroup). Uporabnik, ki zaganja aplikacijo z odjemalcem OPC, mora biti v sistem prijavljen natanko s tem uporabniškim imenom. S takšno konfiguracijo nam je nato končno uspelo vzpostaviti povezavo.
Iz stališča uporabe administracije uporabnikov v operacijskem sistemu nam to pomeni nazadovanje. Možnost prijave različnih uporabnikov v tak sistem, na primer različnih operaterjev SCADA v proizvodnji, nam je s tem odvzeta.

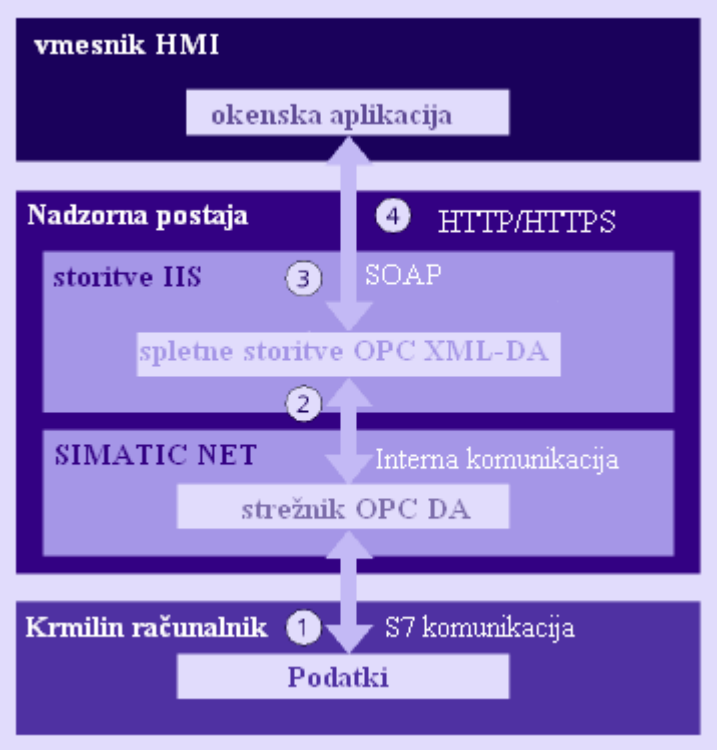
4.2. Klasični OPC DA z vgrajenim prehodom XML: OPC XML-DA

4.2.1. Prehod XML v programskem paketu Siemens SIMATIC NET CD Edition 2008

Prehod OPC XML-DA je v programskem paketu Siemens SIMATIC NET CD Edition 2008 že podprt in deluje v povezavi s spletnim strežnikom Microsoft Windows Internet Information Services (IIS). OPC XML predstavlja vmesnik za klasični OPC DA osnovan na XML (angl. eXtensible Markup Language). Jezik XML se uporablja v spletnih storitvah in ponuja format za opisovanje podatkovnih struktur ter prenos podatkov po omrežju [27].
Podatkovni vmesniki in metode so opisani z jezikom XML. Podroben opis metod določa specifikacija WSDL (angl. Web Service Definition Language), za katero poleg celotne specifikacije OPC XML-DA skrbi Fundacija OPC. Metode so opisane s protokolom SOAP (angl. Simple Object Access Protocol), ki je preprost protokol za izmenjavo informacij po protokolu HTTP (angl. HyperText Transfer Protocol)[28]. Protokol SOAP je osnovan na protokolu XML in ponuja mehanizem za enkapsulacijo dokumentov XML v t.i. ovojnice (angl. envelopes). To predstavlja končni sporočilni sistem med odjemalcem in strežnikom. Slika 4.8. predstavlja opisane odnose med protokoli.

Slika 4.8. – spletna storitev za OPC XML-DA

V poglavju 2.3.5 smo omenili, da je v vmesniku OPC XML-DA protokol DCOM zamenjal protokol SOAP, kar prinaša možnost internetne komunikacije med odjemalci in strežniki ter platformno neodvisnost. Protokol HTTP, ki protokolu SOAP nudi transportno plast, omogoča enostavno komunikacijo prek požarnih zidov, njegova varna izvedenka HTTPS pa omrežno varnost. Administracija sistema, osnovanega na spletnih storitvah je relativno preprosta. Slabost tega je, da imamo na strani strežnika vmesno aplikacijo – spletni strežnik. Z namestitvijo paketa SIMATIC NET se spletna storitev OPC XML vključi v spletni strežnik IIS, ki je del operacijskega sistema Windows. Njegova naloga je prenos podatkov med strežnikom in odjemalcem OPC [18].
Slika 4.8. predstavlja komunikacijsko strukturo naše aplikacije. Na njej je s številko 1 označena komunikacija med krmilnim računalnikom in strežnikom OPC, ki poteka po protokolu S7, s številko 2 spletna storitev OPC XML, ki je vključena v strežnik IIS in prenaša podatke med odjemalcem in strežnikom OPC, s številko 4 izmenjava podatkov med spletno storitvijo OPC XML in priključenimi odjemalci OPC, ki poteka po protokolu SOAP in s številko 4 protokola HTTP ali HTTPS predstavljata transportno plast za izmenjavo podatkov med spletnim strežnikom in povezanimi odjemalci.



Slika 4.9. – komunikacijska struktura aplikacije z OPC XML-DA

Konfiguracija strežnika OPC ostane taka, kot smo jo določili v poglavju 4.1.2. Pred razvojem odjemalca moramo konfigurirati še spletni strežnik IIS.

