Programski okvir za konfiguriranje, prikupljanje i obradu podataka iz industrijskih postrojenja

Rojnić, Domagoj

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA~~R~~ I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

PROGRAMSKI OKVIR ZA KONFIGURIRANJE, PRIKUPLJANJE I OBRADU PODATAKA IZ INDUSTRIJSKIH POSTROJENJA

Diplomski rad

Domagoj Rojnić

Osijek, 2022.

Style Definition: Heading 2
Style Definition: Heading 3

SADRŽAJ

1. UVOD

Već dugi niz godina tehnologija se eksponencijalno razvija i ulazi u svakodnevnicu ljudi. Programska podrška (engl. *software*) postala je ključni dio moderne automatizacije i kontrole sustava različitih postrojenja. Sustavi upravljanja posebne namjene zamijenjeni su onima s više univerzalnosti. Računalne stanice u industriji široko se koriste za prikupljanje različitih podataka, kontrolu procesa kao i za druge ciljeve automatizacije procesa postrojenja. Elementi upravljačkog sustava prelaze s centralizirane u raspodijeljenu arhitekturu pri čemu se analogni i mehanički ~~kontroleri~~ upravljači mijenjaju digitalnima. Posljednjih nekoliko godina broj senzorskih sustava raste vrlo brzo i time se povećava važnost prikupljanja i isporuke važnih informacija u svrhu nadzora i kontrole procesa. Ključni zadatak nadzornog tijela postrojenja je omogućiti jednostavnu integraciju informacija iz upravljačkog sustava te olakšati pristup podacima iz pogona i integrirati ih u poslovne sustave.

Godišnjim testiranjima ustanovilo se kako je vrlo teško, pa čak i nemoguće, dijeliti podatke između različitih uređaja i programa različitih dobavljača. Potrebno je formirati otvorenu i učinkovitu komunikacijsku arhitekturu gdje se usredotočuje na pristup podacima, a ne na vrste podataka. Kao jedno od rješenja, kreiran je OPC (*OLE for Process Control*) protokol.

Cilj ovoga rada je izraditi programsko rješenje kojim se pristupa podacima postrojenja, daje uvid u njihove vrijednosti radi analize te omogućuje spremanje podataka u bazu kao i njihovu ~~statičk~~statističku analizu. Programsko rješenje temelji se na OPC protokolu kao osnovi za pristup i upravljanje podacima različitih ~~kontrolera~~ upravljača odabranog industrijskog postrojenja kojemu se čitaju i spremaju podatci. Izrađeno je pregledno i jednostavno korisničko sučelje sa svim funkcijama koje bi operater postrojenja mogao zatrebati.

Rad je organiziran tako da su, nakon uvoda, opisana već postojeća rješenja i tehnologije za prikupljanje podataka. Trećim poglavljem definirani su zahtjevi na programsko rješenje, tehnologije korištene pri izradu te arhitekturni dizajn aplikacije. Četvrtim poglavljem detaljnije je objašnjeno konkretno programsko rješenje i njegova zadaća dok je petim poglavljem dan zaključak čime se objedinjuje rad u jednu cjelinu.

Commented [ZK1]: termini na stranom jeziku trebaju biti u kurzivu kada ih na ovaj način objašnjavate.

Formatted: Font: Italic

Formatted: Font: Italic

Commented [D2]: Dodano (stat. Analiza ukljucena)

Commented [ZK3]: ovdje dodati detalje za kakvo postrojenje je napraveljno

Commented [D4R3]: U tekstu sam onda sve reference na to "postrojenje" stavio u stilu industrijsko postrojenje kojemu se izvlace podatci

2. PREGLED POSTOJEĆIH RJEŠENJA

Četvrta industrijska revolucija [1, 2] omogućila je sveobuhvatan i međusobno povezan pristup proizvodnji. Njome se spajaju fizičke komponente s digitalnima omogućujući razvoj sustava za prikupljanje podataka kao i bolju suradnju i pristup različitih odjela postrojenja. Uključuje kompleksno dizajniranje specijaliziranih tehnologija kojima se tvrtke moraju prilagođavati kako bi zadržale postojeću poziciju poduzeća.

Commented [ZK5]: reference numerirati onim redoslijedom kako se pojavljuju u tekstu, ovdje bi trebalo biti [1, 2].

Prikupljanje podataka i razvoj različitih rješenja u proizvodnom sustavu kao i njegovom planiranju zahtijeva do dvije trećine ukupnih potreba za vremenskim resursima [3]. Digitalizacija proizvodnih sustava nudi mogućnost automatizacije procesa prikupljanja podataka. Ipak, mala i srednja poduzeća nemaju razvijene automatizirane sustave, kako zbog heterogenih baza podataka tako i zbog nerazvijenih sustava za obradu podataka.

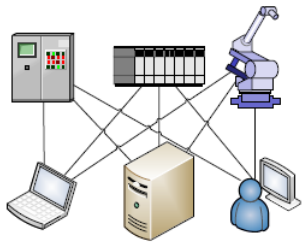
Kako bi se prethodno navedeni problemi nadišli te kako bi se omogućila implementacija jednostavnog sustava za prikupljanje podataka neovisnog o fizičkim komponentama i njihovoj kompatibilnosti, razvijen je OPC protokol. Revolucionarno uvođenje protokola u sustave proizvodnje omogućilo je izradu različitih rješenja pristupa podacima postrojenja.

Commented [ZK6]: Kreiranje -> izrada (search/replace)

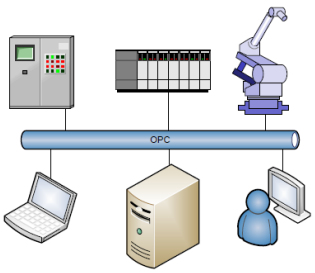
2.1. Povijest prije OPC-a

Prije pojavljivanja OPC protokola [4], programeri su morali razvijati specifične komunikacijske upravljačke programe (engl. *drivers*) za svaki upravljački sustav za kojeg je bilo potrebno imati sučelje. Nije postojao jedinstveni standard pa su različiti dobavljači razvijali vlastitu programsku podršku koja isključivo odgovara njihovim tehnologijama. Dobavljači sučelja čovjek-stroj (engl. *Human-Machine Interface, HMI*) trošili su mnogo vremena i novca na razvoj stotine različitih upravljačkih programa niske razine (engl. *low-level drivers, LLD*) za različite raspodijeljene kontrole sustave (engl. *Distrbuted Control Systems, DCS*) i programabilne logičke upravljače (engl. *Progammable Logic Controllers, PLC*). Kvarovi upravljačkih programa bili su uzrokovani minimalnim promjenama u mogućnostima sklopovlja. S vremenom, dobavljačima je ponestalo vremena i resursa za razvoj tako velikog spektra programske podrške te se pojavila potreba za standardiziranim sučeljem. Tok informacija i međusobna povezanost različitih uređaja prethodno opisanog doba prikazani su slikom 2.1. dok je slikom 2.2. dan prikaz međusobne povezanosti nakon uvođenja OPC standardiziranog sučelja.

Commented [ZK7]: upravljački (s/r)



Slika 2.1. Tok informacija prije pojavljivanja OPC-a
Izvor: [4, str.19.]



Slika 2.2. Tok informacija nakon pojavljivanja OPC-a
Izvor: [4, str.19.]

2.2. Postojeća rješenja

U nastavku je prikazana nekolicina postojećih rješenja koja se bave pristupom, prikupljanjem i pohranom vrijednosti OPC signala. Proučavanjem i korištenjem tih rješenja formiraju se ciljevi rada kao i osnovne funkcionalnosti vlastitog rješenja, čiji će zahtjevi biti detaljno prikazani sljedećim poglavljem.

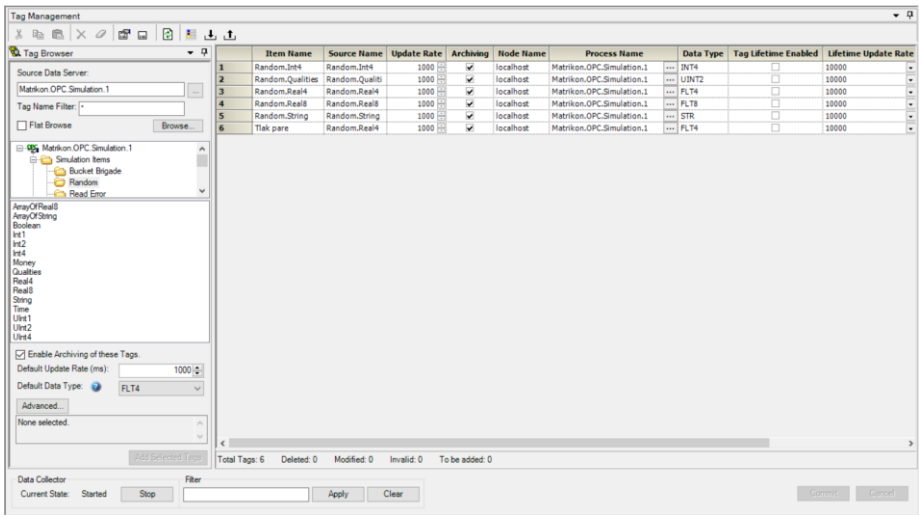
Commented [ZK8]: Kakvih? Ovdje budite konkretni.

2.2.1. Matrikon OPC Desktop Historian

Matrikon OPC Desktop Historian [5] predstavlja pristupačno programsko rješenje za pohranu podataka od vremenske važnosti. Može se koristiti zasebno ili kao dio cjelokupnog sustava prikupljanja poslovnih podataka. Program omogućuje prikupljanje podataka s OPC poslužitelja zadanog postrojenja, omogućujući brz i siguran pristup pohranjenim podacima za analizu i informirano donošenje poslovnih odluka.

Glavna prednost obuhvaća prikupljanje različitih vrsta podataka od različitih proizvođača uz mogućnost višestrukih instanci, balansiranja opterećenja kao i pristup i prijenos povijesnih podataka. Može se koristiti samostalno ili kao dio velikih, distribuiranih arhitektura. Instalacija, konfiguracija i implementacija su jednostavne uz nisko održavanje i visoku razinu podrške. Najveća mana ovoga programa leži u njegovoj cijeni čime se manjim poduzećima stvara barijera prilikom implementacije sustava za prikupljanje podataka.

Slikom 2.3. prikazan je dio podatkovnog pristupa i kreiranja konfiguracije Matrikon Desktop Historiana koji ujedno i predstavlja glavnu funkcionalnost ovog programa temeljem kojih će se formirati zahtjevi vlastitog rješenja. Konkretnije, koristit će se isti način prikazivanja strukture poslužitelja pomoću kretanja kroz stablastu strukturu, prikaz dostupnih OPC stavki unutar zasebnog prozora kao i njihovih karakteristika u obliku tablice.



Slika 2.3. Izrada konfiguracije pomoću Matrikon Dekstop Historiana
Izvor: [5]

2.2.2. MATLAB industrijski komunikacijski alat

OPC protokol može biti dostupan i u obliku biblioteke [4, 6], čijim se uključivanjem u razvojno okruženje stvara mogućnost pristupa i prikupljanjem podacima direktno iz programskog koda. Industrijski komunikacijski alat MATLAB-a, prijašnje poznat kao MATLAB OPC Toolbox, omogućuje programerima razvoj programske podrške za pristup trenutnim i povijesnim podacima

Commented [D9]: Je li ovo dovoljno kao paralela na moje rješenje?

Commented [ZK10R9]: Navedite još detalja, primjerice što konkretno od podatkovnog pristupa ste Vi primijenili kod sebe.

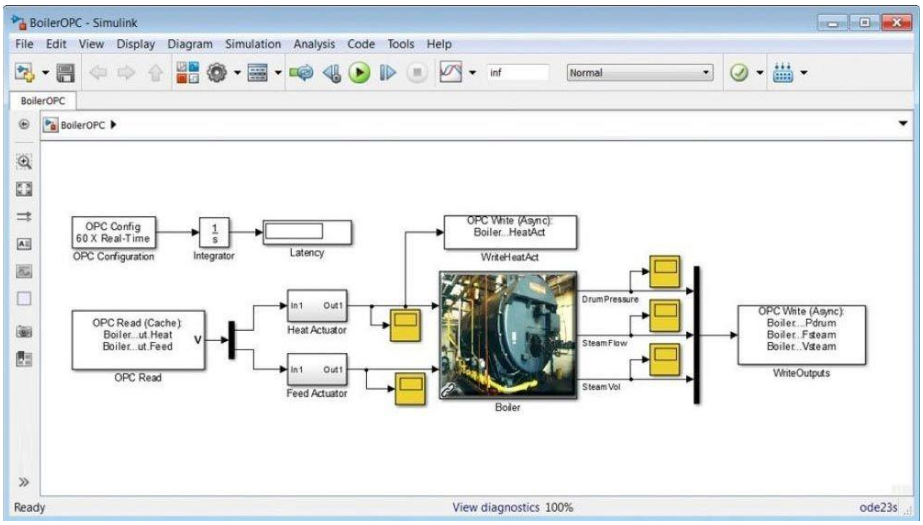
Commented [D11]: Dodano, konkretniji opis

Commented [ZK12]: Ne smijete ići više od 3 razine numeriranja prema uputama za izradu diplomskog rada. Također, slike neka budu numerirane po poglavlju (primjerice, prva slika u poglavlju 2 je 2.1., druga, iako je u potpoglavlju 2.3.3. će biti 2.2. itd.).

Commented [ZK13]: Biti dostupan

industrijskih postrojenja direktno iz MATLAB i Simulink programskih okruženja. Bibliotekom se čitaju i pišu podaci OPC ujedinjene arhitekture (engl. *OPC Unified Architecture, OPC UA*), servisno orijentirane arhitekture neovisne o platformi koja omogućuje integraciju i komunikaciju različitih uređaja, kao što su raspodijeljeni upravljački sustavi, nadzorni sustavi i programabilni logički upravljači, neovisno o korištenim tehnologijama i protokolima. Koristi se u svrhu praćenja, kontrole i poboljšanja poslovnih procesa pristupajući podacima kako s OPC poslužitelja tako i s OSIsoft PI poslužitelja. Komunikacija s rubnim uređajima i poslužiteljima na oblaku ostvaruje se preko Modbus i MQTT (*MQ Telemetry Transport*) protokola. Alat je karakteriziran raznim sigurnosnim protokolima, algoritmima šifriranja pa tako i metodama provjere autentičnosti korisnika.

Biblioteka uključuje Simulink blokove kojima se modeliraju nadzorne kontrole na mreži i testiraju upravljači sklopovlja u petlji (engl. *hardware-in-the-loop testing*) što je prikazano slikom 2.4. Također, omogućena je potvrda algoritama uspostavljanjem sigurne OPC UA veze s postrojenjem te izrada modela za aplikacije interneta stvari (engl. *Internet of Things, IoT*).



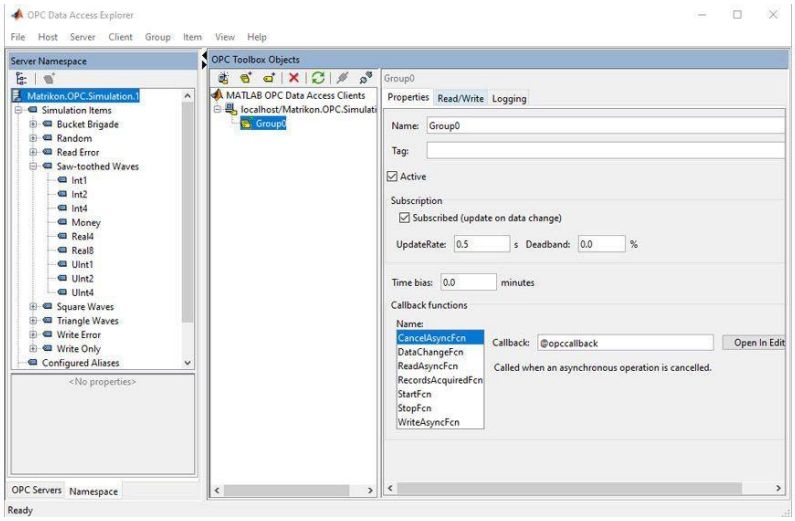
Slika 2.4. Model sklopovlja korištenjem Simulink blokova
Izvor: [6]

Slikom 2.5. prikazana je testna aplikacija razvijena od strane MATLAB-a za pristup i prikupljanje podataka postrojenja koristeći biblioteku za pristup OPC poslužitelju.

Commented [ZK14]: Ovo bi trebalo pojasniti. Ovdje ili ranije, svejedno.

Commented [D15]: dodano

Commented [ZK16]: Ako ste screenshotali ok, ako je slika preuzeta iz nekog izvora onda to navesti.



Slika 2.5. MATLAB aplikacija za prikupljanje podataka
Izvor: [6]

Najveći nedostatak ovoga rješenja je negrupiranost odabranih signala na jednome mjestu što uvelike otežava korisniku rad i preglednost. Također, nemogućnost promatranja signala u stvarnome vremenu onemogućavanju sigurnost u ispravan odabir pravih signala i uvjeravanje u njihovo dobro funkcioniranje. U programskom rješenju izrađenom u okviru ovoga rada implementirati će se funkcionalnost odabira i prikupljanja podataka, ali bez prethodno navedenih nedostataka.

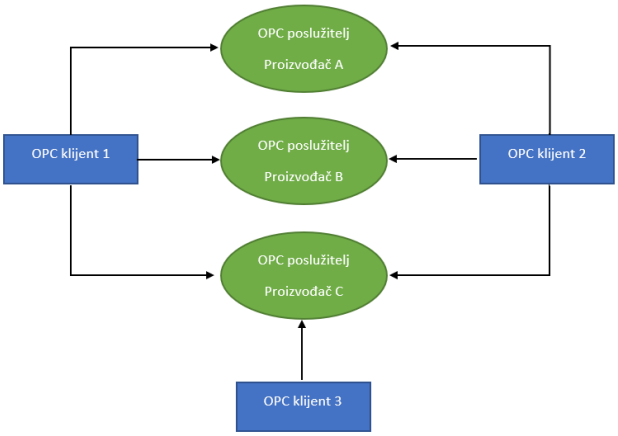
Commented [D17]: Paralela

2.2.3. OPC DA Client Driver

OPC DA Client Driver [7] predstavlja lagan, pouzdan i siguran način spajanja različitih OPC poslužitelja treće strane (engl. *third party*) s bilo kojom OPC DA klijentskom aplikacijom, kao što je i vidljivo na slici 2.6., uključujući različite HMI i SCADA (engl. *Supervisory control and data acquistion*) sustave. Definirano je jedinstveno sučelje kojim se upravlja operacijama pojedinog OPC poslužitelja čime se uklanja potreba prilagođavanja i poznavanja načina rada odabranog poslužitelja treće strane. Također, omogućena je kontrola lokalnih ili udaljenih poslužitelja neovisno o tome podržavaju li DCOM (engl. *The Distributed Component Object Model*) protokol za međusobnu komunikaciju uređaja putem mreže ili ne. Aplikacija kao takva predstavlja OPC grupu kojom se definiraju sve potrebne informacije sustava i pruža podrška grupiranja i logičkog organiziranja dostupnih OPC signala. Funkcionalnost grupiranja OPC

Commented [DR18]: pozivanje na sliku

signala kao i jedinstveno sučelje kojim se upravlja poslužiteljem također će biti implementirani u rješenju izrađenom u okviru ovoga rada. OPC grupe i signali detaljnije su opisani u sljedećem poglavlju.



Slika 2.6. Povezivanje OPC DA Client Driver aplikacije s poslužiteljima različitih proizvođača

2.2.4. Matrikon OPC Explorer

Matrikon OPC Explorer [8] predstavlja OPC klijentsku aplikaciju opće namjene koja omogućuje testiranje OPC poslužitelja te pronalazak i rješavanje problema OPC komunikacijskih veza. Dolazi s naprednim funkcijama za testiranje opterećenja poslužitelja, identifikaciju sigurnih poslužitelja te mogućnosti povezivanja s bilo kojim poslužiteljem kako bi se testirala ispravna konfiguracija. Također omogućuje krajnjim korisnicima sigurnosne protokole za zaštitu od neautoriziranog upada i pristupa podacima odabranog OPC poslužitelja.