4.2.2. Konfiguracija spletnega strežnika IIS

Strežnik IIS (angl. Internet Information Services), integriran v operacijski sistem Windows ponuja storitve za objavo informacij v lokalnem omrežju, intranetu ali internetu. Storitve vključujejo: spletni strežnik, strežnik FTP, storitev SMTP, vrsto možnih razširitev in uporabniški vmesnik za upravljanje. Zagon in administracija spletnega strežnika, ki smo ga uporabili v našem primeru sta relativno nezapleteni opravili.
V strežniku IIS s pomočjo vmesnika za upravljanje ustvarimo navidezni imenik, ki ga vsebuje t.i. standardna spletna stran. Prek nje so dosegljivi podatki, ki jih nudi spletni strežnik. Navidezni imenik predstavlja povezavo s pravim (fizičnim) imenikom na spletnem strežniku. V našem primeru se fizični imenik nahaja v drevesni strukturi imenika, kamor smo namestili programski paket SIMATIC NET. Ustvarjenemu navideznemu imeniku lahko določimo dovoljenja za izvajanje operacij (branje, pisanje, izvajanje skritpt, brskanje po imeniku), urejamo nastavitve po meri uporabnika in določamo dostop ter avtentikacijo. Uporabljamo lahko preprosto avtentikacijo z uporabniškim imenom in geslom, uporabljamo avtentikacijo operacijskega sistema Windows ali omogočimo anonimen dostop. Možnosti v zvezi z administracijo uporabnikov so mnogo širše in udobnejše, kot v primerjavi z administracijo pri klasičnem vmesniku OPC-DA in objektih DCOM.
Z ustvarjenjem navideznega imenika v strežniku IIS so nam procesni podatki OPC-DA na voljo prek spletnega strežnika. Dostop do storitev spletnega strežnika določa naslov URL (ang. Uniform Resource Locators).

4.2.3. Razvoj odjemalca za OPC XML-DA

Metode, ki jih podpira vmesnik OPC XML v programskem paketu SIMATIC NET so enake tistim, ki smo jih opisali v teoretičnem poglavju 2.3.5. V odjemalcu OPC XML-DA smo inplementirali in testirali metode za:

- preverjanje stanja strežnika (metoda *GetStatus*),
- branje vrednosti točk (metoda *Read*),
- pisanje vrednosti točk (metoda *Write*).

Razredi in metode OPC XML, ki bi jih radi uporabili pri izdelavi odjemalca so v programu Visual Studio na voljo prek spletnega sklica (angl. web reference). Vnesti moramo naslov URL navideznega imenika OPC XML v strežniku IIS, ki se v splošnem glasi:
http://<naslov računalnika>/<navidezni imenik OPC SIMATIC NET>/SOPCWeb.asmx?wsdl
Prva metoda, ki smo jo preizkusili je bila `GetStatus` za preverjanje stanja spletnega strežnika in pridobitev specifičnih proizvajalčevih podatkov, ki niso na voljo preko ostalih metod OPC [10].

```
ServerStatus GetStatus(string LocaleID, string ClientRequestHandle, out
serverStatus);
```

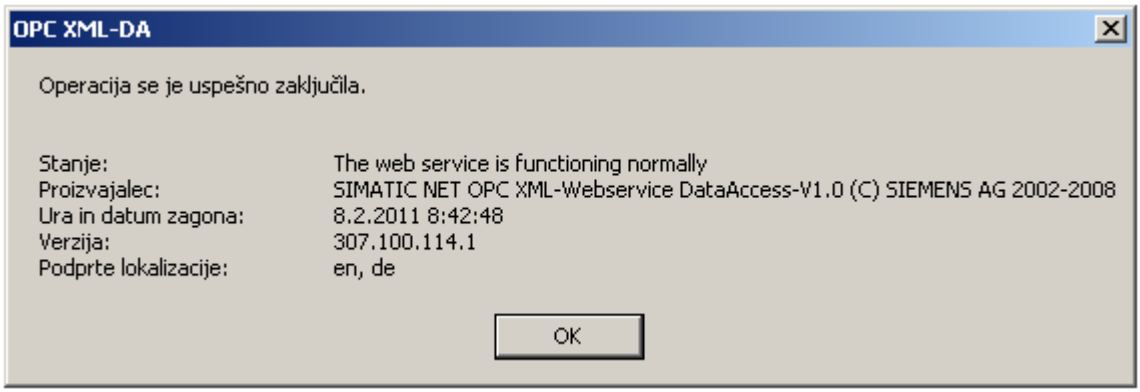
Slika 4.10. – metoda za preverjanje stanja strežnika

Argument `LocaleID` določa jezik, in lokalne nastavitve za podatke, ki jih vrača metoda [29]. Niz `ClientRequestHandle` nam pomaga pomaga pri razločevanju različnih zahtev strežnikom v kompleksnejših sistemih. Enak niz se prenese kot parameter pri odgovoru strežnika odjemalcu. Podatke o stanju strežnika dobimo preko argumenta razreda `serverStatus`. Atributi, ki nas zanimajo so podatki o stanju strežnika, proizvajalcu, času zagona, verziji in podpori lokalnim nastavitvam (lokalizaciji).

```
private void btnServerStatus_Click(object sender, System.EventArgs e)
{
   try
   {
     //Priredimo naslov URL iz tekstovnega polja
     m_OPCXML_DataAcess.Url = txtWebServerURL.Text;
     //Preverimo status strežnika
     ServerStatus Status;
     ReplyBase replay = m_OPCXML_DataAcess.GetStatus ("en", "1", out Status);
     //če naš strežnik teče
     if (replay.ServerState == serverState.running)
     {
        string strText = "Operacija se je uspešno zaključila.\n\n";
        if (Status.StatusInfo != null)
          strText+= "\nStanje:\t\t\t " + Status.StatusInfo;
        if (Status.VendorInfo != null)
          strText+= "\nProizvajalec:\t\t " + Status.VendorInfo;
        strText+=   "\nUra in datum zagona:\t " + Status.StartTime.ToString();
        if (Status.ProductVersion != null)
          strText+= "\nVerzija:\t\t\t " + Status.ProductVersion;
        if (Status.SupportedLocaleIDs != null)
        {
           strText += "\nPodprte lokalizacije:\t\t ";
           foreach (string localeID in Status.SupportedLocaleIDs)
             strText+= localeID + ", ";
           //korekcija izpisa
           strText=strText.Remove(strText.Length - 2);
        }
        //izpšemo vrednosti atributov v sporočilu
        MessageBox.Show (this, strText, this.Text, MessageBoxButtons.OK);
     }
     else
     {
       string strText = "Napaka pri pridobivanju statusa strežnika.\n\n";
       MessageBox.Show (this, strText, this.Text, MessageBoxButtons.OK);
     }
   }
   //obravnavamo vse vrste izjem (splošno)
   catch(Exception ex)
   {
      MessageBox.Show (this, ex.Message, this.Text, MessageBoxButtons.OK);
   }
}
```