Omogućuje pretraživanje lokalnog ili udaljenog računala sadrže li OPC poslužitelje, spajanje na njih te prikazivanje njihovih informacija. Omogućuje izradu OPC grupa te dodavanje i grupiranje OPC signala unutar njih. Odabir signala vrši se kretnjom unutar stablaste strukture poslužitelja te je omogućen prikaz vrijednosti odabranih signala u stvarnom vremenu. Prethodno navedene karakteristike i funkcionalnosti uvelike su usvojene u industriji, stoga su implementirane i u programsko rješenje izrađeno u okviru ovog rada, i to uz mogućnost promatranja stvarnih vrijednosti, što u Matrikon OPC Exploreru nije moguće. Slikom 2.7. prikazano je sučelje aplikacije gdje su jasno vidljive glavne funkcionalnosti.

Commented [D19]: Paralela

Commented [ZK21R20]: U redu je ostaviti ih.

Commented [D20]: Ova slika i slika 3.2. su više-manje jednake. Trebam li izbaciti jednu od njih ili je u redu da ih ostavim jer jedna opisuje posebnu aplikaciju a druga općenito protokol?

Commented [ZK22]: Sliku nacrtati ili prevesti na HR. Uglavnom trebala bi biti na HR jeziku.

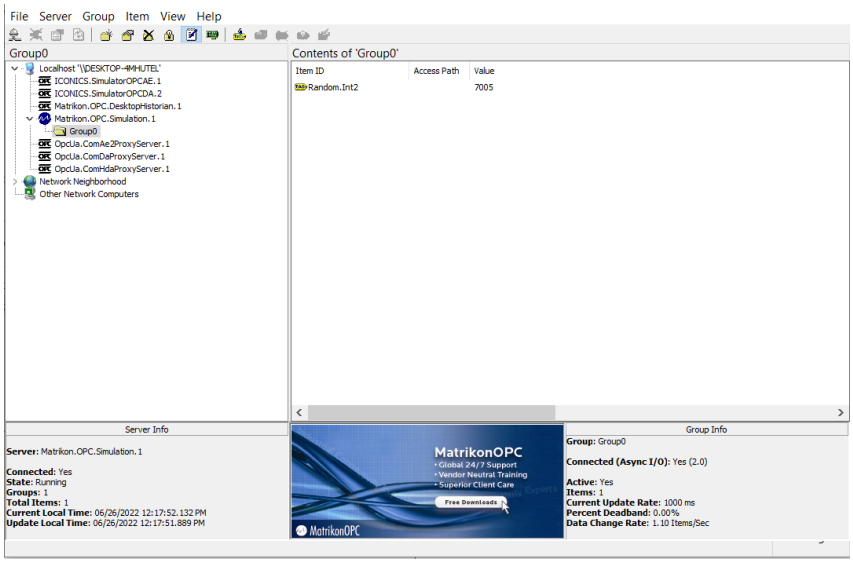
Commented [ZK23]: Ne pozivate se na dijagram u tekstu. A i sada tek primjećujem da to nije slika. Naime, nije uobičajeno da se u radu koristi nešto osim Slika/Tablica/Algoritam, tako da bi tu mogli imati primjedbu.

Commented [ZK24R23]: Važno je također da se na sve slike/tablice/algoritme poziva u tekstu pa to još jednom provjerite.

Commented [ZK25]: Rješavanje problema s OPC poslužiteljem… ?

Commented [D26]: Preformulirano

Commented [D27]: paralela



Slika 2.7. Sučelje Matrikon OPC Explorera

3. MODEL PROGRAMSKOG RJEŠENJA

Zadatak ovog diplomskog rada je osmisliti programsko rješenje kojim će se izraditi konfiguracija nekog industrijskog postrojenja te omogućiti izvlačenje i spremanje podataka postrojenja u bazu uz mogućnost statističke analize odabranih podataka. Rješenje treba sadržavati često korištene i zastupljene značajke već postojećih aplikacija od kojih su neke detaljnije objašnjenje prethodnim poglavljem. Također, postojeća rješenja implementiraju samo neke funkcionalnosti radi kojih su operateri postrojenja prisiljeni koristiti više aplikacija. Jedinstvenost ovog rješenja predstavlja grupiranje svih potrebnih funkcionalnosti u jednu aplikaciju koja omogućuje sve potrebne aktivnosti uz održavanje industrijski usvojenih sučelja i karakteristika.

Sljedećim poglavljem detaljnije su opisani zahtjevi, tehnologije koje su korištene pri izradi te arhitekturni dizajn programskog rješenja.

3.1. Osnovne karakteristike rješenja

Radi lakšeg razumijevanja zahtjeva na programsko rješenje, ovim potpoglavljem će biti opisane osnovne karakteristike programa.

Programsko rješenje omogućuje pretraživanje lokalnog ili udaljenog računala sadrže li OPC poslužitelje, prikaz informacija poslužitelja te spajanje na jednog ili više njih. Spajanjem na određenog poslužitelja započinje izrada konfiguracije. Izrada konfiguracije podrazumijeva navigiranje i pretraživanje OPC poslužitelja u dubinu i odabir željenih OPC signala. Također će biti omogućeno promatranje vrijednosti signala u stvarnom vremenu kako bi se mogla utvrditi njegova ispravnost. Konfiguracija predstavlja listu signala i njihovih karakteristika, poput putanje, grupe OPC poslužitelja kojoj pripada i sl., koji će se promatrati i spremati u bazu podataka. Spremljenim podatcima je omogućen pristup kako bi se vršila statistička analiza. Konfiguracijska lista može se spremiti ili učitati radi lakšeg vođenja evidencije kao i lakše izmjene odabranih signala.

3.2. Zahtjevi na programsko rješenje

Temeljem prethodne analize već postojećih rješenja, kao i zahtjeva projektnog menadžera mentorske tvrtke, definirane su značajke koje programsko rješenje mora sadržavati. Potrebno je odrediti funkcionalnosti te izraditi plan izrade kojeg se treba ugrubo pridržavati tijekom izrade rješenja kako bi se postigli željeni rezultati. U svrhu toga, kreirane su korisničke priče, kratki zapisi kojima se na jednostavan način definiraju zahtjevi programskog rješenja. Prema [9], korisnička

Commented [ZK28]: izraditi

Commented [ZK29R28]: Search/Replace

Commented [ZK30]: konkretnije napisati o kakvom se postrojenbju radi, barem osnovno

Commented [D31R30]: Aplikacija nije rađena za nekakvo određeno postrojenje, ona omogućuje izradu konfiguracije bilo kojeg postrojenja (neovisno o vrsti dobara kojeg proizvodi) koje ima uspostavljenje OPC poslužitelje. Molim dodatno komentirati ako je potrebno nešto više objasniti u radu

Commented [ZK32R30]: Pnda barem napisati industrijsko. Ili konfiguracija postrojenja iz kojeg će se izvlačiti podaci...

Commented [ZK33]: Razlog više za komentar koji sam Vam dao na potpoglavlje 2.2.

Commented [ZK34]: Sada djeluje kao da postojeće rješenje mijenjate samo da "bude drugačije pod svaku cijenu". Taj efekt nikako ne želimo postići. Možete to motivirati činjenicama da postojeća rješenja nemaju ono što Vama treba, ili da postojeća primjena je specifična i zahtijeva neke specifične funkcionalnosti koje ne možete jednostavno pokriti postojećim rješenjima i sl.

Commented [D35]: Dodano (staticka analiza)

Commented [ZK36]: Zahtjevi programskog rješenja -> zahtjevi na programsko rješenje. Ovo prvo je gotovo programsko rješenje koje postavlja zahtjeve. (Izmijeniti i u sadržaju).

priča predstavlja zapis potreba korisnika pa se stoga zapisuju iz njegove perspektive. Svaka priča obično slijedi jednostavan predložak oblika: Kao (korisnik) želim (cilj) kako bi (neka svrha). Tablicom 3.1. prikazane su korisničke priče kojima su specificirani zahtjevi sustava.

Tablica 3.1. Korisničke priče

1	Kao operater, želim se spojiti na lokalni ili udaljeni OPC poslužitelj tako da mogu pristupati njegovim informacijama.
2	Kao operater, želim se moći spojiti istovremeno na više poslužitelja tako da mogu istovremeno raditi više konfiguracija.
3	Kao operater, želim moći dohvatiti informacije o pojedinom OPC poslužitelju kako bih mogao izolirati njegove specifičnosti.
4	Kao operater, želim imati zaseban prostor korisničkog sučelja u kojemu se nalaze sve otvorene konfiguracije koje mogu poravnati i micati po volji radi preglednosti.
5	Kao operater, želim imati mogućnost dodavanja, brisanja i izmjene grupa OPC poslužitelja kako bih mogao grupirati signale i pozivati globalne funkcije nad njima.
6	Kao operater, želim imati uvid u stablastu strukturu poslužitelja i mogućnost kretanja u dubinu radi pronalaska OPC signala kojima kreiram konfiguraciju.
7	Kao operater, želim imati tablicu u kojoj se nalaze svi signali koje sam odabrao kao i njihove karakteristike, kako bih imao jednostavan pregled svih signala potrebnih za ~~kreiranje~~ izradu konfiguracije.
8	Kao operater, želim imati mogućnost promatranja stvarnih vrijednosti odabranih signala kako bih znao poprima li signal prihvatljive vrijednosti te odgovara li onome kojeg želim koristiti pri izradi konfiguracije.
9	Kao operater, želim imati mogućnost spremanja konfiguracije kako bih mogao kasnije nastaviti s radom koristeći tu konfiguraciju. Također želim imati mogućnost učitavanja već postojeće konfiguracije kako bih dodao ili obrisao pojedine signale.
10	Kao operater, želim da se vrijednosti signala kreirane konfiguracije počnu spremati u bazu podataka radi kontrole proizvodnog procesa i uvida u kretnje vrijednosti signala.
11	Kao operater, želim moći pristupiti željenim podatcima u bazi podataka radi vršenja ~~statičk~~statističke analize

Commented [ZK37]: Negdje biste ranije trebali definirati što je konfiguracija (prije korisničkih priča). Razmišljam da stavite jedan poseban odlomak prije tih priča u kojem ćete najaviti neke osnovne karakteristike Vašeg programskog rješenja.

Commented [ZK38]: biti konkretniji, kakav prostor? Negdje u GUI-ju?

Commented [ZK39]: Što su OPC grupe, jesu li to grupe OPC poslužitelja? Ako jesu, onda pisati grupe OPC poslužitelja kako biste bili konzistentni u nomenklaturi.