Slika 4.11. – preverjanje stanja strežnika ob pritisku na gumb `btnServerStatus`

Ob uspešni izvršitvi kode 4.2.4. se nam prikaže sporočilno okno z vsemi navedenimi podatki. Gumb `btnServerStatus` je del sistemskih nastavitev aplikacij, ki tečeta na nadzorni postaji in obeh vmesnikih HMI. Operacija je uporabna predvsem v času testiranja aplikacije ter za diagnostične namene.



Slika 4.12. – sporočilno okno s podatki o stanju spletnega strežnika

Po uspešno testirani povezavi s strežnikom lahko testiramo metode za branje in pisanje. Metoda za branje vrednosti točk je naslednja:

```
ReplyBase Read(RequestOptions Options, ReadRequestItemList ItemList, out
ReplyItemList ItemValues, out OPCError[] Error);
```

Slika 4.13. – metoda `Read` za branje vrednosti točk.

Atributom objekta `Options` lahko postavimo zastavice (logične vrednosti) in časovne omejitve, ki predstavljajo opcijske podatke. Ti se pri strežnikovem odgovoru prenašajo skupaj z vrednostmi točk ali pa se ob zahtevku za branje posredujejo strežniku. Gre za metapodatke, kot so besedilo napake, diagnostični podatki, časovna značka itd. in so vključeni ob odgovoru. Strežniku lahko ob zahtevku posredujemo časovno omejitev `RequestDeadline`, ki določa maksimalen čakalni čas odjemalca na strežnikov odgovor.
Objekt `ItemList` predstavlja vsebovalnik za točke, ki jih hočemo brati. Točke so zbrane v tabeli (angl. array). Točke opišemo s simbolnimi imeni, ki so enaka tistim na krmilnemu računalniku. Simbolni dostop do procesnih spremenljivk smo omogočili v lastnostih strežnika OPC v orodju NetPro v poglavju 4.1.2. Poleg simbolnega je možen tudi dostop do absolutnih (pomnilniških) naslovov. Simbolne spremenljivke v strežniku OPC imajo naslednjo zgradbo:
<krmilniška postaja>.<krmilnik>.<podatkovni blok (opcijsko)>.<spremenljivka>. V našem primeru se procesna spremenljivka, ki v sistemu SCADA predstavlja npr. temperaturo zraka v pisarni glasi: *SIMATIC 300(1).CPU 315-2 DP.temperatureDB.temp*.
Argument `ItemValues` tipa `ReplyItemList` vsebuje tabelo vrednosti točk ob strežnikovem odgovoru. Zadnji argument `OPCError[]` podaja tabelo morebitnih napak, ki so se zgodile pri branju vrednosti točk. Na sliki 4.14. je metoda branja procesnih spremenljivk ob proženju časovnika:

```
private void tmrRead_Tick(object sender, EventArgs e)
{
   try
   {
      // Nov seznam ItemList za ReadRequest
      ReadRequestItemList ItemLists = new ReadRequestItemList();
      ItemLists.Items = new ReadRequestItem[2];
      ItemLists.Items[0] = new ReadRequestItem();
      ItemLists.Items[0].ItemPath  = ""; //ni podprto v SIMATIC NET
      ItemLists.Items[0].ItemName = "SIMATIC 300(1).CPU 315-2
DP.temperatureDB.temp";
      ItemLists.Items[1] = new ReadRequestItem();
      ItemLists.Items[1].ItemPath = "";
      ItemLists.Items[1].ItemName = "SIMATIC 300(1).CPU 315-2
DP.temperatureDB.temp2";
      //po potrebi dodajamo točke

      RequestOptions opt = new RequestOptions(); //pustimo privzete vrednosti
      ReplyItemList ItemValues;
      OPCError[] Errors;
      //klic metode Read (sinhroni)
      m_OPCXML_DataAcess.Read (opt, ItemLists, out ItemValues, out Errors);
      //lastnostim kontrol priredimo vrednosti prebranih točk
      if (ItemValues.Items[0].Value != null && ItemValues.Items[1].Value !=
null)
      {
         txtTemp1.Text = ItemValues.Items[0].Value.ToString();
         txtTemp2.Text = ItemValues.Items[1].Value.ToString();
      }
      else
      {
         txtTemp1.Text = "ni podatka";
         txtTemp2.Text = "ni podatka";
      }
      }
      //napake pri branju
       if(Errors.Length>0)
        {
          MessageBox.Show (this, Errors[0].Text, this.Text,
MessageBoxButtons.OK);
        }
      }
      //obravnavamo vse vrste izjem (splošno)
      catch(Exception excep)
      {
        MessageBox.Show (this, excep.Message, this.Text,
MessageBoxButtons.OK);
      }
   }
```

Slika 4.14. – metoda za branje procesnih spremenljivk ob proženju časovnika

Pri vmesniku OPC XML točke niso organizirane v skupine (angl. OPC groups) kot pri klasičnem OPC, ampak v sezname točk. Ti seznami vsebujejo ime spremenljivke, preko katere beremo ali pišemo vrednosti točk.

Uporaba metode `Read` na sliki 4.14. je primer sinhronega klica. Odjemalec pošlje zahtevo strežniku in čaka, dokler ne dobi odgovora. Med tem časom je izvajanje programa ustavljeno. Po prejemu strežnikovega odgovora se izvajanje programa nadaljuje. Če odjemalec ne dobi odgovora znotraj časovne omejitve se izvajanje metode konča z napako ali pa se sproži izjema. V primeru branja velikih količin podatkov so lahko izvajalni časi dolgi [18].
V našem konkretnem primeru bi bilo bolje, če bi za periodično časovno proženo branje uporabili v programu novo nit (angl. thread), ki bi skrbela samo za to operacijo. Najboljša možna rešitev pa bi bila vpeljava naročnine OPC (angl. subscription) na točke, ki jih moramo periodično osveževati. Prikaz procesnega podatka temperature v grafičnem vmesniku HMI je zagotovo tak primer.
Za pisanje vrednosti točk skrbi metoda `Write`, ki je analogna metodi za branje. Poleg vseh naštetih argumentov metode `Read` vsebuje še logično vrednost `ReturnValuesOnReply`. Argument določa, ali se po izvedbi metode vrednosti točk vrnejo nazaj odjemalcu. Prav tako kot metoda za branje, je tudi metoda za pisanje vrednosti točk sinhrnona.

4.3. Poenotena arhitektura OPC UA

4.3.1. Novosti v programskem paketu SIMATIC NET CD V8.0 (2010)

Aktualni programski paket SIMATIC NET CD V8.0 (2010) je od predhodne verzije SIMATIC NET CD Edition 2008 obdržal večino funkcionalnosti, novosti pa prinaša predvsem na področju poenotene arhitekture OPC Unified Architecture:

- razširitve varnostnih funkci in udobno delo s certifikati,
- optimizirane funkcije za prenos podatkov med odjemalcem in strežnikom UA,
- razširitev konfiguracije na vmesnik za alarme in stanja – OPC UA Alarms & Conditions,
- zastonjska kontrola za preprost, grafičen način razvoja odjemalcev OPC UA,
- omogočena povezava strežnika OPC UA s krmilnimi računalniki serije S7-1200,
- preprosto grafično orodje COML S7 za generiranje in uporabo povezavav po protokolu S7 izven programskega orodja Simatic Step7 in
- podpora samo za 32-bitni operacijski sistem Windows 7.

Konfiguracija strežnika OPC UA za povezavo s krmilnim računalnikom ostane ista, kot za arhitekturi OPC DA in OPC XML-DA. Dodatne nastavitve krmilnega računalnika v okolju Step 7 zato niso potrebne.

4.3.2. Razvoj odjemalca za arhitekturo OPC UA

Za namene testiranja smo v odjemalcu za strežnik OPC UA implementirali:

- metode za povezavo na strežnik,
- metode za branje/pisanje vrednosti atributov spremenljivk, ki predstavljajo procesne podatke,

- metode za naročnino (angl. subscription),
- brskanje spremenljivk v naslovnem prostoru strežnika.

Povezava na strežnik OPC UA je nekoliko kompleksnejši proces, kot pri ostalih opisanih arhitekturah. Strežnik OPC UA dejansko vsebuje dva strežniška procesa in sicer proces za za odkrivanje strežnikov OPC UA (angl. discovery) in dejanski podatkovni strežnik OPC UA. Ta koncept izhaja neposredno iz specifikacije, opisane v poglavju 2.4.1. Oba procesa v našem primeru tečeta na istem računalnuku in v sklopu iste strežniške aplikacije, poslušata pa na različnih vratih (ang. port). Za dostop do podatkovnega strežnika uporabimo strežnik za odkrivanje, lahko pa neposredno dostopamo do njega. V primeru, da imamo podatkovni strežnik OPC UA natančno določen lahko do njega dostopamo neposredno preko enoličnega krajevnika vira (angl. kratica URL).
Strežnik SIMATIC NET OPC UA, ki smo ga uporabili v našem testnem sistemu ima splošno obliko naslova URL *opc.tcp://<naslov računalnika>:4845*. Že sam naslov razkriva, da podatkovni strežnik uporablja protokol TCP in posluša na vratih 4845. Strežnik za odkrivanje uporablja isti protokol in vrata 4840. Poleg naslova URL moramo določiti tudi naslov URI, s katerim izberemo imenski prostor v strežniku. Absolutno naslavljanje izberemo z *S7:*, simbolno pa z *SYM*. Uporabljamo lahko tudi naslavljanje, združljivo s standardom klasičnega OPC DA. V tem primeru naslovu URI določimo kot *S7COM*. Za dostop do procesnih spremenljivk smo v našem primeru smo uporabili absolutno naslavljanje.
Do podatkovnega strežnika OPC UA smo dostopali neposredno, brez uporabe strežnika za odkrivanje. Takšen način dostopa je povsem regularen, le poznati moramo lastnosti strežnika, na katerega bi se radi priključili in podatke o njem.
Programski paket SIMATIC NET v8.0 ponuja metode za povezavo na strežnik OPC UA, ki so neposredna implementacja specifikacije komunikacijskega sklada OPC UA. Te metode so na voljo razvijalcem aplikacij za avtomatizacijo v ogrodju .NET in jeziku C#. Za uspešno vzpostavitev varne povezave do strežnika moramo določiti lastnosti objektov v komunikacijskem skladu OPC UA. To so:

- naslov URL strežnika OPC UA,
- naslov URI
- varnostne mehanizme,
- časovne omejitve (angl. timeouts),
- kodiranje.