Commented [ZK40]: preurediti ovaj dio rečenice

Commented [D41]: dodano

3.3. Korištene tehnologije

Ovim poglavljem detaljnije su opisane korištene tehnologije za izradu programskog rješenja, rukovanje podacima kao i postavljanje baze podataka. Tehnologije i tehnike su odabrane zbog opsežnosti korištenja, jednostavnosti te prijedlogom mentorske tvrtke.

Commented [ZK42]: OK, ali opet za svaku tehnologiju/tehniku/metodu navesti u rečenici dvije zašto Vam baš ona treba, i možda navesti alternative. To pisati u potpoglavljima gdje ih opisujete. Možete na kraj svakog potpoglavlja.

Commented [D43R42]: Za OPC protokol nisam stavio razlog korištenja jer smatram da je intuitivno i očito. Ispravite me ako sam u krivu.

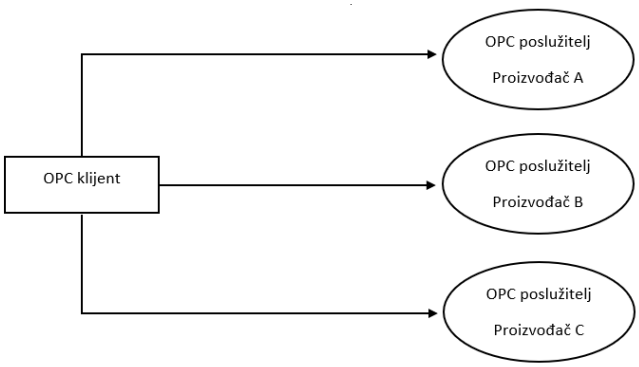
Commented [ZK44R42]: To je u redu

3.3.1. OPC Protokol

OPC (*OLE for Process Control*) [10, 11, 12] predstavlja standard za upravljačku komunikaciju među procesima temeljenu na COM (engl. *Component Object Model*) tehnologiji. Ona omogućuje jednostavnu ponovnu upotrebu programskih komponenti bez znanja o njihovoj unutarnjoj strukturi i implementaciji. COM tehnologija omogućuje korištenje objekata u okruženjima različitima od onih u kojima su napravljeni, pa čak i u drugim operacijskim sustavima. Distribuirani COM (engl. *Distributed COM, DCOM*) proširuje osnovni COM model i omogućuje komunikaciju programskih objekata distribuiranih na različitim računalima putem mreže. COM također predstavlja temeljnu tehnologiju za povezivanje i ugrađivanje objekata (engl. *Object and Linking Embedding, OLE*).

Uz COM, OLE predstavlja drugu tehnologiju koja je temelj OPC protokola. Microsoft Windows OLE tehnologija, kasnije poznata kao ActiveX, prilagođena je za upravljanje i vođenje procesa. Njome je omogućena razmjena podataka između različitih aplikacija, definira standardne metode, objekte pa tako i zahtjeve za interoperabilnost sklopovlja tj. kako bi različiti tipovi sklopovlja mogli međusobno komunicirati.

OPC uveo je standardne metode za komunikaciju različitih izvora podataka, bilo uređaja unutar postrojenja ili pak baza podataka unutar upravljačkih soba. Protokol se sastoji od skupa sučelja, specifikacija i metoda za procesnu kontrolu kao i proizvodnju aplikacija i programske podrške za automatizaciju proizvodnih procesa. Ovime se povezuju proizvođači sklopovlja s proizvođačima programske podrške omogućujući klijentskim aplikacijama pristup bilo kojem sklopovlju koje ima definirano OPC sučelje, što je prikazano slikom 3.2.



Slika 3.2. Spajanje OPC klijenta s OPC poslužiteljem bilo kojeg proizvođača

Tijekom godina razvijeno je više specifičnih sučelja kojima su detaljnije definirane metode i načini pristupa podacima. Prema [13], sučelja se dijele na sučelje za pristup podacima (engl. *Data Access, DA*), sučelje za alarme i događaje (engl. *Alarm&Event, AE*) te sučelje za povijesne podatke (engl. *HistoricalData Access, HDA*).

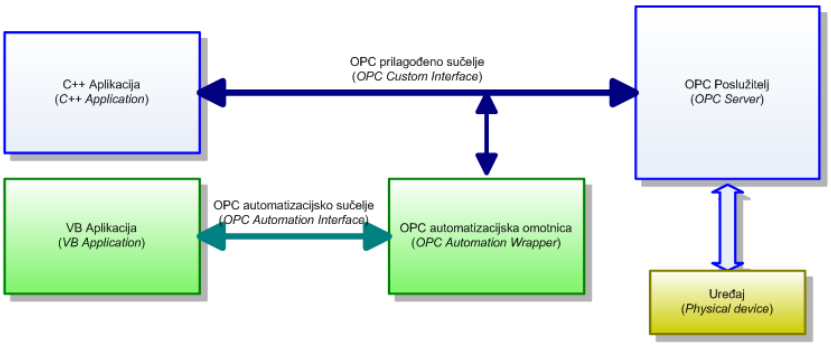
3.3.2. OPC DA

OPC sučelje za pristup podacima (engl. *OPC Data Access, OPC DA*) [10, 11, 12, 14] zapravo se sastoji od 2 sučelja, sučelje za automatizaciju i prilagođeno sučelje. Sučelje za automatizaciju integrirano je u Visal Basic programskom jeziku te svojom jednostavnošću omogućuje lakši razvoj aplikacija za automatizaciju. Prilagođeno sučelje integrirano je u C++ programskom jeziku omogućujući veliku fleksibilnost i više opcija prilikom izrade aplikacija. OPC DA klijent omogućuje pristup lokalnim ili udaljenim OPC poslužiteljima pomoću metoda za čitanje i pisanje čime se uspostavlja komunikacija klijenta i poslužitelja. Slikom 3.3 prikazan je način komunikacije OPC klijenta i odgovarajućeg poslužitelja.

Commented [ZK45]: Prema uputama za izradu diplomskog rada, referenca na literaturu treba biti dio rečenice. U smislu: "...prema [15] izradit će se…", ili "…više je opisano u [15]…" i sl.

Commented [ZK46]: od čega?

Commented [ZK47]: od koga/čega? Ako ne uspoređujete, onda stavite "veliku".

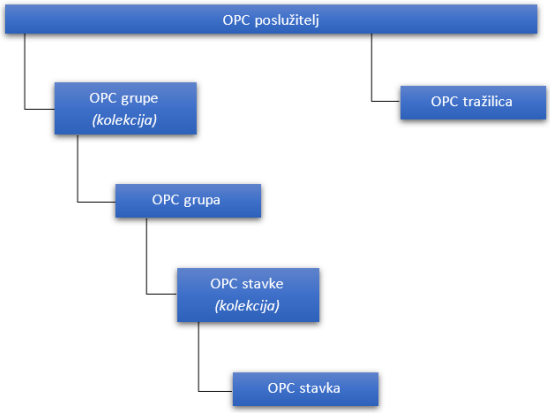


Slika 3.3. Komunikacija OPC klijenta i poslužitelja
Izvor: [14]

Radi pojednostavljenja upravljanja podacima, OPC DA, prema [7], implementira 4 objekta: poslužitelj (engl. *server*), grupa (engl. *group*), stavka (engl. *item*) i tražilica (engl. *browser*). Objekt poslužitelja sadržava informacije o samom poslužitelju te služi kao spremnik objekata grupe. Objekt grupe sadrži informacije o sebi i omogućuje mehanizam za kontrolu i logičko grupiranje različitih objekata stavki. Svaka stavka povezana je s pripadajućim signalom u postrojenju. Objekt tražilice omogućuje izlistavanje svih naziva stavki u konfiguraciji poslužitelja. Svakoj instanci OPC poslužitelja pridružena je samo jedna instanca OPC tražilice. Struktura OPC poslužitelja i odnos između spomenutih objekata prikazan je slikom 3.4.

Commented [D48]: dodano

Commented [ZK49]: naziva



Slika 3.4. Struktura OPC poslužitelja i njegovih pripadajućih objekata

Korištenje OPC DA kao dio programskog rješenja omogućava pristup OPC poslužitelju, njegovoj strukturi kao i njegovim signalima za izradu konfiguracije postrojenja.

Commented [D50]: Razlog korištenja

3.3.3. Programsko okruženje

Za izradu grafičkog korisničkog sučelja, implementaciju svih potrebnih zahtjeva te komunikaciju s bazom podataka korišteno je integrirano razvojnog okruženje (engl. *Integrated Development Environment, IDE*) Microsoft Visual Studio [16]. Okruženje ima ugrađen uređivač izvornog koda (engl. *Code editor*), sustav za pronalaženje pogrešaka (engl. *Debugger*) te vrlo korišteni sustav dizajniranja sučelja za Windows aplikacije (engl. *Windows Forms Designer, WFD*) koji je korišten za izradu svih prozora aplikacije. Visual Studio korišten je zbog svoje jednostavnosti izrade sučelja kao i velikog predznanja o istom, stečenog tijekom studiranja i rada u mentorskoj tvrtki.

Commented [ZK51]: velikog predznanja o istom, stečenog tijekom studiranja i rada u mentorskoj...

Već spomenuti ugrađeni sustavi dio su .Net programskog okvira [17] koji predstavlja posebnu infrastrukturu koja sadrži već gotova i često korištena rješenja i funkcionalnosti. Korištenje gotovih i testiranih rješenja smanjuje kompleksnost izrade aplikacije kao i samo vrijeme potrebno za implementaciju zahtjeva. Najvažniji dio .Net platforme predstavlja CLR (engl. *Common Language Runtime*) sustav kojim se osigurava sigurnost i funkcionalnost pokrenute aplikacije na način da upravlja memorijom, osigurava dodatne biblioteke za razvoj aplikacija različitih funkcionalnosti i još mnogo toga, kao što je i navedeno u [18].

Kod je pisan programskim jezikom C# [18], objektno orijentiranim jezikom temeljenom na .Net programskom okviru. Široko je upotrebljavan pri izradi aplikacija za računala pri čemu omogućuje višenitno (engl. *multi-threading*) izvođenje programskog koda.

Microsoft Visual Studio samo je jedan od mnogih razvojnih okruženja koje se mogu koristiti pri izradi programskog rješenja, a odabran je zbog višegodišnjeg korištenja te preglednosti koda i njegovih struktura. Programski okvir .Net i programski jezik C# odabrani su zbog svoje dobre povezanosti s Visual Studio okruženjem, prethodnog iskustva, visokim performansama te jednostavne izgradnje računalnih aplikacija. Alternativno, moguće je korištenje Java programskog jezika koji također ima veliku potporu pri izgradnji aplikacija za računala.