Po uspešno vzpostavljeni povezavi komunikacijski kanal priredimo seji, kjer omogočimo še pošiljanje signala KeepAlive. Nivo seje omogoča dostop do podatkov v naslovnem prostoru strežnika. Vrsto metod, ki so potrebne za vzpostavitev seje smo večkratno ovili (angl. wrap) v eno samo metodo, ki sprejme le en parameter: naslov URL strežnika OPC UA. Ostale lastnosti objektov so fiksno (angl. hard-coded) določene, saj v jih v našem primeru kasneje ne spreminjamo. Metodi smo dodali še splošno obravnavo napak in prikaz sporočilnega okna z opisom težave v primeru neuspele povezave. Na nivoju seje se v sklopu metode za povezavo na strežnik preveri tudi, ali v naslovnem strežniku dejansko obstaja imenski prostor, ki smo ga določili v naslovu URI. Za to skrbi blok try-catch, v katerem se sprehodimo skozi tabelo imenskih prostorov strežnika in jih primerjamo s tistim, ki smo ga sami določili. Tabela imenskih prostorov v naslovnem prostoru strežnika predstavlja objekt-vozlišče, kar neposredno izhaja iz zahtev po objektni usmerjenosti za standard OPC UA. V odjemalcu za strežnik SIMATIC OPC

UA je vozlišče v splošnem določeno kot objekt razreda `NodeIdCollection`. To je generični razred, iz katerega tvorimo objekt `nodesToRead`, ki vsebuje seznam vozlišč. Tip vozlišča, ki je v tem primeru tabela imenskih prostorov podamo kot argument ob klicu metode `Add`, ki vozlišče doda v seznam. Metoda `ReadValues` sproži branje vozlišč določenih v seznamu s strežnika OPC v objekt `results`. Na sliki 4.15. je prikazana koda celotnega bloka:

```
// Branje tabele imenskih prostorov v naslovnem prostoru strežnika
try
{
      NodeIdCollection nodesToRead = new NodeIdCollection();
      DataValueCollection results;

      nodesToRead.Add(Variables.Server_NamespaceArray);

      // Branje tabele s strežnika
      m_Server.ReadValues(nodesToRead, out results);

      if ((results.Count!=1)||(results[0].Value.GetType()!=typeof(string[])))
            throw new Exception("Pri branju tabele imenskih prostorov je
prišlo do nepričakovane napake!");
      // Preverimo, ali v tabeli obstaja vneseni naslov URI
      string[] nameSpaceArray = (string[])results[0].Value;
      for (int i = 0; i < nameSpaceArray.Length; i++)
      {
            if (nameSpaceArray[i] == txtNamespaceUri.Text)
                  m_NameSpaceIndex = i; // m_NameSpaceIndex je deklarirana kot
spremenljivka na nivoju razreda

      }
      // Preverimo, ali smo našli definiran URI
      if ( m_NameSpaceIndex == 0 )                {
            throw new Exception("Imenski prostor " + txtNamespaceUri.Text + "
ne obstaja v naslovnem prostoru strežnika!");

}
// Splošna obravnava napak
catch (Exception exc)
{
      MessageBox.Show("Branje tabele imenskih prostorov s strežnika je
spodletelo!" + exc.Message);
}
```

Slika 4.15. – blok try-catch za ugotavljanje obstoja vnešenega naslova URI med imenskimi prostori strežnika OPC UA

Uspešno branje tabele imenskih prostorov s strežnika in ujemanje z naslovom URI, določenim s strani uporabnika oziroma razvijalca, omogoča izvajanje metod za delo s podatki v njem.
Branje in pisanje vrednosti atributov spremenljivk v naslovnem prostoru strežnika ravno tako predstavlja branje in pisanje vrednosti vozlišč. S stališča programske arhitekture strežnika in odjemalca OPC se npr. tabela imenskih prostorov strežnika in spremenljivka, ki predstavlja procesni podatek, ne razlikujeta ravno dosti. Za dodajanje spremenljivk v seznam vozlišč `nodesToRead` uporabimo zopet metodo `Add`, ki sprejme argument objekt razreda `NodeId`. Poleg indeksa imenskega prostora `m_NameSpaceIndex` konstruktor razreda `NodeId` sprejme še indetifikator vozlišča. Na ta način je vozlišče enoločno določeno. Splošna oblika niza

indetifikatorja za imenski prostor *S7*, ki predvideva absolutno naslavljanje spremenljivk, sestoji iz: *<ime povezave S7>.<pomnilniško področje>.<odmik>,<podatkovni tip>*. Ime povezave S7 določimo v konfiguraciji omrežnih povezav med strežnikom OPC in krmilnim računalnikom v okolju NetPro.
Primer: V okolju NetPro smo povezavi med krmilnim računalnikom in strežnikom OPC dodelili ime *povezava001*. Indetifikator za branje prve besede (16-bitno število, angl. word) iz pomnilniškega prostora zastavic (M) v krmilnem računalniku se tako glasi: *povezava001.m.0,w*. Slika 4.16. prikazuje del kode, ki izvede branje vrednosti spremenljivke. Indetifikator le-te v tem primeru vnesemo v tekstovno polje `txtIdentifier`.

```
NodeIdCollection nodesToRead = new NodeIdCollection();
DataValueCollection results;
// Enoločno določimo vozlišče in ga dodamo v seznam
nodesToRead.Add(new NodeId(txtIdentifier1.Text, m_NameSpaceIndex));
/* Preberemo vrednosti s strežnika, prebrane vrednosti se shranijo v argument
results */
m_Server.ReadValues(nodesToRead, out results);
```

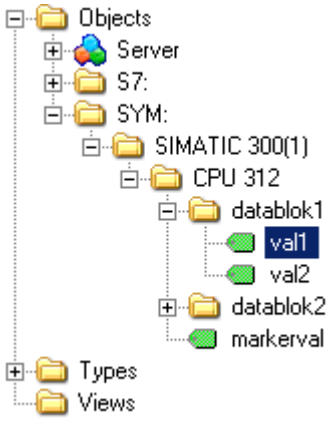
Slika 4.16. – branje atributa vrednosti spremenljivke s strežnika OPC UA