3.3.4. Oracle baza podataka

Commented [ZK52]: Pišete o korištenim tehnologijama, pa bi ovdje trebali napisati o konkretnoj tehnologiji koju koristite za izradu/administriranje baze podataka.

Jedna od funkcionalnosti aplikacije je spremanje podataka u bazu i njihova analiza radi povećanja sposobnosti kontrole procesa proizvodnje. Kontrola je od velike važnosti kako bi se osigurao ispravan rad postrojena praćenjem vrijednosti signala i osiguravanjem da te iste vrijednosti ostanu unutar definiranih granica. U tu svrhu kreirana je Oracle baza podataka kojoj je omogućen pristup

Commented [ZK53]: Zašto spremanje podataka u bazu omogućuje povećanje sposobnosti kontrole… Trebate to malo preciznije napisati, slobodno s rečenicom više ako treba.

Commented [D54]: Povećanje sposobnosti kontrole

putem aplikacije. Odabir Oracle baze određen je mentorskom tvrtkom koja već duži niz godina koristi navedenu bazu podataka. Također, prema [19], Oracle omogućuje koncepte strojnog učenja za nadziranje i upravljanje bazom podataka, ima implementirane stroge sigurnosne protokole te omogućuje jedinstvenu bazu za sve tipove podataka.

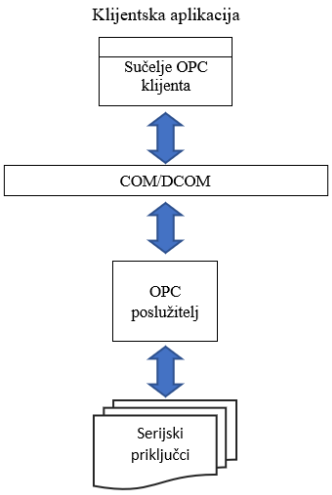
.Net programski okvir omogućuje alate za jednostavno i brzo povezivanje s Oracle bazom podataka te upravljanje podacima pomoću SQL (engl. *Structured Querry Language*) jezika radi čega je ova baza podataka bila primarni izbor. Također, mentorska tvrtka godinama koristi Oracle bazu podataka te je predloženo njeno korištenje radi skalabilnosti i kompatibilnosti s već postojećom strukturom.

Commented [D55]: Razlog korištenja Oracle-a

3.4. Arhitekturalni dizajn rješenja

Prema traženim karakteristikama i zahtjevima aplikacije, koji su definirani korisničkim pričama u poglavlju 3.1., kao i činjenici da se sva OPC sučelja temelje na COM/DCOM tehnologiji te prema [20, 21], izabrana arhitektura jest klijent-poslužitelj. Sustav je strukturiran tako da se na klijentskoj strani nalazi aplikacija dok se na poslužiteljskoj strani nalaze OPC poslužitelji danog postrojenja kao što je i prikazano slikom 3.5. Uspostavom veze klijenta i poslužitelja omogućava se prijenos podataka i pronalazak signala za izradu konfiguracije.

Commented [D56]: dodano

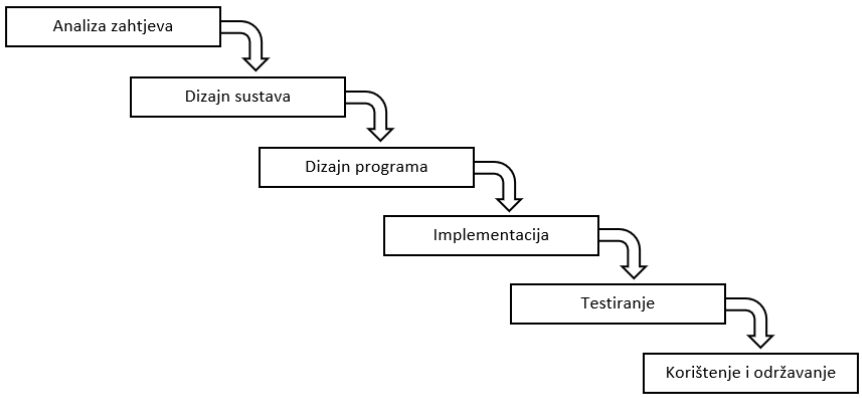


Slika 3.5. Klijent-poslužitelj arhitektura aplikacije

4. IMPLEMENTACIJA RJEŠENJA

Programsko rješenje razvijeno je u svrhu kreiranja konfiguracije postrojenja te je dizajnirano i implementirano za korisnike koji su upoznati s OPC DA sučeljem i njegovim specifikacijama. Kako bi aplikaciju moglo koristiti što više korisnika uz minimalno potrebno znanje o korištenim tehnologijama, aplikacija je ostvarena na način razumljiv korisniku kojemu je također dan priručnik korištenja. Cilj je bio stvoriti intuitivno korisničko sučelje (engl. *User Interface, UI*) bez nepotrebnih informacija koje bi zbunile korisnika. Imena gumbova, lista i ostalih kontrola imaju točno određeno značenje kojim se umanjuje dvosmislenost zadataka.

Programsko rješenje razvijalo se koristeći model vodopada (engl. *Waterfall model*) [22] koji je prikazan slikom 4.1. Model je odabran zbog mogućnosti detaljnog planiranja procesa razvoja aplikacije te konzistentnih i jasno definiranih zahtjeva samoga rješenja. Sustav se dijeli na manje podsustave i module ovisno o funkcionalnostima koje se implementiraju tek nakon što se prethodni podsustavi i koraci izrade u potpunosti.



Slika 4.1. Model vodopada

Model vodopada sastoji se prvotno od planiranja i definiranja zahtjeva programskog rješenja. Temeljem njih izrađuje se dizajn sustava kao i dizajn programa kojeg se treba pridržavati prilikom izrade aplikacije. Implementacijom podsustava ostvaruje se konkretno rješenje spomenutih specifikacija. U ovom dijelu procesa može se ustanoviti kako su potrebne promjene, odnosno preinake sustava ili kako postoji bolji način ostvarenja, stoga prilikom definiranja zahtjeva nije

Commented [ZK57]: Mislim da sam već negdje komentirao, ali nedostaje Vam opis što je to knofiguracija postrojenja.

Commented [ZK58]: Preformulirati. Trenutno nemam ideja kako...

Commented [D59]: Preformulirano

Commented [ZK60]: Navesti zašto.

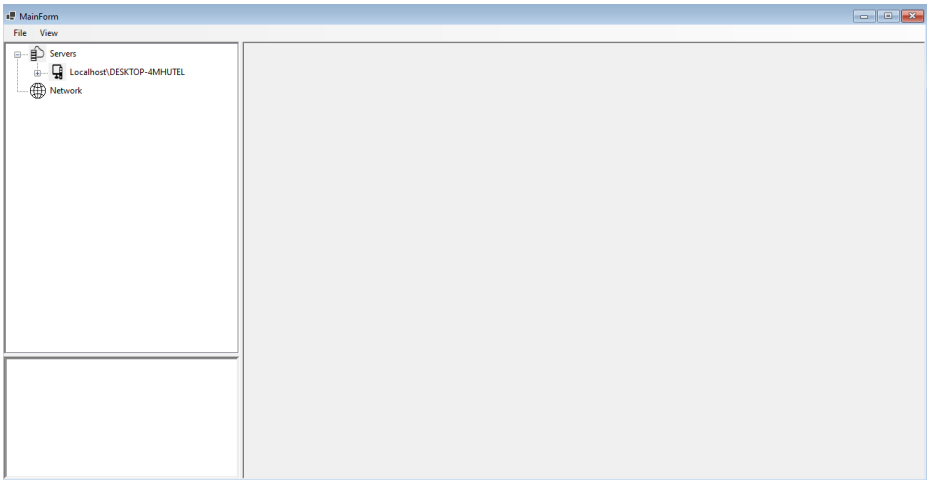
potrebno raditi cjelokupnu i detaljnu analizu istih. Uspješnom implementacijom kreće se u fazu testiranja čime se ispravljaju eventualne pogreške te se omogućuje poboljšanje korisničkog iskustva (engl. *User Experience, UX*). Posljednjim korakom definiraju se operacije i zadaci za uspješno održavanje i korištenje sustava.

4.1. Programsko rješenje

Prethodno definiranim zahtjevima i korisničkim pričama, razvijeno je intuitivno i lako razumljivo rješenje. Aplikacija se temelji na dva korisnička sučelja izrađena korištenjem Windows Formi, glavno i konfiguracijsko sučelje. Svako se sučelje sastoji od više dijelova koji prikazuju različite podatke i obavljaju određene funkcionalnosti. Ti dijelovi se također mogu skalirati te smanjivati i proširivati ovisno o korisnikovim željama, čineći određeni raspored (engl. *layout*).

4.1.1. Glavno sučelje

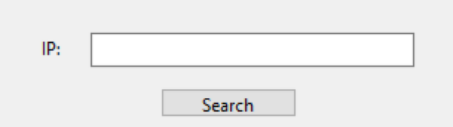
Pokretanjem aplikacije otvara se glavno sučelje prikazano slikom 4.2. Ono je podijeljeno na tri glavna elementa. S lijeve strane nalaze se elementi koji omogućuju pregled i spajanje na OPC poslužitelje te koji pružaju dodatne informacije o odabranom poslužitelju dok se s desne strane nalazi područje u kojem se otvara jedno ili više konfiguracijskih sučelja.



Slika 4.2. Početno sučelje aplikacije

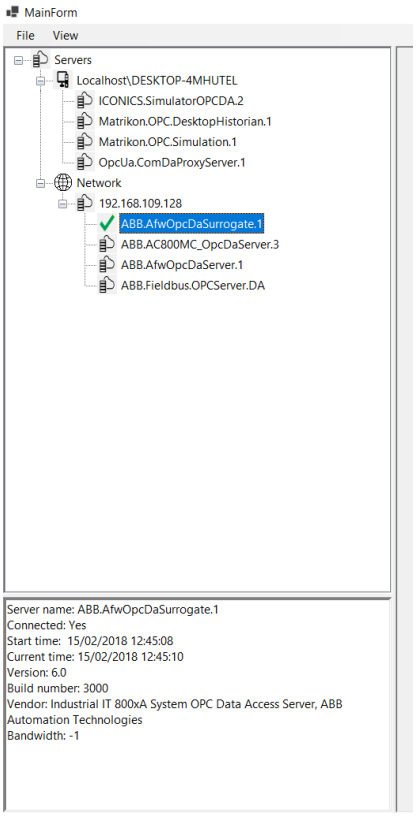
Stablasti prikaz dostupnih OPC poslužitelja omogućuje korisniku simultano spajanje na više poslužitelja u svrhu kreiranja konfiguracija koristeći OPC signale dostupne na poslužitelju.

Prilikom prvog pokretanja aplikacije, u pozadini se izvršava funkcija pretraživanja lokalnog računala za OPC poslužiteljima koji se po pronalasku dodaju kao elementi *Localhost* čvora. Također se, prilikom završetka dodavanja svih pronađenih poslužitelja na lokalnom računalu, dodaje čvor *Network* koji operateru daje mogućnost spajanja na udaljeno računalo. Dvostrukim lijevim klikom miša na čvor *Network* otvara se prozor, prikazan slikom 4.3., u koji se upisuje IP adresa udaljenog računala. Točnim unosom, adresa računala se dodaje kao čvor u listu dostupnih OPC poslužitelja početnog sučelja, dok se kao njegovi elementi dodaju svi OPC poslužitelji pronađeni na tom računalu.



Slika 4.3. Prozor za spajanje na udaljeno računalo

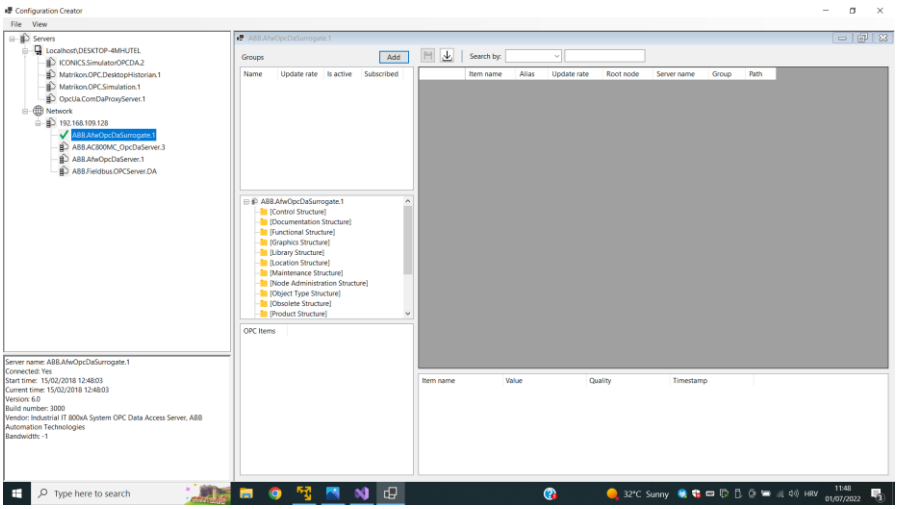
Prethodni postupci omogućuju operateru jedinstven pregled svih dostupnih OPC poslužitelja lokalnog i jednog ili više udaljenih računala. Jednostruki lijevi klik miša na ime poslužitelja omogućuje prikaz detaljnih informacija poslužitelja u zasebnom prozoru. Ovime se operateru daje uvid u zasebnosti odabranog poslužitelja poput vremena spajanja na poslužitelj, verziju poslužitelja, informacije o proizvođaču i sl. Temeljem ovih informacija, operater odabire željenog poslužitelja s kojim uspostavlja vezu. Svi poslužitelji s kojima je uspostavljena veza označeni su ikonom zelene kvačice. Slikom 4.4. prikazana je stablasta struktura dostupnih OPC poslužitelja prilikom spajanja na udaljeno računalo te prikaz informacija o odabranom poslužitelju s kojim je ujedno i uspostavljena veza.



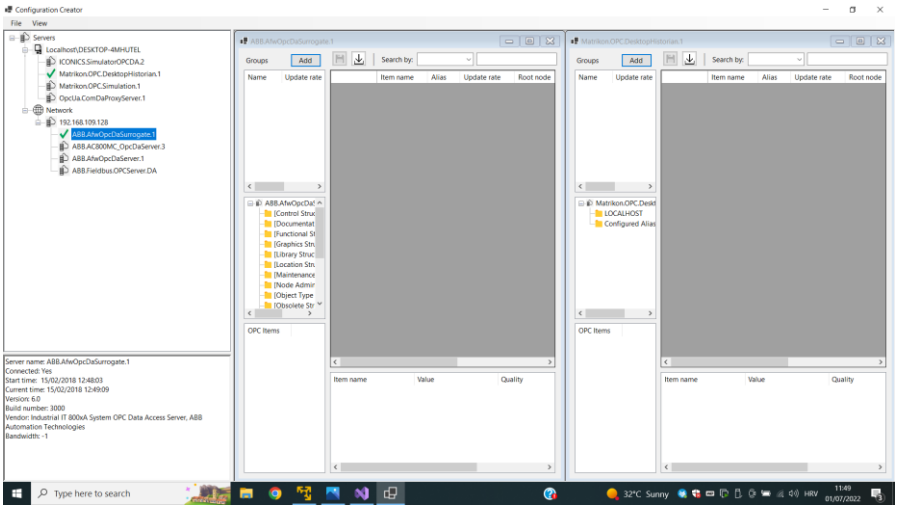
Slika 4.4. Stablasti prikaz dostupnih OPC poslužitelja i njihovih informacija

Nakon spajanja na jednog ili više poslužitelja, operater može započeti s izradom konfiguracije postrojenja dvostrukim lijevim klikom miša na ime poslužitelja. Time se otvara konfiguracijsko sučelje u prethodno spomenutom desnom dijelu glavnog sučelja što je prikazano slikom 4.5.

Operateru je također omogućeno otvaranje više konfiguracijskih sučelja čime može istovremeno izraditi konfiguracije različitih postrojenja, kao što je i vidljivo na slici 4.6. Važno je napomenuti kako se sva otvorena konfiguracijska sučelja mogu proizvoljno povećavati, smanjivati pa tako i horizontalno ili vertikalno rasporediti kako bi se korisniku omogućio jasniji pregled svih otvorenih sučelja.



Slika 4.5. Otvaranje jednog konfiguracijskog sučelja

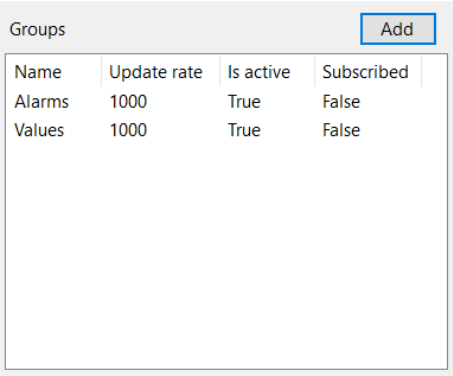


Slika 4.6. Otvaranje više konfiguracijskih sučelja uz horizontalno poravnanje

4.1.2. Konfiguracijsko sučelje

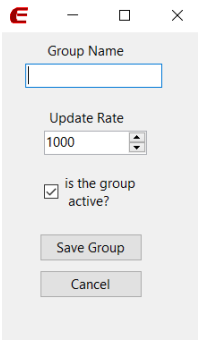
Konfiguracijsko sučelje predstavlja okosnicu kreiranja konfiguracije odabranog postrojenja. Izrada konfiguracije započinje prethodno spomenutom uspostavom veze s OPC poslužiteljem dostupnim iz liste poslužitelja glavnog sučelja. Konfiguracijsko sučelje sastoji se od pet dijelova koji zajedno čine cjelinu koja omogućuje odabir OPC signala i kreiranja konfiguracije.

Kao što je prikazano slikom 3.4., struktura OPC poslužitelja temelji se na grupama OPC poslužitelja. Implementacija rukovanja grupama poslužitelja izvršena je pomoću objekta *Listview* koji sadržava sve grupe i njihove informacije kako je i prikazano slikom 4.7.



Slika 4.7. Prikaz kreiranih grupa OPC poslužitelja i njihovih informacija

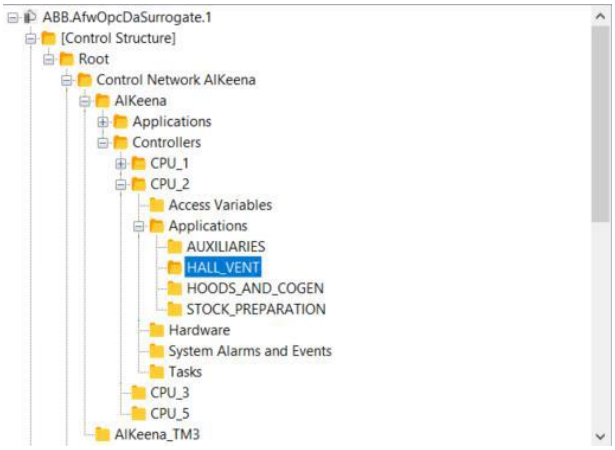
Izrada nove grupe omogućeno je pritiskom gumba *Add* kojim se otvara prozor za unos podataka za izradu, što je prikazano slikom 4.8. Korisniku je omogućeno dodavanje proizvoljnog broja grupa.



Slika 4.8. Sučelje za izradu nove grupe

Svaka grupa jednoznačno je određena svojim imenom te je implementiran mehanizam nemogućnosti dodavanja grupe s već korištenim imenom. Svojstvo *Update Rate* predstavlja period sinkrone razmjene podataka u milisekundama. Opcija *is the group active?* omogućuje korisniku odabir hoće bi grupa biti aktivna ili ne. Aktivna grupa je ona koja ima mogućnost dohvata podataka dok neaktivna nema tu mogućnost osim u slučaju eksplicitnog korištenja metoda za sinkrono čitanje i pisanje. Klikom na gumb *Save group* grupa se dodaje u listu svih grupa u dodatno generiranje svojstva *isSubscribed*. Ono govori je li grupa pretplaćena na asinkrono primanje podataka prilikom promjene vrijednosti pripadajućih stavki na poslužitelju. Početna zadana vrijednost je *False* zbog toga što unutar grupe inicijalno nema stavki .