Izhodni argument funkcije za branje `results` je objekt razreda `DataValueCollection`. S pomočjo metod, ki jih ta razred ponuja, lahko iz strežnikovega odgovora izluščimo vrednost spremenljivke in različne metapodatke. Nas je poleg vrednosti spremenljivke najbolj zanimala kvaliteta prebranega podatka, saj so bili le tisti, označeni s kvaliteto dobro (angl. good), merodajni in uporabni v sistemu SCADA.
Metoda za pisanje vrednosti spremenljivk deluje na podoben način kot metoda za branje. Namesto metode `ReadValues` uporablja `WriteValues`. Metoda sprejme dva argumenta; seznam vozlišč, katerih vrednosti bomo vpisovali, in izhodni argument. Pri metodi za branje je izhodni argument razreda `StatusCodeCollection`, ki ponuja metode za ugotavljanje uspešnosti vpisa vrednosti vozlišč na strežnik OPC UA.
Metode za naročnino smo se poslužili v primeru spremljanja vrednosti procesnega podatka zunanje in notranje temperature. Ti se lahko v času relativno hitro spreminjata, zato je konstantno spremljanje tovrstnih procesnih podatkov idealna rešitev. Strežnikov objekt pozna metode za dodajanje in odstranjevanje naročnine na določeno spremenljivko in imajo naslednje argumente:

- seznam vozlišč,
- metodo, ki se pokliče ob spremembi vrednosti opazovane spremenljivke in
- frekvenco spremljanja sprememb vrednosti opazovane spremenljivke.

Metoda, ki se pokliče ob spremembi vrednosti spremenljivke poskrbi za osveževanje procesnega podatka v vmesniku HMI in zapis vrednosti v podatkovno bazo.
Zadnja funkcionalnost, ki smo jo testirali, je brskanje vozlišč po naslovnem prostoru strežnika. Brskalnik omogoča uporabniku prijazno brskanje po vseh naslovnih prostorih in vseh vozliščih, ki so na voljo v naslovnem prostoru strežnika OPC UA. Za izbrano vozlišče so na voljo podatki o njegovih atributih. Brskanje vozlišč je predvsem uporabno v času razvoja, kot orodje za pomoč, orientiranje in testiranje, nekoliko manj pa v sami aplikaciji v avtomatiziranem sistemu. Tako ga v končno verzijo testne aplikacije nismo vključili kot uporabno funkcijo. V našo testno aplikacijo smo v času razvoja vključili brskalnik, ki je dostopen v testnih primerkih aplikacije odjemalca za

strežnik OPC UA v programskem paketu SIMATIC NET CD V8.0. [31]. Brskalnik podprira brskanje z uporabo miškinega klika in grafični prikaz. Za to uporablja razred `TreeView`, ki prikaže hierarhično zbirko objektov. Metoda za izgradnjo drevesnega pogleda se sprehodi skozi celoten naslovni prostor strežnika OPC UA in postopoma gradi strukturo. Rezultat je prikazan na sliki 4.17.



Slika 4.17. – drevesni pogled objektov v strežniku OPC UA

Na sliki 4.17. smo na primer v imenskem prostoru *SYM* izbrskali spremenljivko s prikaznim imenom *val1*, ki se v nahaja v projektu *SIMATIC 300(1)*, natančneje v krmilnem računalniku z imenom *CPU 312*. Vsebovana je v v podatkovnem bloku *datablok1*. Ob kliku na vozlišče se nam prikažejo njegovi atributi in vrednosti le-teh:

Attributes	Value
NodeId	ns=4;s=SIMATIC 300(1).CPU 312.datablok1.val1
NodeClass	Variable
BrowseName	4:val1
DisplayName	val1
Description	BadAttributeIdInvalid
WriteMask	None
UserWriteMask	None
Value	0
DataType	Int16
ValueRank	-1
ArrayDimensions	
AccessLevel	Readable, Writeable
UserAccessLevel	Readable, Writeable
MinimumSamplingInterval	BadAttributeIdInvalid

Slika 4.18. – atributi vozlišča na strežniku OPC UA

4.3.3. Testiranje delovanja aplikacije z odjemalcem OPC UA

Med testiranjem aplikacije OPC UA nas je najbolj zanimalo delovanje arhitekture odjemalec-strežnik, kjer je odjemalec prek interneta povezan s strežnikom. Za to je več razlogov; internetna komunikacija je bila ena od zahtev naši testni aplikaciji, zmožnost delovanja prek interneta pa se poudarja kot ena pomembnih funkcij arhitetekture OPC UA in prinaša velik napredek v primerjavi s klasičnim OPC.
Konfiguriranje odjemalca in strežnika se je res izkazalo za preprosto opravilo, odjemalec mora poznati le naslov IP in vrata, na katerih posluša strežnik. Na poti od odjemalca do strežnika moramo ta ista vrata odpreti tudi v vseh požarnih zidovih, preko katerih poteka promet. Nekaj dodatne konfiguracije na strani strežnika je zahtevalo dejstvo, da je naše testno lokalno omrežje ločeno od zunanjega sveta z omrežnim usmerjevalnikom (angl. router) s funkcijo prevajanja omrežnega naslova NAT (ang. network address translation). Odjemalec SIMATIC OPC UA se sam po sebi funkcije NAT ne zaveda, to moramo ročno določiti v strežnikovi konfiguraciji. Strežnikova konfiguracija je shranjene v datoteki. Posamezne nastavitve so določene v obliki oznak XML (angl. XML tags). Za odjemalčevo uspešno povezovanje na strežnik, skrit za funkcijo NAT, smo morali v nastavitve dodati še eno oznako XML. V njej določimo napravo, ki je za odjemalca OPC UA vidna od zunaj. V našem konkretnem primeru je to omrežni usmerjevalnik s svojim zunanjim naslovom WAN (angl. wide area network) . Osnovna konfiguracija strežnika OPC UA, ki se določi ob njegovi namestitvi, vsebuje namreč le njegov lokalen naslov. Popravljena konfuguracija nastavitve je prikazana na sliki 4.19.

```
<UaEndpoint>
      <Url> opc.tcp://95.176.141.246:4845</Url>
      <StackUrl>opc.tcp://192.168.1.100:4845</StackUrl>
</UaEndpoint>
```

Slika 4.19. – popravljena konfuguracija strežnika OPC UA za komunikacijo preko funkcije NAT

V oznaki `<Url>` sta določena določen naslov in vrata, ki sta vidna iz omrežja WAN, v oznaki `<StackUrl>` pa naslov in vrata strežnika OPC UA v lokalnem omrežju. V primeru, da bi na poti med odjemalcem in strežnikom OPC UA imeli več prevedb omrežnega naslova oziroma več naprav, ki izvajajo funkcijo NAT, bi to enostavno dodali v konfiguracijo z dodatnimi oznakami `<StackUrl>` . Ta scenarij je v realnosti zelo možen, saj so omrežja na nivoju korporacij večkrat ločena na podomrežja in zaščitena s požarnimi zidovi. V poglavju 2.4. je primer takega omrežja prikazan na sliki 2.4.
Naša internetna aplikacija z arhitekturo odjemalec-strežnik OPC UA deluje pri uporabi širokopasovnega dostopa do interneta praktično enako, kot tista v lokalnem omrežju. Prenos podatkov je s stališča uporabnika enako hiter, latence pa nezaznavne.

5. Zaključek

V diplomski nalogi smo v lasten nadzorni sistem SCADA vpeljali tri različne standarde arhitekture odjemalcev-strežnikov OPC: klasični OPC DA, klasični OPC DA z vgrajenim prehodom XML OPC XML-DA in novo, poenoteno arhitekturo OPC UA. Testirali smo njihove funkcije in skušali predstaviti njihove prednosti in slabosti. Na podlagi testiranj bi lahko rekli, da ima nova arhitekura OPC UA dobre možnosti, da nasledi in sčasoma zamenja klasično. Uvajanje OPC UA ne pomeni ukinitev obstoječih investicij v OPC, saj sta arhitekturi med seboj kompatibilni in lahko sobivata v istem avtomatiziranem sistemu. Omrežne in internetne povezave ter varnostne funkcije so dobro podprte, kar je dandanes za tovrstne sisteme že samoumevno nujno. Internetno komunikacijo v standardu OPC UA smo dodobra stestirali tudi v okviru tega diplomskega dela. Ugotovili smo, da predstavlja velik napredek v primerjavi s klasično arhitekturo, kjer komunikacija poteka preko objektov DCOM. Omrežna komunikacija je možna samo ob določenih nehotenih omejitvah.

S stališča aplikativnega inženirja, ki se posluži standarda OPC za razvoj sodobnih aplikacij v avtomatizaciji, objektna usmerjenost OPC UA kar sama kliče po uporabi objektno usmerjenih programskih jezikov in platform za njihov razvoj. Tudi sami smo se poslužili jezika C#, in ogrodja .NET ter bili pri tem zadovoljni in uspešni. Foundacija OPC, ki v splošnem skrbi za celoten standard OPC, pa ponuja razvojne komplete za ustvarjanje aplikacij osnovanih na OPC UA tudi za platformo Java in programska jezika C in C++.

Platformna neodvisnost, ki je tudi ena od zahtev v standardu OPC UA, v našem delu ni prišla do izraza, saj smo v vseh testnih primerih uporabili strežnik proizvajalca Siemens, ki teče samo na operacijskih sistemih Windows. Ogrodje .NET, v katerem smo v okviru razvoja sistema SCADA implementirali odjemalca OPC, pa je prvinsko ravno tako podprto le v operacijskem sistemu Windows. Sam standard OPC UA predvideva popolno neodvisnost od določene platforme. Na trgu produktov OPC najdemo produkte, ki temeljijo na primer na operacijskem sistemu Linux in vgradnih sistemih. Slednji so nas tekom razvoja tudi zelo zanimali in predstavljajo nadaljevanje dela na tem področju. Na primer krmilni računalnik na osnovi vgradnega sistema, na katerem poleg krmilnega programa teče še strežnik OPC UA je v industrijski avtomatizaciji dobrodošla novost. V avtomatiziran sistem vpeljuje dodatne poenostavitve strojne in programske opreme, ki bi po našem mnenju dvignile nivo robustnosti takega sistema sistema. To je pa v neprijaznih industrijskih okoljih še kako pomemben faktor.

Vse arhitekure OPC, ki smo jih testirali, so vključevale samo osnovni vmesnik za dostop do procesnih podatkov. Poleg tega arhitektura OPC podpira še vmesnike za dostop do arhivskih podatkov, vmesnike za alarme in dogodke ter ostale izpeljanke osnovnega. Vpeljava le-teh predstavlja določeno količino dela, ki bi moralo biti opravljeno pri vzpostavitvi kakšnega bolj kompleksnega sistema, kot je bil naš testni sistem SCADA, v osnovi namenjen nadzoru ogrevanja v industrijski stavbi.

Klasični vmesnik OPC DA je bil prvotno zamišljen kot gonilnik med krmilnimi računalniki na nižjem nivoju in napravami na višjih nivojih vodenja. V našem primeru smo ga bolj kot ne uporabili zgolj za to. Aktualni standard OPC UA pa predstavlja vmesnik med različnimi sistemi vodenja, v mislih imamo predvsem sisteme za upravljanje proizvodnje MES in sisteme za planiranje proizvodnje ERP. Nadaljevanje dela na tem področju iz te diplomske naloge nedvomno sledi in od nas zahteva odstiranje novih področij.

6. Viri in literatura

[1] Wolfgang Mahnke, Stefan-Helmut Leitner, Matthias Damm, *OPC Unified Architecture*, Berlin, Heidelberg: Springer-Verlag, 2009, poglavje 1.