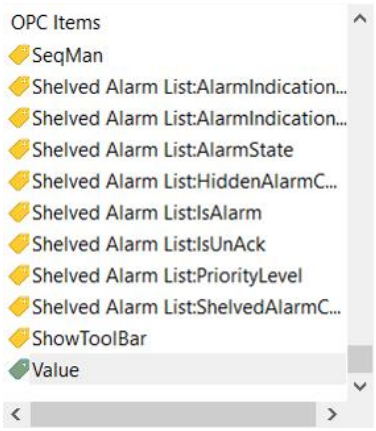
Izradom jedne ili više grupa korisniku je omogućeno dodavanje OPC stavki u željene grupe. Kao što je prethodno navedeno, svaka stavka predstavlja jedan signal unutar nekog postrojenja, poput razine vode u nekakvom spremniku, temperature zraka, tlaka, brzine vrtnje turbine i sl. Sve stavke nalaze se na jedinstvenim mjestima unutar stablaste strukture OPC poslužitelja. Kako bi se učinkovito pronašle željene stavke, implementiran je mehanizam pretraživanja strukture poslužitelja pomoću objekta *TreeView*, prikazano slikom 4.8.



Slika 4.8. Stablasta struktura odabranog OPC poslužitelja

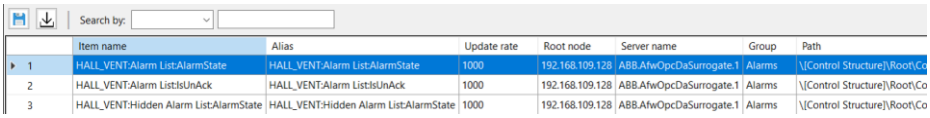
Kretanje po granama omogućeno je objektom tražilice (engl. *OPC Browser*), korištenjem metode *OPCBrowser.MoveTo()*. Klikom na ime grane, tj. datoteke, tražilica se pomiče na tu granu te poziva metodu *OPCBrowser.ShowBranches()*. Ovom metodom otkrivaju se sve podgrane odabrane datoteke koje se dodaju kao elementi djeca unutar grafičkog prikaza strukture kako bi se

moglo izvršavati daljnje pretraživanje u dubinu. Također se, klikom na odabranu granu, poziva metoda *OPCBrowser.ShowLeafs()*. Ovom metodom se odabrana grana pretražuje i pronalazi sve dostupne OPC stavke. Sve pronađene stavke prikazuju se u zasebnoj listi koja je dana slikom 4.9. Lista stavki se automatski osvježava prilikom svake kretnje po strukturi poslužitelja kako bi se prikazale samo one stavke koje se lokacijski nalaze na odabranoj grani.



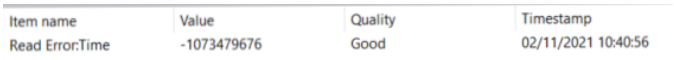
Slika 4.9. Lista dostupnih OPC stavki prethodno odabrane grane.

Nadalje, odabirom jedne ili više stavki, metodom povuci i ispusti (engl. *Drag and Drop*) omogućeno je odvlačenje stavki u zasebnu tablicu koja predstavlja konfiguraciju. Tablica je strukturirana na način da se za svakom pojedinačnom signalu spreme i prikažu njegove karakteristike. Karakteristike signala su sljedeće: ime, alias (koji je ujedno i jedina karakteristika koju korisnik može mijenjati i proizvoljno upisati), ime grupe kojoj signal pripada i njeno vrijeme osvježavanja, ime korijenskog čvora poslužitelja u kojem se nalazi signal, ime poslužitelja te apsolutna putanja do signala unutar strukture poslužitelja. Ime poslužitelja kao i apsolutna putanja signala potrebno je zabilježiti radi potreba upisivanja u bazu podataka. Također, implementirana je funkcionalnost filtriranja signala po ključu čime se prikazuju samo oni signali koji sadrže uneseni tekst. Primjer izrađene tablice konfiguracije, prikaz signala i njihovih značajki kao i kontrola za filtriranje, spremanje i učitavanje prikazane se slikom 4.10.



Slika 4.10. Tablica odabranih signala

Prilikom odabira signala i kreiranja konfiguracije može doći do zabune i ljudske pogreške kod odabira ispravnih signala. Kako bi se smanjila vjerojatnost pogreške i kreiranja manjkave konfiguracije, implementiran je mehanizam promatranja vrijednosti odabranih signala u stvarnome vremenu. Izrada liste signala za promatranje omogućava korisniku detaljan uvid u ponašanje i periodično mijenjanje vrijednosti signala kao i uvid u njegovu kvalitetu. Stupci liste određuju karakteristike koje se prikazuju za svaki pojedinačni signal. Pod stupcima su uključeni ime signala, trenutna vrijednost signala, kvaliteta signala te vrijeme zadnjeg čitanja vrijednosti, kao što je prikazano primjerom na slici 4.11.



Item name	Value	Quality	Timestamp
Read Error:Time	-1073479676	Good	02/11/2021 10:40:56

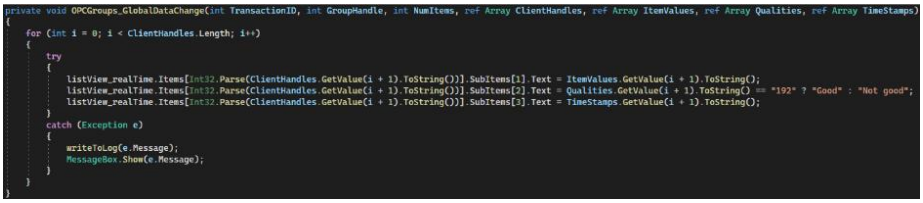
Slika 4.11. Lista za promatranje stvarnih vrijednosti signala

Dodavanje novog elementa u listu okida dva događaja. Prvi obuhvaća provjeru nalazi li se signal već u listi i promatra li se već njegova vrijednost. Ukoliko ne, jedan ili više odabranih signala dodaju se u listu na kraj u obliku novih redaka te okidaju drugi događaj. Drugi događaj predstavlja osnovnu funkcionalnost i upotrebu OPC komunikacijskog protokola. Služeći se prethodno spremljenim karakteristikama signala u tablici signala konfiguracije, točnije imenom poslužitelja, imenom grupe te apsolutnom putanjom do signala, kreira se događaj slušanja promjene vrijednosti. To se izvršava na način da se na odabranom OPC poslužitelju kreira grupa prethodno odabranog imena te se u nju smješta sam signal. Dodavanje signala u grupu prikazano je programskim kodom na slici 4.12.

```
OPCItem addedOPCItem = this.tempOPCServer.OPCGroups.GetOPCGroup(row.Cells["group"].Value.ToString())
                       .OPCItems.AddItem(this.browser.GetItemID(itemName), listView_realTime.Items.Count);
addedOPCItem.Read((short)OPCDataSource.OPCDevice, out value, out quality, out timestamp);
isValid = true;
```

Slika 4.12. Mehanizam dodavanja signala u grupu OPC poslužitelja

Nakon dodavanja svih signala u odgovarajuće grupe kreira se događaj *GlobalDataChange*, čija je implementacija prikazana slikom 4.13., a koji osluškuje sve promjene koje se događaju na odabranim signalima na stvarnom poslužitelju. Svakom promjenom događaj se okida i zatim prikazuje sve promijenjene vrijednosti u listi za praćenje. Brisanjem signala iz liste briše se također i signal iz njegove grupe čime prestaje praćenje promjena njegovih vrijednosti.



```
private void OPCGroups_GlobalDataChange(int TransactionID, int GroupHandle, int NumItems, ref Array ClientHandles, ref Array ItemValues, ref Array Qualities, ref Array TimeStamps)
{
    for (int i = 0; i < ClientHandles.Length; i++)
    {
        try
        {
            listView_realTime.Items[Int32.Parse(ClientHandles.GetValue(i + 1).ToString())].SubItems[1].Text = ItemValues.GetValue(i + 1).ToString();
            listView_realTime.Items[Int32.Parse(ClientHandles.GetValue(i + 1).ToString())].SubItems[2].Text = Qualities.GetValue(i + 1).ToString() == "192" ? "Good" : "Not good";
            listView_realTime.Items[Int32.Parse(ClientHandles.GetValue(i + 1).ToString())].SubItems[3].Text = TimeStamps.GetValue(i + 1).ToString();
        }
        catch (Exception e)
        {
            writeToLog(e.Message);
            MessageBox.Show(e.Message);
        }
    }
}
```

Slika 4.13. Implementacija rukovanja događajem GlobalDataChange

Konačno, kako bi se konfiguraciju moglo dijeliti s drugima te kako bi se moglo nastaviti s radom i/ili izmijeniti već postojeće konfiguracije, implementirane su funkcije spremanja i učitavanja konfiguracija. Konfiguracije se spremaju u datoteku JSON formata čime se signali formiraju i strukturiraju kao pojedinačni JSON objekti. Primjer jedne takve datoteke dan je slikom 4.14.


```
[
    {
        "itemName": "AIC1:Value",
        "alias": "AIC1:Value",
        "itemPath": "\\[Control Structure]\\Root\\Control Network AlKeena\\AlKeena\\Applications\\test\\Programs\\OPCTest\\AIC1:Value",
        "rootNodeName": "192.168.109.128",
        "serverName": "ABB.AfwOpcDaSurrogate.1",
        "groupName": "group1",
        "groupUpdateRate": 1000
    },
    {
        "itemName": "AIC2:Value",
        "alias": "AIC2:Value",
        "itemPath": "\\[Control Structure]\\Root\\Control Network AlKeena\\AlKeena\\Applications\\test\\Programs\\OPCTest\\AIC2:Value",
        "rootNodeName": "192.168.109.128",
        "serverName": "ABB.AfwOpcDaSurrogate.1",
        "groupName": "group1",
        "groupUpdateRate": 1000
    }
]
```

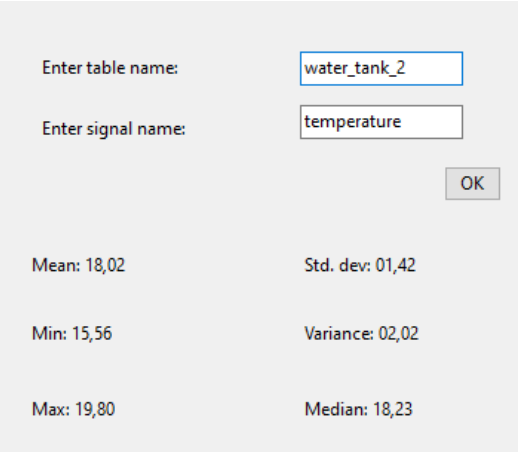

Slika 4.14. Primjer konfiguracije zapisane u JSON datoteci

4.1.3. Upisivanje u bazu podataka

Svrha kreiranja konfiguracije postrojenja je odabir signala čije se vrijednosti žele promatrati i spremati u bazu podataka. Kako bi se spremanje omogućilo, implementirano je povezivanje na Oracle bazu podataka putem Windows servisa (engl. *Windows Service*) te periodično dohvaćanje vrijednosti signala sa poslužitelja i njihovo spremanje. Dohvaćanje podataka s poslužitelja odvija se na isti način kao i kod promatranja stvarnih vrijednosti signala, odnosno na OPC poslužitelju se kreira grupa u koju se smještaju pripadajući signali nad kojima se kreira *GlobalDataChange* događaj koji osluškuje promjene vrijednosti signala. Prilikom detektirane promjene, koristeći SQL jezik, podatci se upisuju u pripadajuće tablice i stupce.

4.1.4. Statistička analiza podataka

Spremljenim podatcima omogućen je pristup preko glavnog sučelja aplikacije odabirom *File -> Analysis.* Time se otvara zaseban prozor koji omogućuje upis imena tablice kojoj se pristupa te imena spremljenog signala. Klikom na gumb *OK* željeni podatci se dohvaćaju te se nad njima izvršava jednostavna statistička analiza, prikazujući srednju vrijednost, minimalni i maksimalnu vrijednost, standardnu devijaciju, medijan te varijancu, kao što je i vidljivo na slici 4.15.



Slika 4.15. Sučelje za statističku analizu odabranog signala

4.2. Nedostatci i mogućnosti nadogradnje

Aplikacija izrađena u sklopu ovoga rada omogućuje izradu konfiguracije postrojenja koji imaju uspostavljene poslužitelje koristeći OPC DA protokol. OPC DA protokol je stariji protokol kojega sve više zamjenjuje ujedinjena arhitektura, OPC UA. Nadogradnja ove aplikacije bi omogućila spajanje na poslužitelje koji su modernizirani i temeljeni na UA protokolu.

Nedostatak ovoga rješenja leži u nemogućnosti povezivanja s uređajima i poslužiteljima koji se ne temelje na OPC protokolu. No, kako je i prethodno navedeno, OPC se koristi kako se ne bi morala razvijati zasebna programska podrška koja odgovara isključivo jednom proizvođaču.

Nastavno na izrađeno rješenje, moguće je izraditi napredniji sustav za analizu te izraditi sustav za vizualizaciju podataka upisanih u bazu koji će davati detaljniji uvid praćenja vrijednosti signala kao i njihove interakcije i međuovisnosti. Također, nadogradnju bi obuhvaćala i integracija

Windows Servisa, koji vrši upisivanje u bazu podataka, s glavnom aplikacijom čime bi se formiralo jedinstveno i samostalno rješenje.

5. ZAKLJUČAK

Unatoč brojnima aplikacijama koje se bave povezivanjem s industrijskim postrojenjima, kao i proučavanjem njihovih signala, nedostatak sveobuhvatnih aplikacija koje omogućuju sve značajke potrebne jednome operateru predstavlja inspiraciju ovoga rada.

Ovime radom izrađena je aplikacija koja omogućuje operaterima pronalazak i spajanje na OPC poslužitelje lokalnog ili udaljenog računala, uvid u njihovu strukturu i pronalazak OPC signala potrebnih za izradu konfiguracije. Izradom konfiguracije odabiru se signali čije se vrijednosti spremaju u bazu podataka u svrhu povećanja nadzora i kontrole proizvodnog procesa.

Zadani zahtjevi aplikacije su u potpunosti ispunjeni i pokriveni s izrađena dva jednostavna sučelja. Prvim je omogućen pregled svih dostupnih poslužitelja kao i njihovih informacija dok je drugim dan jasan pregled strukture odabranog poslužitelja, dostupnih signala kao i njihovih detaljnih informacija. Također je omogućeno promatranje stvarnih vrijednosti signala čime se utvrđuje njihova ispravnost.

Završetkom rada kreirana je aplikacija koju se može nadograđivati i proširivati različitim funkcionalnostima, primjerice omogućavanjem naprednije statističke analize spremljenih podataka kao i njihove vizualizacije.

LITERATURA

[1] S. I. Shafiq, E. Szczerbicki, C. Sanin, „Proposition of the methodology for Dana Acquisition, Analysis and Visualization in support of Industry 4.0“, Procedia Computer Science, 2019.

[2] R. Geissbauer, S. Schrauf, V. Koch, S. Kuge, „Industry 4.0 – Opportunities and Challenges of the Industrial Internet“, PricewaterhouseCoopers, 2014.

[3] T. H.-J. Uhlemann, C. Schock, C. Lehmann, S. Freiberger, R. Steinhilper, „The Digital Twin: Demonstrating the potential of real time dana acquisition in production systems“, Procedia Manufacturing, Germany, 2017.

[4] Bajer, Marcin, „Control Systems Integration using OPC Standard“, lipanj 2008.

[5] Matrikon OPC Desktop Historian [online], Honeywell International Inc., dostupno na: https://www.matrikonopc.com/products/opc-archiving/opc-desktop-historian.aspx [09.09.2022.]

[6] Industrial Communication Toolbox [online], The MathWorks, Inc., dostupno na: https://uk.mathworks.com/products/industrial-communication.html [09.09.2022.]

[7] OPC DA Client [online], PTC Inc., dostupno na: https://www.kepware.com/en-us/products/kepserverex/drivers/opc-da-client/ [09.09.2022.]

[8] Matrikon OPCExplorer [online], Honeywell International Inc., dostupno na: https://www.matrikonopc.com/products/opc-desktop-tools/opc-explorer.aspx [09.09.2022.]

[9] G. Lucassen, „Understanding User Stories“, SIKS Dissertation Series, 2017.

[10] K. H. Johnson, A.-K. Nordekvist, „Client for OPC History Data Access“, Departments of Computer Engineering of Mälardalen University, Västerås, Švedska, 2001.

[11] Elektrotehnički fakultet u Sarajevu, „Specijalna poglavlja softverskih sistema, OPC specifikacije“, rujan 2004.

[12] Z. X. Guo, X. Q. Xie and Z. G. Ni, "The application of OPC DA in factory data acquisition," 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE), 2012, pp. 209-212, kolovoz 2012.

[13] OPC Classic [online], OPC Foundation, dostupno na: https://opcfoundation.org/about/opc-technologies/opc-classic/ [09.09.2022.]

[14] OPC Data Access Server [online], dostupno na: https://quercus-lab.com/opc-data-acces-server/index.html [09.09.2022.]

[15] OPC Foundation, „OPC Data Access Automation Specification“, Version 2.02., veljača 1999.

[16] Visual studio: IDE and Code Editor for Software developers and Teams [online], Microsoft, dostupno na: https://visualstudio.microsoft.com/ [09.09.2022.]

[17] What is .Net [online], Microsoft, dostupno na: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet [09.09.2022.]

[18] A tour of C# language [online], Microsoft, dostupno na: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ [09.09.2022.]

[19] Introduction to Oracle Database [online], Oracle, dostupno na: https://docs.oracle.com/en/database/oracle/oracle-database/19/cncpt/introduction-to-oracle-database.html#GUID-A42A6EF0-20F8-4F4B-AFF7-09C100AE581E [09.09.2022.]

[20] OPC Client/Server Architecture [online], General Electric Company, dostupno na: https://www.ge.com/digital/documentation/cimplicity/version10/oxy_ex-2/device_communications/topics/g_cimplicity_device_communications_opc_client_server_architecture.html [09.09.2022.]

[21] OPC Server and OPC Client Relationship [online], Honeywell International Inc., dostupno na: https://www.matrikonopc.com/resources/opc-server.aspx [09.09.2022.]

[22] H. Gomaa, „Software modeling and design: UML, Use Cases, Patterns, and Software Architectures“, Cambridge University Press, New York, 2011.

SAŽETAK

Ubrzani razvoj tehnologije omogućio je industrijskim postrojenjima prikupljanje informacija o samom proizvodnom procesu. Nastavno tome, kreirane su brojne aplikacije koje omogućuju spajanje na poslužitelje postrojenja i prikupljanje njegovih informacija.

Ovime radom analizirane su postojeće aplikacije temeljene na OPC protokolu za izradu konfiguracije postrojenja. Definirani su korisnički zahtjevi koji će operateru pružati sve potrebne mogućnosti. Zatim je implementirano programsko rješenje koristeći C# programski jezik kao i .Net razvojno okruženje.

Rješenje omogućuje pronalazak i spajanje na lokalne ili udaljene OPC poslužitelje kojima se promatra unutarnja struktura u svrhu pronalaska dostupnih OPC signala. Odabirom željenih signala kreira se konfiguracija postrojenja. Također, omogućeno je promatranje stvarnih vrijednosti signala kao i njihovo spremanje u Oracle bazu podataka.

Ključne riječi: aplikacija, konfiguracija, OPC, postrojenje

Commented [ZK61]: Poredajte po abecedi.

ABSTRACT

Commented [ZK62]: abstract staviti na novu stranicu

Framework for configuring, collecting and processing data from industrial plants

Commented [ZK63]: Naslov iz Mak-a

The accelerated development of technology has enabled industrial plants to collect information about the production process itself. Subsequently, numerous applications have been created which enable connecting to the plant's servers and collecting its information.

This paper analysed the existing applications based on the OPC protocol for creating the plants configuration. User requirements have been defined that will provide the operator with all the necessary options. A programming solution was implemented using the C# programming language as well as the .Net development environment.

The solution enables finding and connecting to local or remote OPC servers that observe the internal structure in order to find available OPC signals. By selecting the desired signals, the plant configuration is created. It is also possible to observe the actual values of the signals as well as save them in the Oracle database.

Key words: application, configuration, OPC, plant

ŽIVOTOPIS

Domagoj Rojnić rođen je 27. lipnja 1998. godine u Osijeku. Osnovnu školu Višnjevac završava 2013. čime upisuje Prirodoslovno-matematičku gimnaziju (III. gimnazija) u Osijeku. Završenom srednjom školom 2017., iste godine upisuje preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek u Osijeku. Preddiplomski studij završava 2020. godine nakon čega te iste godine upisuje diplomski sveučilišni studij informacijskih i podatkovnih znanosti na istome fakultetu.

Potpis autora:

_____

PRILOZI

[P1] DVD s programskim kodom