[2] (2011) OPC foundation, Member list. Dostopno na: http://www.opcfoundation.org/About/MemberList.aspx.

[3] (2011) Senzorji in aktuatorji. Dostopno na: http://colos1.fri.uni-lj.si/ERI/RAC_SISTEMI_OMREZJA/html/RSO-OKOLJE/Senzorji_aktuatorji.html.

[4] (2011) Globally unique identifier, from Wikipedia, the free encyclopedia. Dostopno na: http://en.wikipedia.org/wiki/Globally_unique_identifier.

[5] (2011) OPC, iz Wikipedije, proste enciklopedije. Dostopno na: http://sl.wikipedia.org/wiki/OPC.

[6] (2008) Siemens SIMATIC S7-200 Programmable Controller System Manual. Dostopno na: http://support.automation.siemens.com/WW/llisapi.dll/csfetch/1109582/s7200_system_manual_en-US.pdf.

[7] Valentin Bogdan, *Industrijski ethernet*, Maribor: Univerza v Mariboru, 2003, poglavje 2.

[8] (2010) Siemens SIMATIC Controller Software, Tools for configuring and programming SIMATIC Controllers. Dostopno na: http://www.automation.siemens.com/salesmaterial-as/brochure/en/brochure_simatic-industrial-software_en.pdf.

[9] (2011)Siemens Simatic Controllers Brochure. Dostpno na: http://www.automation.siemens.com/salesmaterial-as/brochure/en/brochure_simatic-controller_en.pdf.

[10] (2010) Siemens, SIMATIC NET Industrial Communication with PG/PC Volume 2 – Interfaces. Dostopno na: http://support.automation.siemens.com/WW/llisapi.dll/csfetch/42783660/PGH_opc2_76.pdf.

[11] (2011) About Hirschmann. Dostopno na: http://www.beldensolutions.com/en/Brands-Products/Hirschmann_Produkte/About_Hirschmann/index.phtml.

[12] (2011) Siemens S7-SCL. Dostopno na: http://www.sea.siemens.com/us/Products/Automation/Engineering-Software/s7-scl/Pages/s7-scl.aspx.

[13] (2011) SCADA systems. Dostopno na: http://www.scadasystems.net/.

[14] (2011) Microsoft Visual Studio 2010. Dostopno na: http://www.microsoft.com/visualstudio/en-us .

[15] (2011) Microsoft .NET Framework: Overview. Dostopno na: http://www.microsoft.com/net/overview.aspx.

[16] (2011) Integrated development environment, From Wikipedia, the free encyclopedia. Dostopno na: http://en.wikipedia.org/wiki/Integrated_development_environment.

[17] (2011) What is Mono. Dostopno na: http://mono-project.com/What_is_Mono.

[18] (2005) Siemens, Using the XML-DA interface of the SIMATIC NET OPC server with Visual Basic .NET. Dostopno na: http://support.automation.siemens.com/WW/adsearch/resultset.aspx?region=WW&lang=en&netmode=internet&ui=NDAwMDAxNwAA&term=opc+xml+da+programming&ID=21402169&ehbid=21402169.

[19] (2008) Siemens SIMATIC HMI, WinCC V7.0 System Description. Dostopno na: http://support.automation.siemens.com/WW/llisapi.dll/csfetch/32107026/SIMATIC_WinCC_System_Description_en-US.pdf.

[20] (2011) Advosol, OPC DA .Net Client Development Component. Dostopno na: http://www.advosol.com/pc-1-3-opcdanet.aspx.

[21] (2008) Siemens, Readme file for the "SIMATIC NET PC Software CD, Edition 2008" with important information on the products. Dostopno na: http://cache.automation.siemens.com/dnl/TI/TIzNzE2NzcA_31675909_Akt/Readme.htm.

[22] (2002) Siemens, Configuring PC Stations with SIMATIC NCM PC, Manual and Quick Start. Dostopno na: http://support.automation.siemens.com/WW/llisapi.dll/csfetch/13654633/mn_ncm-pc_76.pdf.

[23] (2001) Siemens, SIMATIC NET, S7 Programming Interface . Dostopno na: http://support.automation.siemens.com/WW/llisapi.dll/csfetch/13649203/mn_s7api_e.pdf.

[24] (2011) Advosol, OPCDA.NET Reference Manual. Dostopno na: http://advosol.us/manuals/opcdanet/webframe.html.

[25] (2011) Software Toolbox, Recommended steps for XP SP2 DCOM and Firewall Setup. Dostopno na: http://www.softwaretoolbox.com/xpsp2/html/xpsp2_dcomsetup.html.

[26] (2011) Software Toolbox, OPC Common Questions, COM/DCOM Error Codes. Dostopno na: http://support.softwaretoolbox.com/ci/fattach/get/21815/1249418484/redirect/1/filename/OPC%20Common%20Questions%20-%20COM%20ErrorCodes.pdf.

[27] (2011) XML, Iz Wikipedije, proste enciklopedije. Dostopno na: http://sl.wikipedia.org/wiki/XML.

[28] (2011) SOAP tutorial. Dostopno na: http://www.w3schools.com/soap/default.asp.

[29] (2011) Microsoft, Locale ID (LCID) Chart . Dostopno na: http://msdn.microsoft.com/en-us/library/0h88fahh%28v=vs.85%29.aspx.

[30] (2011) AD, avtomatizacija industrijskih procesov, Poskusite v živo! Dostopno na: http://adavtomatizacija.wordpress.com/poskusite-v-zivo/.

[31] (2010) Siemens, Programming an OPC UA .NET Client with C# for the SIMATIC NET OPC UA Server. Dostopno na: http://support.automation.siemens.com/WW/adsearch/resultset.aspx?region=WW&lang=en&netmode=internet&ui=NDAwMDAxNwAA&term=opc+ua+client+programming&ID=42014088&ehbid=42014088.